IBM SPSS Modeler 15
CLEF-Entwicklerhandbuch

Hinweis: Lesen Sie zunächst die allgemeinen Informationen unter Hinweise auf S. , bevor Sie dieses Informationsmaterial sowie das zugehörige Produkt verwenden.

Diese Ausgabe bezieht sich auf IBM SPSS Modeler 15 und alle nachfolgenden Versionen sowie Anpassungen, sofern dies in neuen Ausgaben nicht anders angegeben ist.

Vorwort

IBM® SPSS® Modeler ist die auf Unternehmensebene einsetzbare Data-Mining-Workbench von IBM Corp.. Mit SPSS Modeler können Unternehmen und Organisationen die Beziehungen zu ihren Kunden bzw. zu den Bürgern durch ein tief greifendes Verständnis der Daten verbessern. Organisationen benutzen die mithilfe von SPSS Modeler gewonnenen Erkenntnisse zur Bindung profitabler Kunden, zur Ermittlung von Cross-Selling-Möglichkeiten, zur Gewinnung neuer Kunden, zur Ermittlung von Betrugsfällen, zur Reduzierung von Risiken und zur Verbesserung der Verfügbarkeit öffentlicher Dienstleistungen.

Die visuelle Benutzeroberfläche von SPSS Modeler erleichtert die Anwendung des spezifischen Geschäftswissens der Benutzer, was zu leistungsstärkeren Vorhersagemodellen führt und die Zeit bis zur Lösungserstellung verkürzt. SPSS Modeler bietet zahlreiche Modellierungsverfahren, beispielsweise Algorithmen für Vorhersage, Klassifizierung, Segmentierung und Assoziationserkennung. Nach der Modellerstellung ermöglicht IBM® SPSS® Modeler Solution Publisher die unternehmensweite Bereitstellung für Entscheidungsträger oder in einer Datenbank.

Über IBM Business Analytics

IBM Business Analytics-Software bietet vollständige, einheitliche und genaue Informationen, auf die Entscheidungsträger vertrauen, um die Unternehmensleistung zu steigern. Ein umfassendes Portfolio von Anwendungen für Unternehmensinformationen, Vorhersageanalysen, Verwaltung der Finanzleistung und Strategie sowie Analysen bietet sofort klare und umsetzbare Einblicke in die aktuelle Leistung und ermöglicht die Vorhersage zukünftiger Ergebnisse. In Kombination mit umfassenden Branchenlösungen, bewährten Vorgehensweisen und professionellen Dienstleistungen können Unternehmen jeder Größe optimale Produktivität erreichen, die Entscheidungsfindung zuverlässig automatisieren und bessere Ergebnisse erzielen.

Als Teil dieses Portfolios unterstützt die IBM SPSS Predictive Analytics-Software Unternehmen dabei, zukünftige Ereignisse vorherzusagen und aktiv auf diese Erkenntnisse zu reagieren, um bessere Geschäftsergebnisse zu erzielen. Kunden aus den Bereichen Wirtschaft, Behörden und Bildung aus aller Welt verlassen sich auf die IBM SPSS-Technologie. Sie bringt Ihnen beim Gewinnen, Halten und Ausbauen neuer Kundenbeziehungen einen Wettbewerbsvorteil und verringert gleichzeitig das Betrugs- sowie andere Risiken. Durch Integration der IBM SPSS-Software in den täglichen Betrieb können diese Unternehmen qualifizierte Vorhersagen treffen und dadurch die Entscheidungsfindung so ausrichten und automatisieren, dass Geschäftsziele erreicht werden und ein messbarer Wettbewerbsvorteil entsteht. Wenn Sie weitere Informationen wünschen oder einen Mitarbeiter kontaktieren möchten, ist dies unter *http://www.ibm.com/spss* möglich.

Technischer Support

Kunden mit Wartungsvertrag können den technischen Support in Anspruch nehmen. Kunden können sich an den technischen Support wenden, wenn sie Hilfe bei der Arbeit mit IBM Corp.-Produkten oder bei der Installation in einer der unterstützten Hardware-Umgebungen benötigen. Die Kontaktdaten des Technischen Supports finden Sie auf der IBM Corp.-Website

unter *http://www.ibm.com/support*. Sie müssen bei der Kontaktaufnahme Ihren Namen, Ihre Organisation und Ihre Supportvereinbarung angeben.

Inhalt

1 Übersicht 1

Einführung in CLEF . . . 1
Systemarchitektur . . . 1
Clientseitige Komponenten . . . 1
Serverseitige Komponenten . . . 2
Funktionen von CLEF . . . 3
Spezifikationsdatei . . . 4
Knoten . . . 4
Data Model . . . 4
Eingabe- und Ausgabedateien . . . 5
Anwendungsprogrammierschnittstellen (APIs) . . . 5
Dateistruktur . . . 6
Clientseitige Komponenten . . . 6
Serverseitige Komponenten . . . 8

2 Knoten 10

Überblick über die Knoten . . . 10
Datenleserknoten . . . 11
Datentransformationsknoten . . . 12
Modellerstellungsknoten . . . 12
Dokumenterstellungsknoten . . . 14
Modellanwendungsknoten . . . 14
Datenschreiberknoten . . . 15
Menüs, Symbolleisten und Paletten . . . 15
Menüs und Untermenüs . . . 15
Symbolleisten . . . 16
Paletten und Unterpaletten . . . 17
Erstellen von Knotensymbolen . . . 18
Rahmen . . . 19
Hintergründe . . . 20
Grafikanforderungen . . . 20
Erstellen benutzerdefinierter Bilder . . . 21
Hinzufügen der Bilddateien in die Knotenspezifikation . . . 21
Erstellen von Dialogfeldern . . . 22
Informationen zu Knotendialogfeldern . . . 22
Richtlinien für die Erstellung von Dialogfeldern . . . 23
Dialogfeldkomponenten . . . 24
Erstellen von Ausgabefenstern . . . 30

3 CLEF-Beispiele 33

Informationen zu den Beispielen . . . 33
Aktivieren der Beispiele . . . 33
Datenleserknoten (Apache Log Reader) . . . 34
Datentransformationsknoten (URL Parser) . . . 35
Dokumenterstellungsknoten (Web Status Report) . . . 36
Modellerstellungsknoten (Interaction) . . . 36
Untersuchen der Spezifikationsdateien . . . 37
Untersuchen des Quellcodes . . . 38
Entfernen der Beispiele . . . 38

4 Spezifikationsdatei 39

Überblick über Spezifikationsdateien . . . 39
Beispiel für eine Spezifizierungsdatei . . . 40
XML-Deklaration . . . 42
Erweiterungselement . . . 42
Abschnitt 'Erweiterungsdetails' . . . 42
Abschnitt 'Ressourcen' . . . 43
- Pakete . . . 44
- Jar-Dateien . . . 44
- Freigegebene Bibliotheken . . . 45
- Hilfeinformationen . . . 46

Abschnitt 'Gemeinsame Objekte' . . . 46
- Eigenschaftstypen . . . 47
- Eigenschaftssätze . . . 48
- Containertypen . . . 49
- Aktionen . . . 51
- Kataloge . . . 52

Abschnitt 'Benutzeroberfläche (Paletten)' . . . 55
Abschnitt 'Objektdefinition' . . . 61
- Objekt-ID . . . 62
- Modellerstellung (ModelBuilder) . . . 65
- Dokumentersteller (DocumentBuilder) . . . 65
- Modell-Provider (ModelProvider) . . . 65
- Eigenschaften . . . 66
- Container (Containers) . . . 68
- Benutzeroberfläche . . . 69

Ausführung . . . 70
Ausgabedatenmodell (OutputDataModel) . . . 75
Konstruktoren . . . 76
Allgemeine Funktionen . . . 76
Werttypen . . . 76
Evaluierte Zeichenketten . . . 81
Vorgänge . . . 81
Felder und Feldmetadaten . . . 87
Feld-Sets . . . 88
Rollen . . . 89
Logische Operatoren . . . 90
Bedingungen . . . 91
Verwenden von CLEF-Knoten in Skripts . . . 96
Erhaltung der Abwärtskompatibilität . . . 97

5 Erstellen von Modellen und Dokumenten 99

Einführung in die Modell- und Dokumenterstellung . . . 99
Modelle . . . 99
Dokumente . . . 100
Konstruktoren . . . 100
Erstellen von Modellen . . . 100
Modellerstellung (ModelBuilder) . . . 101
Modellausgabe . . . 110
Erstellen interaktiver Modelle . . . 112
Automatisierte Modellierung . . . 116
Anwenden von Modellen . . . 124
Erstellen von Dokumenten . . . 124
Dokumentersteller (DocumentBuilder) . . . 124
Dokumentausgabe . . . 125
Verwenden von Konstruktoren . . . 126
CreateModelOutput (Erstellen der Modellausgabe) . . . 127
CreateDocumentOutput (Erstellen der Dokumentausgabe) . . . 128
CreateInteractiveModelBuilder (Erstellen der interaktiven Modellerstellung) . . . 129
CreateModelApplier (Erstellen des Modellanwenders) . . . 129

6 Erstellen von Benutzeroberflächen 131

Info zu Benutzeroberflächen . . . 131
Abschnitt 'Benutzeroberfläche' . . . 134

Symbole . . . 137
Steuerelemente . . . 138
Menüs . . . 138
Menüelemente . . . 140
Symbolleistenelemente . . . 141
Beispiel: Hinzufügen zum Hauptfenster . . . 142
Registerkarten . . . 143
Zugriffstasten und Tastenkombinationen . . . 145
Fensterspezifizierungen . . . 147
Textbrowserfenster . . . 147
Erweiterungsobjektfenster . . . 149
Eigenschaftsfenster . . . 150
Modell-Viewer-Fenster . . . 152
Spezifikationen von Eigenschaftssteuerelementen . . . 154
Steuerelemente der Benutzeroberflächenkomponente . . . 154
Steuerelemente des Eigenschaftsfensters . . . 158
Controller . . . 161
Layouts für Eigenschaftssteuerelemente . . . 187
Standardsteuerelementlayout . . . 187
Benutzerdefiniertes Steuerelementlayout . . . 189
Benutzerdefinierte Ausgabefenster . . . 204

7 Hinzufügen eines Hilfesystems 206

Hilfesystemtypen . . . 206
HTML Help . . . 206
JavaHelp . . . 207
Implementieren eines Hilfesystems . . . 208
Festlegen von Speicherort und Typ des Hilfesystems . . . 208
Festlegen des anzuzeigenden Hilfethemas . . . 209

8 Lokalisierung und Eingabehilfen 211

Einführung . . . 211
Lokalisierung . . . 211
Eigenschaftendateien . . . 212
Hilfedateien . . . 216
Testen eines lokalisierten CLEF-Knotens . . . 217
Zugriffsmöglichkeiten . . . 218

9 Programmierung 220

Informationen zur Programmierung von CLEF-Knoten . . . 220
Dokumentation zu den CLEF-APIs . . . 220
Clientseitige API . . . 221
Clientseitige API-Klassen . . . 221
Verwenden der clientseitigen API . . . 221
Predictive Server-API (PSAPI) . . . 222
Serverseitige API . . . 222
Architektur . . . 222
Servicefunktionen . . . 223
Callback-Funktionen . . . 225
Prozessfluss . . . 227
Serverseitige API-Funktionen . . . 229
Fehlerbehandlung . . . 245
XML Parsing-API . . . 246
Verwenden der serverseitigen API . . . 246
Richtlinien für die Server-seitige Programmierung . . . 246

10 Test und Verteilung 251

Testen von CLEF-Erweiterungen . . . 251
Testen einer CLEF-Erweiterung . . . 251
Fehlersuche in einer CLEF-Erweiterung . . . 252
Verteilen von CLEF-Erweiterungen . . . 255
Installieren von CLEF-Erweiterungen . . . 256
Deinstallieren von CLEF-Erweiterungen . . . 256

Anhänge

A CLEF XML-Schema 257

CLEF-Elementreferenz . . . 257
Action Element . . . 257
ActionButton Element . . . 258
Actions Element . . . 258
AddField Element . . . 259
And Element . . . 263

Attribute Element 263
BinaryFormat Element 264
Catalog Element 264
Catalogs Element 265
ChangeField Element 265
CheckBoxControl Element 269
CheckBoxGroupControl Element 270
ClientDirectoryChooserControl Element 271
ClientFileChooserControl Element 272
ComboBoxControl Element 273
Command Element 275
CommonObjects Element 275
Condition Element 276
Constraint Element 278
Constructors Element 279
Container Element 279
ContainerFile Element 280
ContainerTypes Element 280
Controls Element 280
CreateDocument Element 281
CreateDocumentOutput Element 282
CreateInteractiveDocumentBuilder Element 282
CreateInteractiveModelBuilder Element 283
CreateModel Element 284
CreateModelApplier Element 285
CreateModelOutput Element 286
DatabaseConnectionValue Element 287
DataFile Element 288
DataFormat Element 288
DataModel Element 288
DBConnectionChooserControl Element 295
DBTableChooserControl Element 296
DefaultValue Element 297
DelimitedDataFormat Element 299
DisplayLabel Element 300
DocumentBuilder Element 300
DocumentOutput Element 301
DocumentType Element 302
Enabled Element 303
Enumeration Element 304
ErrorDetail Element 304
Executable Element 306
Execution Element 307
Extension Element 307
ExtensionDetails Element 308

ExtensionObjectPanel Element 308
Field Element 309
FieldFormats Element 312
FieldGroup Element 314
FieldGroups Element 315
FileFormatType Element 316
FileFormatTypes Element 317
ForEach Element 317
Icon Element 318
Icons Element 319
InputFiles Element 319
InteractiveDocumentBuilder Element 320
InteractiveModelBuilder Element 321
Layout Element 322
License Element 324
ListValue Element 324
MapValue Element 325
Menu Element 329
MenuItem Element 331
MissingValues Element 332
ModelBuilder Element 335
ModelOutput Element 344
ModelProvider Element 345
ModelType Element 345
ModelViewerPanel Element 346
Module Element 347
MultiFieldChooserControl Element 347
MultiItemChooserControl Element 349
Node Element 350
Not Element 352
NumberFormat Element 353
NumericInfo Element 354
Option Element 354
OptionCode Element 355
Or Element 355
OutputDataModel Element 356
OutputFiles Element 357
Palette Element 357
Parameters Element 358
PasswordBoxControl Element 360
Properties Element 361
PropertiesPanel Element 361
PropertiesSubPanel Element 363
Property Element 364
PropertyControl Element 366

PropertyGroup Element 367
PropertySets Element 367
PropertyType Element 368
PropertyTypes Element 369
RadioButtonGroupControl Element. 369
Range Element 371
Range Element 371
RemoveField Element. 371
Resources Element 372
Run Element 374
SelectorPanel Element 375
ServerDirectoryChooserControl Element 376
ServerFileChooserControl Element. 377
SetContainer Element 378
SetProperty Element 379
SingleFieldChooserControl Element 379
SingleFieldValueChooserControl Element. 381
SingleItemChooserControl Element 382
SpinnerControl Element 383
SPSSDataFormat Element 384
StaticText Element. 385
StatusCode Element. 385
StatusCodes Element. 386
StatusDetail Element 386
Structure Element 388
StructuredValue Element 389
SystemControls Element 391
Tab Element. 392
TabbedPanel Element 392
TableControl Element. 393
Tabs Element. 394
TextAreaControl Element 395
TextBoxControl Element. 396
TextBrowserPanel Element 397
ToolbarItem Element 398
UserInterface Element. 399
UTF8Format Element 399
Value Element 399
Values Element 400
Values Element 401
Visible Element 402
Erweitert Typen 403

B Hinweise *404*

Index *407*

Kapitel 1

Übersicht

Einführung in CLEF

Component-Level Extension Framework (CLEF) ist ein Mechanismus, mit dem der Standardfunktionalität von IBM® SPSS® Modeler benutzerdefinierte Erweiterungen hinzugefügt werden können. Eine Erweiterung enthält in der Regel eine gemeinsam verwendete Bibliothek, z. B. eine Datenverarbeitungsroutine oder einen Modellierungsalgorithmus, die SPSS Modeler hinzugefügt und als neuer Menüeintrag oder als neuer Knoten in der Knotenpalette zur Verfügung gestellt wird.

Dazu benötigt SPSS Modeler Informationen über das benutzerdefinierte Programm, unter anderem dessen Namen, die an SPSS Modeler übergebenen Parameter und Informationen darüber, wie SPSS Modeler dem Programm Optionen und dem Benutzer Ergebnisse bereitstellen soll. Diese Informationen stellen Sie in einer XML-Datei, der **Spezifikationsdatei**, bereit. SPSS Modeler überträgt die Informationen dieser Datei in einen neuen Menüeintrag oder eine Knotendefinition.

CLEF bietet unter anderem folgende Vorteile:

- Es stellt eine einfach zu verwendende, äußerst flexible und robuste Umgebung bereit, mit der Systementwickler, Berater und Endbenutzer neue Funktionen in SPSS Modeler integrieren können.
- Es stellt sicher, dass Erweiterungsmodule genauso aussehen und sich genauso verhalten wie systemeigene SPSS Modeler-Module.
- Es stellt sicher, dass die Geschwindigkeit und Effizienz der Erweiterungsknoten so nahe wie möglich an die der systemeigenen SPSS Modeler-Knoten herankommen.

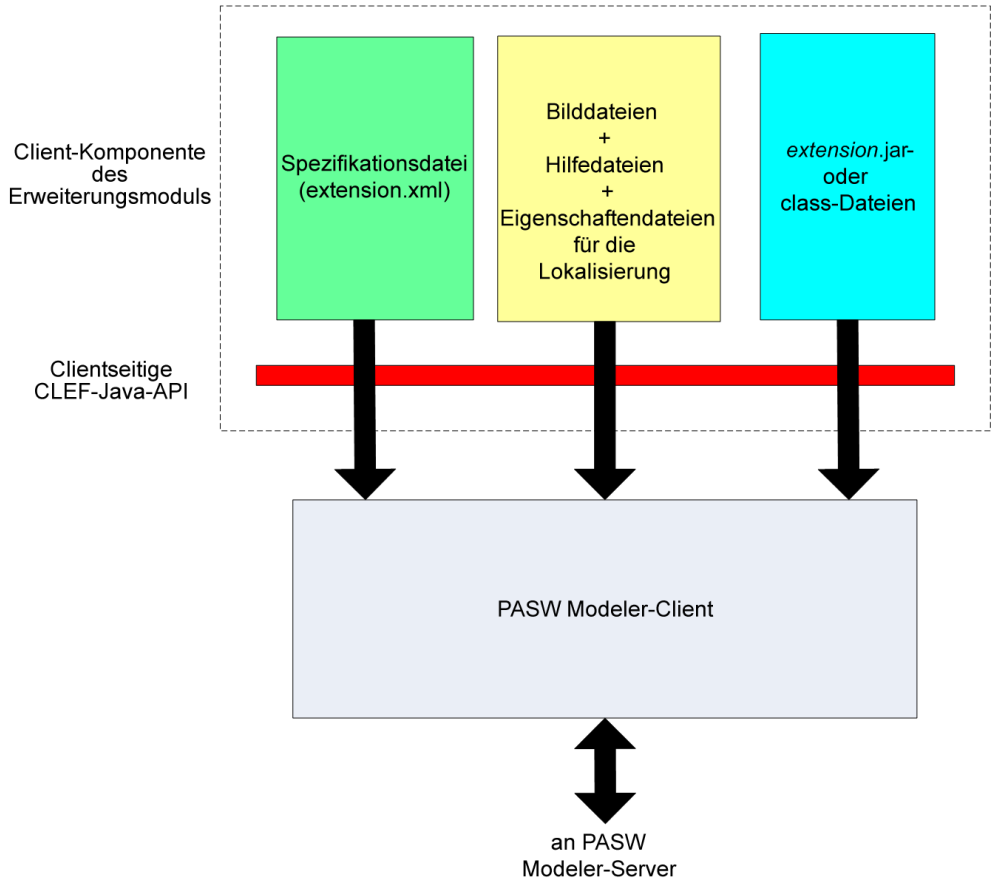
Systemarchitektur

Wie IBM® SPSS® Modeler verwendet CLEF eine zweischichtige Client-/Serverarchitektur, wobei sich die Schichten auf dem gleichen oder auf zwei verschiedenen Computern befinden können.

Clientseitige Komponenten

Die Komponenten der Clientschicht werden in nachfolgender Abbildung gezeigt.

Abbildung 1-1
Clientseitige Komponenten

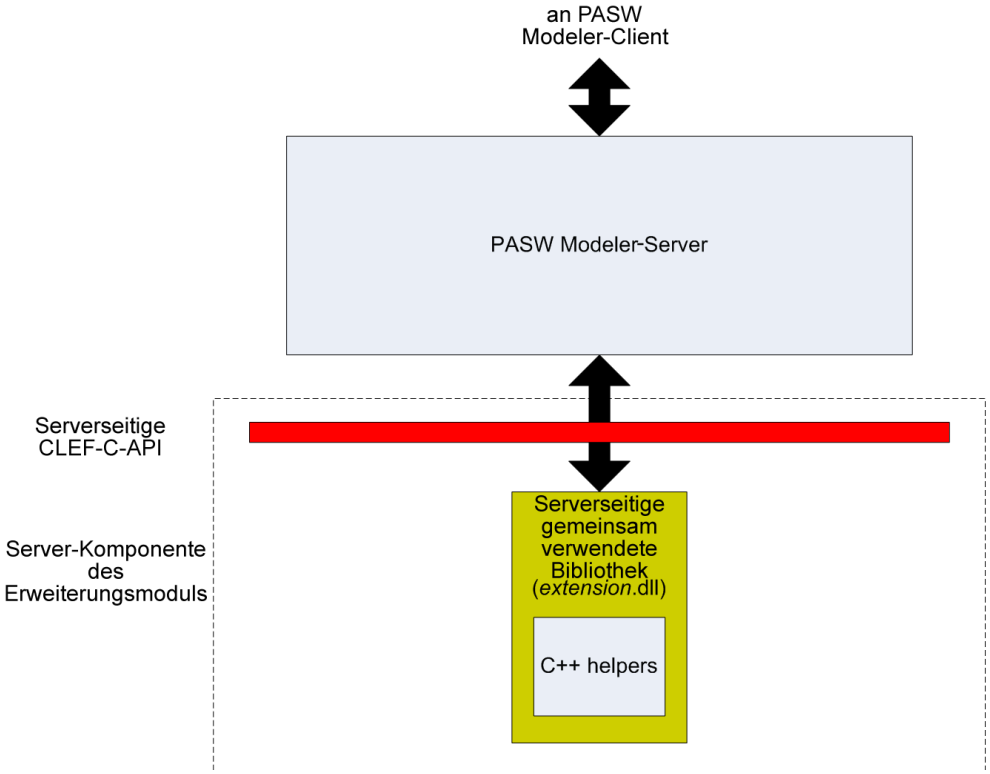


- **Spezifikationsdatei:** Definiert die Eigenschaften, Formate, Datenmodelländerungen, Steuerelemente und andere Merkmale der Erweiterung.
- **Bilddateien:** Enthalten die Bilder für die Darstellung eines Knotens in der Erweiterung.
- **Hilfedateien:** Enthalten die Hilfeinformationen für die Erweiterung.
- **Eigenschaftendateien:** Enthalten Textstrings mit den Namen, Beschriftungen und Meldungen, die von der Erweiterung auf dem Bildschirm angezeigt werden.
- **Java-Dateien (.jar oder .class):** Enthalten die von der Erweiterung verwendeten Java-Ressourcen.
- **Java-API (Anwendungsprogrammierschnittstelle):** Kann von Erweiterungen verwendet werden, für die Steuerelemente, Benutzeroberflächenkomponenten oder Interaktivitätselemente erforderlich sind, die nicht in der Spezifikationsdatei bereitgestellt werden können.

Serverseitige Komponenten

Die Komponenten der Serverschicht werden in nachfolgender Abbildung gezeigt.

Abbildung 1-2
Serverseitige Komponenten



- **C-basierte API für gemeinsam verwendete Bibliotheken:** Definiert Aspekte wie die Festlegung und das Abrufen von Ausführungseinstellungen, die Persistenz dieser Einstellungen, Ausführungsrückmeldungen, die Auftragssteuerung (z. B. Unterbrechung der Ausführung), die SQL-Generierung und die zurückgegebenen Objekte.
- **Serverseitige gemeinsam verwendete Bibliothek:** Eine DLL (dynamische Verknüpfungsbibliothek), die die Knotenausführung unterstützt. Die C++-Hilfselemente sind Wrapper für einige der C-basierten APIs, die als Quellcode bereitgestellt werden und problemlos in C++-Module für CLEF kompiliert werden können.

Funktionen von CLEF

In den folgenden Abschnitten werden die wichtigsten Funktionen von CLEF vorgestellt:

- Spezifikationsdatei
- Knoten
- Datenmodell
- Eingabe- und Ausgabedateien
- Anwendungsprogrammierschnittstellen (APIs)

Spezifikationsdatei

Bei der Spezifikationsdatei von CLEF handelt es sich um eine XML-Datei mit strukturierten Spezifikationen, die das Verhalten der neuen Erweiterung beschreiben. Eine Spezifikationsdatei beschreibt folgende Komponenten:

- Die von der Erweiterung benötigten, gemeinsam verwendeten Ressourcen (z. B. lokalisierte Textbündel und serverseitige gemeinsam verwendete Bibliotheken)
- Allgemeine Definitionen wie Datei- oder Eigenschaftstypen
- Neue Objekte, wie Knoten und Ausgabemodelle, die vom Endbenutzer verwendet werden können

Beim Starten von IBM® SPSS® Modeler werden die Spezifikationsdateien aus ihrem jeweiligen Speicherort geladen, damit die in den Dateien definierten Funktionen sofort zur Verfügung stehen.

Weitere Informationen finden Sie in Kapitel 4, *Spezifikationsdatei* auf S. 39.

Knoten

Wenn Sie IBM® SPSS® Modeler eine Erweiterung hinzufügen, die einen neuen Knoten implementieren soll, müssen Sie zunächst entscheiden, welchen Knotentyp Sie erstellen möchten (ob der Knoten z. B. ein Modell generieren oder lediglich Daten transformieren soll). Für weitere Informationen siehe Thema Überblick über die Knoten in Kapitel 2 auf S. 10.

Nachdem Sie die Spezifikationsdatei und alle erforderlichen Java-Klassen und gemeinsam verwendeten Bibliotheken erstellt haben, kopieren Sie die Dateien in bestimmte Verzeichnisse, in denen SPSS Modeler die Dateien lesen kann. Wenn Sie SPSS Modeler danach das nächste Mal starten, wird der neue, betriebsbereite Knoten der entsprechenden Palette hinzugefügt.

Data Model

Das **Datenmodell** stellt die Struktur der Daten dar, die durch den IBM® SPSS® Modeler-Stream fließen. Da das Modell die Daten an diesem Punkt des Streams beschreibt, entspricht es den im Typknoten angezeigten Informationen. Es listet die Namen der an einem bestimmten Punkt vorhandenen Felder auf und beschreibt deren Typ.

Beachten Sie beim Hinzufügen eines Knotens zu SPSS Modeler, wie das an den Knoten übergebene Datenmodell das Verhalten dieses Knotens beeinflusst. Ein Ableitungsknoten nimmt beispielsweise ein Eingabedatenmodell an, fügt ein neues Feld hinzu und erstellt ein Ausgabedatenmodell, das an den nächsten Knoten im SPSS Modeler-Stream übergeben wird. Ein Diagrammknoten hingegen nimmt ein Eingabedatenmodell an, erstellt aber kein Ausgabedatenmodell, da keine Daten an nachfolgende Knoten weitergeleitet werden müssen. SPSS Modeler muss darüber informiert werden, was mit dem Datenmodell geschieht, damit nachfolgende Knoten korrekte Informationen über die verfügbaren Felder angeben können. Mit den Datenmodellinformationen der Spezifikationsdatei erhält SPSS Modeler die notwendigen Informationen, um das Datenmodell im gesamten Stream konsistent zu halten.

Je nachdem, ob die Daten in den Knoten, aus dem Knoten oder durch den Knoten fließen, muss die Spezifikationsdatei das Datenmodell für die Eingabe, die Ausgabe oder beides beschreiben. Ein CLEF-Knoten kann das Datenmodell auf zweierlei Weise beeinflussen: durch Hinzufügen

neuer Felder zu den Feldern, die an den Knoten übergeben werden, oder durch Ersetzen der an den Knoten übergebenen Felder durch neue Felder, die vom Programm selbst erzeugt werden. Die Auswirkungen eines CLEF-Knotens auf das Datenmodell werden im Element OutputDataModel der Spezifikationsdatei definiert. Für weitere Informationen siehe Thema Ausgabedatenmodell (OutputDataModel) in Kapitel 4 auf S. 75.

Eingabe- und Ausgabedateien

Sie können festlegen, dass vor der Ausführung eines CLEF-Knotens eine oder mehrere temporäre Dateien generiert werden. Diese Dateien werden als **Eingabedateien** bezeichnet, da sie die Eingabe für die Ausführung des Knotens auf dem Server bereitstellen. Ein Modellerstellungsknoten kann beispielsweise über einen Modellcontainer verfügen, dessen Inhalt bei der Ausführung des Knotens in eine bestimmte Eingabedatei übertragen wird. Für weitere Informationen siehe Thema Eingabedateien (InputFiles) in Kapitel 4 auf S. 71.

Andere temporäre Dateien werden während der Ausführung des Knotens auf dem Server generiert. Diese enthalten z. B. das Ergebnis eines Modell- oder Dokumenterstellungsknotens. Diese Dateien werden als **Ausgabedateien** bezeichnet. Sie werden nach der Knotenausführung an den Client zurückgeleitet. Für weitere Informationen siehe Thema Ausgabedateien in Kapitel 4 auf S. 72.

Anwendungsprogrammierschnittstellen (APIs)

Je nachdem, welche Funktion Ihre Erweiterung haben soll, müssen Sie zu deren Entwicklung unter Umständen auch eine Anwendungsprogrammierschnittstelle (API) verwenden. Für die Festlegung der Verarbeitungsschritte einer einfachen Datentransformation reicht unter Umständen die Spezifikationsdatei aus. Für erweiterte Funktionen müssen Sie jedoch eine oder mehrere der verfügbaren APIs hinzuziehen:

- Clientseitige CLEF-API
- Serverseitige CLEF-API
- Predictive Server-API (PSAPI)

Bei der **clientseitigen API** von CLEF handelt es sich um eine Java-API, die von Erweiterungen verwendet werden kann, für die Steuerelemente, Benutzeroberflächenkomponenten oder Interaktivitätselemente erforderlich sind, die nicht in der Spezifikationsdatei bereitgestellt werden.

Bei der **serverseitigen API** von CLEF handelt es sich um eine C-basierte API, in der Aspekte wie die Festlegung und das Abrufen von Ausführungseinstellungen, die Persistenz dieser Einstellungen, Ausführungsrückmeldungen, die Auftragssteuerung (z. B. Unterbrechung der Ausführung), die SQL-Generierung und die zurückgegebenen Objekte definiert werden.

Bei der **Predictive Server API** handelt es sich um eine Java-API, die Anwendungen, die Data-Mining-Funktionen und Mechanismen für die Vorhersageanalyse benötigen, die Funktionalität von IBM® SPSS® Modeler bereitstellt.

Weitere Informationen finden Sie in Kapitel 9, *Programmierung* auf S. 220.

Dateistruktur

Eine CLEF-Erweiterung besteht aus zwei Komponentensätzen:

- Clientseitige Komponenten
- Serverseitige Komponenten

Zu den **clientseitigen Komponenten** gehören die Spezifikationsdatei der Erweiterung, Java-Klassen und .jar-Dateien, Eigenschaftenbündel mit lokalisierbaren Ressourcen sowie Bild- und Hilfedateien.

Die **serverseitigen Komponenten** umfassen gemeinsam verwendete Bibliotheken und DLLs, die bei der Ausführung eines Erweiterungsknotens erforderlich sind.

Clientseitige Komponenten

Clientseitige Komponenten werden im Installationsverzeichnis von IBM® SPSS® Modeler im Ordner \ext\lib installiert. Es gibt folgende clientseitige Komponenten:

- Spezifikationsdatei
- Java-Klassen und .jar-Dateien
- Eigenschaftendateien
- Bilddateien
- Hilfedateien

Erweiterungsordner

Jede Erweiterung wird in einem eigenen **Erweiterungsordner** direkt unter dem Verzeichnis \ext\lib gespeichert.

Für Erweiterungsordner wird folgendes Namensschema empfohlen:

providerTag.id

Dabei ist *providerTag* die Kennung des Providers aus dem Element ExtensionDetails der Spezifikationsdatei und *id* ist die Erweiterungskennung aus dem gleichen Element.

Beispiel: Das Element ExtensionDetails beginnt wie folgt:

```
<ExtensionDetails providerTag="myco" id="sorter" ... />
```

Der Erweiterungsordner sollte dann den Namen myco.sorter erhalten.

Spezifikationsdatei

Die Spezifikationsdatei muss den Namen extension.xml erhalten und sie muss in der obersten Ebene des Erweiterungsordners gespeichert werden. Für das vorangegangene Beispiel würde der Pfad der Erweiterungsdatei relativ zum Installationsverzeichnis von IBM® SPSS® Modeler wie folgt lauten:

```
\ext\lib\myco.sorter\extension.xml
```

Java-Klassen und .jar-Dateien

Erweiterungen, die die clientseitige Java-API verwenden, enthalten kompilierten Java-Code. Dieser Code kann als .class-Dateiensatz oder verpackt in einer .jar-Datei bereitgestellt werden.

Die .class-Dateien von Java werden relativ zum obersten Erweiterungsordner angegeben. Eine Klasse, die die Schnittstelle ActionHandler implementiert, kann beispielsweise folgenden Pfad haben:

```
com.my_example.my_extension.MyActionHandler
```

In diesem Fall muss sich die .class-Datei im folgenden Ordner im Installationsverzeichnis von SPSS Modeler befinden:

```
\
Erweiterungsordner\com\my_example\my_extension\MyActionHandler.class
```

Eine .jar-Datei kann sich in einem beliebigen Verzeichnis des Erweiterungsordners befinden. Diesen Speicherort der .jar-Datei legen Sie mit dem Element JarFile der Spezifikationsdatei fest. Verwendet eine Erweiterung beispielsweise eine .jar-Datei mit dem folgenden Pfad:

```
\
Erweiterungsordner\lib\common-utilities.jar
```

dann muss das Element Resources der Spezifikationsdatei folgenden Eintrag enthalten:

```
<Resources>
 <JarFile id="util" path="lib\common-utilities.jar"/>
 ...
</Resources>
```

Für weitere Informationen siehe Thema Jar-Dateien in Kapitel 4 auf S. 44.

Eigenschaftendateien

Lokalisierte Ressourcen (z. B. der auf dem Bildschirm angezeigte Text, Fehlermeldungen und die entsprechenden Übersetzungen) werden in Dateien mit der Erweiterung .properties gespeichert. Diese Dateien können in einem beliebigen Verzeichnis im Erweiterungsordner abgelegt werden.
Für weitere Informationen siehe Thema Eigenschaftendateien in Kapitel 8 auf S. 212.

Bild- und Hilfedateien

Die Dateien mit den Grafiken für die angezeigten Symbole und die Dateien, die das Hilfesystem enthalten, können in einem beliebigen Verzeichnis im Erweiterungsordner abgelegt werden. Es empfiehlt sich jedoch, Bild- und Hilfedateien in separaten Unterordnern zu speichern.

Den Speicherort einer Bilddatei deklarieren Sie in der Spezifikationsdatei mit dem Attribut imagePath eines Icon-Elements. Für weitere Informationen siehe Thema Symbole in Kapitel 6 auf S. 137.

Den Speicherort eines Hilfesystems deklarieren Sie ebenso in der Spezifikationsdatei, allerdings mit dem Attribut path eines HelpInfo-Elements. Für weitere Informationen siehe Thema Festlegen von Speicherort und Typ des Hilfesystems in Kapitel 7 auf S. 208.

Beispiel

Die clientseitige Dateistruktur mit diesen Komponenten kann z. B. wie folgt aussehen:

```
\ext\lib\myco.sorter
\ext\lib\myco.sorter\extension.xml
\ext\lib\myco.sorter\sorter_en.properties
\ext\lib\myco.sorter\sorter_fr.properties
\ext\lib\myco.sorter\sorter_it.properties
\ext\lib\myco.sorter\com\my_example\my_extension\MyActionHandler.class
\ext\lib\myco.sorter\help\sorter.chm
\ext\lib\myco.sorter\images\lg_sorter.gif
 \ext\lib\myco.sorter\images\sm_sorter.gif
\ext\lib\myco.sorter\lib\common-utilities.jar
```

Serverseitige Komponenten

Die für die Ausführung erforderlichen gemeinsam verwendeten Bibliotheken müssen sich in einem Unterordner des Ordners \ext\bin im Installationsverzeichnis von IBM® SPSS® Modeler befinden. Beispiel:

Installationsverzeichnis\ext\bin\myco.sorter\my_lib.dll

Die gemeinsam verwendeten Bibliotheken dürfen keinesfalls direkt in den Ordner \ext\bin gestellt werden.

Den Speicherort von gemeinsam verwendeten Bibliotheken, die SPSS Modeler während der Ausführung aufruft, deklarieren Sie in einem SharedLibrary-Element der Spezifikationsdatei. Für weitere Informationen siehe Thema Freigegebene Bibliotheken in Kapitel 4 auf S. 45.

Die gemeinsam verwendete Hauptbibliothek muss eventuell auf weitere, abhängige Bibliotheken zugreifen. Damit die Hauptbibliothek diese Bibliotheken findet, müssen sich die abhängigen Bibliotheken im gleichen Verzeichnis befinden wie die Hauptbibliothek.

Beispiel

Die serverseitige Dateistruktur kann z. B. wie folgt aussehen:

```
\ext\bin\myco.sorter\my_lib.dll
\ext\bin\myco.sorter\my_lib2.dll
```

Kapitel
2

Knoten

Überblick über die Knoten

Wenn Sie eine Erweiterung mit einem neuen Knoten erstellen, sollten Sie sich zunächst mit den Merkmalen der IBM® SPSS® Modeler-Knoten vertraut machen. Dies hilft Ihnen, Ihren neuen Knoten korrekt in der Spezifikationsdatei zu definieren.

SPSS Modeler-Knoten werden entsprechend ihrer Funktion als Quellen-, Prozess-, Ausgabe- und Modellierungsknoten klassifiziert. In CLEF werden die Knoten nach ihrem Typ unterschieden. In der folgenden Tabelle sind Knoten der beiden Systeme mit ähnlichen Funktionen einander gegenübergestellt:

Tabelle 2-1
CLEF-Knotentypen

SPSS Modeler-Klassifizierung	Palette	CLEF-Knotentyp
Quellenknoten	Quellen	Data reader (Datenleser)
Prozessknoten	Datensatzoperationen	Data transformer (Datentransformer)
	Feldoperationen	
Ausgabeknoten	Grafiken	Document builder (Dokumentersteller)
	Ausgabe (Berichtsknoten)	
	Exportieren	Data writer (Datenschreiber)
Modellierungsknoten	Modellierung	Model builder (Modellersteller)

Bei der Erstellung eines neuen CLEF-Knotens weisen Sie ihm einen der Knotentypen von CLEF zu. Für welchen Knotentyp Sie sich entscheiden, richtet sich nach der Hauptfunktion des Knotens.

Tabelle 2-2
Knotentypen und ihre Funktionen

CLEF-Knotentyp	Beschreibung	Zugehörige Knotenpalette	Symbol
Data reader (Datenleser)	Importiert Daten aus einem anderen Format in SPSS Modeler.	Quellen	
Data transformer (Datentransformer)	Liest Daten aus SPSS Modeler ein, modifiziert sie und gibt die modifizierten Daten in den SPSS Modeler-Stream zurück.	Datensatzoperationen, Feldoperationen	

CLEF-Knotentyp	Beschreibung	Zugehörige Knotenpalette	Symbol
Model builder (Modellersteller)	Generiert aus SPSS Modeler-Daten Modelle.	Modellierung	
Document builder (Dokumentersteller)	Generiert aus SPSS Modeler-Daten Diagramme oder Berichte.	Grafiken	
		Ausgabe (Berichtsknoten)	
Modellanwender (auch als “Modell-Nugget” bezeichnet)	Definiert einen Container für ein generiertes Modell, das wieder auf dem SPSS Modeler-Zeichen-bereich abgelegt wurde.	–	
Data writer (Datenschreiber)	Exportiert Daten aus dem SPSS Modeler-Format in ein für eine andere Anwendung geeignetes Format.	Exportieren	

Den Knotentyp legen Sie gemeinsam mit anderen Attributen in der Spezifikationsdatei in einem Node-Element fest. Beispiel:

```
<Node name="sort_process" type="dataTransformer"
    palette="recordOp" ... >
   -- node elements --
</Node>
```

Das Attribut palette bestimmt die Palette im Hauptfenster von SPSS Modeler, aus der die Benutzer den Knoten aufrufen können – in diesem Beispiel die Palette “Datensatzoperationen”. Falls dieses Attribut fehlt, wird der Knoten der Palette “Feldoperationen” hinzugefügt.

IBM® SPSS® Modeler stellt verschiedene Beispielknoten zur Verfügung. Für weitere Informationen siehe Thema Informationen zu den Beispielen in Kapitel 3 auf S. 33.

Datenleserknoten

Ein Datenleserknoten ermöglicht das Einlesen von Daten aus einer externen Quelle in einen IBM® SPSS® Modeler-Stream. Bei den Knoten der SPSS Modeler-Palette “Datenquellen” handelt es sich um Datenleserknoten. Sie sind durch ein Kreissymbol gekennzeichnet.

Die Spezifikation eines Datenleserknotens enthält folgende Details:

- Datenquelle (z. B. eine Datei oder eine Datenbank)

- Datensatzvorverarbeitung (z. B. die Behandlung von führenden und abschließenden Leerzeichen oder die Festlegung des Datensatztrennzeichens)
- Ob Datensatzfelder ausgefiltert werden
- Datentyp (z. B. Bereich, Set oder Flag) und Speichertyp (String, ganze Zahl oder reelle Zahl) jedes einzelnen Felds
- Ob das Eingabedatenmodell geändert wird

Der Datenleserknoten kann die Logik zum Einlesen der Quelldatensätze enthalten. In SPSS Modeler kann diese Logik aber auch erst später anhand eines Typknotens definiert werden.

IBM® SPSS® Modeler enthält ein Beispiel für einen Datenleserknoten. Für weitere Informationen siehe Thema Informationen zu den Beispielen in Kapitel 3 auf S. 33.

Datentransformationsknoten

Ein Datentransformationsknoten liest die Daten eines IBM® SPSS® Modeler-Streams ein, modifiziert diese Daten auf eine bestimmte Weise und gibt die modifizierten Daten wieder an den Stream zurück. Bei den Knoten der SPSS Modeler-Paletten "Datensatzoperationen" und "Feldoperationen" handelt es sich um Datentransformationsknoten. Sie sind durch ein sechseckiges Symbol gekennzeichnet.

Die Spezifikation eines Datentransformationsknotens enthält folgende Details:

- Welche Datensätze oder Felder transformiert werden sollen
- Wie die Daten transformiert werden sollen

IBM® SPSS® Modeler enthält ein Beispiel für einen Datentransformationsknoten. Für weitere Informationen siehe Thema Informationen zu den Beispielen in Kapitel 3 auf S. 33.

Modellerstellungsknoten

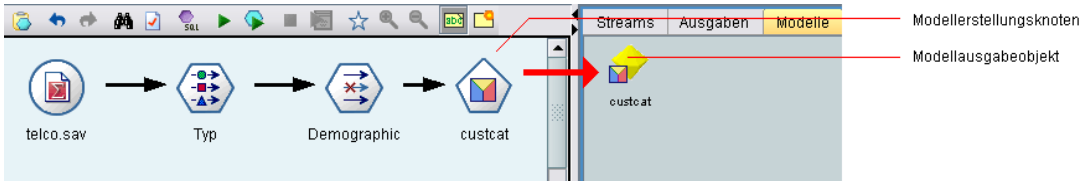
Eine Einführung in die Modellerstellung mit IBM® SPSS® Modeler erhalten Sie im Abschnitt "Einführung in die Modellierung" im *SPSS Modeler 15-Anwendungshandbuch*.

Modellerstellungsknoten generieren Objekte, die auf der Registerkarte "Modelle" bzw. auf der Registerkarte "Ausgaben" des Managerfensters im Hauptfenster von SPSS Modeler angezeigt werden.

Bei den Knoten der SPSS Modeler-Palette "Modellierung" handelt es sich um Modellerstellungsknoten. Sie sind durch ein fünfeckiges Symbol gekennzeichnet.

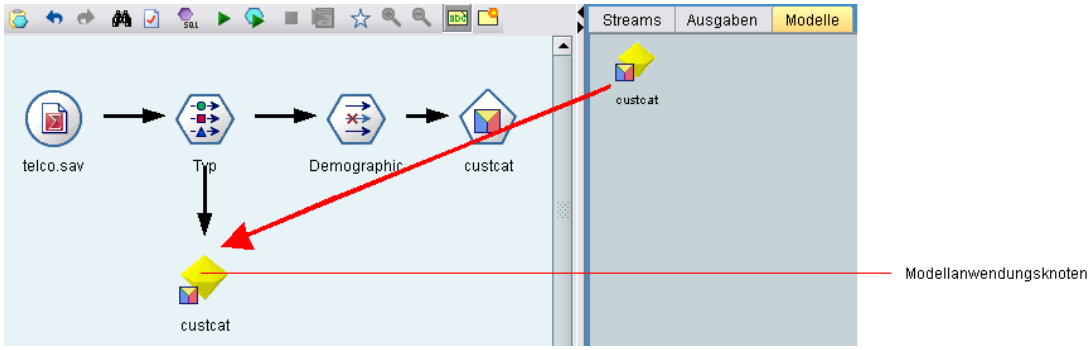
Ein Modellerstellungsknoten generiert bei seiner Ausführung ein **Modellausgabeobjekt** (auch als "Modell-Nugget" bezeichnet), das auf der Registerkarte "Modelle" des Managerfensters angezeigt wird.

Abbildung 2-1
Modellerstellungsknoten bei der Generierung eines Modellausgabeobjekts (Modell-Nugget)



Wenn ein durch einen Modellerstellungsknoten generiertes Modell auf den Zeichenbereich gezogen wird, nimmt es die Form eines Modellanwendungsknotens an.

Abbildung 2-2
Modellausgabeobjekt wird dem Zeichenbereich als Modellanwendungsknoten hinzugefügt



Die Spezifikation eines Modellerstellungsknotens enthält folgende Details:

- Modellerstellungsdetails, beispielsweise der Algorithmus für die Generierung des Modells sowie die Eingabe- und Ausgabefelder, die für das Scoring der Daten im Modell verwendet werden
- Vom Modell verwendete Eigenschaften
- Container für die Ausgabeobjekte
- Benutzeroberfläche des Knotendialogfelds
- Eigenschaften und Dateien, die bei der Ausführung des Knotens verwendet werden
- Auswirkung der Knotenausführung auf das Eingabedatenmodell
- Kennung des Modellausgabeobjekts sowie die Kennungen aller anderen bei der Knotenausführung erstellten Objekte
- Kennung des Modellanwendungsknotens (siehe Modellanwendungsknoten auf S. 14)

Hinweis: Bei der Definition eines Modellerstellungsknotens fügen Sie die tatsächliche Definition des Modellausgabeobjekts und des Modellanwendungsknotens an einer anderen Stelle der gleichen Spezifikationsdatei ein.

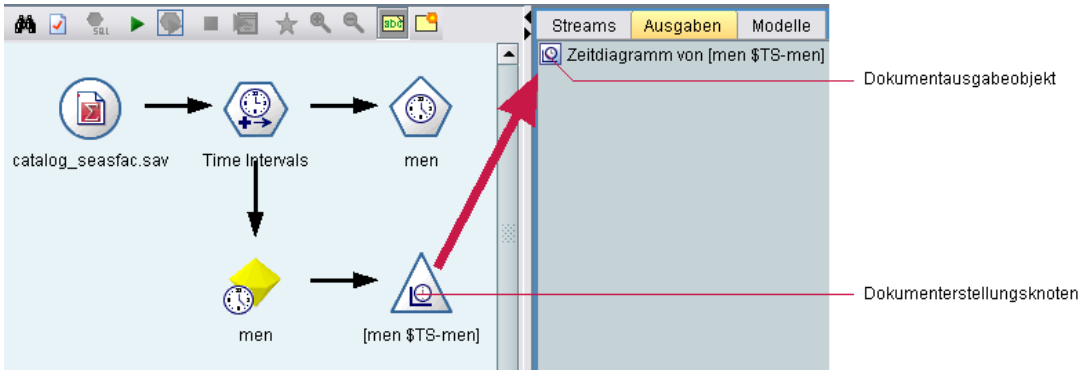
IBM® SPSS® Modeler enthält ein Beispiel für einen Modellerstellungsknoten. Für weitere Informationen siehe Thema Informationen zu den Beispielen in Kapitel 3 auf S. 33.

Dokumenterstellungsknoten

Dokumenterstellungsknoten generieren Objekte, die auf der Registerkarte “Ausgaben” des Managerfensters im Hauptfenster von IBM® SPSS® Modeler angezeigt werden. Bei den Knoten der Palette “Diagramme” handelt es sich um Dokumenterstellungsknoten. Sie sind durch ein dreieckiges Symbol gekennzeichnet.

Ein Dokumenterstellungsknoten generiert bei seiner Ausführung ein **Dokumentausgabeobjekt**, das auf der Registerkarte “Ausgaben” des Managerfensters angezeigt wird.

Abbildung 2-3
Dokumenterstellungsknoten bei der Generierung eines Dokumentausgabeobjekts



Im Gegensatz zu einem Modellausgabeobjekt kann ein Dokumentausgabeobjekt nicht zurück auf den SPSS Modeler-Zeichenbereich gezogen werden.

Die Spezifikation eines Dokumenterstellungsknotens enthält folgende Details:

- Dokumenterstellungsdetails, beispielsweise die Registerkarte, auf der sich die Steuerelemente für die Generierung des Dokuments befinden
- Vom Dokument verwendete Eigenschaften
- Container für die Ausgabeobjekte
- Benutzeroberfläche des Knotendialogfelds
- Eigenschaften und Dateien, die bei der Ausführung des Knotens verwendet werden
- Kennung des Dokumentausgabeobjekts sowie die Kennungen aller anderen bei der Knotenausführung erstellten Objekte

Hinweis: Bei der Definition eines Dokumenterstellungsknotens fügen Sie die tatsächliche Definition des Dokumentausgabeobjekts an einer anderen Stelle der gleichen Spezifikationsdatei ein.

Modellanwendungsknoten

Ein Modellanwendungsknoten legt den Container eines generierten Modells fest, in dem das Modell von der Registerkarte “Modelle” des Managerfensters auf den IBM® SPSS® Modeler-Zeichenbereich gezogen wird.

Die Spezifikation eines Modellanwendungsknotens enthält folgende Details:

- Container für das Modell (bzw. mehrere Container, wenn die Modellausgabe in mehreren Formaten wie Text und HTML erstellt werden kann)
- Details der Benutzeroberfläche des Dialogfelds, das angezeigt wird, wenn der Benutzer den Anwendungsknoten auf der Registerkarte "Modelle" durchsucht oder ihn im Zeichenbereich öffnet
- Ausgabedatenmodell
- Die bei der Ausführung des Streams, der den Knoten enthält, durchzuführenden Verarbeitungsschritte
- Konstruktoren für die Behandlung der Objekte, die bei der Ausführung des Streams, der den Knoten enthält, erstellt werden

Datenschreiberknoten

Ein Datenschreiberknoten exportiert Daten aus dem IBM® SPSS® Modeler-Format in ein für eine andere Anwendung geeignetes Format. Bei den Knoten der SPSS Modeler-Palette "Exportieren" handelt es sich um Datenschreiberknoten. Sie sind durch ein rechteckiges Symbol gekennzeichnet.

Die Spezifikation eines Datenschreiberknotens enthält folgende Details:

- Angaben zur Datei oder Datenbank, in die die Stream-Daten geschrieben werden
- Optional, ob der gesamte Stream veröffentlicht werden soll, damit er in eine externe Anwendung eingebettet werden kann

Menüs, Symbolleisten und Paletten

Der Zugriff auf eine Erweiterung kann über ein IBM® SPSS® Modeler-Menü, die Symbolleiste oder eine Palette erfolgen. Eine Erweiterung kann entweder einen Knoten implementieren oder eine bestimmte Aktion ausführen.

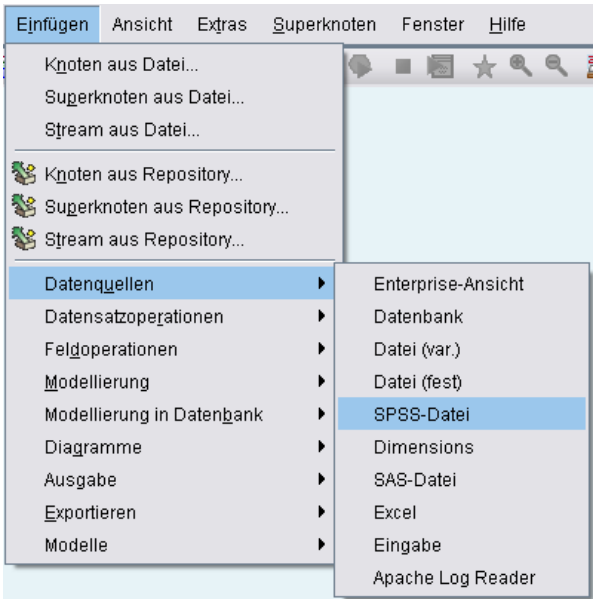
Eine Erweiterung (Knoten oder Aktion), die über ein ausdrücklich angegebenes Menü aufgerufen wird, kann auch über die Symbolleiste zugänglich gemacht werden. Das Gleiche gilt auch im umgekehrten Fall.

Für einen Knoten, der über eine Palette aufgerufen wird, steht automatisch auch ein entsprechendes Menüelement im Menü "Einfügen" zur Verfügung.

Menüs und Untermenüs

Die Standardknoten von IBM® SPSS® Modeler können über das Menü "Einfügen" aufgerufen werden. Jedes Menüelement der letzten Gruppe dieses Menüs (mit Ausnahme des Menüelements "Modelle") verfügt über ein Untermenü, das Zugriff auf einen Satz verwandter Knoten bietet.

Abbildung 2-4
Menü \"Einfügen\"



Die Elemente dieses Menüs entsprechen den Einträgen der Knotenpaletten. Ein Knoten, der einer Palette hinzugefügt wird, erscheint automatisch in der entsprechenden Gruppe im Menü “Einfügen”.

Falls Ihre Erweiterung eine Aktion definiert, die nicht über einen Knoten aufgerufen werden kann, können Sie die Erweiterung durch Hinzufügen eines der folgenden Elemente zugänglich machen:

- Neues Element in einem Systemmenü oder einem Untermenü
- Neues Menü in SPSS Modeler
- Neues Element in der Symbolleiste (siehe Symbolleisten auf S. 16)

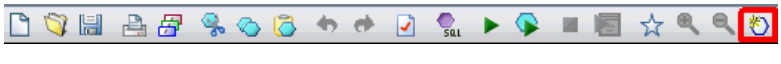
Dem neuen Menü oder Menüelement können Sie optional auch das Symbol der Erweiterung hinzufügen, wie dies bei einigen Elementen des Menüs “Einfügen” bereits der Fall ist.

Weitere Informationen finden Sie in den Abschnitten Menüs auf S. 138 und Menüelemente auf S. 140.

Symbolleisten

Falls Ihre Erweiterung eine Aktion definiert, die nicht über einen Knoten aufgerufen werden kann, können Sie die Erweiterung der Hauptsymbolleiste von IBM® SPSS® Modeler hinzufügen, um sie auf diese Weise zugänglich zu machen.

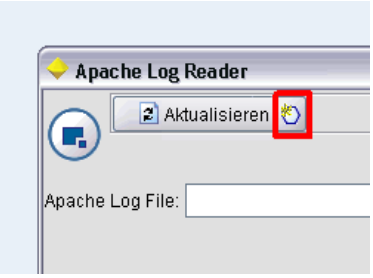
Abbildung 2-5
Hinzufügen eines Elements zur Hauptsymbolleiste



In diesem Fall empfiehlt es sich, die Beschriftung der Aktion auszublenden.

Sie können ein Element auch zur Symbolleiste eines Knotendialogfelds oder eines Ausgabefensters hinzufügen. Die Elementbeschriftung können Sie anzeigen oder wie in diesem Beispiel ausblenden:

Abbildung 2-6
Hinzufügen eines Elements zur Symbolleiste eines Dialogfelds

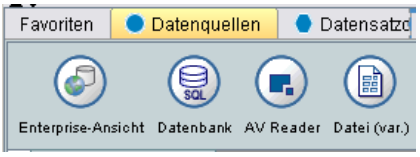


Für weitere Informationen siehe Thema Symbolleistenelemente in Kapitel 6 auf S. 141.

Paletten und Unterpaletten

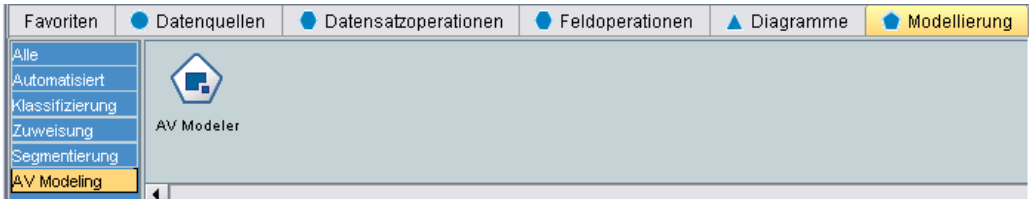
Wenn Ihre Erweiterung einen neuen Knoten definiert, können Sie den Knoten an einer beliebigen Stelle in einer der Standardpaletten oder Unterpaletten von IBM® SPSS® Modeler einfügen.

Abbildung 2-7
Neuer Knoten in einer Standardpalette



Sie können einer Standard-Unterpalette einen Eintrag hinzufügen und den neuen Knoten dort einfügen.

Abbildung 2-8
Neuer Knoten in benutzerdefiniertem Eintrag einer Standard-Unterpalette



Sie können eine benutzerdefinierte Palette erstellen und den neuen Knoten dort einfügen.

Abbildung 2-9
Neuer Knoten in einer benutzerdefinierten Palette

Einer benutzerdefinierten Palette können Sie benutzerdefinierte Unterpaletten hinzufügen.

Abbildung 2-10
Neuer Knoten in einer benutzerdefinierten Unterpalette einer benutzerdefinierten Palette

Weitere Informationen finden Sie in den Abschnitten Knoten auf S. 62 und Abschnitt ‘Benutzeroberfläche (Paletten)’ auf S. 55.

Erstellen von Knotensymbolen

Sie können für jeden neuen Knoten, den Sie in CLEF erstellen, ein Bild bereitstellen, das in der Mitte des Symbols, das den Knoten auf dem Bildschirm darstellt, angezeigt wird.

Hinweis: Sie müssen kein Bild bereitstellen – IBM® SPSS® Modeler stellt ein Standardbild zur Verfügung, das angezeigt wird, wenn Sie kein eigenes Bild festlegen (dies kann z. B. in der anfänglichen Entwicklungsphase eines Knotens recht praktisch sein).

Abbildung 2-11
Standardbild für CLEF-Symbole

Die Standardsymbole von SPSS Modeler bestehen aus drei Schichten:

- Rahmen
- Hintergrund
- Bild in der Mitte

Für das Symbol eines neuen Knotens brauchen Sie nur das Bild in der Mitte des Symbols bereitzustellen (auch als **Glyphe** bezeichnet). Die Rahmen- und Hintergrundverarbeitung wird von SPSS Modeler übernommen. Das Glyphenbild muss einen transparenten Hintergrund haben, damit die Hintergrundschicht des Symbols nicht verdeckt wird. Die Glyphen in diesem Abschnitt sind mit einem farbigen Hintergrund dargestellt, der Transparenz darstellen soll.

Abbildung 2-12
Glyphenschicht mit farbigem Hintergrund, der Transparenz darstellen soll

Ein typisches Modellierungssymbol setzt sich in SPSS Modeler wie folgt zusammen.

Tabelle 2-3
Zusammensetzung der Symbole für Knoten und generierte Modelle

	Knotensymbol	**Symbol für generiertes Modell**
Rahmen		None
Hintergrund		
Glyphe		
Bild angezeigt als		

Rahmen

Die Funktion des Knotens wird durch die Form des Symbolrahmens angezeigt. Für weitere Informationen siehe Thema Überblick über die Knoten auf S. 10.

Wenn für einen Knoten Caching aktiviert ist, wird der Rahmenform ein kleines Dokumentsymbol hinzugefügt. Ein weißes Dokumentsymbol an einem Knoten weist darauf hin, dass sein Cache leer ist. Wenn der Cache voll ist, wird das Dokumentsymbol grün.

Tabelle 2-4
Knotenrahmen und Caching-Status

Caching-Status	**Beispiel**
Kein Caching	
Caching aktiviert	
Cache voll	

Die verschiedenen Rahmensymbole werden vom System bereitgestellt. IBM® SPSS® Modeler sorgt dafür, dass die erforderliche Verarbeitung ausgeführt wird, um das richtige Symbol zur richtigen Zeit anzuzeigen.

Hintergründe

Die Hintergrundfarbe der Knotensymbole ändert sich, um den jeweiligen Status anzuzeigen. Hiervon ausgenommen sind Symbole für generierte Modelle und Symbole von Modellanwendungsknoten.

Tabelle 2-5
Knotenhintergründe

State	Farbe	Beispiel
Nicht ausgewählt	Grau	
Selected	Blau	
Fehler	Rot	
Aktion wird an einer Datenbank ausgeführt	Violett	

Auch die Hintergrundbilder werden vom System bereitgestellt und hier sorgt IBM® SPSS® Modeler wiederum dafür, dass die erforderliche Verarbeitung ausgeführt wird, um den richtigen Hintergrund für die jeweilige Situation anzuzeigen.

Grafikanforderungen

Erstellen Sie für jeden neuen CLEF-Knoten die folgenden Versionen der Glyphenbildschicht:

- Großes Format (49 x 49 Pixel) für das Knotensymbol im Stream-Zeichenbereich
- Kleines Format (38 x 38 Pixel) für das Knotensymbol im Palettenmanager am unteren Bildschirmrand

Wenn Sie das Symbol in einem Menü oder einer Symbolleiste oder in der Titelleiste eines Browser- oder Ausgabefensters anzeigen möchten, müssen Sie zusätzlich folgende Bildschicht erstellen:

- Miniaturformat (16 x 16 Pixel)

Wenn der Knoten ein Modell generiert, müssen Sie zusätzlich folgende Bildschicht erstellen:

- Kleines Format (38 x 38 Pixel), wobei die Grafik in die linke untere Ecke verschoben werden muss, damit die Bildschicht über das Goldnugget-Symbol eines generierten Modells gelegt werden kann

Hinweis: Bilder, die größer als die angegebenen Formate sind, werden in IBM® SPSS® Modeler abgeschnitten.

Für weitere Informationen siehe Thema Symbole in Kapitel 6 auf S. 137.

Erstellen benutzerdefinierter Bilder

Das Bild, das Sie für einen Knoten erstellen, sollte die Hauptfunktion des Knotens grafisch vermitteln. Wenn Sie Ihre Erweiterung einem internationalen Publikum bereitstellen möchten, sollten Sie darauf achten, dass Sie keine landestypischen Symbole verwenden und dass die Bilder in anderen Kulturen und Sprachräumen nicht missverstanden werden.

So erstellen Sie ein benutzerdefiniertes Bild für CLEF:

- Verwenden Sie zur Erstellung Ihres Bildes eine Grafikanwendung, die Transparenz unterstützt, und richten Sie einen Zeichenbereich mit dem erforderlichen Format ein. Zeichnen Sie danach die einzelnen Bildversionen.
- Speichern Sie jede der erforderlichen Versionen (groß, klein usw.) als separate *.gif*-Datei mit folgenden Eigenschaften:
  - Hintergrund: Transparent
  - Farbtiefe: 16 Farben (4 Bit) oder höher

Auf welche Art und Weise Sie den Bildhintergrund transparent gestalten, richtet sich nach der verwendeten Grafikanwendung. Möglicherweise können Sie die Hintergrundfarbe ohne Umwege transparent einstellen oder Sie müssen zunächst eine Transparenzfarbe festlegen und den Bildhintergrund danach mit dieser Farbe "anmalen".

Bei der Benennung der Bilddateien empfiehlt es sich, den von IBM® SPSS® Modeler intern verwendeten Dateinamenskonventionen zu folgen:

Tabelle 2-6
Dateinamenskonventionen für Bilddateien

Bildtyp	Dateiname
Large	lg_*knoten*.gif
Klein	sm_*knoten*.gif
Miniatur	*knoten*16.gif
Generiertes Modell	sm_gm_*knoten*.gif

- Die Wirkung Ihres Bildes können Sie testen, indem Sie in der Spezifikationsdatei auf die Bilddateien verweisen (siehe Hinzufügen der Bilddateien in die Knotenspezifikation) und den neuen Knoten zu SPSS Modeler hinzufügen (siehe Testen einer CLEF-Erweiterung auf S. 251).

Hinzufügen der Bilddateien in die Knotenspezifikation

Nachdem Sie die Bilddateien erstellt haben, kopieren Sie diese in einen Ordner auf dem Computer, auf dem Sie IBM® SPSS® Modeler ausführen. In der Spezifikationsdatei müssen Sie den Bildpfad relativ zu dem Ordner \ext\lib*provider.knotenname* im Installationsverzeichnis von SPSS Modeler angeben. Sie sollten die Dateien daher in einen Ordner kopieren, der von diesem Speicherort leicht zu erreichen ist. Für weitere Informationen siehe Thema Symbole in Kapitel 6 auf S. 137.

Die Zuordnung der Bilddateien mit dem großen und dem kleinen Symbol zu einem benutzerdefinierten Knoten erfolgt in der Spezifikationsdatei über das Element Icons im Abschnitt UserInterface der Node-Spezifikation. Beispiel:

```
<Icons>
   <Icon type="standardNode" imagePath="images/lg_mynode.gif" />
   <Icon type="smallNode" imagePath="images/sm_mynode.gif" />
</Icons>
```

Bei einem Modell- oder Dokumenterstellungsknoten müssen Sie zusätzlich auf die Miniaturversion (16 x 16 Pixel) verweisen. Bei einem Modellerstellungsknoten erfolgt dieser Verweis im Abschnitt UserInterface der ModelOutput-Spezifikation bzw. der DocumentOutput-Spezifikation (bei einem Dokumenterstellungsknoten). Beispiel:

```
<Icons>
   <Icon type="standardWindow" imagePath="images/mynode16.gif" />
</Icons>
```

Bei einem Modellanwendungsknoten müssen Sie im Abschnitt UserInterface der Node-Spezifikation zusätzlich auf die Bildversion für das generierte Modell verweisen. Beispiel:

```
<Icons>
   <Icon type="standardNode" imagePath="images/lg_gm_mynode.gif" />
   <Icon type="smallNode" imagePath="images/sm_gm_mynode.gif" />
</Icons>
```

Erstellen von Dialogfeldern

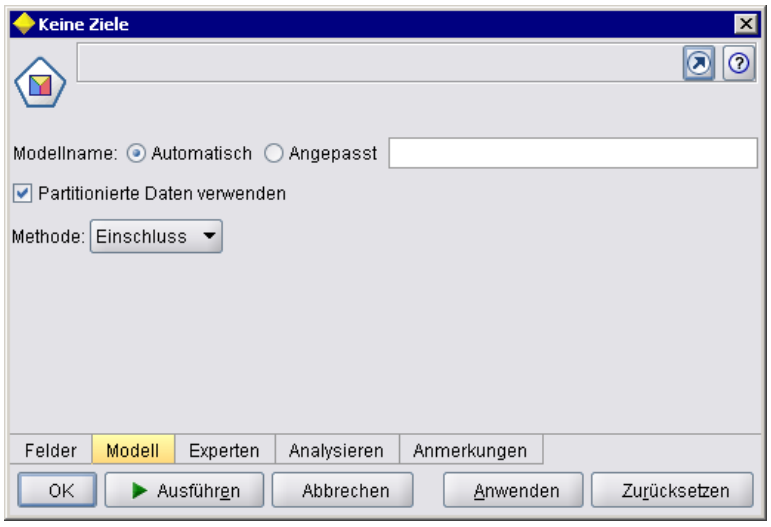
In diesem Abschnitt werden die Merkmale der Standard-Knotendialogfelder von IBM® SPSS® Modeler beschrieben. Der Abschnitt soll Ihnen dabei helfen, auch in CLEF konsistente Dialogfelder zu erstellen.

Informationen zu Knotendialogfeldern

Ein Knotendialogfeld bietet eine Benutzeroberfläche, mit der der Endbenutzer Ausführungseinstellungen ändern kann. Die Darstellung eines Dialogfelds ist äußerst wichtig, da hier das Knotenverhalten geändert wird. Die Benutzeroberfläche muss alle notwendigen Informationen enthalten und benutzerfreundlich sein.

Das Knotenverhalten wird durch verschiedene, für Dialogfelder typische **Steuerelemente**, d. h. durch Benutzeroberflächenelemente, mit denen der Benutzer interagieren kann, geändert. Ein Dialogfeld kann zahlreiche Steuerelemente wie Optionsschaltflächen, Kontrollkästchen, Textfelder und Menüs enthalten. CLEF bietet eine große Auswahl an Steuerelementen, die Sie in Ihre Dialogfelder einfügen können. Für weitere Informationen siehe Thema Spezifikationen von Eigenschaftssteuerelementen in Kapitel 6 auf S. 154.

Abbildung 2-13
Beispiel für ein Dialogfeld mit Optionsschaltflächen, einem Kontrollkästchen, einem Kombinationsfeld und Registerkarten



Der durch ein Steuerelement geänderte Parametertyp bestimmt, welches Steuerelement im Dialogfeld angezeigt wird, wobei einige Typen alternative Steuerelemente bieten. Verwandte Optionen können Sie mittels des Tab-Elements der Spezifikationsdatei auf neuen Registerkarten zusammenfassen. Für weitere Informationen siehe Thema Registerkartenbereich auf S. 27.

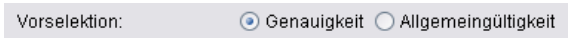
Hinweis: Das Erscheinungsbild der Benutzeroberfläche einer Erweiterung können Sie testen, noch bevor Sie die durch die Erweiterung auszuführende Verarbeitung festgelegt haben. Für weitere Informationen siehe Thema Testen von CLEF-Erweiterungen in Kapitel 10 auf S. 251.

Richtlinien für die Erstellung von Dialogfeldern

Beachten Sie bei der Festlegung der Steuerelemente für ein Dialogfeld die folgenden Richtlinien:

- Überlegen Sie sich den Text für die Beschriftung eines Steuerelements gut. Der Text sollte möglichst knapp und präzise sein und gleichzeitig korrekte Informationen vermitteln. Wenn Sie Ihre Erweiterung für den internationalen Markt entwickeln, sollten Sie ausreichend Platz für die Übersetzung lassen, die häufig erheblich länger ist als das Original.
- Verwenden Sie das richtige Steuerelement für einen Parameter. Auch wenn ein Parameter nur zwei Werte haben kann, ist ein Kontrollkästchen nicht immer die beste Wahl. Das Dialogfeld des C5.0-Knotens von IBM® SPSS® Modeler verwendet beispielsweise Optionsschaltflächen für die Auswahl des Ausgabetyps, obwohl dieser nur einen der beiden folgenden Werte annehmen kann — Entscheidungsbaum oder Regelmenge.

Abbildung 2-14
Auswahl zwischen zwei Werten



Diese Einstellung könnte durch ein Kontrollkästchen mit der Beschriftung Entscheidungsbaum dargestellt werden. Wenn das Kontrollkästchen aktiviert ist, ist der Ausgabetyp ein Entscheidungsbaum. Ist es nicht aktiviert, handelt es sich bei der Ausgabe um eine

Regelmenge. Auch wenn das Ergebnis dasselbe wäre, so wären die zur Verfügung stehenden Optionen durch Verwendung von Optionsschaltflächen leichter zu verstehen.

- Steuerelemente für Dateinamen werden im Allgemeinen im oberen Bereich eines Dialogfelds platziert.
- Steuerelemente, die den Fokus des Knotens bilden, werden oben im Dialogfeld platziert. Diagrammknoten zeigen beispielsweise Felder aus den Daten an. Da die Auswahl dieser Felder die Hauptfunktion des Dialogfelds darstellt, werden Feldparameter oben platziert.
- Kontrollkästchen oder Optionsschaltflächen ermöglichen dem Benutzer häufig die Auswahl einer Option, die weitere Informationen erfordert. Beispiel: Die Auswahl der Option Verstärkung verwenden im Dialogfeld des C5.0-Knotens erfordert, dass die Analyse eine Zahl für die Anzahl der Versuche enthält.

Abbildung 2-15
Kontrollkästchen mit zugehörigem Steuerelement

Die Zusatzinformationen werden immer nach der Optionsauswahl platziert, und zwar entweder rechts oder direkt darunter.

In den Dialogfeldern von CLEF wird die Commit-Bearbeitung von SPSS Modeler auf die gleiche Weise verwendet wie in den Standarddialogfeldern von SPSS Modeler: Die in den Dialogfeldern angezeigten Werte werden erst in den Knoten kopiert, wenn der Benutzer auf OK, Anwenden oder bei Endknoten auf Ausführen klickt. Ebenso werden die im Dialogfeld angezeigten Informationen erst aktualisiert (z. B. wenn sich die Eingabefelder des Knotens aufgrund von Vorgängen weiter oben im Stream des aktuellen Knotens geändert haben), wenn der Benutzer das Dialogfeld abbricht und erneut anzeigt oder auf die Schaltfläche Aktualisieren klickt.

Dialogfeldkomponenten

Dialogfelder enthalten die folgenden Komponenten:

- Titelleiste
- Symbolbereich
- Symbolleisten- und Menübereich mit:

  – Datei, Generieren, Ansicht, Vorschau, Aktualisieren und andere Schaltflächen (abhängig vom Knoten)

  – Schaltfläche “Maximieren/Normale Größe”

  – Hilfeschaltfläche
- Statusbereich
- Fensterbereich
- Registerkartenbereich
- Schaltflächenbereich

Jeder benutzerdefinierte Knoten benötigt ein Dialogfeld, das angezeigt wird, sobald der Benutzer den Knoten öffnet. Vorausgesetzt, Ihre Spezifikationsdatei enthält ein Node-Element mit dem Abschnitt UserInterface, der wiederum das Element Tabs enthält, dann werden beim Öffnen des Knotens alle oben aufgelisteten Dialogfeldkomponenten angezeigt. Abhängig vom Knotentyp enthalten der Registerkartenbereich und der Schaltflächenbereich mindestens die folgenden Komponenten:

Tabelle 2-7
Standardkomponenten des Registerkarten- und Schaltflächenbereichs für verschiedene Knotentypen

Knotentyp	Registerkarten	Schaltflächen
Data reader (Datenleser)	Anmerkungen (mit Schaltfläche "Aktualisieren" im Symbolleistenbereich)	OK, Abbrechen, Anwenden, Zurücksetzen
Data transformer (Datentransformer)	Anmerkungen	OK, Abbrechen, Anwenden, Zurücksetzen
Data writer (Datenschreiber)	Veröffentlichen, Anmerkungen	OK, Abbrechen, Ausführen, Anwenden, Zurücksetzen
Model builder (Modellersteller)	Anmerkungen	OK, Abbrechen, Ausführen, Anwenden, Zurücksetzen
Document builder (Dokumentersteller)	Anmerkungen	OK, Abbrechen, Ausführen, Anwenden, Zurücksetzen
Model applier (Modellanwender)	Übersicht, Anmerkungen	OK, Abbrechen, Anwenden, Zurücksetzen

Knotendialogfelder sind zunächst so positioniert, dass das Knotensymbol beim Öffnen des Knotens über den Knoten, den es darstellt, gelegt wird. Der Benutzer kann das Dialogfeld verschieben, die neue Position wird aber nicht gespeichert und beim nächsten Öffnen des Knotens nicht wiederhergestellt. Wenn der Benutzer das Dialogfeld verschoben hat und es danach teilweise oder vollständig durch ein anderes Dialogfeld verdeckt wird, kann er es durch Doppelklicken auf den Originalknoten im Zeichenbereich wieder in den Vordergrund verschieben. Das Dialogfeld ist moduslos (d. h., die gleiche Benutzereingabe bewirkt immer die gleiche Aktion) und seine Größe kann geändert werden.

Alle bearbeitbaren Felder im Dialogfeld unterstützen die folgenden Direktzugriffstasten:

Direktzugriffstaste	Effekt
STRG-C	Kopieren
STRG-V	Einfügen
STRG-X	Ausschneiden

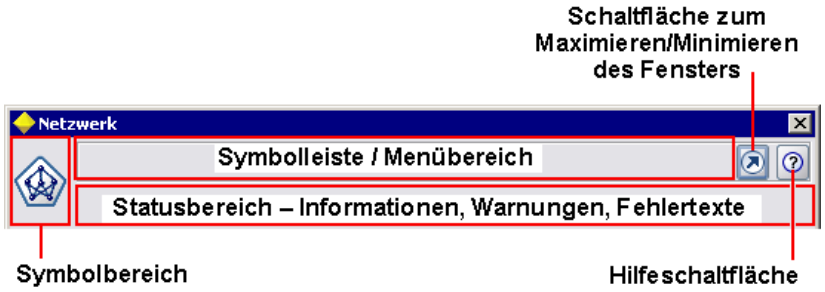
Titelleiste

Die Titelleiste eines Knotendialogfelds enthält eine Miniaturausgabe des Goldnugget-Symbols von IBM® SPSS® Modeler gefolgt vom Namen des Modells. Der Text wird der Einstellung der Steuerelemente für den Modellnamen entnommen. Auf der rechten Seite der Titelleiste befindet sich standardmäßig die Schaltfläche "Schließen" (X).

Symbolbereich

Das Knotensymbol befindet sich links oben im Dialogfeld im Symbolbereich.

Abbildung 2-16
Oberer Teil des Knotendialogfelds



Hier wird die kleine Version des Symbols (38 x 38 Pixel) angezeigt, d. h. die gleiche Version, die auch unten im Hauptfenster in der Knotenpalette angezeigt wird (nicht die größere, für den Zeichenbereich vorgesehene Version).

Hinweis: Die Miniaturausgabe des Goldnugget-Symbols auf der linken Seite der Titelleiste ist in alle Knotendialogfelder fest einkodiert.

Symbolleisten- und Menübereich

Der obere Bereich des Dialogfelds ist für die Symbolleiste und den Menübereich reserviert.

Die Dialogfelder für Datenleser- oder Datentransformationsknoten verfügen in diesem Bereich über die Schaltfläche “Vorschau”, mit deren Hilfe ein Beispiel der Eingabedaten angezeigt werden kann.

Dialogfelder für Datenleserknoten enthalten auch die Schaltfläche “Aktualisieren”, die die vom Knoten angezeigten Informationen aktualisiert (z. B. wenn die Eingabefelder des Knotens geändert wurden).

Modellanwendungsknoten verfügen über die Schaltflächen “Datei”, “Generieren” und “Ansicht”, mit deren Hilfe Benutzer zahlreiche Operationen ausführen können, z. B. Exportieren eines Modells oder Generieren neuer Knoten. Modellanwendungsknoten besitzen auch eine Vorschau-Schaltfläche, die in diesem Fall ein Beispiel der Eingabedaten zusammen mit den zusätzlichen Spalten anzeigt, die beim Anwenden des Knotens erstellt werden.

Auf der rechten Seite dieses Bereichs befinden sich in jedem Knotendialogfeld zwei Schaltflächen:

- Schaltfläche “Maximieren/Normale Größe”
- Hilfe (Schaltfläche)

Schaltfläche "Maximieren/Normale Größe"

Mit dieser Schaltfläche kann das Dialogfeld auf Vollbildgröße vergrößert werden. Wenn das Dialogfeld bereits mit maximaler Größe angezeigt wird, wird es bei Betätigen der Schaltfläche wieder auf die vorherige Größe verkleinert.

Hilfeschaltfläche

Diese Schaltfläche ruft die kontextsensitive Hilfe für den Knoten auf. Bei einem Dialogfeld mit Registerkarten oder einem Ausgabefenster wird die Hilfe für die jeweilige Registerkarte bzw. die Hilfe für das Ausgabefenster angezeigt. Die gleiche Hilfe können Sie auch mit der Taste F1 aufrufen.

Statusbereich

Der verbleibende Teil des oberen Dialogfeldbereichs ist für Informationen, Warnungen oder Fehlermeldungen reserviert. Bei Quellenknoten wird hier der vollständige Pfad und Dateiname der Datenquelle angezeigt. In diesem Bereich werden je nach Knotentyp unterschiedliche Informationen angezeigt. Jeglicher für diesen Bereich vorgesehene Text sollte sich auf zwei Zeilen beschränken.

Fensterbereich

Dies ist der Hauptbereich des Dialogfelds. Er enthält alle Steuerelemente und Anzeigebereiche des Knotens. Der Fensterbereich sieht auf jeder Registerkarte anders aus. Es gibt folgende Fensterbereichtypen:

- Textbrowser
- Erweiterungsobjekt
- Eigenschaften

Sie können auch Unterfensterbereiche definieren. Dies sind eigene Dialogfelder, die über Aktionsschaltflächen im Fensterbereich aufgerufen und in einem neuen Fenster geöffnet werden.
Für weitere Informationen siehe Thema Fensterspezifizierungen in Kapitel 6 auf S. 147.

Registerkartenbereich

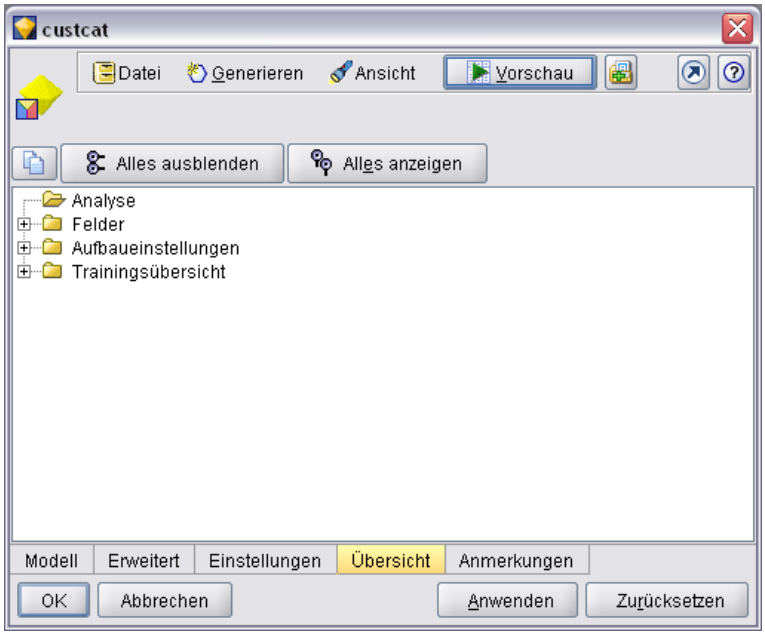
Knotendialogfelder enthalten die folgenden Registerkarten:

- Eine oder mehrere benutzerdefinierte knotenspezifische Registerkarten
- Registerkarte "Übersicht" (nur bei Modellausgabeobjekten und Modellanwendungsknoten)
- Registerkarte "Anmerkungen"

Die knotenspezifischen Registerkarten werden im Abschnitt Tabs der Spezifikationsdatei von CLEF definiert. Für weitere Informationen siehe Thema Registerkarten in Kapitel 6 auf S. 143.

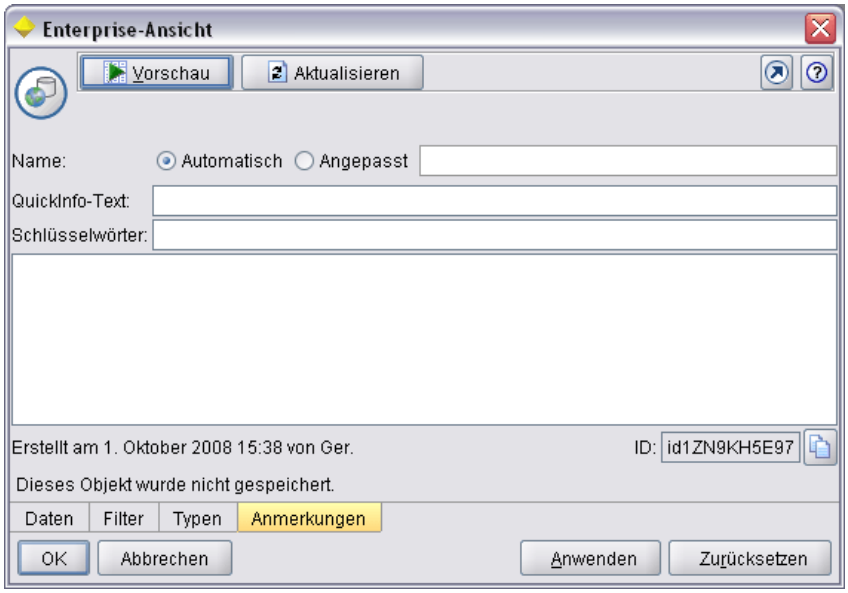
Den Dialogfeldern von Modellausgabeobjekten und Modellanwendungsknoten wird vom System automatisch die Registerkarte “Übersicht” hinzugefügt. Diese Registerkarte enthält zusammengefasste Informationen über das generierte Modell, einschließlich der Felder, der Generierungseinstellungen und des verwendeten Modellschätzungsprozesses. Die Ergebnisse werden in einer Baumansicht dargestellt, die durch Klicken auf bestimmte Elemente erweitert bzw. reduziert werden kann.

Abbildung 2-17
Registerkarte “Übersicht”.



Die Registerkarte “Anmerkungen” wird allen Knotendialogfeldern automatisch vom System hinzugefügt. Hier können Benutzer Informationen zum jeweiligen Knoten wie den Knotennamen, QuickInfo-Text und einen längeren Kommentar eingeben.

Abbildung 2-18
Registerkarte "Anmerkungen"



Name: Der Standardknotenname wird im Element "Node" der Spezifikationsdatei mit dem Attribut Label angegeben (siehe Knoten auf S. 62). Dieser Name kann vom Benutzer geändert werden. Dazu wählt er Benutzerdefiniert aus, gibt einen Namen im Bearbeitungsfeld "Benutzerdefiniert" ein und klickt auf Anwenden oder OK. Der neue Name wird auch in späteren Sitzungen beibehalten. Allerdings kann der Benutzer jederzeit zum Standardnamen zurückkehren, indem er auf Auto klickt. Ein auf der Registerkarte "Anmerkungen" angegebener benutzerdefinierter Name überschreibt jeden benutzerdefinierten Namen, der auf einer anderen Registerkarte des Dialogfelds festgelegt wurde.

QuickInfo-Text: Der hier angegebene Text wird im Zeichenbereich als QuickInfo für den Knoten angezeigt. Falls kein Text angegeben ist, wird keine QuickInfo eingeblendet, wenn der Benutzer mit dem Cursor auf den Knoten zeigt.

Schlüsselwörter: Hier kann der Benutzer Schlüsselwörter eingeben, die in Projektberichten oder beim Suchen oder Verfolgen von Objekten im IBM® SPSS® Collaboration and Deployment Services Repository verwendet werden.

Kommentarbereich: In diesem Bereich kann der Benutzer Kommentare eingeben.

Erstell- und Speicherinformationen: In diesem nicht bearbeitbaren Textbereich werden Informationen zur Erstellung, der Dateiname sowie Speicherdatum und -uhrzeit einer Datei angezeigt (das Datum-/Zeitformat ist von der Ländereinstellung abhängig). Falls das Element noch nicht gespeichert wurde, enthält dieses Feld die Meldung "This item has not been saved" (Dieses Element wurde noch nicht gespeichert).

Schaltflächenbereich

Im unteren Bereich jedes Dialogfelds befinden sich die Schaltflächen Anwenden, Zurücksetzen, OK und Abbrechen. Bei einem Endknoten (ein ausführbarer Knoten, der die Stream-Daten verarbeitet) befindet sich dort zusätzlich die Schaltfläche Ausführen.

Abbildung 2-19
Dialogfeldschaltflächen

Abbildung 2-20
Dialogfeldschaltflächen für Endknoten

OK: Übernimmt die Einstellungen und schließt das Dialogfeld. Direkt nach dem Öffnen des Dialogfelds aus dem Knoten ist diese Schaltfläche durch ein blaues Rechteck hervorgehoben, d. h., sie hat den Fokus. Wenn Sie die Eingabetaste drücken, solange die Schaltfläche den Fokus hat, wird die Funktion der Schaltfläche sofort ausgeführt.

Abbrechen: Schließt das Dialogfeld, wobei die neuen Einstellungen nicht gespeichert werden. Es werden also die Einstellungen übernommen, die das Dialogfeld beim Öffnen hatte bzw. die beim letzten Betätigen der Schaltfläche “Anwenden” gespeichert wurden. Die Funktion der Schaltfläche “Abbrechen” können Sie auch mit der Esc-Taste ausführen, sofern das gesamte Dialogfeld den Fokus hat.

Ausführen: Wendet die aktuellen Einstellungen an, schließt das Dialogfeld und führt den Endknoten aus.

Anwenden: Speichert die aktuellen Einstellungen des Dialogfelds, damit sie von den nächsten Operationen übernommen werden können.

Zurücksetzen: Setzt die Einstellungen des Dialogfelds auf diejenigen Einstellungen zurück, die das Dialogfeld beim Öffnen hatte bzw. die beim letzten Betätigen der Schaltfläche “Anwenden” gespeichert wurden.

Erstellen von Ausgabefenstern

In diesem Abschnitt werden die Merkmale der Standardausgabefenster von IBM® SPSS® Modeler beschrieben. Der Abschnitt soll Ihnen dabei helfen, auch in CLEF konsistente Ausgabefenster zu erstellen.

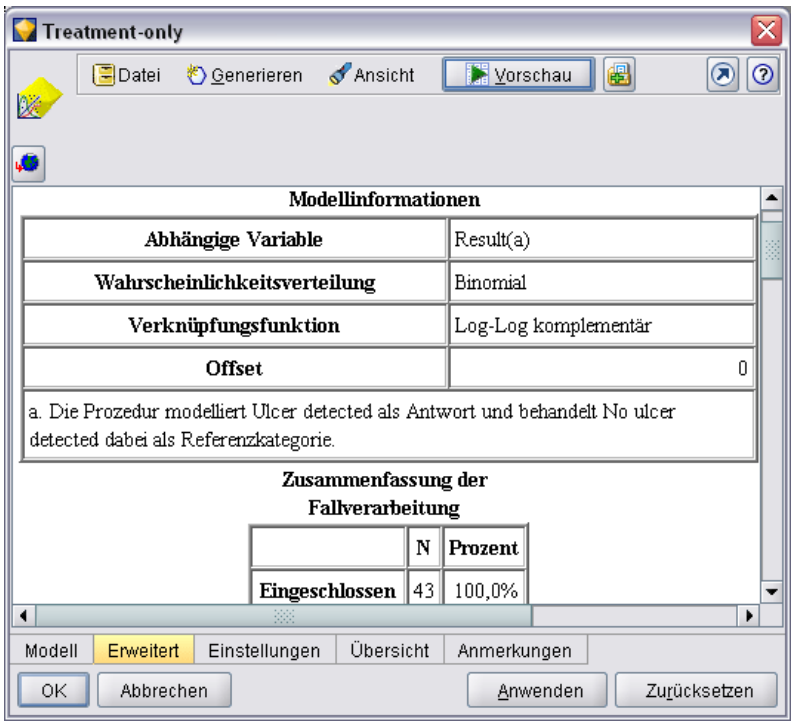
In Ausgabefenstern kann die Ausgabe folgender Elemente angezeigt werden:

- Die Ausgabe eines Modells, z. B. das Ergebnis aus dem Scoring eines Daten-Sets (durch Scoring wird ein Modell angewendet)
- Die Ausgabe eines Dokuments, z. B. ein Diagramm oder ein Bericht

Für weitere Informationen siehe Thema Info zu Benutzeroberflächen in Kapitel 6 auf S. 131.

Ein typisches Ausgabefenster sieht wie folgt aus:

Abbildung 2-21
Ausgabefenster eines Modells



Ausgabefenster unterscheiden sich nur in den folgenden Punkten von Knotendialogfeldern:

- Die Titelleiste enthält statt des allgemeinen Goldnugget-Symbols ein knotenspezifisches Miniatursymbol.
- Das Symbol des Hauptknotens fehlt.
- Im Symbolleisten- und Menübereich fehlt die Schaltfläche “Maximieren/Normale Größe”. Diese kann in einem Dokumentausgabefenster durch die Schaltfläche “Schließen” oder “Löschen” ersetzt sein. Das Fenster kann jedoch mit der Maus vergrößert und verkleinert werden.
- Der Statusbereich fehlt.
- Die typischen Registerkarten sind:

  Registerkarte “Modell” (für Modellausgabefenster), um die Daten zur Bedeutsamkeit des Prädiktors anzuzeigen, falls diese Option am Modellknoten ausgewählt wurde

  eine einzelne Registerkarte für die Ausgabe

  Registerkarte “Übersicht” (für Modellausgabefenster), um die zusammenfassenden Details über das Modell anzuzeigen

  Registerkarte “Anmerkungen” (Die Anmerkungswerte werden von dem Knoten übernommen, der die Ausgabe generiert hat.)
- Der Schaltflächenbereich enthält die Schaltflächen “OK”, “Abbrechen”, “Anwenden” und “Zurücksetzen”.

CLEF stellt Standardfenster für die Modell- und Dokumentausgabe bereit, die dem oben abgebildeten Fenster in etwa entsprechen. Diese Elemente werden normalerweise angezeigt, wenn in der Spezifikationsdatei ein ModelOutput- bzw. ein DocumentOutput-Element verwendet wird. Für weitere Informationen siehe Thema Objekt-ID in Kapitel 4 auf S. 62.

Sie können das Element ModelOutput bzw. DocumentOutput jedoch auch so definieren, dass statt des Standardausgabefensters ein von Ihnen selbst erstelltes Ausgabefenster angezeigt wird. Für weitere Informationen siehe Thema Benutzerdefinierte Ausgabefenster in Kapitel 6 auf S. 204.

Kapitel
3

CLEF-Beispiele

Informationen zu den Beispielen

Damit Sie schneller mit CLEF vertraut werden, beinhaltet die IBM® SPSS® Modeler-Installation verschiedene Beispielknoten mit vollständigem Quellcode. Es handelt sich hier um grundlegende Knoten mit eingeschränkter Funktionalität, die Ihnen lediglich helfen sollen, die Funktionsweise und Verwendung von CLEF zu verstehen. Sie können diese Knoten sofort oder wann immer Sie Zeit dazu finden, ausprobieren.

Folgende Beispiele werden bereitgestellt:

- Datenleserknoten (mit dem Namen “Apache Log Reader”)
- Datentransformationsknoten (mit dem Namen “URL Parser”)
- Dokumenterstellungsknoten (mit dem Namen “Web Status Report”)
- Modellerstellungsknoten (mit dem Namen “Interaction”)

Die Beispiele müssen vor der Verwendung aktiviert werden.

Aktivieren der Beispiele

Die Beispiele werden bei der Installation von IBM® SPSS® Modeler in komprimiertem Format im Verzeichnis *Demos* installiert. Zur Aktivierung der Beispiele müssen Sie die Dateien in die korrekten Verzeichnisse extrahieren.

Führen Sie dazu auf dem Computer, auf dem SPSS Modeler installiert ist, die folgenden Schritte aus:

▶ Beenden Sie SPSS Modeler, falls der Client ausgeführt wird.

▶ Suchen Sie die Datei *clef_examples_ext_lib.zip* im Verzeichnis *Demos* der SPSS Modeler-Installation.

▶ Extrahieren Sie den Inhalt der Datei *clef_examples_ext_lib.zip* in den Ordner *\ext\lib* des Installationsverzeichnisses von SPSS Modeler.

Wenn nur SPSS Modeler installiert ist:

▶ Extrahieren Sie den Inhalt der Datei *clef_examples_ext_bin.zip* in den Ordner *\ext\bin* des Installationsverzeichnisses von SPSS Modeler.

Bei einer separaten SPSS Modeler- und IBM® SPSS® Modeler Server-Installation:

▶ Extrahieren Sie den Inhalt der Datei *clef_examples_ext_bin.zip* in den Ordner *\ext\bin* die Installationsverzeichnisse von SPSS Modeler sowie SPSS Modeler Server.

- (Nur bei UNIX- und 64-Bit-Windows-Servern:) Verwenden Sie zusätzlich zum vorstehenden Schritt die in der Datei *clef_examples_ext_bin.zip* bereitgestellte Projektdatei (Makefile), um den Quellcode für die Beispiele zu kompilieren. Für weitere Informationen siehe Thema Untersuchen des Quellcodes auf S. 38.

Bei allen Installationen:

- Starten Sie SPSS Modeler und vergewissern Sie sich, dass die Knotenpalette folgende Knoten enthält:

Registerkarte "Palette"	**Knoten**
Quellen	Apache Log Reader
Feldoperationen	URL Parser
Modellierung	Wechselwirkung
Ausgabe	Web Status Report

Datenleserknoten (Apache Log Reader)

Der Datenleserknoten ist ein Datenquellenknoten, der die Daten aus der Zugriffsprotokolldatei eines Apache-HTTP-Webservers einliest. Das Zugriffsprotokoll enthält Informationen zu allen Anforderungen, die vom Webserver verarbeitet wurden. Die Protokolldatensätze sind im kombinierten Protokollformat (Combined Log Format) gespeichert. Beispiel:

IP_address - - [09/Jul/2007:07:57:38 +0000] "GET /lsearch.php?county_id=3 HTTP/1.1" 200 16348
"http://www.google.co.uk/search?q=thunderbirds+cliveden&hl=en&start=10&sa=N"
"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322)"

Mit diesem Beispielknoten können Sie die Protokolldatensätze in ein Tabellenformat konvertieren, das leichter zu lesen ist.

So verwenden Sie den Apache Log Reader-Knoten:

- Wenn Sie die Beispiele von CLEF noch nicht aktiviert haben, tun Sie es jetzt. Für weitere Informationen siehe Thema Aktivieren der Beispiele auf S. 33.
- Öffnen Sie IBM® SPSS® Modeler.
- Wählen Sie auf der Registerkarte "Datenquellen" der Knotenpalette Apache Log Reader aus und fügen Sie den Knoten dem Zeichenbereich hinzu.
- Bearbeiten Sie den Knoten. Geben Sie auf der Registerkarte "Option" im Feld "Apache Log File" (Apache-Protokolldatei) Folgendes ein:

Demo-Ordner\combined_log_format.txt

Dabei ist *Demo-Ordner* der Speicherort des Ordners *Demos* im Installationsverzeichnis von IBM® SPSS® Modeler (verwenden Sie nicht das Format *$CLEO_DEMOS*).

- Klicken Sie auf OK.

- Fügen Sie einen Typknoten zum Stream hinzu.
- Bearbeiten Sie den Typknoten. Klicken Sie auf Werte lesen, um die Daten einzulesen, und klicken Sie anschließend auf OK.
- Verbinden Sie einen Tabellenknoten mit dem Typknoten und führen Sie den Stream aus. Der Inhalt der Protokolldatei wird nun in einer Tabelle angezeigt.
- Speichern Sie den Stream für die nächsten beiden Beispiele.

Datentransformationsknoten (URL Parser)

Im Beispiel für den Datentransformationsknoten werden die im vorherigen Beispiel zurückgegebenen Daten weiter verarbeitet. Sie wählen ein ID-Feld aus (das für jede Zeile einen eindeutigen Wert enthalten sollte) sowie ein Eingabefeld mit einer URL. Die vom Knoten generierte Ausgabe besteht aus diesen beiden Feldern sowie den URL-Daten, die zusätzlich analysiert und in separat generierte Felder ausgegeben werden. Beispiel: Wenn ein URL-Datensatz einen Abfragestring wie den folgenden enthält:

http://www.dummydomain.co.uk/resource.php?res_id=89

wird der Datensatz wie folgt analysiert:

Generiertes Feld	**Inhalte**
*URLfield*_server	*http://www.dummydomain.co.uk*
*URLfield*_path	*/resource.php*
*URLfield*_field	*res_id*
*URLfield*_value	*89*

So verwenden Sie den URL Parser-Knoten:

- Falls Sie den Stream des vorangegangenen Beispiels geschlossen haben, öffnen Sie ihn wieder. Der Stream enthält die Knoten “Apache Log Reader” und “Typ”.
- Verbinden Sie auf der Registerkarte “Feldoperationen” der Knotenpalette einen URL Parser-Knoten mit dem Typknoten.
- Bearbeiten Sie den URL Parser-Knoten. Wählen Sie in der Dropdown-Liste “ID-Feld” die Option ReturnedContentSize aus. Wählen Sie in der Dropdown-Liste “URL-Feld” die Option ReferralURL aus. Klicken Sie auf OK.
- Verbinden Sie einen Tabellenknoten mit dem URL Parser-Knoten und führen Sie den Stream aus. Die Felder ReturnedContentSize und ReferralURL werden angezeigt, wobei ReferralURL zusätzlich in die folgenden vier separat generierten Felder analysiert wird: ReferralURL_server, ReferralURL_path, ReferralURL_field und ReferralURL_value.

Dokumenterstellungsknoten (Web Status Report)

Im Beispiel für den Dokumenterstellungsknoten werden die vom Webserverprotokoll übergebenen Daten eingelesen. Aus den eingelesenen Daten wird ein Bericht im Format HTML generiert. Der Bericht besteht aus einer Tabelle, die angibt, mit welcher Wahrscheinlichkeit (in Prozent) die einzelnen Protokolldatensätze verschiedene HTTP-Statuscodes zurückgeben (z. B. 200, 302, 404 usw.).

So verwenden Sie den Web Status Report-Knoten:

- Falls Sie den Stream des ersten Beispiels geschlossen haben, öffnen Sie ihn wieder. Es handelt sich hier um den Stream, der die Knoten "Apache Log Reader" und "Typ" enthält. Wenn Ihr Stream bereits den URL Parser-Knoten des zweiten Beispiels enthält, wird dieser Knoten in diesem Beispiel ignoriert.
- Verbinden Sie auf der Registerkarte "Ausgabe" der Knotenpalette einen Web Status Report-Knoten mit dem Typknoten.
- Bearbeiten Sie den Web Status Report-Knoten. Wählen Sie in der Dropdown-Liste "Status Code Field" (Statuscode-Feld) die Option StatusCode aus. Klicken Sie auf Ausführen. Ein Ausgabefenster mit dem Inhalt des Berichts wird angezeigt.

Modellerstellungsknoten (Interaction)

Das Beispiel für den Modellerstellungsknoten ist unabhängig von den anderen Beispielen. Es ermöglicht die Erstellung eines einfachen Modells mit dem nicht interaktiven Standardverfahren bzw. die Generierung eines Modells nach voriger Benutzerinteraktion. Das Modell erstellt Vorhersagen über die Kundenabwanderung für ein Telekommunikationsunternehmen.

So verwenden Sie den Interaction-Knoten:

- Wenn Sie die Beispiele von CLEF noch nicht aktiviert haben, tun Sie es jetzt. Für weitere Informationen siehe Thema Aktivieren der Beispiele auf S. 33.
- Erstellen Sie in IBM® SPSS® Modeler einen neuen Stream.
- Fügen Sie einen Statistikdatei-Quellenknoten hinzu, der die Datei *telco.sav* aus dem Verzeichnis *Demos* importiert.
- Klicken Sie auf der Registerkarte "Typen" auf Werte lesen und zur Bestätigung im Meldungsfenster auf OK.
- Setzen Sie die Rolle des Felds Churn ("Abwanderung", das letzte Feld in der Liste) auf Ziel und klicken Sie dann auf OK.
- Verbinden Sie auf der Registerkarte "Modellierung" der Knotenpalette einen Interaction-Knoten mit dem Quellenknoten.

So testen Sie das nicht interaktive Standardverfahren der Modellerstellung:

- Führen Sie den Stream aus, um ein Modell-Nugget im Stream und in der Modellpalette oben rechts im Fenster zu erstellen.

- Verbinden Sie einen Tabellenknoten mit dem Modell-Nugget.
- Führen Sie den Tabellenknoten aus. Blättern Sie im Tabellenausgabefenster nach rechts, um die Verlustvorhersagen anzuzeigen. Im Feld $I-churn werden die Prognosen angezeigt, während das Feld $IP-churn die Konfidenzwerte für die einzelnen Prognosen enthält (von 0,0 bis 0,1).

So testen Sie die interaktive Modellerstellung:

- Wählen Sie auf der Registerkarte "Modell" des Dialogfelds "Interaction Model Builder" (Modellerstellung durch den Knoten 'Interaction') die Option Start an interactive session (Interaktive Sitzung starten) aus.
- Klicken Sie auf Ausführen, um das Dialogfeld "Interaction Test" (Interaktionstest) zu öffnen.
- Klicken Sie im Dialogfeld "Interaction Test" (Interaktionstest) auf Start Build Task (Erstellungstask starten), um den Fortschritt der Modellerstellung anzuzeigen.
- Wählen Sie nach Beendigung der Modellerstellung die Zeile aus, die im Dialogfeld der Tabelle mit den Erstellungstasks hinzugefügt wurde.
- Klicken Sie oben im Symbolleistenbereich des Dialogfelds auf die Schaltfläche mit dem gelben, kristallförmigen Symbol. Dadurch wird das Modellausgabeobjekt (mit dem Namen model_1) in der Modellpalette oben rechts im Fenster erstellt.

  Das interaktiv generierte Modell ist mit Ausnahme seines Namens identisch mit dem zuvor erstellten Modell. Wenn Sie diesen Vorgang ab Start Build Task (Erstellungstask starten) wiederholen, wird ein weiteres identisches Modell mit dem Namen model_2 erstellt.

Untersuchen der Spezifikationsdateien

Wenn Sie wissen möchten, wie CLEF funktioniert, sollten Sie sich die Spezifikationsdateien der bereitgestellten Beispiele näher ansehen. Diese Dateien befinden sich in folgenden Verzeichnissen:

install_dir\ext\lib\
Erweiterungsordner\extension.xml

Dabei ist *Installationsverzeichnis* das Installationsverzeichnis von IBM® SPSS® Modeler und *Erweiterungsordner* ist einer der folgenden Ordner:

- *spss.apachelogreader*
- *spss.interaction*
- *spss.urlparser*
- *spss.webstatusreport*

Eventuell befinden sich unter *\ext\lib* noch weitere Erweiterungsordner. Diese gehören zu IBM® SPSS® Modeler-Knoten, die während der Verwendung von CLEF vom System erstellt wurden. Welche Knoten in Ihrer Installation vorhanden sind, richtet sich nach den von Ihnen lizenzierten SPSS Modeler-Modulen. Sie können sich diese Spezifikationsdateien gerne ansehen, um mehr über das Produkt zu erfahren, jedoch sollten Sie **diese Dateien in keiner Weise bearbeiten**. Anderenfalls besteht die Gefahr, dass die Knoten nicht mehr richtig funktionieren. Sie müssten in

diesem Fall die betreffenden SPSS Modeler-Produkte neu installieren. Änderungen an den vom System bereitgestellten Dateien werden von IBM Corp. nicht unterstützt.

Untersuchen des Quellcodes

Als Referenz wird der vollständige Quellcode der Beispielknoten bereitgestellt. Alle Beispielknoten verwenden serverseitige C++-Bibliotheken. Der Interaction-Knoten verwendet zusätzlich clientseitige Java-Klassen.

Die Quellcodedateien werden, sobald Sie die Beispiele aktivieren, automatisch extrahiert und in folgenden Verzeichnissen installiert:

Ort	Inhalte
...\ext\lib\spss.interaction\src	Java-Quellcode für die .class-Dateien in der Datei ui.jar im übergeordneten Ordner
...\ext\bin\spss.apachelogreader\src *...\ext\bin\spss.interaction\src* *...\ext\bin\spss.urlparser\src* *...\ext\bin\spss.webstatusreport\src*	C++-Quell- und Projektdateien für die DLLs im übergeordneten Ordner

Entfernen der Beispiele

Wenn Sie in IBM® SPSS® Modeler keine Beispielknoten anzeigen möchten, können Sie diese wie folgt entfernen:

- Beenden Sie IBM® SPSS® Modeler.
- Löschen Sie die Beispielordner sowohl aus dem Verzeichnis *\ext\bin* als auch aus dem Verzeichnis *\ext\lib* Ihrer SPSS Modeler-Installation. Achten Sie darauf, nicht versehentlich die SPSS Modeler-Standardordner zu löschen. In diesem Fall müssten Sie das betreffende SPSS Modeler-Produkt neu installieren. Folgende Ordner dürfen gelöscht werden:
  - *spss.apachelogreader*
  - *spss.urlparser*
  - *spss.webstatusreport*
  - *spss.interaction*

  Die Änderungen werden wirksam, sobald Sie SPSS Modeler das nächste Mal starten.

Kapitel
4

Spezifikationsdatei

Überblick über Spezifikationsdateien

Alle CLEF-Erweiterungen müssen eine XML-Datei besitzen, in der sämtliche Erweiterungsmerkmale definiert sind. Diese Datei wird als **Spezifikationsdatei** bezeichnet und hat stets den Namen extension.xml. Eine Spezifikationsdatei enthält folgende Abschnitte:

- **XML-Deklaration**. Optionale Deklaration der XML-Version und weitere Informationen.
- **Erweiterungselement**. Hauptteil der Datei; enthält alle nachfolgenden Abschnitte.
- **Abschnitt 'Erweiterungsdetails'**. Legt grundlegende Informationen über die Erweiterung fest.
- **Abschnitt 'Ressourcen'**. Spezifiziert externe Ressourcen, die für das Funktionieren der Erweiterung erforderlich sind, wie Ressourcenpakete, JAR-Dateien und freigegebene Bibliotheken.
- **Abschnitt 'Gemeinsame Objekte'**. (Optional) Definiert Elemente, die von anderen Objekten in der Erweiterung verwendet oder referenziert werden können, wie Modelle, Dokumente und Eigenschaftstypen.
- **Abschnitt 'Benutzeroberfläche (Paletten)'**. (Optional) Definiert eine benutzerdefinierte Palette oder Unterpalette, auf der ein Knoten erscheint.
- **Abschnitt 'Objektdefinition'**. Identifiziert die durch die Erweiterung definierten Objekte, wie Knoten, Modellausgaben und Dokumentausgaben.

Jeder Abschnitt kann statische Deklarationen enthalten (wie Komponenten in einem Element) oder einfache dynamische Prozesse (wie die Berechnung eines Ausgabedatenmodells eines Knotens) oder beide. Das Gesamtformat einer CLEF-Spezifikationsdatei sieht wie folgt aus:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Extension ... >
  <ExtensionDetails ... />
  <Ressourcen
    Abschnitt 'Ressourcen'
  </Resources>
 <CommonObjects>
    Abschnitt 'Gemeinsame Objekte'
  </CommonObjects>
 <UserInterface>
    Abschnitt 'Benutzeroberfläche (Paletten)'
  </UserInterface>
  Abschnitt 'Objektdefinition'
    Objektdefinition
    Objektdefinition
    Objektdefinition
   ...
</Extension>
```

Kommentarzeilen

In der Spezifikationsdatei können Sie jederzeit eine Kommentarzeile im folgenden Format einfügen:

<!-- *Kommentartext* -->

Erforderlich oder optional?

In den Elementdefinitionen in nachfolgenden Abschnitten (normalerweise durch die Überschrift **Format** gekennzeichnet) sind Elementattribute und untergeordnete Elemente optional, sofern sie nicht als "(erforderlich)" gekennzeichnet sind. Die vollständige Elementsyntax ist in Anhang A*CLEF XML-Schema* auf S. 257 beschrieben.

Beispiel für eine Spezifizierungsdatei

Hier sehen Sie ein vollständiges Beispiel für eine CLEF-Spezifizierungsdatei für einen einfachen Datentransformationsknoten.

```
<?xml version="1.0" encoding="UTF-8"?>
<Extension version="1.0" debug="true">
   <ExtensionDetails id="urlparser" providerTag="spss" label="URL CLEF Module" version="1.0"
provider="IBM Corp." copyright="(c) 2005-2011 IBM Corp." description="A Url Transform CLEF Extension"/>
   <Resources>
      <SharedLibrary id="urlparser_library" path="spss.urlparser/urlparser" />
   </Resources>
   <Node id="urlparser_node" type="dataTransformer" palette="fieldOp" label="URL Parser">
      <Properties>
         <Property name="id_fieldname" valueType="integer" label="ID field" />
         <Property name="url_fieldname" valueType="string" label="URL field" />
      </Properties>
      <UserInterface>
         <Icons />
         <Tabs>
            <Tab label="Types" labelKey="optionsTab.LABEL">
               <PropertiesPanel>
                  <SingleFieldChooserControl property="id_fieldname"  storage="integer" />
                  <SingleFieldChooserControl property="url_fieldname"  storage="string" />
               </PropertiesPanel>
            </Tab>
         </Tabs>
         <Controls />
      </UserInterface>
      <Execution>
         <Module libraryId="urlparser_library" name="">
            <StatusCodes>
               <StatusCode code="0" status="error" message="Cannot initialise a peer" />
               <StatusCode code="1" status="error" message="error reading input data" />
               <StatusCode code="2" status="error" message="Internal Error" />
               <StatusCode code="3" status="error" message="Input Field Does Not Exist" />
            </StatusCodes>
```

```
        </Module>
      </Execution>
      <OutputDataModel mode="replace">
        <AddField name="${id_fieldname}" fieldRef="${id_fieldname}"/>
        <AddField name="${url_fieldname}" fieldRef="${url_fieldname}"/>
        <AddField name="${url_fieldname}_server" storage="string" />
        <AddField name="${url_fieldname}_path" storage="string" />
        <AddField name="${url_fieldname}_field" storage="string" />
        <AddField name="${url_fieldname}_value" storage="string" />
      </OutputDataModel>
    </Node>
</Extension>
```

Das Element ExtensionDetails liefert grundlegende Informationen über die Erweiterung, die intern durch IBM® SPSS® Modeler verwendet wird.

Das Element Resources gibt den Speicherort einer serverseitigen Bibliothek an, die später in der Datei referenziert wird. Die Pfadspezifizierung gibt an, dass sich die Bibliothek im SPSS Modeler-Installationsverzeichnis unter \ext\bin\spss.urlparser\urlparser.dll befindet.

Diese Spezifikationsdatei enthält kein CommonObjects-Element.

Das Element Node spezifiziert alle Informationen über den Knoten als solchen:

- Unter Properties werden anfangs zwei Eigenschaften deklariert, die später auf einer Registerkarte des Knotendialogfelds verwendet werden.
- Das Element UserInterface definiert das Aussehen und das Layout der Registerkarte des Kontendialogfelds, das zu dieser Erweiterung gehört (weitere Registerkarten werden durch SPSS Modeler bereitgestellt).
- Das Element Execution definiert Elemente, die beim Ausführen des Knotens verwendet werden. In diesem Fall handelt es sich bei den Elementen um die serverseitige Bibliothek, die zuvor in der Datei deklariert wurde, und um einen Satz Meldungen, die angezeigt werden, wenn bei der Ausführung ein bestimmter Statuscode zurückgegeben wird.
- Das Element OutputDataModel definiert die Datentransformation, die dieser Knoten durchführt. Es legt fest, dass das Eingabedatenmodel (der Satz an Feldern, der als Eingabe für den Knoten verwendet wird) durch den hier definierten Satz an Feldern ersetzt werden soll, wodurch das Ausgabedatenmodell erstellt wird (der Satz an Feldern, der von hier aus an alle nachfolgenden Knoten übergeben wird, sofern das Modell anschließend nicht weiter angepasst wird). In diesem Beispiel übergibt der Knoten die beiden Originalfelder (id_fieldname und url_fieldname) unverändert, fügt aber vier weitere Felder hinzu, deren Namen von url_fieldname abgeleitet sind.

Die spezielle Spezifikationsdatei wird von einem der Beispielknoten übernommen, die im Rahmen der IBM® SPSS® Modeler-Installation bereitgestellt werden. Für weitere Informationen siehe Thema Datentransformationsknoten (URL Parser) in Kapitel 3 auf S. 35.

XML-Deklaration

Die XML-Deklaration ist optional und gibt an, welche XML-Version verwendet sowie Details des Zeichenkodierungsformats.

Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
```

Erweiterungselement

Das Erweiterungselement bildet den Hauptteil der Datei und enthält alle anderen Abschnitte. Format:

```
<Extension version="version_number" debug="true_false">
   Abschnitt 'Erweiterungsdetails'
   Abschnitt 'Ressourcen'
   Abschnitt 'Gemeinsame Objekte'
   Abschnitt 'Benutzeroberfläche (Paletten)'
   Abschnitt 'Objektdefinition'
</Extension>
```

Dabei gilt:

version ist die Versionsnummer der Erweiterung.

debug ist optional; wenn true festgelegt ist, wird zu allen CLEF-Knoten oder -Ausgaben zugeordneten Dialogfeldern oder Frames eine Registerkarte Fehlersuche hinzugefügt, die Zugriff auf die für das Objekt definierten Eigenschaften und Container bietet. Der Standardwert ist false. Für weitere Informationen siehe Thema Verwenden der Registerkarte “Fehlersuche” in Kapitel 10 auf S. 253.

Abschnitt ‘Erweiterungsdetails’

Der Abschnitt ‘Erweiterungsdetails’ enthält grundlegende Informationen zur Erweiterung.

Format

```
<ExtensionDetails providerTag="extension_provider_tag"
   id="extension_unique_identifier"
   label="display_name" version="extension_version_number"
   provider="extension_provider" copyright="copyright_notice"
    description="extension_description"/>
```

Dabei gilt:

providerTag (erforderlich) ist der Name, der den Provider dieser Erweiterung eindeutig definiert. Achten Sie darauf, dass die Zeichenfolge spss innerhalb des Werts nicht vorkommen darf, da diese für die interne Verwendung reserviert ist.

id (erforderlich) ist ein Name, der diese Erweiterung eindeutig identifiziert und in Systemmeldungen verwendet wird. In der Regel wird die Erweiterungsdatei in einem Ordner namens \ext\lib*providerTag.id* im IBM® SPSS® Modeler-Installationsverzeichnis gespeichert.

label (erforderlich) ist die Anzeigebezeichnung der Erweiterung. Dieser Text wird im Feld 'Name' des Paletten-Managers angezeigt, wenn der Knoten hinzugefügt wird. Für weitere Informationen siehe Thema Testen einer CLEF-Erweiterung in Kapitel 10 auf S. 251.

version ist die Versionsnummer der Erweiterung.

provider ist eine Zeichenkette, die den Provider der Erweiterung angibt. Dieser Text wird im Feld 'Provider' des Paletten-Managers angezeigt, wenn der Knoten hinzugefügt wird. Der Standardwert ist die Zeichenkette (unknown).

copyright ist der Copyrighthinweis für die Erweiterung. Dieser Text wird im Feld 'Copyright' des Paletten-Managers angezeigt, wenn der Knoten hinzugefügt wird.

description ist eine kurze Beschreibung des Zwecks der Erweiterung. Dieser Text wird im Feld 'Beschreibung' des Paletten-Managers angezeigt, wenn der Knoten hinzugefügt wird.

Beispiel

```
<ExtensionDetails providerTag="myco" id="sorter" name="Sort Data" version="1.2"
   provider="My Company Inc." copyright="(c) 2005-2006 My Company Inc."
   description="Ein Beispiel für eine Erweiterung, die Daten mittels integrierter Betriebssystembefehle sortiert."/>
```

Abschnitt 'Ressourcen'

Dieser Abschnitt definiert, welche externen Ressourcen erforderlich sind, damit die Erweiterung korrekt funktioniert.

Format

```
<Resources>
   <Bundle .../>
   ...
   <JarFile .../>
   ...
   <SharedLibrary .../>
   ...
   <HelpInfo .../>
</Resources>
```

Dabei gilt:

Bundle identifiziert einen Satz clientseitiger lokalisierter Ressourcen. Für weitere Informationen siehe Thema Pakete auf S. 44.

JarFile identifiziert eine clientseitige Java-JAR-Datei. Für weitere Informationen siehe Thema Jar-Dateien auf S. 44.

SharedLibrary identifiziert eine serverseitige Bibliothek oder DLL. Für weitere Informationen siehe Thema Freigegebene Bibliotheken auf S. 45.

HelpInfo spezifiziert den Typ der Hilfeinformationen für die Erweiterung, sofern welche vorliegen. Für weitere Informationen siehe Thema Implementieren eines Hilfesystems in Kapitel 7 auf S. 208.

Beispiel

```
<Resources>
   <SharedLibrary id="discriminantnode" path="spss.xd/Discriminant"/>
   <Bundle id="translations.discrim" type="properties" path="messages"/>
   <JarFile id="java" path="discriminant.jar"/>
   <HelpInfo id="help" type="native"/>
</Resources>
```

Pakete

Das Element Bundle spezifiziert ein clientseitiges Ressourcenpaket (wie einen Satz Meldungtexte für die Lokalisierung), das entweder als eine .properties-Datei oder eine Java-.class-Datei implementiert ist. Für weitere Informationen siehe Thema Lokalisierung in Kapitel 8 auf S. 211.

Format

<Bundle id="*identifier*" path="*path*"/>

Dabei gilt:

id (erforderlich) ist ein eindeutiger Bezeichner für dieses Paket.

path (erforderlich) spezifiziert den Speicherort der Paketdatei relativ zum übergeordneten Ordner der Spezifikationsdatei. Wenn sich das Paket auf eine .properties-Datei bezieht, muss der Pfad keine Spracherweiterungen oder den .properties-Suffix enthalten.

Beispiel

```
<Bundle id="translations.discrim" path="messages"/>
```

Dies gibt an, dass eine Ressourcenpaket in einer Datei namens messages.properties in demselben Ordner wie die Spezifikationsdatei vorhanden ist.

Jar-Dateien

Das Element JarFile spezifiziert eine clientseitige Java-Archivdatei (.jar), die Java-Klassen und sonstige clientseitige Ressourcen für die Erweiterung bereitstellt.

Format

<JarFile id="*identifier*" path="*path*"/>

Dabei gilt:

id (erforderlich) ist ein eindeutiger Bezeichner für diese .jar-Datei.

path (erforderlich) spezifiziert den Speicherort der .jar-Datei relativ zum übergeordneten Ordner der Spezifikationsdatei.

Beispiel

```
<JarFile id="java" path="coxreg_model_terms.jar"/>
```

Dies gibt an, dass sich die .jar-Datei für diese Erweiterung in demselben Ordner wie die Spezifikationsdatei befindet.

Freigegebene Bibliotheken

Das Element SharedLibrary spezifiziert eine serverseitige freigegebene Bibliothek oder DLL. Dies ist in der Regel nur zur Unterstützung der Knotenausführung erforderlich. Wenn eine Bibliothek mehrere Module implementiert, identifiziert ein Module-Element im Ausführungsabschnitt der Knotenspezifikation ein bestimmtes Modul innerhalb der Bibliothek.

Format

```
<SharedLibrary id="identifier" path="path"/>
```

Dabei gilt:

id (erforderlich) ist ein eindeutiger Bezeichner für die freigegebene Bibliothek.

path (erforderlich) spezifiziert den Speicherort der freigegebenen Bibliothek relativ zum Ordner \ext\bin im serverseitigen Installationsverzeichnis. Beachten Sie, dass der Pfad die Dateierweiterung (z. B. .dll) der freigegebenen Bibliothek nicht enthalten muss.

Beispiel

Folgende Deklaration einer freigegebenen Bibliothek:

```
<SharedLibrary id="Binning" path="spss.binning/Binning" />
```

spezifiziert, dass die freigegebene Bibliothek geladen wird aus:

install_dir\ext\bin\spss.binning\Binning.dll

wobei *install_dir* das Verzeichnis ist, in dem die serverseitigen CLEF-Komponenten installiert sind. Da diese Bibliothek mehr als ein Modul implementiert, wird das konkret benötigte Modul (supervisedBinning) mittels eines Module-Elements in der Spezifikationsdatei des Erstellungsknotens identifiziert, wobei das Bibliothekskennzeichen wie folgt referenziert wird:

```
<Execution>
 <Module libraryId="Binning" name="supervisedBinning" .../>
 ...
```

```
</Execution>
```

Hilfeinformationen

Das optionale Element HelpInfo gibt an, welche der möglichen Hilfetypen für die Erweiterung bereitgestellt werden. Für weitere Informationen siehe Thema Implementieren eines Hilfesystems in Kapitel 7 auf S. 208.

Abschnitt 'Gemeinsame Objekte'

Der optionale Abschnitt 'Gemeinsame Objekte' definiert Objekte, die von an anderer Stelle in der Spezifikationsdatei definierten Elementen gemeinsam genutzt werden können. Einige Objekttypen in diesem Abschnitt (wie Eigenschaftsaufzählungen) können auch dort, wo sie benötigt werden, lokal definiert werden, während andere (wie Modelle und Dokumente) nur hier definiert werden können.

Format

```
<CommonObjects>
   <PropertyTypes .../>
   <PropertySets .../>
   <ContainerTypes .../>
   <Actions .../>
   <Catalogs .../>
</CommonObjects>
```

Dabei gilt:

PropertyTypes erlaubt, dass allgemeine Eigenschaftsdefinitionen von Objekten gemeinsam verwendet werden. Für weitere Informationen siehe Thema Eigenschaftstypen auf S. 47.

PropertySets wird in der Regel verwendet, wenn Modellierungsknoten, Modellausgabeobjekte und Modellanwendungsknoten denselben Satz an Eigenschaften enthalten. Für weitere Informationen siehe Thema Eigenschaftssätze auf S. 48.

ContainerTypes definiert Typen von Containern, bei denen es sich um Objekte handelt, die komplexe Datenstrukturen aufnehmen können. Für weitere Informationen siehe Thema Containertypen auf S. 49.

Actions definieret grundlegende Informationen zu Benutzerinteraktionen, z. B. mittels Menüs oder Symbolleisten. Für weitere Informationen siehe Thema Aktionen auf S. 51.

Catalogs implementieren eine Steuerung, mit deren Hilfe Sie eine oder mehrere Optionen aus einer Werteliste auswählen können, die der Server dynamisch erzeugt.Für weitere Informationen siehe Thema Kataloge auf S. 52.

Beispiel

```
<CommonObjects>
   <ContainerTypes>
      <ModelType id="discriminant_model" format="utf8" />
      <DocumentType id="html_output" />
      <DocumentType id="zip_outputType" format="binary"/>
   </ContainerTypes>
</CommonObjects>
```

Eigenschaftstypen

Der optionale Abschnitt ‘Eigenschaftstypen’ erlaubt die gemeinsame Verwendung von Eigenschaftsdefinitionen durch Objekte. Dies dient teilweise einer leichteren Wartung - so kann die Definition einer Eigenschaft z. B. an einer einzelnen Stelle hinterlegt sein und muss nicht an mehrere Stellen kopiert werden. Die Freigabe von Definitionen wird außerdem verwendet, um die Kompatibilität von Eigenschaften in verschiedenen Objekten sicherzustellen, wenn deren Werte kopiert werden, sobald eine neue Instanz eines Objekts erstellt wird.

Eigenschaftstypen können nur im Abschnitt ‘Gemeinsame Objekte’ definiert werden.

Format

```
<PropertyTypes>
 <PropertyType id="identifier" isKeyed="true_false" isList="true_false" max="max_value"
     min="min_value" valueType="value_type">
       <Enumeration ... />
       <Structure ... />
       <DefaultValue ... />
    </PropertyType>
 <PropertyType ... />
    ...
</PropertyTypes>
```

Die PropertyType-Attribute sind im Folgenden aufgeführt.

id (erforderlich) ist ein eindeutiger Bezeichner für den Eigenschaftstyp.

isKeyed gibt, sofern die Einstellung true vorgenommen wurde, an, dass der Eigenschaftstyp verschlüsselt ist. Eine Schlüsseleigenschaft verknüpft eine Gruppe von Operationen durch ein benutzerdefiniertes Steuerelement mit einem Feld (siehe Eigenschaftssteuerelement auf S. 175). Wenn isKeyed auf true gesetzt ist, muss das Attribut valueType auf structure gesetzt sein. Weitere Informationen zu strukturierten Eigenschaften erhalten Sie unter Strukturierte Eigenschaften auf S. 78 .

isList spezifiziert, ob es sich bei der Eigenschaft um eine Liste von Werten des spezifizierten Werttyps (true) handelt oder um einen einzelnen Wert (false).

max und min geben den maximalen und den minimalen Wert für einen Bereich an.

valueType kann einen der folgenden Werte besitzen:

- string
- encryptedString
- fieldName
- integer
- double
- boolean
- date
- enum (siehe Aufgezählte Eigenschaften auf S. 77)
- structure (siehe Strukturierte Eigenschaften auf S. 78)
- databaseConnection

Die untergeordneten Elemente Enumeration und Structure schließen sich gegenseitig aus. Die untergeordneten Elemente Enumeration, Structure und DefaultValue werden in speziellen Fällen verwendet. Siehe hierzu Aufgezählte Eigenschaften auf S. 77, Strukturierte Eigenschaften auf S. 78 und Standardwerte auf S. 81.

Eigenschaftssätze

Eigenschaftssätze werden in der Regel verwendet, wenn Modellierungsknoten, Modellausgabeobjekte und Modellanwendungsknoten denselben Satz an Eigenschaften enthalten. Beispiel: Ein Modellierungsknoten kann einen Standardsatz von Eigenschaften definieren, die im Builder festgelegt werden können, die aber erst bei der Modellanwendung verwendet werden. Damit sie automatisch übertragen werden, müssen sie außerdem in der Modellausgabe enthalten sein.

Format

```
<PropertySets>
 <PropertySet id="identifier">
       <Property ... />
       <Property ... />
        ...
   </PropertySet>
     ...
</PropertySets>
```

wobei id ein eindeutiger Bezeichner für den Eigenschaftssatz ist.

Eine Beschreibung des Elements ‘Property’ finden Sie unter Eigenschaften auf S. 66.

Beispiel

Dieses Beispiel demonstriert die Definition eines Satzes von zwei Eigenschaften: die Anzahl der zu erzeugenden Vorhersagen und ob Wahrscheinlichkeiten enthalten sein sollen. Im Abschnitt ‘Gemeinsame Objekte’ erfolgt folgende Definition:

```
<PropertySets>
 <PropertySet id="common_model_properties">
      <Property name="prediction_count" valueType="integer" min="1" max="10"/>
      <Property name="include_probabilities" valueType="boolean" defaultValue="false"/>
   </PropertySet>
    ...
</PropertySets>
```

Anschließend wird in alle Definitionen für den Modellierungsknoten, das Modellausgabeobjekt und den Modellanwendungsknoten ein includePropertySets-Attribut eingefügt, das wie folgt aussieht (diese Definition gilt nur für den Modellierungsknoten):

```
<Node id="my_builder" type="modelBuilder" ... >
   <Properties includePropertySets="[common_model_properties]">
    ...
   </Properties>
    ...
</Node>
```

Containertypen

Container sind Objekte, die als Platzhalter für komplexe Datenstrukturen wie Modelle und Dokumente agieren. Ein Container wird als bestimmter Containertyp definiert. Im Folgenden sind die Typdefinitionen aufgeführt. Folgende Containertypen können definiert werden:

- Modelltypen
- Dokumenttypen

Containertypen können zwischen Client und Server übertragen werden, geklont und in einer Datei oder einem Inhalts-Repository gespeichert werden. Ein Modell wird geklont, wenn aus einem Modellausgabeobjekt ein Modellanwendungsknoten generiert wird.

Jeder Containertyp besitzt einen vordefinierten Satz von Eigenschaften, obwohl benutzerdefinierte Eigenschaften hinzugefügt werden können. Containertypen können nur im Abschnitt ‘Gemeinsame Objekte’ definiert werden.

Format

Das Format des Abschnitts ‘Containertypen’ sieht wie folgt aus:

```
<ContainerTypes>
 <ModelType ... />
    ...
   <DocumentType ... />
```

```
   ...
</ContainerTypes>
```

Dabei gilt:

ModelType spezifiziert das Format für einen bestimmten Modelltyp. Für weitere Informationen siehe Thema Modelltypen auf S. 50.

DocumentType spezifiziert das Format für einen bestimmten Dokumenttyp. Für weitere Informationen siehe Thema Dokumenttypen auf S. 50.

Beispiel

```
<ContainerTypes>
   <ModelType id="discriminant_model" format="utf8">
   <DocumentType id="html_output" />
   <DocumentType id="zip_outputType" format="binary"/>
</ContainerTypes>
```

Modelltypen

Ein Modell muss Informationen bereitstellen - wie Algorithmusname, Modelltyp sowie Eingabe- und Ausgabedatenmodell. Eine Modelltypdefinition spezifiziert das Format für einen bestimmten Modelltyp.

Die Modelltypdefinition kann hier statisch in der Spezifikationsdatei festgelegt sein oder dynamisch, wenn das Modell durch den Modellierungsknoten errichtet wird.

Format

```
<ModelType id="identifier" format="model_type_format" />
```

Dabei gilt:

- id (erforderlich) ist ein eindeutiger Bezeichner für den Modelltyp.
- format (erforderlich) ist das Format des Modelltyps. Es kann entweder utf8 (Text) oder binary sein. Das Modellformat muss als Teil der statischen Informationen spezifiziert sein.

Beispiel

```
<ModelType id="my_model" format="utf8" />
```

Dokumenttypen

Ein **Dokument** ist ein Ausgabeobjekt wie ein Diagramm oder ein Bericht. Eine Dokumenttypdefinition spezifiziert das Format für einen bestimmten Dokumenttyp.

Format

```
<DocumentType id="identifier" format="document_type_format" />
```

Dabei gilt:

- id (erforderlich) ist ein eindeutiger Bezeichner für den Dokumenttyp.
- format (erforderlich) ist das Format des Dokumenttyps. Es kann entweder utf8 (Text) oder binary sein.

Beispiele

```
<DocumentType id="html_output" format="utf8" />
<DocumentType id="zip_outputType" format="binary"/>
```

Aktionen

Aktionen definieren grundlegende Informationen zu Benutzerinteraktionen, z. B. mittels Menüs oder Symbolleisten. Jede Aktion definiert, wie sie auf der Benutzeroberfläche dargestellt werden soll, wie z. B. als Beschriftung, QuickInfo oder Symbol. Eine Sammlung von Aktionen wird durch eine clientseitige Java-Klasse behandelt, die für jede Aktionsgruppe definiert ist. Aktionen können auch innerhalb bestimmter Objekte definiert werden.

Format

```
<Actions>
 <Action id="identifier" label="display_label" labelKey="label_key" description="action_description"
       descriptionKey="description_key" imagePath="image_path" imagePathKey="image_path_key"
       mnemonic="mnemonic_char" mnemonicKey="mnemonic_key" shortcut="shortcut_string"
       shortcutKey="shortcut_key" />
    ...
</Actions>
```

Dabei gilt:

id (erforderlich) ist ein eindeutiger Bezeichner für die Aktion.

label (erforderlich) ist der Anzeigename für die Aktion, mit dem sie auf der Benutzeroberfläche angezeigt wird.

labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke.

description ist eine Beschreibung der Aktion — Beispiel: für ein benutzerdefiniertes Menüelement oder ein Aktionsschaltflächensymbol in einer Symbolleiste wäre dies der Text der QuickInfo für das Menüelement oder die Schaltfläche.

descriptionKey kennzeichnet die Beschreibung für Lokalisierungszwecke.

imagePath ist der Speicherort der Grafikdatei, wie beispielsweise ein Symbolbild. Der Speicherort wird relativ zu dem Verzeichnis angegeben, in dem die Spezifikationsdatei installiert ist.

imagePathKey gibt den Bildpfad für Lokalisierungszwecke an.

mnemonic ist der Buchstabe, der zusammen mit der Alt-Taste gedrückt wird, um diese Steuerung zu aktivieren (wenn z. B. der Wert S angegeben ist, kann der Benutzer diese Steuerung über Alt+S aktivieren).

mnemonicKey kennzeichnet die mnemonische Taste für Lokalisierungszwecke. Wenn weder mnemonic noch mnemonicKey verwendet wird, ist keine Direktzugriffstaste für diese Aktion verfügbar. Für weitere Informationen siehe Thema Zugriffstasten und Tastenkombinationen in Kapitel 6 auf S. 145.

shortcut ist ein String (Zeichenfolge), der Direktzugriffstasten angibt (z. B. CTRL+SHIFT+A), die zum Initiieren dieser Aktion verwendet werden können.

shortcutKey kennzeichnet die Direktzugriffstaste für Lokalisierungszwecke. Wenn weder shortcut noch shortcutKey verwendet wird, ist keine Direktzugriffstaste für diese Aktion verfügbar. Für weitere Informationen siehe Thema Zugriffstasten und Tastenkombinationen in Kapitel 6 auf S. 145.

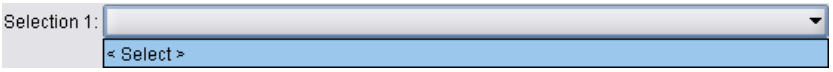
Beispiel

```
<Actions>
   <Action id="generateSelect" label="Select Node..." labelKey="generate.selectNode.LABEL"
    imagePath="images/generate.gif" description="Generates a select node"
    descriptionKey="generate.selectNode.TOOLTIP"/>
   <Action id="generateDerive" label="Derive Node..." labelKey="generate.deriveNode.LABEL"
    imagePath="images/generate.gif" description="Generates a derive node"
    descriptionKey="generate.deriveNode.TOOLTIP"/>
</Actions>
```

Kataloge

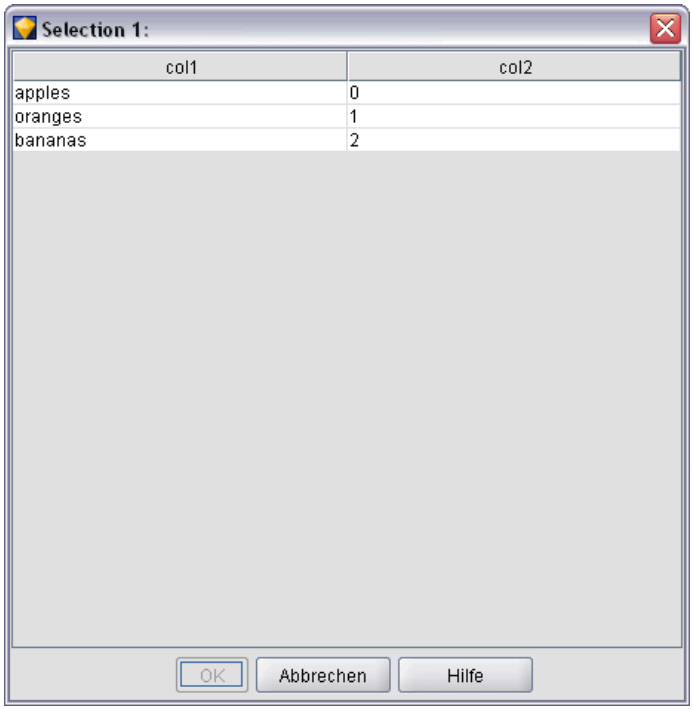
Kataloge ermöglichen Ihnen, eine Eigenschaft mit einem Steuerelement zu verknüpfen, damit der Benutzer eine oder mehrere Optionen aus einer Werteliste wählen kann, die der Server dynamisch erzeugt.

Abbildung 4-1
Steuerung zur Auswahl aus Werteliste



Die Werte werden im Steuerelement als Popup-Liste angezeigt, wenn der Benutzer auf den Eintrag <Auswahl> klickt.

Abbildung 4-2
Liste der angezeigten Werte



Wenn der Benutzer eine Zeile in der Liste auswählt, wird der Zeilenwert aus einer Spalte, die im Catalog-Element angegeben ist, im Steuerelement platziert.

Abbildung 4-3
Auswahl aus Werteliste

Format

```
<CommonObjects>
 <Catalogs>
 <Catalog id="identifier" valueColumn="integer">
          <Attribute label="display_name" />
           ...
       </Catalog>
        ...
   </Catalogs>
</CommonObjects>
```

Dabei gilt:

id (erforderlich) ist ein eindeutiger Bezeichner für den Katalog.

valueColumn (erforderlich) ist die Nummer der Spalte, deren Wert in das Steuerelement übertragen wird, sobald der Benutzer eine Zeile auswählt. Die Spaltennummerierung beginnt bei 1.

Verwenden Sie ein Attribute-Element pro Spalte in Spaltenreihenfolge (siehe folgendes Beispiel).

Wenn der Benutzer ein Steuerelement aktiviert, das mit einem Katalog verknüpft ist, wird der Katalog mit der Werteliste durch einen Aufruf der Funktion getCatalogInformation vom Server abgerufen. Diese Funktion gibt ein XML-Dokument zurück, das die Werteliste enthält. Für weitere Informationen siehe Thema Peer-Funktionen in Kapitel 9 auf S. 224.

Beispiel

Dieses Beispiel zeigt etwas von dem Code zur Definition der oben dargestellten Katalogsteuerungen. Es werden drei Kataloge definiert und mit drei verschiedenen Steuerungen auf der Registerkarte eines Dialogfelds verbunden.

Zuerst werden die Kataloge im Abschnitt ‘CommonObjects’ definiert.

```
<CommonObjects>
 <Catalogs>
 <Catalog id="cat1" valueColumn="1">
 <Attribute label="col1" />
 <Attribute label="col2" />
 </Catalog>
 <Catalog id="cat2" valueColumn="2">
 <Attribute label="col1" />
 <Attribute label="col2" />
 <Attribute label="col3" />
 </Catalog>
 <Catalog id="cat3" valueColumn="1">
          <Attribute label="col1" />
       </Catalog>
    </Catalogs>
</CommonObjects>
```

Als Nächstes werden die Eigenschaften, die mit den Steuerelementen verknüpft werden sollen, im Abschnitt “Properties” der Knotendefinition definiert:

```
<Node id="catalognode" type="dataReader" palette="import" label="Catalog">
 <Properties>
 <Property name="sometext" valueType="string" label="Some Text" />
 <Property name="selection1" valueType="string" label="Selection 1" />
 <Property name="selection2" valueType="string" isList="true" label="Selection 2" />
 <Property name="selection3" valueType="string" label="Selection 3" />
    </Properties>
```

Im Abschnitt “UserInterface” der Knotendefinition werden die Steuerungen definiert und durch Verweise auf die Eigenschaften mit den Katalogdefinitionen verknüpft:

```
<UserInterface>
 <Tabs>
 <Tab label="Catalog Controls" labelKey="Catalog.LABEL" >
 <PropertiesPanel>
 <TextBoxControl property="sometext" />
 <SingleItemChooserControl property="selection1" catalog="cat1" />
```

```
        <MultiItemChooserControl property="selection2" catalog="cat2" />
        <SingleItemChooserControl property="selection3" catalog="cat3" />
      </PropertiesPanel>
</Tab>
```

Abschnitt ‘Benutzeroberfläche (Paletten)’

Dies ist ein optionaler Abschnitt, der nur dann angegeben wird, wenn die Erweiterung eine benutzerdefinierte Palette oder Unterpalette definiert, auf der der Knoten erscheinen soll.

Wenn eine Erweiterung eine benutzerdefinierte Palette oder Unterpalette definiert, kann bei nachfolgend geladenen Erweiterungen, die Knoten definieren, die auf derselben Palette oder Unterpalette enthalten sein sollen, der Abschnitt ‘Benutzeroberfläche (Paletten)’ entfallen — in diesem Fall ist es lediglich erforderlich, dass für das Element Node ein Attribut customPalette angegeben ist, das die Palette referenziert. Erweiterungen werden in alphabetischer Reihenfolge des Werts für *providerTag.id* geladen, wobei dies die Werte der Attribute providerTag und id des Elements ExtensionDetails für die Erweiterung sind (siehe Abschnitt ‘Erweiterungsdetails’ auf S. 42). So wird die Erweiterung myco.abc z. B. vor der Erweiterung myco.def geladen.

Anmerkung: Der Abschnitt ‘Benutzeroberfläche (Paletten) unterscheidet sich vom Hauptabschnitt ‘Benutzeroberfläche’, der als Teil einer einzelnen Objektdefinition angegeben wird und der beschrieben ist unter Kapitel 6, *Erstellen von Benutzeroberflächen* auf S. 131.

Format

Das Format des Abschnitts ‘Benutzeroberfläche (Paletten)’ sieht wie folgt aus:

```
<UserInterface>
 <Palettes>
 <Palette id="name" systemPalette="palette_name" customPalette="palette_name"
        relativePosition="Lage" relativeTo="Palette" label="display_label"
        labelKey="label_key" description="description" descriptionKey="description_key"
        imagePath="image_path" />
      <Palette ... />
   ...
  </Palettes>
</UserInterface>
```

Tabelle 4-1
Palettenattribute

Attribut	Beschreibung
id	(erforderlich) Ein eindeutige Kennzeichen für die Palette oder Unterpalette, die definiert wird.
systemPalette	Wird nur verwendet, wenn eine Unterpalette zu einer Systempalette hinzugefügt wird. Gibt die Systempalette an, in der die Unterpalette vorkommen soll: import - Quellen recordOp - Datensatzoperationen fieldOp - Feldoperationen graph - Grafiken modeling - Modellierung (siehe unten) dbModeling - Datenbank-Modellierung output - Ausgabe export - Exportieren
customPalette	Wird nur verwendet, wenn eine Unterpalette zu einer benutzerdefinierten Palette hinzugefügt wird. Gibt die benutzerdefinierte Palette an, in der die Unterpalette vorkommen soll. Der Wert des id-Attributs des Elements Palette, das die benutzerdefinierte Palette definiert.
relativePosition	Wird nur verwendet, wenn eine benutzerdefinierte Palette definiert wird. Legt die Position auf dem Palettenstreifen am unteren Bildschirmrand fest. Mögliche Werte: first last before after Wenn der Wert before (vor) oderafter (nach) ist, wird außerdem das Attribut relativeTo benötigt (siehe unten). Wenn relativePosition nicht angegeben wird, wird die Palette am Ende des Streifens platziert.
relativeTo	Wenn der Wert für relativePositionbefore (vor) oder after (nach) lautet, wird mit relativeTo das Kennzeichen der Palette angegeben, die vor oder nach dieser benutzerdefinierten Palette erscheint. Paletten-IDs werden als Werte des Palettenattributs des Knotenelements aufgeführt (siehe Knoten auf S. 62).
label	(erforderlich) Der Anzeigename für die Palette oder Unterpalette, der auf der Benutzeroberfläche erscheint.
labelKey	Kennzeichnet die Beschriftung für Lokalisierungszwecke.
description	Der Text der QuickInfo, die angezeigt wird, wenn der Kursor über die Registerkarte ‘Palette’ bewegt wird (wird für Unterpaletten nicht verwendet). Dieser Wert wird auch als lange aufrufbare Beschreibung des Steuerelements verwendet. Für weitere Informationen siehe Thema Zugriffsmöglichkeiten in Kapitel 8 auf S. 218.

Attribut	Beschreibung
descriptionKey	Kennzeichnet die Beschreibung für Lokalisierungszwecke.
imagePath	Gibt den Speicherort des Bildes an, das auf der Registerkarte 'Palette' verwendet wird (wird für Unterpaletten nicht verwendet). Der Speicherort wird relativ zu dem Verzeichnis angegeben, in dem die Spezifikationsdatei installiert ist. Wenn dieses Attribut nicht angegeben ist, wird kein Bild verwendet.

Beispiel - Hinzufügen eines Knotens zu einer Systempalette

Angenommen ihr Unternehmen hat einen neuen Algorithmus zum Mining von Audio- und Videodaten entwickelt und Sie möchten den Algorithmus in IBM® SPSS® Modeler integrieren. Sie definieren zuerst einen benutzerdefinierten Knoten zum Lesen von Daten, der die eingegebenen Audio- und Videodateien liest.

Zuerst beschließen Sie, Ihren neuen Datenleseknoten zur Systempalette 'Quellen' hinzuzufügen. Hierzu müssen Sie lediglich die Palette 'Quellen' mithilfe des Attributs palette des Elements Node angeben. Für weitere Informationen siehe Thema Knoten auf S. 62.

Um den Knoten nach dem Knoten 'Datenbank' zur Palette 'Quellen' hinzuzufügen, erstellen Sie folgende Anweisung:

```
<Node id="AVreader" type="dataReader" palette="import" relativePosition="after" relativeTo="database"
   label="AV Reader">
```

Sie erhalten folgendes Ergebnis:

Abbildung 4-4
Neuer Knoten in einer Standardpalette

Beispiel - Hinzufügen einer benutzerdefinierten Palette

Es bietet sich zwar an eine Standard-IBM® SPSS® Modeler-Palette zu verwenden, Sie möchten Ihrem neuen Knoten aber zu mehr Prominenz verhelfen. Sie beschließen für ihn eine benutzerdefinierte Palette zu definieren, die Sie nach der Favoritenpalette aber vor den Quellen platzieren möchten. Sie müssen zuerst einen Abschnitt 'Benutzeroberfläche (Paletten)' hinzufügen, um die benutzerdefinierte Palette wie folgt zu definieren:

```
<UserInterface>
 <Palettes>
 <Palette id="AV_mining" label="AV Mining" relativePosition="before" relativeTo="import"
        description="Audio video mining"  />
   </Palettes>
</UserInterface>
```

Das Attribut relativeTo verwendet die interne ID der Quellenpalette, die import lautet.

Dann verändern Sie die Node-Definition wie folgt:

```
<Node id="AVreader" type="dataReader" customPalette="AV_mining" label="AV Reader">
```

Dadurch wird die AV Mining-Palette zwischen die Favoriten- und die Quellenpalette platziert.

Abbildung 4-5
Neuer Knoten auf neuer Palette

Beispiel - Hinzufügen einer benutzerdefinierten Unterpalette zu einer benutzerdefinierten Palette

Ausgehend vom vorherigen Beispiel, beschließen Sie nun beispielsweise, dass Sie den Datenleseknoten lieber in seine eigene AV Sources-Unterpalette der AV Mining-Palette eintragen möchten. Hierzu müssen Sie zuerst die Unterpalette spezifizieren, indem Sie ein zum Abschnitt ‘Benutzeroberfläche (Paletten)’ ein zweites Palette-Element hinzufügen:

```
<UserInterface>
 <Palettes>
 <Palette id="AV_mining" label="AV Mining" description="Audio video mining" />
      <Palette id="AV_mining.sources" customPalette="AV_mining" label="AV Sources" />
   </Palettes>
<UserInterface>
```

Dann ändern Sie das Node-Element dahingehend, dass es die Unterpaletten-ID referenziert:

```
<Node id="AVreader" type="dataReader" customPalette="AV_mining.sources" label="AV Reader">
```

Wenn der Benutzer jetzt auf die Registerkarte AV-Mining klickt, werden zwei Unterpaletten angezeigt, eine mit der Bezeichnung Alle und eine mit AV-Quellen. Der AV Reader-Knoten wird auf beiden angezeigt:

Abbildung 4-6
Neuer Knoten auf neuer Unterpalette

Wenn Sie einen weiteren neuen Knoten zu einer anderen neuen Unterpalette von AV-Mining hinzufügen, wird der neue Knoten unter Alle und auf der neuen Unterpalette angezeigt, aber nicht auf der Unterpalette AV-Quellen.

Beispiel - Hinzufügen eines Knotens zu einer Systemunterpalette

Um die Audio- und Videoquellendaten zu verarbeiten, definieren Sie jetzt einen Modellierungsknoten. Sie beschließen, ihn zur Standardmodellierungspalette hinzuzufügen, die eine Anzahl von Standardunterpaletten enthält. Sie beschließen, ihn zur Unterpalette 'Klassifizierung' hinzuzufügen und direkt vor dem Knoten für neuronale Netze zu platzieren:

```
<Node id="AVmodeler" type="modelBuilder" palette="modeling.classification" relativePosition="before"
   relativeTo="neuralnet" label="AV Modeler">
```

Die Unterpalette 'Klassifizierung' sieht jetzt folgendermaßen aus:

Abbildung 4-7
Neuer Knoten auf Standardunterpalette

Beachten Sie, dass der Knoten in derselben relativen Position auch auf der Modellierungspalette in der Unterpalette 'Alle' hinzugefügt wird.

Beispiel - Hinzufügen einer benutzerdefinierten Unterpalette zu einer Systempalette

Wenn Sie sich die Anzahl der Modellierungsknoten auf der Unterpalette 'Klassifizierung' noch einmal ansehen, werden Sie feststellen, dass es für Benutzer nicht einfach ist, Ihren neuen Knoten zu erkennen. Um Ihren Knoten sichtbarer zu machen, können Sie Ihre eigene Unterpalette zur Modellierungspalette hinzufügen und den Knoten dort platzieren.

Hierzu müssen Sie zuerst Ihre benutzerdefinierte Unterpalette definieren, indem Sie die Datei um einen Abschnitt 'Benutzeroberfläche (Paletten)' ergänzen:

```
<UserInterface>
 <Palettes>
 <Palette id="modeling.av_modeling" systemPalette="modeling" label="AV Modeling"
        labelKey="av_modeling.LABEL" description="Contains AV mining-related modeling nodes"
        descriptionKey="av_modeling.TOOLTIP"/>
   </Palettes>
</UserInterface>
```

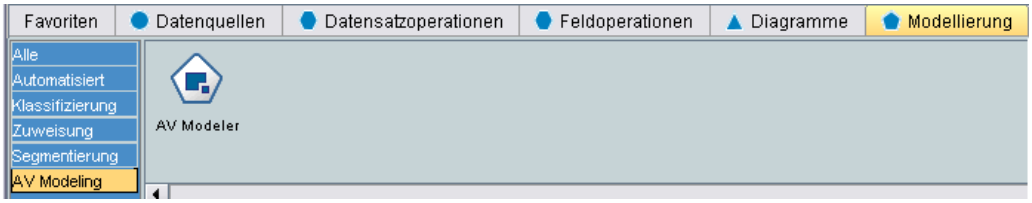
Beachten Sie, dass Sie systemPalette explizit angeben müssen, um die Systempalette festzulegen, die Sie erweitern.

Anschließend spezifizieren Sie im Hauptabschnitt 'Benuteroberfläche' für den Knoten, dass sie auf dieser Unterpalette erscheinen soll:

```
<Node id="my.avmodeler" type="modelBuilder" customPalette="modeling.av_modeling" label="AV Modeler">
```

Benutzerdefinierte Unterpaletten werden immer nach den Systemunterpaletten platziert:

Abbildung 4-8
Neuer Knoten auf benutzerdefinierter Erweiterung einer Standardunterpalette



Anmerkung: Wenn Sie weitere Knoten zur AV-Modellierungsunterpalette hinzufügen möchten, benötigen deren Spezifikationsdateien **keinen** Abschnitt ‘Benutzeroberfläche (Paletten)’, sofern die AV-Modellierungserweiterung zuvor geladen wurde.

Ausblenden oder Löschen einer benutzerdefinierten Palette oder Unterpalette

Wenn eine benutzerdefinierte Palette oder Unterpalette nicht mehr angezeigt werden soll, können Sie diese mit dem IBM® SPSS® Modeler Paletten-Manager entweder ausblenden oder löschen.

Denken Sie daran, dass das Ausblenden für alle SPSS Modeler-Sitzungen gilt, über das entsprechende Kontrollkästchen aber wieder rückgängig gemacht werden kann. Das Löschen kann innerhalb einer Sitzung nicht rückgängig gemacht werden, beim Neustart von SPSS Modeler wird das Symbol allerdings so lange wieder angezeigt, bis Sie es aus der Spezifikationsdatei entfernen oder die gesamte Erweiterung löschen. Für weitere Informationen siehe Thema Deinstallieren von CLEF-Erweiterungen in Kapitel 10 auf S. 256.

So blenden Sie eine Palette aus oder löschen sie:

- Wählen Sie im SPSS Modeler-Hauptmenü Folgendes:
  Werkzeuge > Paletten verwalten
- Wählen Sie im Feld ‘Palettenname’ eine Palette aus und verfahren Sie wie folgt:
  - Um die Palette auszublenden, deaktivieren Sie das entsprechende Kontrollkästchen ‘Angezeigt?’
  - Um die Palette zu löschen, klicken Sie auf die Schaltfläche ‘Auswahl löschen’
- Klicken Sie auf ‘OK’.

So blenden Sie eine Unterpalette aus oder löschen sie:

- Wählen Sie im SPSS Modeler-Hauptmenü Folgendes:
  Werkzeuge > Paletten verwalten
- Wählen Sie im Feld ‘Palettenname’ eine Palette aus:
- Klicken Sie auf die Schaltfläche ‘Unterpaletten’.

- Wählen Sie im Feld ‘Unterpalettenname’ eine Unterpalette aus und verfahren Sie wie folgt:
  - Um die Unterpalette auszublenden, deaktivieren Sie das entsprechende Kontrollkästchen ‘Angezeigt?’
  - Um die Unterpalette zu löschen, klicken Sie auf die Schaltfläche ‘Auswahl löschen’
- Klicken Sie auf ‘OK’.

Abschnitt ‘Objektdefinition’

Elemente stellen die sichtbarsten Bestandteile einer Erweiterung dar. Der Abschnitt ‘Objektdefinition’ bildet den Abschluss der CLEF-Spezifikationsdatei und wird verwendet, um die verschiedenen Objekte in der Erweiterung zu definieren. Folgende Objekttypen können definiert werden:

- Knoten
- Modellausgabeobjekte
- Dokumentausgabeobjekte
- interaktive Ausgabeobjekte

Knoten sind Objekte, die in einem Stream erscheinen. **Modellausgabeobjekte** werden durch Modellierungsknoten generiert und im Hauptfenster unter der Registerkarte ‘Modelle’ im Managerfenster angezeigt. Auf ähnliche Weise werden **Dokumentausgabeobjekte** durch Dokumenterstellungsknoten generiert und in demselben Bereich unter der Registerkarte ‘Ausgaben’ angezeigt. **Interaktive Ausgabeobjekte** werden durch interaktive Modellierungsknoten generiert und im Managerfenster unter der Registerkarte ‘Ausgaben’ angezeigt.

Der Abschnitt ‘Objektdefinition’ besteht aus einer oder mehreren solcher Objektdefinitionen.

Die Elemente, die für die verschiedenen Objekttypen definiert werden können, werden in den folgenden Abschnitten beschrieben. Einige dieser Elemente haben alle Objekttypen gemeinsam, während andere spezifisch für Knoten- oder Modellausgabedefinitionen sind. Objektspezifische Elemente werden im Text als solche gekennzeichnet.

- Objekt-ID
- Modellersteller
- Dokumentersteller
- Eigenschaften
- Container
- Benutzeroberfläche
- Ausführung
- Ausgabedatenmodell
- Konstruktoren

Objekt-ID

Die Objekt-ID gibt den Objekttyp an, der wie folgt aussehen kann:

<Node .../>

<ModelOutput .../>

<DocumentOutput .../>

<InteractiveModelBuilder .../>

Die Objekt-ID bietet außerdem Informationen darüber, wie das Objekt in Skripts zur Verfügung steht. Das Attribut scriptName stellt einen eindeutigen Namen des Objekts dar. Skripts können dieses Attribut verwenden, um ein bestimmtes Objekt zu spezifizieren (z. B. einen Knoten in einem Stream oder eine Ausgabe auf der Registerkarte ‘Ausgaben’).

Knoten

Eine Knotendefinition beschreibt ein Objekt, das in einem Stream vorkommen kann.

Format

```
<Node id="identifier" type="node_type" palette="Palette" customPalette="custom_palette"
        relativePosition="Lage" relativeTo="Knoten" label="display_label" labelKey="label_key"
   scriptName="script_name" helpLink="topic_id" description="description"
      descriptionKey="description_key">
   <ModelBuilder ... >
    ...
   </ModelBuilder>
   <DocumentBuilder ... >
    ...
   </DocumentBuilder>
   <ModelProvider ... />
   <Properties>
    ...
   </Properties>
   <Containers>
    ...
   </Containers>
   <UserInterface>
    ...
   </UserInterface>
   <Execution>
    ...
   </Execution>
 <OutputDataModel ...>
    ...
   </OutputDataModel>
   <Constructors>
    ...
```

```
  </Constructors>
</Node>
```

Die in der Knotendefinition zulässigen Elemente werden in den Abschnitten ab Eigenschaften auf S. 66 beschrieben.

Tabelle 4-2
Knotenattribute

Attribut	Beschreibung
id	(erforderlich) Ein Kennzeichen für den Knoten als Textzeichenfolge.
type	(erforderlich) Der Knotentyp: dataReader - Knoten, der Daten liest (z. B. Quellpalettenknoten) dataWriter - Knoten, der Daten schreibt (z. B. Exportpalettenknoten) dataTransformer - Knoten, der Daten umwandelt (z. B. Datensatz-/Feldoperationsknoten) modelBuilder - Modellierungsknoten (z. B. Modellierungspalettenknoten) documentBuilder - Knoten, der ein Diagramm oder einen Bericht erstellt modelApplier - Knoten, der ein generiertes Modell enthält Der Knotentyp legt die Form des Knotensymbols in der Palette und im Zeichenbereich fest. Für weitere Informationen siehe Thema Überblick über die Knoten in Kapitel 2 auf S. 10. Wenn der Knotentyp modelBuilder lautet, muss die Knotendefinition ein ModelBuilder-Element enthalten - siehe Modellerstellung (ModelBuilder) auf S. 65. Wenn der Knotentyp documentBuilder lautet, muss die Knotendefinition ein DocumentBuilder-Element enthalten - siehe Dokumentersteller (DocumentBuilder) auf S. 65.
palette	Die ID einer der standardmäßigen IBM® SPSS® Modeler-Paletten oder Unterpaletten, auf denen der Konten erscheint: import - Quellen recordOp - Datensatzoperationen fieldOp - Feldoperationen graph - Grafiken modeling - Modellierung (siehe unten) dbModeling - Datenbank-Modellierung output - Ausgabe export - Exportieren Die Modellierungspalette besitzt eine Anzahl von Standardunterpaletten: modeling.classification - Klassifikation modeling.association - Assoziation modeling.segmentation - Segmentierung modeling.auto - Automatisiert Falls das Attribut palette fehlt, wird der Knoten der Palette “Feldoperationen” hinzugefügt. Anmerkung: Das Attribut palette wird nur für Modellierungsknoten verwendet.

Attribut	Beschreibung
customPalette	Die ID einer benutzerdefinierten Palette oder Unterpalette, auf der der Knoten erscheint. Dies ist der Wert des id-Attributs eines Palette-Elements, der im Abschnitt 'Benutzeroberfläche (Paletten)' der Datei spezifiziert ist. Für weitere Informationen siehe Thema Abschnitt 'Benutzeroberfläche (Paletten)' auf S. 55.
relativePosition	Gibt die Position des Knotens innerhalb der Palette an. Mögliche Werte: first last before after Wenn der Wert before (vor) oderafter (nach) ist, wird außerdem das Attribut relativeTo benötigt (siehe unten). Wenn relativePosition nicht angegeben wird, wird der Knoten an das Ende der Palette platziert.
relativeTo	Wenn der Wert für relativePositionbefore(vor) oder after (nach) lautet, wird mit relativeTo der Knoten in der Palette angegeben, der vor oder nach diesem Knoten erscheint. Der Wert von relativeTo ist der Skriptname des Knotens. Für einen standardmäßigen SPSS Modeler-Knoten finden Sie den Skriptnamen im Abschnitt "Eigenschaftsverweis" im*SPSS ModelerHandbuch für die Skripterstellung und Automatisierung*, allerdings ohne das Suffix ...node (für den Datenbankknoten verwenden Sie z. B. database und nicht databasenode). Für einen CLEF-Knoten ist dies der Wert des Attributs scriptName des Knotens.
label	(erforderlich) Der Anzeigename für den Knoten, der in der Palette, dem Zeichenbereich und den Dialogfeldern angezeigt wird.
labelKey	Kennzeichnet die Beschriftung für Lokalisierungszwecke.
scriptName	Wird zur eindeutigen Kennzeichnung des Knotens verwendet, wenn ein Skript auf diesen verweist. Für weitere Informationen siehe Thema Verwenden von CLEF-Knoten in Skripts auf S. 96.
helpLink	Eine optionale ID für ein Hilfethema, das angezeigt wird, wenn der Benutzer das Hilfesystem aufruft, sofern vorhanden. Das Format der ID hängt vom Hilfesystemtyp ab (siehe Kapitel 7, *Hinzufügen eines Hilfesystems* auf S. 208): HTML-Hilfe - URL des Hilfethemas JavaHelp - ID des Themas
description	Eine Textbeschreibung des Knotens.
descriptionKey	Kennzeichnet die Beschreibung für Lokalisierungszwecke.

Die in der Knotendefinition zulässigen Elemente werden in den Abschnitten ab Modellerstellung (ModelBuilder) auf S. 65 beschrieben.

Beispiel

Ein Beispiel für eine Knotendefinition finden Sie unter Beispiel für eine Spezifizierungsdatei auf S. 40.

Modellausgabe

Eine Modellausgabedefinition beschreibt ein generiertes Modell — ein Objekt, das nach der Ausführung eines Streams im Managerfenster unter der Registerkarte ‘Modelle’ erscheint.

Alle Details zur Kodierung dieses Teils der Datei finden Sie unter Modellausgabe auf S. 110.

Dokumentausgabe

Eine Dokumentausgabedefinition beschreibt ein Objekt, wie eine generierte Tabelle oder ein generiertes Diagramm, das nach der Ausführung eines Streams im Managerfenster unter der Registerkarte ‘Ausgaben’ erscheint.

Alle Details zur Kodierung dieses Teils der Datei finden Sie unter Dokumentausgabe auf S. 125.

Interaktive Modellerstellung

Alle Details zur Kodierung dieses Teils der Datei finden Sie unter Erstellen interaktiver Modelle auf S. 112.

Modellerstellung (ModelBuilder)

Dieses Element wird nur in Knotenelementdefinitionen verwendet.

Alle Details zur Kodierung dieses Teils der Datei finden Sie unter Kapitel 5.*Erstellen von Modellen und Dokumenten* auf S. 99.

Dokumentersteller (DocumentBuilder)

Dieses Element wird nur in Knotenelementdefinitionen verwendet.

Alle Details zur Kodierung dieses Teils der Datei finden Sie unter Kapitel 5.*Erstellen von Modellen und Dokumenten* auf S. 99.

Modell-Provider (ModelProvider)

Dieses Element wird nur in Knotenelementdefinitionen verwendet.

Wenn Sie ein Modellausgabeobjekt und einen Modellanwendungsknoten definieren, können Sie mit dem Element ModelProvider den Container angeben, in den das Element aufgenommen wird. Sie können außerdem festlegen, ob das Modell im PMML-Format gespeichert wird. PMML-Modelle können entweder über einen benutzerdefinierten Viewer oder mit dem standardmäßigen IBM® SPSS® Modeler-Modellausgabe-Viewer angezeigt werden, der über das Element ModelViewerPanel bereitgestellt wird. Für weitere Informationen siehe Thema Modell-Viewer-Fenster in Kapitel 6 auf S. 152.

Format

<ModelProvider container="*container_name*" isPMML="*true_false*" />

Dabei gilt:

container ist der Name des Containers, der das Modell enthält.

isPMML gibt an, ob das Modell im PMML-Format gespeichert wird.

Beispiel

```
<ModelProvider container="model" isPMML="true" />
```

Ein Beispiel für die Verwendung von ModelProvider im Kontext eines Modellanwendungsknotens finden Sie unter Modell-Viewer-Fenster auf S. 152.

Eigenschaften

Eine Eigenschaftsdefinition besteht aus einem Satz von Namens- und Wertepaaren. Einzelne Eigenschaftsdefinitionen, von denen es eine Vielzahl geben kann, werden in einem einzigen Abschnitt ‘Eigenschaften’ aufgeführt.

Anmerkung: Wenn eine Eigenschaft im Abschnitt ‘Eigenschaften’ definiert ist, ist es nicht erforderlich, sie für ein einzelnes Eigenschaftssteuerelement zu definieren, da die Definitionen des Abschnitts ‘Eigenschaften’ vorrangig sind. Aus diesem Grund empfiehlt es sich, Eigenschaften im Abschnitt ‘Eigenschaften’ zu definieren.

Die einzige Ausnahme von dieser Regel betrifft das Attribut label. Wenn das Attribut label für eine Eigenschaftssteuerelement definiert wird, hat *jede* innerhalb der Deklaration vorhandene Eigenschaftsdefinition (nicht nur die Definition für label) Vorrang vor der entsprechenden Definition im Abschnitt ‘Eigenschaften’. Beachten Sie, dass diese Ausnahme nur für Eigenschaftssteuerelemente gilt und nicht für andere Steuerelementtypen wie Menüs, Menüelemente und Symbolleistenelemente. Für diese muss explizit eine Beschriftung definiert sein, entweder direkt (Menüs) oder indirekt über ein Action-Element (Menüelemente und Symbolleistenelemente).

Format

```
<Properties>
 <Property name="name" scriptName="script_name" valueType="value_type" isList="true_false"
       defaultValue="default_value" label="display_label" labelKey="label_key" description="description"
       descriptionKey="description_key" />
       <Enumeration ... />
       <Structure ... />
       <DefaultValue ... />
    ...
</Properties>
```

Die untergeordneten Elemente Enumeration, Structure und DefaultValue werden in speziellen Fällen verwendet. Für weitere Informationen siehe Thema Werttypen auf S. 76.

Das Attribute des Property-Elements sind im Folgenden aufgeführt:

Tabelle 4-3
Eigenschaftsattribute

Attribut	Beschreibung
name	(erforderlich) Ein eindeutiger Name für die Eigenschaft.
scriptName	Der Name, über den die Eigenschaft in einem Skript referenziert wird. Für weitere Informationen siehe Thema Verwenden von CLEF-Knoten in Skripts auf S. 96.
valueType	Legt den Typ des Werts fest, den die Eigenschaft annehmen kann: string encryptedString fieldName integer double boolean date enum structure databaseConnection Für weitere Informationen siehe Thema Werttypen auf S. 76.
isList	Spezifiziert, ob es sich bei der Eigenschaft um eine Liste von Werten des spezifizierten Werttyps (true) handelt oder um einen einzelnen Wert (false).
defaultValue	Der Standardwert für die Eigenschaft. Dieser kann als einfaches Wertattribut ausgedrückt werden oder als zusammengesetztes Element und muss mit den angegebenen gültigen Werten übereinstimmen.
label	Der Anzeigename für den Eigenschaftswert, der auf der Benutzeroberfläche angezeigt wird.
labelKey	Kennzeichnet die Beschriftung für Lokalisierungszwecke.
description	Eine Beschreibung der Eigenschaft.
descriptionKey	Kennzeichnet die Beschreibung für Lokalisierungszwecke.

Eigenschaften können optional deklarieren, wie gültige Bereiche ermittelt werden:

- Für numerische Werte sind dies der Minimal- und/oder Maximalwert.
- Für Zeichenketten ist dies in der Regel eine Feldauswahl (z. B. alle Felder, alle numerischen Felder, alle diskreten Felder etc.), es kann aber auch eine Dateiauswahl sein.
- Für Aufzählungen ist dies der Satz der gültigen Werte.

Für verschlüsselte Eigenschaften muss außerdem deklariert werden, wie gültige Schlüssel ermittelt werden. Beachten Sie, dass der Schlüsseltyp einer verschlüsselten Eigenschaft entweder eine Zeichenkette oder eine Aufzählung sein muss. Für weitere Informationen siehe Thema Eigenschaftstypen auf S. 47.

Der der Eigenschaft zugeordnete optionale Standardwert wird evaluiert, wenn das zugeordnete Objekt erstellt wird. Beispiel: Standardwerte für Knoteneigenschaften werden jedes Mal evaluiert, wenn eine neue Instanz des Knotens erstellt wird; Ausführungseigenschaften werden jedes Mal

evaluiert, wenn der Knoten ausgeführt wird. Die Evaluierung erfolgt in der Reihenfolge, in der die Eigenschaften deklariert sind.

Beachten Sie, dass eine Eigenschaftsdefinition einen Eigenschaftstyp referenzieren kann, der im Abschintt ‘Gemeinsame Objekte’ deklariert ist.

Container (Containers)

Ein Container ist ein Platzhalter für ein Ausgabeobjekt, dessen Generierung im Abschnitt ‘Konstruktoren’ definiert ist.

Format

```
<Containers>
 <Container name="container_name" />
    ...
</Containers>
```

Dabei gilt:

name entspricht dem Wert des Zielattributs eines CreateModel- oder CreateDocument-Elements (siehe Verwenden von Konstruktoren auf S. 126) und ordnet den Container indirekt einem der im Abschnitt ‘Gemeinsame Objekte’ deklarierten Containertypen zu.

Beispiel

Zuerst werden die Containertypen im Abschnitt ‘Gemeinsame Objekte’ deklariert. Es gibt einen Containertyp für Modelle, im Textformat, und zwei Containertypen für Dokumentausgabeobjekte, einer im standardmäßigen Textformat für HTML-Ausgaben und einer im Binärformat für komprimierte Zip-Ausgaben.

```
<CommonObjects>
 <ContainerTypes>
 <ModelType id="my_model" format="utf8" />
 <DocumentType id="html_output" />
       <DocumentType id="zip_outputType" format="binary" />
   </ContainerTypes>
</CommonObjects>
```

Im Abschnitt ‘Ausführung’ der Knotendefinition werden die Ausgabedateien als Containerdateien definiert, mit Containertypen, die den im Abschnitt ‘Gemeinsame Objekte’ angegebenen IDs entsprechen:

```
<Node id="mynode" ... >
 ...
    <Execution>
     ...
        <OutputFiles>
 <ContainerFile id="pmml" path="${tempfile}.pmml" containerType="my_model" />
           <ContainerFile id="htmloutput" path="${tempfile}.html" containerType="html_output" />
```

```
        <ContainerFile id="zipoutput" path="${tempfile}.zip" containerType="zip_outputType" />
      </OutputFiles>
```

Anschließend werden im Abschnitt ‘Konstruktoren’ die Ausgabeobjekte definiert, die bei der Ausführung des Knotens generiert werden. Hier haben die Elemente CreateModel und CreateDocument ein sourceFile-Attribut, das einer Containerdatei entspricht, die zuvor im Abschnitt ‘Ausgabedateien’ festgelegt wurde:

```
      <Constructors>
 <CreateModelOutput type="myoutput">
 <CreateModel target="Modell" sourceFile="pmml" />
            <CreateDocument target="advanced_output" sourceFile="htmloutput" />
            <CreateDocument target="zip_output" sourceFile="zipoutput" />
         </CreateModelOutput>
      </Constructors>
   </Execution>
</Node>
```

Zuletzt erfolgt im Abschnitt ‘Modellausgabe’ die Zuordnung eines Containers zu einem Modellausgabe- oder einem Dokumentausgabeobjekt. Im Element Container entspricht das Attribut name dem Attribut target in den sobeben angegebenen CreateModel- und CreateDocument-Elementen:

```
<ModelOutput id="myoutput" label="My Model">
 <Containers>
 <Container name="Modell" />
      <Container name="advanced_output" />
      <Container name="zip_output" />
   </Containers>
    ...
</ModelOutput>
```

Benutzeroberfläche

Die Spezifikationsdatei unterstützt eine Palette von Benutzeroberflächenkomponenten, die die Anzeige von Objekten ermöglichen, sowie Steuerelemente und Eigenschaften, die geändert werden sollen. Es gibt Funktionen für die Angabe des Layouts und die Anpassung des Verhaltens der Komponente sowie zur Aktivierung oder Anzeige der Komponente, wenn andere Steuerelemente geändert werden.

Der Abschnitt ‘Benutzeroberfläche’ spezifiziert die sichtbare Erscheinung eines Objekts. Die Spezifizierung kann verwendet werden, um eine grundlegende Komponente der Benutzeroberfläche anzupassen, wie das Eigenschaftsdialogfeld eines Knotens oder ein Ausgabefenster.

Der Abschnitt ‘Benutzeroberfläche’ ist als Teil der Spezifizierung des Node-Elements erforderlich.

Alle Details zur Kodierung dieses Teils der Datei finden Sie unter Kapitel 6.*Erstellen von Benutzeroberflächen* auf S. 131.

Ausführung

Dieses Element wird nur in Knotenelementdefinitionen verwendet.

Der Abschnitt ‘Ausführung’ definiert Eigenschaften und Dateien, die verwendet werden, wenn ein Knoten ausgeführt wird.

Format

```
<Execution>
   <Properties>
    ...
   </Properties>
 <InputFiles>
 <ContainerFile ... />
       ...
   </InputFiles>
   <OutputFiles>
      <ContainerFile ... />
       ...
   </OutputFiles>
   <Module ... >
      <StatusCodes ... />
   </Module>
 <Constructors ... />
</Execution>
```

Der Abschnitt ‘Ausführung’ enthält die Definition eines Satzes von Eigenschaften, die jedes Mal neu erstellt werden, wenn der Knoten ausgeführt wird, und nur während der Ausführung des Knotens verfügbar sind.

Die Ausführungsinformationen können auch den Satz der Eingabedateien definieren, die vor der Ausführung des Knotens generiert werden, sowie alle während der Ausführung generierten Ausgabedateien.

Es kann eine beliebige Anzahl von Ein- und Ausgabedateien angegeben werden. Jede Eingabedatei wird einem durch den Knoten definierten Container zugeordnet. Jede Ausgabedatei wird in der Regel verwendet, um Container für generierte Objekte zu erstellen. Das Format einer Eingabe- oder Ausgabedatei wird durch die Deklaration des Containers im Abschnitt ‘Allgemeine Objekte’ festgelegt.

Beispiel

Ein Beispiel für den Abschnitt ‘Ausführung’ finden Sie unter Beispiel für eine Spezifizierungsdatei auf S. 40.

Properties (Runtime)

Dieser Abschnitt definiert die Laufzeiteigenschaften, die nur dann verfügbar sind, wenn der Knoten ausgeführt wird.

Format

Das Format ist ähnlich wie das des Abschnitts Properties im Hauptteil der Elementdefinition. Für weitere Informationen siehe Thema Eigenschaften auf S. 66.

Während der Ausführung eines Modellierungs- oder Dokumenterstellungsknotens wird eine **temporäre Serverdatei** erstellt, in der das Modellausgabe- oder Dokumentausgabeobjekt gespeichert wird. Der Server greift auf diese Datei zu und bringt das Objekt auf den Client, wo es in einem Container eingeschlossen wird. Sie müssen diese Datei hier angeben.

Beispiel

Dieses Beispiel zeigt, wie die temporäre Serverdatei spezifiziert wird.

```
<Properties>
   <Property name="tempfile" valueType="string">
      <DefaultValue>
         <ServerTempFile basename="datatmp"/>
      </DefaultValue>
   </Property>
</Properties>
```

Eingabedateien (InputFiles)

Dieser Abschnitt definiert den Satz der Eingabedateien, der vor der Ausführung des Knotens erzeugt wird. Eingabedateien sind in diesem Kontext Dateien, die als Eingabe für die Ausführung eines Knotens auf dem Server dienen. Beispiel: Ein Modellanwendungsknoten hat einen Modellcontainer, der an die angegebene Eingabedatei für die Ausführung des Knotens übertragen wird.

Format

```
<InputFiles>
 <ContainerFile id="identifier" path="path" container="Container">
    ...
</InputFiles>
```

Im Element ContainerFile für eine Eingabedatei gibt es die in der folgenden Tabelle aufgeführten Attribute.

Tabelle 4-4
Containerdateiattribute - Eingabedateien

Attribut	Beschreibung
id	Eine eindeutige ID für die Containerdatei.
path	Der Speicherort auf dem Server, an dem die Eingabedatei erzeugt werden soll (z. B. der Speicherort einer temporären Serverdatei - siehe Properties (Runtime) auf S. 70).
container	Die ID des Containers, der das Objekt enthält, das als Eingabe an den Server gesendet werden soll.

Beispiel

```
<InputFiles>
   <ContainerFile id="pmml" path="${tempfile}.pmml" container="model"/>
</InputFiles>
```

Ausgabedateien

In diesem Abschnitt werden die Ausgabedateien festgelegt, die während des Ausführens des Knotens auf dem Server erstellt werden. Ausgabedateien (z. B. die Ergebnisse der Ausführung des Modellierungs- oder Dokumenterstellungsknotens) werden nach der Ausführung zurück auf den Client übertragen.

Format

```
<OutputFiles>
 <ContainerFile id="identifier" path="path" containerType="Container">
    ...
</OutputFiles>
```

Im Element ContainerFile gibt es die in der folgenden Tabelle aufgeführten Attribute.

Tabelle 4-5
Containerdateiattribute - Ausgabedateien

Attribut	Beschreibung
id	Eine eindeutige ID für die Containerdatei.
path	Der Speicherort des Objekts, das auf den Client übertragen wird, auf dem Server (z. B. der Speicherort einer temporären Serverdatei - siehe Properties (Runtime) auf S. 70).
containerType	Die ID des Containertyps für das Objekt (d. h. die ID des Modelltyps oder Dokumenttyps), über die die Übertragung des Objekts im richtigen Format aktiviert wird. Für weitere Informationen siehe Thema Containertypen auf S. 49..

Beispiel

```
<OutputFiles>
   <ContainerFile id="pmml" path="${tempfile}.pmml" containerType="mynode_model" />
   <ContainerFile id="htmloutput" path="${tempfile}.html" containerType="html_output" />
   <ContainerFile id="zipoutput" path="${tempfile}.zip" containerType="zip_outputType" />
</OutputFiles>
```

Module

In diesem Abschnitt wird festgelegt, dass während der Ausführung eine serverseitige freigegebene Bibliothek verwendet wird (z. B. eine DLL, die in den Hauptspeicher geladen wird).

Format

```
<Module libraryId="shared_library_identifier" name="node_name">
   <StatusCodes ... />
```

```
</Module>
```

Dabei gilt:

libraryId ist die ID einer Bibliothek, die im Element ‘Shared Library’ im Abschnitt ‘Ressourcen’ deklariert wird. Für weitere Informationen siehe Thema Freigegebene Bibliotheken auf S. 45.

name wird verwendet, wenn die Bibliothek durch mehrere Knoten verwendet wird. Hierüber wird der auszuführende Knoten identifiziert. Wenn die Bibliothek nur von einem Knoten verwendet wird, muss der Name nicht angegeben werden.

Beispiel

```
<Module libraryId="mynode1" name="mynode">
   <StatusCodes>
      <StatusCode code="0" status="error" message="An exception occurred" />
      <StatusCode code="1" status="error" message="Error reading input data" />
       ...
   </StatusCodes>
</Module>
```

Statuscodes (StatusCodes)

Die meisten Programme führen eine Art Fehlerprüfung durch und zeigen dem Benutzer alle erforderlichen Meldungen an. In der Regel werden dabei Ganzzahlen zurück gegeben, die den erfolgreichen Abschluss oder einen anderen Status anzeigen. Die serverseitige API kann nach der Ausführung eines Streams, der den Knoten enthält, einen Stauscode zurück geben. Für weitere Informationen siehe Thema Statusdetaildokument (StatusDetail-Dokument) in Kapitel 9 auf S. 243.

Im Abschnitt ‘Statuscode’ können Sie einem bestimmten Statuscode eine Meldung zuordnen, die für Benutzer angezeigt wird.

Format

```
<StatusCodes>
 <StatusCode code="codenum" status="status" message="message_text"
    messageKey="message_key" />
   ...
</StatusCodes>
```

Die folgende Tabelle enthält die Statuscodeattribute.

Tabelle 4-6
Statuscodeattribute

Attribut	Beschreibung
code	Der Statuscode (ein ganzzahliger Wert), dem die Meldung zugeordnet wird.
status	Die Statusklassifizierung: success - Knoten wurde erfolgreich ausgeführt warning - Knoten wurde mit Warnungen ausgeführt error - Knoten wurde nicht erfolgreich ausgeführt

Attribut	Beschreibung
message	Die Meldung, die angezeigt werden soll, wenn der entsprechende Statuscode zurückgegeben wird.
messageKey	Kennzeichnet die Meldung für Lokalisierungszwecke.

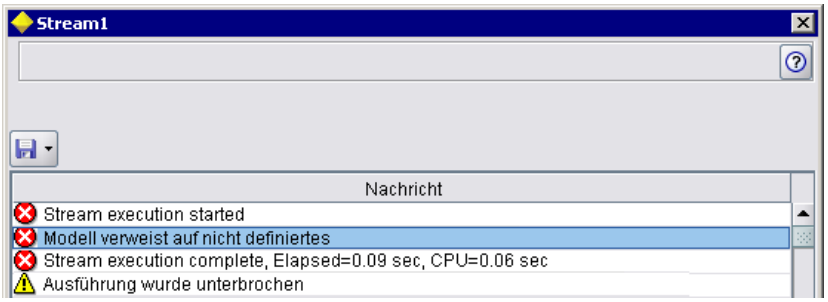
Beispiel

In diesem Beispiel ist der Text der Fehlermeldung im Element StatusCode enthalten:

```
<StatusCodes>
   <StatusCode code="0" status="error" message="Cannot initialise a peer" />
   <StatusCode code="1" status="error" message="Error reading input data" />
   <StatusCode code="2" status="error" message="Internal Error" />
   <StatusCode code="3" status="error" message="Input Field Does Not Exist" />
</StatusCodes>
```

Bei der Ausführung erhält der Benutzer folgende Anzeige, wenn die serverseitige API den Statuscode ‘3’ zurückgibt:

Abbildung 4-9
Fehlermeldungsanzeige



Im nächsten Beispiel werden die Fehlermeldungstexte durch ein messageKey-Attribut referenziert:

```
<StatusCodes>
   <StatusCode code="0" status="error" messageKey="initErrMsg.LABEL"/>
   <StatusCode code="1" status="error" messageKey="inputErrMsg.LABEL"/>
   <StatusCode code="2" status="error" messageKey="internalErrMsg.LABEL"/>
   <StatusCode code="3" status="error" messageKey="invalidMetadataErrMsg.LABEL"/>
    ...
</StatusCodes>
```

Eine Eigenschaftsdatei (z. B. messages.properties), die sich in demselben Ordner befindet wie die Spezifikationsdatei, enthält den Meldungstext zusammen mit anderen Anzeigetexten:

```
 ...
initErrMsg.LABEL=Initialisation failed.
inputErrMsg.LABEL=Error when reading input data.
internalErrMsg.LABEL=Internal error.
invalidMetadataErrMsg.LABEL=Metadata (on input/output fields) not valid.
 ...
```

Dieses Verfahren ist nützlich, wenn der Anzeigetext z. B. für ausländische Märkte lokalisiert werden muss, da sich der gesamte zu übersetzende Text in einer einzigen Datei befindet. Für weitere Informationen siehe Thema Lokalisierung in Kapitel 8 auf S. 211.

Ausgabedatenmodell (OutputDataModel)

Dieses Element wird nur in Knotenelementdefinitionen verwendet.

Im Abschnitt ‘Ausgabedatenmodell’ wird festgelegt, wie sich verschiedene Eigenschaften auf das Datenmodell auswirken.

Das Ausgabedatenmodell kann auf drei verschiedenen Weisen bestimmt werden:

- Mithilfe der Definitionsfunktionen für Feld-Sets in der Spezifikationsdatei. Für weitere Informationen siehe Thema Feld-Sets auf S. 88.
- Mithilfe einer clientseitigen Java-Klasse, die eine Datenmodellproviderschnittstelle implementiert, die einen Satz Eigenschaften und das Eingabedatenmodell empfängt und eine Datenmodellinstanz zurück gibt.
- Mithilfe einer serverseitigen freigegebenen Bibliothekskomponente, die einen Satz Eigenschaften und ein Eingabedatenmodell empfängt und ein Metadatenmodell zurück gibt.

Im Abschnitt ‘Ausgabedatenmodell’ wird definiert wie sich die Eigenschaften eines Knotens auf die durch den Knoten fließenden Felder auswirken. Das Ausgabedatenmodell bietet folgende Optionen:

- Das Eingabedatenmodell unverändert lassen
- Das Eingabedatenmodell ändern
- Das Eingabedatenmodell durch ein anderes Eingabedatenmodell ersetzen

Beispiel: Ein Sortierknoten wirkt sich nicht auf die Eigenschaften als solche aus, er sortiert sie lediglich. Ein Ableitungsknoten ändert das Datenmodell, indem er ein neues Feld hinzufügt. Ein Aggregatknoten ersetzt das Datenmodell vollständig.

In den Fällen, in denen das Eingabedatenmodell geändert wird, können über die Definition neue Felder hinzugefügt oder vorhandene Felder geändert oder entfernt werden. Wenn das Datenmodell ersetzt wird, können nur neue Felder hinzugefügt werden. Die Spezifikationsdatei unterstützt diese Basisoperationen (auch die Möglichkeit, neue Felder zu erstellen, deren Typen auf Eingabefeldern basieren) sowie die Möglichkeit, das Eingabefeld-Set oder einen Schlüssel oder eine Listeneigenschaft zu durchlaufen, die eine Feldgruppe im Eingabefeld-Set darstellen.

Format

Das allgemeine Format des Abschnitts ‘Ausgabedatenmodell’ sieht wir folgt aus, spezifische Formate für diese Fälle finden Sie aber in den Abschnitten Steuerelement für die Auswahl mehrerer Felder auf S. 171 und Steuerelement für die Auswahl eines einzelnen Felds auf S. 180.

```
<OutputDataModel mode="mode" libraryId="container_name">
   -- data model operations --
</OutputDataModel>
```

Die Ausgabdatenmodellattribute sind in der folgenden Tabelle dargestellt.

Tabelle 4-7
Ausgabedatenmodellattribute

Attribut	Beschreibung
mode	Auswirkung auf das Datenmodell: extend - fügt ein neues Feld zum vorhandenen Modell hinzu fixed - unverändert modify - ändert vorhandene Felder (z. B. entfernen oder umbenennen) replace - ersetzt das vorhandene Modell
libraryId	Der Name eines serverseitigen Containers, aus dem das Datenmodell erhalten wird.

Datenmodelloperationen sind die Operationen, die neue Felder hinzufügen oder vorhandene Felder ändern oder entfernen. Für weitere Informationen siehe Thema Datenmodelloperationen auf S. 82.

Beispiel

Ein OutputDataModel-Element ist im Beispiel für eine Spezifikationsdatei enthalten. Für weitere Informationen siehe Thema Beispiel für eine Spezifizierungsdatei auf S. 40.

Konstruktoren

Konstruktoren definieren die Objekte, die als Ergebnis der Ausführung eines Knotens in einem Stream oder beim Generieren eines Objekts zurück in den Stream produziert werden.

Alle Details zur Kodierung dieses Teils der Datei finden Sie in den Abschnitten ab Verwenden von Konstruktoren auf S. 126.

Allgemeine Funktionen

Einige Funktionen können in mehreren Abschnitten der Spezifikationsdatei verwendet werden:

- Wertetypen
- evaluierte Zeichenketten
- Operationen
- Felder und Feldmetadaten
- Feld-Sets
- Rollen
- logische Operatoren
- Bedingungen

Werttypen

Werttypdeklarationen geben den Typ des Werts an, den eine Spalten-, Eigenschafts- oder Eigenschaftstypspezifikation annehmen kann.

Zeichenketten und verschlüsselte Zeichenketten

Das Format valueType="string" gibt an, dass der Wert eine Textzeichenkette ist. Die Deklaration valueType="encryptedString" wird für eine Eigenschaft verwendet, die sich auf ein Feld bezieht, dessen Inhalt bei der Eingabe durch den Benutzer ausgeblendet werden, wie z. B. ein Passwortfeld.

Feldnamen

Wenn ein Wert die Form eines Feldnamens annimmt, verwenden Sie das Format valueType="fieldName".

Mathematische, logische und Datumsausdrücke

Wenn der Wert ein mathematischer (Ganzzahl oder doppelte Genauigkeit), logischer (wahr/falsch) oder ein Datumsausdruck ist, geben Sie für valueType entsprechend integer, double, boolean oder date an.

Aufgezählte Eigenschaften

Aufgezählte Eigenschaften sind im Abschnitt ‘Aufzählung’ enthalten, der direkt auf eine valueType="enum"-Deklaration folgt. Für weitere Informationen siehe Thema Aufgezählte Eigenschaften auf S. 77.

Strukturdeklarationen

Eine valueType="structure"-Deklaration spezifiziert einen zusammengesetzten Wert, der andere benannte Attribute enthält. Attribute sind ähnlich wie Eigenschaften, können jedoch nicht strukturiert oder verschlüsselt werden. Für weitere Informationen siehe Thema Strukturierte Eigenschaften auf S. 78.

- **Verschlüsselter Indikator.** Spezifiziert, ob die Eigenschaft ein einzelner Wert oder eine Hash-Tabelle ist, in der alle Werte den angegebenen Werttyp besitzen.
- **Wert-Set.** Legt fest, wie der Satz der verfügbaren Werte ermittelt wird.
- **Schlüssel-Set.** Für verschlüsselte Eigenschaften. Wird verwendet, um festzulegen wie der Satz der verfügbaren Schlüssel bestimmt wird. Diese Information wird außerdem verwendet, um an der Benutzeroberfläche Hinweise zum Controller-Typ zu geben, der verwendet werden sollte.

Datenbankverbindungen

Dies sind Verbindungszeichenketten mit denen sich Benutzer bei einer Datenbank anmelden können, wie z. B. user1@testdb. Die Anmeldedetails müssen für die Datenbank bereits definiert sein. Für weitere Informationen siehe Thema Steuerelement für die Auswahl der Datenbankverbindung in Kapitel 6 auf S. 168.

Aufgezählte Eigenschaften

Eine aufgezählte Eigenschaft kann einen Wert aus einer vordefinierten Werteliste annehmen.

Format

Das Format für aufgezählte Eigenschaften verwendet einen Abschnitt Enumeration, in dem die Liste der Werte definiert ist:

```
<PropertyTypes>
 <PropertyType id="identifier" valueType="enum">
 <Enumeration>
 <Enum value="value" label="display_label" labelKey="label_key"
   description="description" descriptionKey="description_key" />
      ...
     </Enumeration>
   </PropertyType>
<PropertyTypes>
```

wobei die PropertyType-Attribute wie folgt aussehen:

- id ist ein eindeutiges Kennzeichen für den Eigenschaftstyp.
- valueType gibt an, dass der Eigenschaftstyp aufgezählt ist.

Die Enum-Attribute sehen wie folgt aus:

- value (erforderlich) ist der Eigenschaftswert, der in der Werteliste auftaucht.
- label (erforderlich) ist der Anzeigename für die Eigenschaft, mit dem sie auf der Benutzeroberfläche angezeigt wird.
- labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke.
- description ist eine Beschreibung des aufgezählten Werts.
- descriptionKey kennzeichnet die Beschreibung für Lokalisierungszwecke.

Beispiel

```
<PropertyTypes>
   <PropertyType id="shared_enum1" valueType="enum">
      <Enumeration>
         <Enum value="value1" label="Value 5.1" labelKey="enum5.value1.LABEL" />
         <Enum value="value2" label="Value 5.2" labelKey="enum5.value2.LABEL" />
         <Enum value="value3" label="Value 5.3" labelKey="enum5.value3.LABEL" />
      </Enumeration>
   </PropertyType>
<PropertyTypes>
```

Strukturierte Eigenschaften

Eine strukturierte Eigenschaft wird in einer gitterartigen Struktur verwendet, wie ein Tabellensteuerelement in einem Dialogfeld.

Format

Das Format für strukturierte Eigenschaften verwendet einen Abschnitt Structure, in dem die Struktur definiert wird, und besteht aus einer Reihe von Attribute-Elementen, die wie folgt aussehen:

```
<PropertyTypes>
 <PropertyType id="identifier" valueType="structure" isList="true_false">
       <Structure>
 <Attribute name="column_ID" valueType="value_type" isList="true_false" label="column_label"
         labelKey="label_key" defaultValue="value" description="description"
       descriptionKey="description_key" />
        ...
       </Structure>
   </PropertyType>
</PropertyTypes>
```

wobei die Attribute des Elements PropertyType wie folgt aussehen:

- id ist ein eindeutiges Kennzeichen für den Eigenschaftstyp.
- valueType gibt an, dass der Eigenschaftstyp strukturiert ist.
- isList gibt an, ob es sich bei der Eigenschaft um eine Liste von Werten des spezifizierten Werttyps (true) handelt oder um einen einzelnen Wert (false).

Die Attribute des Attribute-Elements sehen wie folgt aus:

- name (erforderlich) ist die ID der Spalte.
- valueType gibt den Werttyp an, den die Inhalte der Spalte annehmen können:

  string

  encryptedString

  integer

  double

  boolean

  date

  enum
- isList gibt an, ob es sich bei dem Attribut um eine Liste von Werten des spezifizierten Werttyps (true) handelt oder um einen einzelnen Wert (false). Auf diese Weise kann eine Schlüsseleigenschaft mit einem festen Satz an bekannten Attributen (z. B. Boolean-Attribute für verschiedene Aggregationsvorgänge, die an einem bestimmten Feld ausgeführt werden sollen) oder mit einer Werteliste (z. B. Verbinden einer Feldnamenliste mit anderen Feldnamen) verknüpft werden.
- label (erforderlich) ist der Anzeigename für die Spalte, mit dem sie auf der Benutzeroberfläche angezeigt wird.
- labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke.
- defaultValue ist ein Wert, der in der Spalte erschient, wenn sie angezeigt wird.

- description ist eine Beschreibung der Spalte.
- descriptionKey kennzeichnet die Beschreibung für Lokalisierungszwecke.

Beispiel - Tabellensteuerelement

Ein Beispiel zur Verwendung strukturierter Eigenschaften in einem Tabellensteuerelement finden Sie unter Tabellensteuerelement auf S. 184.

Beispiele - Schlüsseleigenschaftstypen

Die ersten dieser Beispiele illustrieren die Verwendung eines Schlüsseleigenschaftstyps, bei dem jeder verknüpfte Wert eine Struktur darstellt, die angibt, welcher Aggregationsvorgang aus einer vorgegebenen Menge an Vorgängen auf ein Feld angewendet wird:

```
<PropertyType id="aggregateOps" isKeyed="true" valueType="structure">
 <Structure>
  <Attribute name="MIN" valueType="boolean" label="Min" />
  <Attribute name="MAX" valueType="boolean" label="Max" defaultValue="true"/>
  <Attribute name="SUM" valueType="boolean" label="Sum" defaultValue="false"/>
  <Attribute name="MEAN" valueType="boolean" label="Mean" defaultValue="false"/>
  <Attribute name="SDEV" valueType="boolean" label="SDev" defaultValue="false"/>
 </Structure>
</PropertyType>
```

Daher könnte eine Eigenschaft, die für die Verwendung des Eigenschaftstyps aggregateOps deklariert wurde, wie folgt lauten:

```
<Property name="aggregationSettings" scriptName="aggregation_settings" type="aggregateOps"/>
```

Hier besteht die Eigenschaft aus mehreren Werten, wobei jeder Wert über einen anderen Schlüssel verfügt. Der Schlüssel name beispielsweise ist der Name eines Felds (MIN, MAX usw.).

Im nächsten Beispiel eines Schlüsseleigenschaftstyps stellt jeder verknüpfte Wert eine Struktur dar, die ein einzelnes Attribut enthält. In diesem Fall ist das Attribut eine Liste von zweistelligen Ausdrücken, die sich als Multiplikatoren auf ein Feld anwenden lassen:

```
<PropertyType id="multiplierOps" isKeyed="true" valueType="structure">
 <Structure>
  <Attribute name="multipliers" valueType="double" isList="true"/>
 </Structure>
</PropertyType>
```

Eine Eigenschaft, die für die Verwendung des Eigenschaftstyps multiplierOps deklariert wurde, könnte Folgendes sein:

```
<Property name="multiplierSettings" scriptName="multiplier_settings" type="multiplierOps"/>
```

Standardwerte

Das Element DefaultValue wird verwendet, um ein temporäres Serververzeichnis, eine temporäre Serverdatei oder beides anzugeben. Diese werden erstellt, um ein Modellausgabe- oder ein Dokumentausgabeobjekt zu speichern.

Format

```
<DefaultValue>
 <ServerTempDir basename="name"/>
   <ServerTempFile basename="name"/>
</DefaultValue>
```

wobei basename (erforderlich) der Name des temporären Verzeichnisses oder der temporären Datei ist.

Beispiel

```
<DefaultValue>
   <ServerTempFile basename="datatmp"/>
</DefaultValue>
```

Evaluierte Zeichenketten

Einige in der Spezifikationsdatei deklarierte Zeichenketten enthalten möglicherweise Referenzen auf Eigenschaftsnamen. Diese Zeichenketten werden als evaluierte Zeichenketten bezeichnet.

Die Syntax für eine Eigenschaftsreferenz lautet:

"${*property_name*}"

Wenn auf eine evaluierte Zeichenkette zugegriffen wird, werden alle Eigenschaftsreferenzen durch den Wert der referenzierten Eigenschaft ersetzt. Wenn die Eigenschaft nicht vorhanden ist, tritt ein Fehler auf. Beispiel: Wenn ein neues Feld hinzugefügt wird, ist in der Knotendefinition möglicherweise ein Feld namens my_new_field vorhanden und im Abschnitt ‘Benutzeroberfläche’ gibt es ein Steuerelement, das dem Benutzer die Möglichkeit gibt, den Wert dieser Eigenschaft zu bearbeiten.

Beispiel

```
<AddField name="${my_new_field}" ... >
```

Vorgänge

Bestimmte Abschnitte der Spezifikationsdatei unterstützen Operationen wie das Hinzufügen von Feldern, das Erstellen von Komponenten und das Initialisieren von Eigenschaften. Zu den Abschnitten, die Operationen unterstützen, gehören:

- Ausgabedatenmodell (Quell- und Prozessknoten)

- Eingabe- und Ausgabedatenmodell (Komponenten)
- Ausgabeobjekterstellung (Modellierungs- und Dokumenterstellungsknoten)
- Modellanwendungserstellung (Modellausgaben)

Operationen werden in folgende Typen eingeteilt:

- Datenmodeloperationen: AddField, ChangeField, RemoveField
- Iteration: ForEach

Datenmodelloperationen

Folgende Operationen können Sie auf einem Datenmodell durchführen:

- Hinzufügen eines neuen Felds zu einem vorhandenen Datenmodell
- Ändern eines vorhandenen Felds in einem Datenmodell
- Entfernen eines Felds aus einem Datenmodell

Feld hinzufügen

Das Element AddField ermöglicht das Hinzufügen eines neuen Felds zu einem vorhandenen Datenmodell.

Format

```
<AddField prefix="prefix" name="name" direction="field_role" directionRef="field_role_ref"
   fieldRef="field_ref" group="group_id" label="label" missingValuesRef="mval_ref"
   storage="storage_type" storageRef="storage_ref" targetField="target_field" type="data_type"
   typeRef="type_ref" role="Rolle" tag="propensity_type" value="value" >
   <Range min="min_value" max="max_value" />
</AddField>
```

Die Attribute für AddField sind im Folgenden aufgeführt.

Tabelle 4-8
AddField-Attribute

Attribut	Beschreibung
prefix	Ein Präfix, das zum Feldnamen hinzugefügt wird, um z. B. ein Modellausgabefeld anzugeben.
name	(erforderlich) Der Name des Felds, das hinzugefügt wird. Dies kann entweder ein hartkodierter String (z. B. field8) oder ein evaluierter String (z. B. ${target}) sein, der auf ein Feld verweist. Für weitere Informationen siehe Thema Evaluierte Zeichenketten auf S. 81.
direction	Die Feldrolle, d. h., ob das Feld eine Eingabe oder ein Ziel ist. in, out, both, partition oder none. Für weitere Informationen siehe Thema Festlegen der Feldrolle in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.
directionRef	Gibt an, dass die Feldrichtung von dem Feld übernommen werden soll, das durch einen evaluierten String (z. B. ${field1}) identifiziert wird und auf ein Feld verweist. Für weitere Informationen siehe Thema Evaluierte Zeichenketten auf S. 81.

Attribut	Beschreibung
fieldRef	Gibt an, dass alle referenzierten Werte (directionRef, missingValuesRef, storageRef und typeRef) von den entsprechenden Werten des Felds übernommen werden sollen, das durch einen evaluierten String (z. B. ${field1}) identifiziert wird, der auf ein Feld verweist. Für weitere Informationen siehe Thema Evaluierte Zeichenketten auf S. 81.
group	Gibt an, dass das Feld einer Feldgruppe angehört. Für weitere Informationen siehe Thema Modellfelder (ModelFields) in Kapitel 5 auf S. 108.
label	Eine Beschriftung, die dem Feld hinzugefügt werden soll.
missingVal-uesRef	Gibt an, dass die Behandlung von fehlenden Werten gemäß der Spezifikation fehlender Werte des Felds übernommen werden soll, das durch einen evaluierten String (z. B. ${field1}) identifiziert wird, der auf ein Feld verweist. Für weitere Informationen siehe Thema Evaluierte Zeichenketten auf S. 81.
storage	Der Datenspeichertyp für den Feldwert - integer, real, string, date, time, timestamp oder unknown.
storageRef	Gibt an, dass der Speichertyp von dem Feld übernommen werden soll, das durch einen evaluierten String (z. B. ${field1}) identifiziert wird, der auf ein Feld verweist. Für weitere Informationen siehe Thema Evaluierte Zeichenketten auf S. 81.
targetField	Gibt für ein Modellausgabefeld das Zielfeld an, aus dem Daten für dieses neue Feld übernommen werden. Dies kann entweder ein hartkodierter String (z. B. field8) oder ein evaluierter String (z. B. ${target}) sein, der auf ein Feld verweist. Für weitere Informationen siehe Thema Evaluierte Zeichenketten auf S. 81.
type	Der Datentyp des Felds - auto, range, discrete, set, orderedSet, flag oder typeless. Für weitere Informationen siehe Thema Messniveaus in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.
typeRef	Gibt an, dass der Datentyp von dem Feld übernommen werden soll, das durch einen evaluierten String (z. B. ${field1}) identifiziert wird, der auf ein Feld verweist. Für weitere Informationen siehe Thema Evaluierte Zeichenketten auf S. 81.
role	Die Art der Daten, die sich im Modellausgabefeld befinden - unknown, predictedValue, predictedDisplayValue, probability, residual, standardError, entityId, entityAffinity, upperConfidenceLimit, lowerConfidenceLimit, propensity, value oder supplementary. Für weitere Informationen siehe Thema Rollen auf S. 89.
tag	Wird nur verwendet, wenn role den Wert propensity aufweist; gibt den Neigungstyp an und ist entweder RAW oder ADJUSTED.
value	Gibt an, dass die Werte, die das neue Feld enthalten soll, von dem Feld übernommen werden sollen, das durch einen evaluierten String (z. B. ${field1}) identifiziert wird, der auf ein Feld verweist.Für weitere Informationen siehe Thema Evaluierte Zeichenketten auf S. 81.

Die Attribute für Range (Bereich) sind im Folgenden aufgeführt.

Tabelle 4-9
Bereichsattribute

Attribut	Beschreibung
min	Der kleinste Wert, den das Feld annehmen kann.
max	Der größte Wert, den das Feld annehmen kann.

Beispiele

Das folgende Beispiel fügt ein Zeichenkettenfeld namens field8 hinzu:

```
<AddField name="field8" storage="string" />
```

Das nächste Beispiel zeigt, wie beim Hinzufügen eines Felds eine Referenz auf einen Eigenschaftsnamen verwendet werden kann. Das Feld wird hier mit einem Namen hinzugefügt, der mit dem Wert der zuvor definierten Eigenschaft prop1 übereinstimmt:

```
<AddField name="${prop1}" ... />
```

Wenn im folgenden Beispiel das Zielfeld den Namen field1 aufweist, erstellt das Modell ein Ausgabefeld namens $S-field1, das den vorhergesagten Wert für field1 aufnimmt:

```
<AddField prefix="$S" name="${target}" role="predictedValue" targetField="${target}"/>
```

Das nächste Beispiel fügt ein Modellausgabefeld hinzu, das einen Wahrscheinlichkeits-Score zwischen 0 und 1 enthält:

```
<AddField prefix="$SC" name="${target}" storage="real" role="probability" targetField="${target}">
   <Range min="0.0" max="1.0"/>
</AddField>
```

Im letzten Beispiel wird für jedes Modellausgabefeld ein Ausgabefeld hinzugefügt, das einen Wahrscheinlichkeits-Score zwischen 0.0 und 1.0 enthält und dessen Wert aus der Variablen fieldValue übernommen wird:

```
<ForEach var="fieldValue" inFieldValues="${field}">
   <AddField prefix="$SP" name="${fieldValue}" storage="real" role="probability" targetField="${field}"
   value="${fieldValue}">
      <Range min="0.0" max="1.0"/>
   </AddField>
</ForEach>
```

Für weitere Informationen siehe Thema Evaluierte Zeichenketten auf S. 81.

Feld ändern

Mit dem Element ChangeField kann ein vorhandenes Feld in einem Datenmodell geändert werden.

Format

```
<ChangeField
name="name" fieldRef="field_reference" direction="field_role" storage="storage_type" type="data_type" >
   <Range min="min_value" max="max_value" />
</ChangeField>
```

Die Attribute für ChangeField sind im Folgenden aufgeführt.

Tabelle 4-10
ChangeField-Attribute

Attribut	Beschreibung
name	(erforderlich) Der Name des Felds, das geändert wird.
fieldRef	Ein Referenzwert für das Feld.

Attribut	Beschreibung
direction	Die Feldrolle, d. h., ob das Feld eine Eingabe oder ein Ziel ist. in, out, both, partition oder none. Für weitere Informationen siehe Thema Festlegen der Feldrolle in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.
storage	Der Datenspeichertyp für den Feldwert - integer, real, string, date, time, timestamp oder unknown.
type	Der Datentyp des Felds - auto, range, discrete, set, orderedSet, flag oder typeless. Für weitere Informationen siehe Thema Messniveaus in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Die Attribute für Range (Bereich) sind im Folgenden aufgeführt.

Tabelle 4-11
Bereichsattribute

Attribut	Beschreibung
min	Der kleinste Wert, den das Feld annehmen kann.
max	Der größte Wert, den das Feld annehmen kann.

Feld entfernen

Mit dem Element 'RemoveField' kann ein Feld aus einem Datenmodell entfernt werden.

Format

```
<RemoveField fieldRef="field_reference" />
```

wobei fieldRef ein Referenzwert für das Feld ist.

Iteration mit dem Element 'ForEach'

An manchen Stellen ist es nützlich, wenn man bestimmte Operationen wiederholt durchführen kann, um alle Elemente eines Werte-Sets zu verarbeiten. Die Spezifikationsdatei unterstützt einen einfachen ForEach-Iterator, der jeden Wert eines angegebenen Sets mit einer temporären Eigenschaft verbindet. Die ForEach-Schleife kann auf eine der folgenden Weisen eingerichtet werden:

- zwischen zwei ganzzahligen Werten mit einer optionalen Schrittweite
- über Werte in einer Listeneigenschaft
- über Schlüssel in einer verschlüsselten Eigenschaft
- über Felder in einer Feldgruppe

Format

```
<ForEach var="field_name" from="integer_exp" to="integer_exp" step="integer_exp"
inFields="Felder" inFieldValues="field_name" inProperty="property_name" >
  -- data model operation --
</ForEach>
```

Dabei gilt:

var (erforderlich) spezifiziert das Feld, das die Werte enthält, auf die die Iteration angewendet wird.

from uand to geben Ganzzahlen an (oder einen Ausdruck, dessen Auswertung Ganzzahlen liefert), die die untere und die obere Iterationsgrenze bestimmen, mit dem optionalen Attribut step, das eine ganzzahlige Schrittweite angibt.

inFields, inFieldValues und inProperty sind Alternativen zum Format from/to/step:

- inFields spezifiziert ein Feld-Set, auf dem die Iteration durchgeführt wird, das eines der folgenden Formate besitzt:

  inputs - die Eingabefelder für den Knoten

  outputs - die Ausgabefelder aus dem Knoten

  modelInput - die in der Modellsignatur spezifizierten Eingabefelder

  modelOutput - die in der Modellsignatur spezifizierten Ausgabefelder
- inFieldValues spezifiziert einen Feldnamen (oder eine Eigenschaft, die für einen Feldnamen steht) und iteriert durch die Werte in den Metadaten für das Feld
- inProperty gibt den Namen einer Eigenschaft an, für die die Iteration durchgeführt wird

Die Datenmodeloperation, die in einem ForEach-Element angegeben werden kann, ist eines der Elemente AddField, ChangeField oder RemoveField. Für weitere Informationen siehe Thema Datenmodelloperationen auf S. 82.ForEach-Elemente können auch geschachtelt werden.

Beispiele

Das folgende Beispiel führt eine Operation zehn Mal durch:

```
<ForEach var="val" from="1" to="10">
 ...
</ForEach>
```

Das folgende Beispiel führt eine Operation so oft durch, wie durch eine ganzzahlige Eigenschaft angegeben:

```
<ForEach var="val" from="1" to="${history_count}">
 ...
</ForEach>
```

Im nächsten Beispiel iteriert die Verarbeitung durch die Werte in den Ausgabefeldern für den Knoten:

```
<ForEach var="field" inFields="outputs">
 ...
</ForEach>
```

Das folgende Beispiel iteriert durch die Werte in den Metadaten für die Felder, die durch ${field} abgegeben werden:

```
<ForEach var="fieldValue" inFieldValues="${field}">
 ...
</ForEach>
```

Das nächste Beispiel iteriert durch die Werte in einer Listeneigenschaft:

```
<ForEach var="val" inProperty="my_list_property">
 ...
</ForEach>
```

Das folgende Beispiel iteriert durch die Schlüsselwerte in einer verschlüsselten Eigenschaft:

```
<ForEach var="key" inProperty="my_keyed_property">
 ...
</ForEach>
```

Felder und Feldmetadaten

Knoten, Modelle und Datenquellen agieren als **Datenmodell-Provider** — sie können Feldmetadaten definieren, auf die von anderen Objekten zugegriffen werden kann.

Datenmodel-Provider haben ein Eingabedatenmodell und ein Ausgabedatenmodell. Ein Ausgabedatenmodell kann abhängig vom Eingabedatenmodell beschrieben werden, indem z. B. ein Eingabedatenmodell um ein Feld erweitert wird oder indem ein vorhandenes Modell geändert wird.

Alle Objekte haben leicht unterschiedliche Anforderungen.

Knoten. Das Eingabedatenmodell kann referenziert werden, ist aber nicht änderbar. Das Ausgabedatenmodell kann auf dem Eingabedatenmodell basieren oder dieses ersetzen. Das Ausgabedatenmodell wird jedes Mal neu berechnet, wenn die Knoteneigenschaften oder das Eingabedatenmodell geändert werden. Das Ausgabedatenmodell eines Modellanwendungsknotens kann auch das Ausgabedatenmodell der Modellkomponente referenzieren.

Modelle. Standardmäßig basieren die Eingabe- und Ausgabedatenmodelle (die Modellsignatur) auf den Eingabe- und Ausgabefeldeinstellungen, die beim Erstellen des Modells verwendet werden. Idealerweise gibt der Modellerstellungsprozess eine Metadatendatei zurück, die die erforderlichen Eingabefelder und die generierten Ausgabefelder definiert. Die Modellsignatur kann nach ihrer Definition nicht mehr geändert werden. Die Eigenschaften in einem Modellanwendungsknoten können jedoch die Datenmodellausgabe des Anwendungsknotens verändern. Dise Eigenschaften können z. B. definieren, ob eine Cluster-ID als Zeichenkette oder als Ganzzahl zurückgegeben wird, oder wie viele Sequenz-IDs generiert werden. Zusätzlich spezifiziert die Modellsignatur in der Regel Ausgaben mit der Feldrolle (direction) "out", während der Knoten diese eher mit der Feldrolle "in" generiert.

Datenquellen. Datenquellen, die in Datenleseknoten verwendet werden, können ein Ausgabedatenmodell angeben. Das Eingabedatenmodell ist immer leer.

Feld-Sets

Ein Feld-Set kann an verschiedenen Stellen verwendet werden, um eine Untermenge von Feldern aus einem Datenmodell-Provider auszuwählen. Der Datenmodell-Provider kann entweder das umschließende Objekt oder ein Container des umschließenden Objekts sein. Im Ausgangszustand kann ein Feldfilter entweder alle verfügbaren Felder einschließen und dann bestimmte Feldtypen ausschließen oder mit einem leeren Feldsatz beginnen, um dann die erforderlichen Felder einzuschließen oder neue Felder hinzuzufügen.

Das folgende Beispiel zeigt, wie ein Erweiterungsknoten das Ausgabedatenmodell spezifizieren kann. Die Schlüsselfelder werden durch eine Eigenschaftsliste namens keys angegeben, auf die ein optionales Datensatzzählerfeld folgt, das generiert werden kann und dessen Name auch durch eine Eigenschaft angegeben wird.

```
<OutputDataModel mode="replace">
   <ForEach var="field" inProperty="keys">
      <AddField name="${field}" fieldRef="${field}"/>
   </ForEach>
   <AddField name="${record_count_name}" storage="integer">
      <Condition property="include_record_count" op="equals" value="true"/>
   </AddField>
</OutputDataModel>
```

Feld-Sets und Modellierung

Das folgende Beispiel zeigt, wie ein Modellanwender die Informationen in der zuvor erstellten Modellkomponente verwenden kann, um seine Ausgabefelder zu erstellen:

```
<OutputDataModel mode="modify">
   <AddField provider="model" dataModel="output">
</OutputDataModel>
```

Sowohl AddField als auch ForEach geben einen Datenmodell-Provider an und spezifizieren, welche Eingabe- oder Ausgabedatenmodelle verwendet werden sollen. Sie bieten einen Mechanismus zur Angabe eines Sets (oder einer Teilmenge) von Feldern vom Datenmodell-Provider. Der Standard-Provider ist this, was für das einschließende Element steht (d. h. nicht das zu erstellende Objekt), wobei standardmäßig das n Eingabefeld-Set verwendet wird. Wenn kein Feld-Set angegeben ist, werden alle verfügbaren Felder verwendet.

Feld-Sets können auf Speicherung, Typ, Feldrolle oder Namen basieren. Wenn sie auf Namen basieren, ist eine Referenz auf eine Listeneigenschaft erforderlich. Das Feld-Set kann entweder voll sein (Standard) oder leer; im ersten Fall können Felder ausgeschlossen werden, im letzten eingeschlossen. Für jeden der einzelnen Filter können mehrere Werte festgelegt werden, die als “intersection”- oder “and”-Operatoren arbeiten, wie z. B.:

```
<FieldSet include="none">
   <Include direction="in" storage="string"/>
</FieldSet>
```

Dieses Beispiel beginnt mit einem leeren Feld-Set (durch include="none" angegeben) und schließt dann Felder ein, die die Feldrolle (direction) "in" aufweisen und als Zeichenkette gespeichert werden.

Weiteres Beispiel:

```
<FieldSet include="all">
   <Exclude type="typeless"/>
</FieldSet>
```

Dieses Beispiel schließt alle verfügbaren Felder ein (festgelegt durch das Attribut include="all", was das Standardverhalten ist) und schließt dann alle Felder mit dem Typ typeless aus. Die schließt auch Felder ein, bei denen direction auf "in" oder "both" gesetzt ist.

Es können auch mehrere Filter angegeben werden, die als “union”- oder “or”-Operatoren arbeiten, wie z. B.:

```
<FieldSet include="all">
   <Exclude type="discrete" storage="real"/>
   <Exclude type="discrete" storage="integer"/>
</FieldSet>
```

Dies schließt Felder aus, die entweder diskrete Zahlen sind, die als reelle gespeichert werden, oder diskrete Zahlen, die als ganze Zahlen gespeichert werden.

Beachten Sie, dass die Reihenfolge der include-Anweisungen beim Einschließen von Feldern in ein anfangs leeres Feld-Set in der Regel keine Auswirkung auf die Reihenfolge hat, in der die Felder eingeschlossen werden. Dies bedeutet, dass Felder von einem Feld-Set-Provider in ihrer natürlichen Reihenfolge in Bezug auf die Bedingung ausgewertet werden, um festzustellen, ob sie in das Feld-Set eingeschlossen werden.

Rollen

Eine Rolle beschreibt die Art der Daten, die im Ausgabefeld eines Datenmodells enthalten sind. Eine Rolle kann durch ein AddField-Element angegeben und durch ein Condition-Element getestet werden.

Im Folgenden sind die möglichen Rollen aufgeführt.

Tabelle 4-12
Modellausgaberollen

Rolle	Bedeutung
unknown	Keine Rolle angegeben (sollte nicht vorkommen).
predictedValue	Dieses Feld enthält den vorhergesagten Wert des Zielfelds.
probability	Die Wahscheinlichkeit oder Konfidenz der Vorhersage.
residual	Der Wert des Residuums.
standardError	Der Standardfehler der Vorhersage.

Rolle	Bedeutung
entityId	Die ID des Elements — stellt in der Regel die Cluster-ID in einem Clustermodell dar.
entityAffinity	Die Affinität des Elements — stellt in der Regel die Entfernung vom Cluster-Zentrum in einem Modell dar.
upperConfidenceLimit	Die obere Konfidenzgrenze der Vorhersage.
lowerConfidenceLimit	Die untere Konfidenzgrenze der Vorhersage.
propensity	Der Neigungs-Score. Ein zusätzliches tag-Attribut gibt an, ob sich diese Angabe auf eine rohe oder auf eine angepasste Neigung bezieht.
value	Wird für die Darstellung eines Werts für Modelle verwendet, denen eine andere Ausgabe zugeordnet ist (siehe unten).
supplementary	Vom Modell erzeugte Informationen, die nicht durch andere Rollen abgedeckt werden.

Beispiel für value: Das Anomalieerkennungsmodell generiert Gruppen von Feldern, bei denen jede Gruppe aus zwei Feldern besteht, von denen das eine einen Feldnamen angibt und das andere eine Maßzahl zur Anomalie des Felds. In diesem Fall ist value der Feldname.

Logische Operatoren

Eine Anzahl von Elementen kann die logischen Operatoren And, Or und Not verwenden, um verschiedene Verarbeitungsarten anzugeben, wie wenn z. B. zusammengesetzte Bedingungen festgelegt werden siehe Zusammengesetzte Bedingungen auf S. 95).

Format

Das Format des And-Elements sieht wie folgt aus. Das Format der Elemente Or und Not ist nahezu identisch, der einzige Unterschied besteht in den einschließenden Tags <Or>...</Or> bzw. <Not>...</Not>.

```
<And>
 <Condition .../>
 <And ... />
   <Or ... />
   <Not ... />
</And>
```

Das Element Condition gibt eine zu testende Bedingung an. Für weitere Informationen siehe Thema Bedingungen auf S. 91.

Beachten Sie, dass die untergeordneten Elemente And, Or und Not geschachtelt werden können.

Bedingungen

Das Verhalten einiger Objekte kann mithilfe von Bedingungen geändert werden. Diese werden in Condition-Elementen angegeben (die IF-Anweisungen entsprechen). Beispiel: Ein Knoten, der durch einen Befehl ausgeführt wird, kann zur Ausführungsinformation eine Bedingung hinzufügen, wie z. B., dass eine bestimmte Option nur dann eingeschlossen wird, wenn eine Eigenschaft einen bestimmten Wert hat. Genauso kann ein Eigenschaftssteuerelement auf der Benutzeroberfläche nur dann aktiviert oder sichtbar sein, wenn ein anderes Steuerelement einen bestimmten Wert hat.

Bedingungen können einfach oder zusammengesetzt sein. Eine **einfache Bedingung** hat folgende Bestandteile:

- eine Wertquelle (entweder eine Eigenschaft oder ein Steuerelement)
- einen Test
- einen optionalen Testwert

Eine **zusammengesetzte Bedingung** ermöglicht die Kombination von Bedingungen zu komplexen logischen Bedingungen. Für zusammengesetzte Bedingungen kommen zum Einsatz:

- And
- Or
- Not

Format

```
<Condition container="container_name" control="prop_name" property="name" op="operator"
   value="value" />
```

Dabei gilt:

container spezifiziert den Namen eines bestimmten Containers, dessen Wert durch die Bedingung getestet wird.

control gibt ein Eigenschaftssteuerelement an, dessen Wert durch die Bedingung getestet wird. *prop_name* ist der Wert des property-Attributs des Elements, in dem das Steuerelement definiert ist (z. B. in einem Eigenschaftsfenster auf einer Dialogregisterkarte).

property gibt eine Eigenschaft an, deren Wert durch die Bedingung getestet wird. *name* ist der Wert des name-Attributs des Property-Elements, in dem die Eigenschaft definiert ist.

op ist der Bedingungsoperator. Für weitere Informationen siehe Thema Bedingungsoperatoren auf S. 92.

value ist der spezifische Wert, der durch die Bedingung getestet wird.

Beispiele

Beispiele für das Festlegen von Bedingungen finden Sie hier Einfache Bedingungen auf S. 94 und hier Zusammengesetzte Bedingungen auf S. 95.

Bedingungsoperatoren

Es steht ein Satz Operatoren zur Verfügung, der die Formulierung der meisten Bedingungen ermöglicht.

Tabelle 4-13
Für beliebige Werte unterstützte Tests

Operator	Wert	Beschreibung
equals	*value*	Wahr, wenn die Eigenschaft gleich dem angegebenen Wert ist (bei Zeichenketten wird zwischen Groß- und Kleinschreibung unterschieden).
notEquals	*value*	Wahr, wenn die Eigenschaft nicht dem angegebenen Wert entspricht (bei Zeichenketten wird zwischen Groß- und Kleinschreibung unterschieden).
in	*Werteliste*	Wahr, wenn sich die Eigenschaft in der angegebenen Werteliste befindet.

Tabelle 4-14
Für numerische Werte unterstützte Tests

Operator	Wert	Beschreibung
lessThan	*number*	Wahr, wenn die Eigenschaft kleiner als die angegebene Zahl ist.
lessOrEquals	*number*	Wahr, wenn die Eigenschaft kleiner oder gleich die angegebene Zahl ist.
greaterThan	*number*	Wahr, wenn die Eigenschaft größer als die angegebene Zahl ist.
greaterOrEquals	*number*	Wahr, wenn die Eigenschaft größer oder gleich die angegebene Zahl ist.

Tabelle 4-15
Für Zeichenketten unterstützte Tests

Operator	Wert	Beschreibung
isEmpty	-	Wahr, wenn die Eigenschaft eine Zeichenkette der Länge null ist.
isNotEmpty	-	Wahr, wenn die Eigenschaft keine Zeichenkette der Länge null ist.
startsWith	*string*	Wahr, wenn die Eigenschaft mit der angegebenen Zeichenfolge beginnt (Groß- und Kleinschreibung wird unterschieden).
startsWithIgnoreCase	*string*	Wahr, wenn die Eigenschaft mit der angegebenen Zeichenfolge beginnt, wobei die Groß- und Kleinschreibung ignoriert wird.
endsWith	*string*	Wahr, wenn die Eigenschaft auf die angegebene Zeichenfolge endet (Groß- und Kleinschreibung wird unterschieden).

Operator	Wert	Beschreibung
endsWithIgnoreCase	*string*	Wahr, wenn die Eigenschaft auf die angegebene Zeichenfolge endet, wobei die Groß- und Kleinschreibung ignoriert wird.
equalsIgnoreCase	*string*	Wahr, wenn die Eigenschaft gleich der angegebenen Zeichenfolge ist, wobei die Groß- und Kleinschreibung ignoriert wird.
hasSubstring	*string*	Wahr, wenn die Eigenschaft die angegebene Zeichenfolge enthält (Groß- und Kleinschreibung wird unterschieden).
hasSubstringIgnoreCase	*string*	Wahr, wenn die Eigenschaft die angegebene Zeichenfolge enthält, wobei die Groß- und Kleinschreibung ignoriert wird.
isSubstring	*string*	Wahr, wenn die Eigenschaft eine Teilzeichenfolge der angegebenen Zeichenfolge ist (Groß- und Kleinschreibung wird unterschieden).
isSubstringIgnoreCase	*string*	Wahr, wenn die Eigenschaft eine Teilzeichenfolge der angegebenen Zeichenfolge ist, wobei die Groß- und Kleinschreibung ignoriert wird.

Tabelle 4-16
Für Listeneigenschaften unterstützte Tests

Operator	Wert	Beschreibung
isEmpty	-	Wahr, wenn die Anzahl der in der Liste vorhandenen Elemente 0 ist.
isNotEmpty	-	Wahr, wenn die Anzahl der in der Liste vorhandenen Elemente ungleich 0 ist.
countEquals	*number*	Wahr, wenn die Anzahl der in der Liste enthaltenen Elemente gleich dem angegebenen Wert ist.
countLessThan	*number*	Wahr, wenn die Anzahl der in der Liste enthaltenen Elemente kleiner als der angegebene Wert ist.
countLessOrEquals	*number*	Wahr, wenn die Anzahl der in der Liste enthaltenen Elemente kleiner oder gleich der angegebene Wert ist.
countGreaterThan	*number*	Wahr, wenn die Anzahl der in der Liste enthaltenen Elemente größer als der angegebene Wert ist.
countGreaterOrEquals	*number*	Wahr, wenn die Anzahl der in der Liste enthaltenen Elemente größer oder gleich der angegebene Wert ist.
contains	*value*	Wahr, wenn das angegebene Element in der Liste enthalten ist.

Tabelle 4-17
Für Feldeigenschaften unterstützte Tests

Operator	Beschreibung
storageEquals	Wahr, wenn der Speicher gleich dem angegebenen Wert ist.
typeEquals	Wahr, wenn der Typ gleich dem angegebenen Wert ist.
directionEquals	Wahr, wenn die Feldrolle (direction) gleich dem angegebenen Wert ist.

Operator	Beschreibung
isMeasureDiscrete	Wahr, wenn der Felddatentyp discrete ist (er kann also nur set, flag oder orderedSet sein).
isMeasureContinuous	Wahr, wenn der Felddatentyp range ist.
isMeasureTypeless	Wahr, wenn der Felddatentyp typeless ist.
isMeasureUnknown	Wahr, wenn der Felddatentyp unknown ist.
isStorageString	Wahr, wenn der Speichertyp string ist.
isStorageNumeric	Wahr, wenn der Speichertyp numeric ist.
isStorageDatetime	Wahr, wenn der Speichertyp datetime ist.
isStorageUnknown	Wahr, wenn der Speichertyp unknown ist.
isModelOutput	Wahr, wenn das Feld ein Modellausgabefeld ist.
modelOutputRoleEquals	Wahr, wenn die Rolle für das Feld eine der in der nächsten Tabelle enthaltenen gültigen Rollen ist.
modelOutputTargetFieldEquals	Wahr, wenn das Zielfeld gleich der angegebenen Wert (Zeichenfolge) ist.
modelOutputHasValue	Wahr, wenn das Feld ein Modellausgabefeld ist, dem ein Wert zugeordnet ist.
modelOutputTagEquals	Wahr, wenn das Tag gleich der angegebenen Wert (Zeichenfolge) ist.

Bedingungsoperatoren, die Schlüsseleigenschaften unterstützen:

- isEmpty
- isNotEmpty
- countEquals
- countLessThan
- countLessOrEquals
- countGreaterThan
- countGreaterOrEquals
- contains

Einfache Bedingungen

Eine einfache Bedingung besteht aus einer ersten Wertquelle, die getestet werden soll (entweder eine Eigenschaft oder ein Controllername oder ein ausgewerteter Ausdruck), dem durchzuführenden Test und optional einem Wert, gegen den der Test durchgeführt werden soll.

Beispiele

Die folgende Bedingung liefert das Ergebnis ‘True’ (Wahr), wenn eine boolesche Eigenschaft namens values_grouped wahr ist:

```
<Condition property="values_grouped" op="equals" value="true"/>
```

Das nächste Beispiel liefert das Ergebnis ‘True’ (Wahr), wenn ein Steuerelement namens values_grouped, das boolesche Werte anzeigt, aktiviert ist:

```
<Condition control="values_grouped" op="equals" value="true"/>
```

Das folgende Beispiel liefert das Ergebnis ‘True’ (Wahr), wenn eine Listeneigenschaft namens plot_fields mindestens einen Wert enthält:

```
<Condition property="plot_fields" op="countGreaterThan" value="0"/>
```

Das nächste Beispiel liefert das Ergebnis ‘True’ (Wahr), wenn eine Listeneigenschaft namens input_fields nur instanziierte Werte enthält:

```
<Condition property="input_fields" op="instantiated" listMode="all"/>
```

Das letzte Beispiel liefert das Ergebnis ‘True’ (Wahr), wenn eine Listeneigenschaft namens input_fields mindestens einen Wert hat, der ein nicht instanziiertes Feld darstellt:

```
<Condition property="input_fields" op="uninstantiated" listMode="any "/>
```

Zusammengesetzte Bedingungen

Gruppen einfacher Bedingungen können mithilfe logischer Operatoren kombiniert werden.

Beispiele

Die folgende Bedingung liefert das Ergebnis ‘True’ (Wahr), wenn die boolesche Eigenschaft values_grouped wahr ist und group_fields mindestens einen Wert enthält:

```
<And>
   <Condition property="values_grouped" op="equals" value="true"/>
   <Condition property="group_fields" op="countGreaterThan" value="0"/>
</And>
```

Die folgende Bedingung liefert das Ergebnis ‘True’ (Wahr), wenn die boolesche Eigenschaft values_grouped wahr ist oder wenn group_fields mindestens einen Wert enthält:

```
<Or>
   <Condition property="values_grouped" op="equals" value="true"/>
   <Condition property="group_fields" op="countGreaterThan" value="0"/>
</Or>
```

Das folgende Beispiel liefert das Ergebnis ‘True’ (Wahr), wenn group_fields mindestens einen Wert enthält:

```
<Not>
   <Condition property="group_fields" op="equals" value="0"/>
</Not>
```

Zusammengesetzte Bedingungen können geschachtelt werden, um beliebige Kombinationen von Bedingungen zu realisieren.

Verwenden von CLEF-Knoten in Skripts

Sie können in einem Skript auf einen CLEF-Knoten verweisen, indem Sie das Attribut scriptName des Elements Node verwenden. Auf die gleiche Weise können Sie in einem Skript auf eine Eigenschaft des Knotens mithilfe des Attributs scriptName des Elements Property verweisen.

Das Attribut scriptName ist in beiden Fällen optional, seine Verwendung wird jedoch empfohlen, um Namenskonflikte zwischen Erweiterungen und Eigenschaften zu vermeiden.

Wenn Sie den Skriptnamen in einer Knotendefinition weglassen, kann das Skript mithilfe des Werts vom Attribut id, dem der Erweiterungsname als Präfix vorangestellt ist, auf den Knoten verweisen. Beispiel: Eine Erweiterung namens myext, die einen Datenleseknoten mit der ID import definiert, kann in einem Skript als myextimport auf den Knoten verweisen.

Wenn Sie den Skriptnamen in einer Eigenschaftsdefinition unterdrücken, kann ein Skript auf die Eigenschaft durch den Wert ihres Attributs name verweisen.

Weitere Informationen finden Sie im *IBM® SPSS® ModelerHandbuch für Skripterstellung und Automatisierung*.

Beispiel - Bearbeiten und Ausführen eines Knotens

Das folgende Beispiel zeigt, wie Sie mithilfe eines Skripts die Aufgaben der Bearbeitung und Ausführung des Beispiel-Datenleseknotens (siehe Datenleserknoten (Apache Log Reader) auf S. 34) automatisieren können.

In der Spezifikationsdatei für den Knoten “Apache Log Reader” beginnt die Knotenspezifikation wie folgt:

```
<Node id="apachelogreader" type="dataReader" palette="import" labelKey="apacheLogReader.LABEL">
 <Properties>
 <Property name="log_filename" valueType="string" labelKey="logfileName.LABEL" />
   </Properties>
```

Im Skript würden Sie wie folgt auf den Knoten und die Eigenschaft verweisen:

```
erstellen apachelogreader
set :apachelogreader.log_filename='Installationsverzeichnis\Demos\combined_log_format.txt'
create tablenode at 200 100
connect :apachelogreader to :tablenode
execute :tablenode
```

Dabei bezeichnet *installation_directory* das Verzeichnis, in dem IBM® SPSS® Modeler installiert ist.

Ausführen dieses Skripts:

- erstellt den Datenleserknoten.
- gibt *combined_log_format.txt* als zu lesende Apache-Protokolldatei an.
- erstellt einen Tabellenknoten.

- verbindet den Datenleserknoten mit dem Tabellenknoten.
- führt den Tabellenknoten aus.

Beispiel - Schlüsseleigenschaften

Schlüsseleigenschaften unterstützen Standard-Skriptingsyntax. Beispielsweise könnte die Struktur aus dem ersten Beispiel für Schlüsseleigenschaften unter Strukturierte Eigenschaften auf S. 78. in einem Skript wie folgt definiert werden:

```
set :mynode.aggregation_settings.Age = {true true false false false}
```

Ein einzelnes Attribut könnte wie folgt geändert werden:

```
set :mynode.aggregation_settings.Age.MIN = true
```

Erhaltung der Abwärtskompatibilität

Beim Planen von Aktualisierungen für eine vorhandene Erweiterung müssen Sie darauf achten, dass die Kompatibilität zu vorher verteilten Versionen der Erweiterung erhalten bleibt. Einige Änderungen haben keine problematischen Auswirkungen, einige stellen ein signifikantes Risiko dar und bei anderen ist bekannt, dass sie die Kompatibilität aufheben und daher vermieden werden sollten.

Änderungen ohne Risiko

Die folgenden Änderungen haben keine Auswirkungen auf die Abwärtskompatibilität:

- Hinzufügen neuer Node-, ModelOutput-, DocumentOutput- oder InteractiveModelBuilder-Elemente
- Hinzufügen neuer Property-Definitionen und der diesen Elementen zugeordneten neuen Steuerelemente
- Hinzufügen neuer Container dieser Elemente*
- Hinzufügen neuer Werte zu einer vorhandenen Aufzählungseigenschaft

* Beachten Sie, dass jeder Code, der diese neuen Container verwendet, zulassen sollte, dass diese Container für Objekte, die mit der vorherigen Version der Erweiterung erstellt wurden, leer sind.

Änderungen mit signifikantem Risiko

Änderungen an vorhandenen Deklarationen bergen ein signifikantes Risiko hinsichtlich der Kompatibilität. Sie sollten vor der Freigabe sorgfältig getestet werden.

Zu vermeidende Änderungen

Bei den folgenden Änderungen ist bekannt, dass sie die Kompatibilität beeinträchtigen. Sie sollten daher vermieden werden:

- Änderung des Werts des id- oder providerTag-Attributs im ExtensionDetail-Element

- Änderung des Werts des id-Attributs in einem Node-, ModelOutput-, DocumentOutput- oder InteractiveModelBuilder-Element
- Entfernen eines Node-, ModelOutput-, DocumentOutput- oder InteractiveModelBuilder-Elements aus der Erweiterung
- Änderung des Werts des valueType-Attributs in einem Property- oder PropertyType-Element

Kapitel 5

Erstellen von Modellen und Dokumenten

Einführung in die Modell- und Dokumenterstellung

Die Standardmodule von IBM® SPSS® Modeler umfassen Knoten für die Generierung bzw. Erstellung der verschiedensten Modelle und Diagramme. CLEF ermöglicht die Definition weiterer Knoten für die Generierung von Modellen und Dokumenten (Diagrammen und Berichten), die nicht durch die Standardmodule bereitgestellt werden.

Bei der Definition von Modell- oder Dokumenterstellungsknoten müssen Sie auch die Objekte definieren, die bei der Ausführung dieser Knoten erstellt werden. Dazu verwenden Sie Elemente, die als "Konstruktoren" bezeichnet werden.

In den folgenden Abschnitten wird dieser Vorgang ausführlich beschrieben.

Modelle

Ein **Modell** ist eine Regelmenge, eine Formel oder eine Gleichung, die ein Ergebnis auf Grundlage von bestimmten Eingabefeldern vorhersagen können. Die Vorhersage eines Ergebnisses ist das zentrale Ziel der Prognoseanalyse. In IBM® SPSS® Modeler erreichen Sie dies durch folgende Schritte:

- Generieren eines Modells aus vorhandenen Daten
- Anwenden des generierten Modells auf die Daten, die für die Vorhersage hinzugezogen werden

Die Generierung eines Modells wird auch als "Modellerstellung" bezeichnet. In SPSS Modeler erstellen Sie ein Modell mithilfe eines Modellierungsknotens. In CLEF werden Modellierungsknoten als **Model Builder-Knoten** bezeichnet, zu Deutsch "Modellerstellungsknoten". Diese Bezeichnung leitet sich von der Syntax der XML-Anweisung ab, die zur Definition dieser Knoten verwendet wird. Für weitere Informationen siehe Thema Modellerstellungsknoten in Kapitel 2 auf S. 12.

Die Anwendung eines Modells auf Daten wird auch als "Daten-Scoring" bezeichnet. Beim Scoring werden die aus der Modellerstellung gewonnenen Informationen für Vorhersagen für neue Datensätze verwendet. In SPSS Modeler ziehen Sie dazu das Symbol eines generierten Modells auf den Stream-Zeichenbereich. Da das Symbol wie ein Goldnugget aussieht, wird ein generiertes Modell in SPSS Modeler auch als "Modell-Nugget" bezeichnet. In CLEF wird ein Modell-Nugget auf der Registerkarte "Modelle" des Managerfensters auch als **Modellausgabeobjekt** bezeichnet, während es im Zeichenbereich als **Modellanwendungsknoten** bezeichnet wird. Für weitere Informationen siehe Thema Modellanwendungsknoten in Kapitel 2 auf S. 14.

Dokumente

Vermutlich möchten Sie nicht nur Modelle, sondern auch andere Objekte, beispielsweise Diagramme oder Berichte, erstellen. In IBM® SPSS® Modeler werden diese Objekte als **Dokumente** bezeichnet. Sie werden mithilfe eines **Dokumenterstellungsknotens** generiert. Für weitere Informationen siehe Thema Dokumenterstellungsknoten in Kapitel 2 auf S. 14.

Konstruktoren

Konstruktoren definieren die Objekte, die entweder als Ergebnis der Ausführung eines Knotens in einen Stream zurückgeliefert oder als Ergebnis der Rückgabe eines Objekts in den Stream erstellt werden.

Konstruktoren können für die folgenden Elemente definiert werden:

- Modellerstellungsknoten
- Dokumenterstellungsknoten
- Modellanwendungsknoten
- Modellausgabeobjekt

Bei **Modell-** und **Dokumenterstellungsknoten** legt der Konstruktor fest, wie das Ausgabeobjekt bei der Ausführung des Knotens generiert wird. Die Definition eines Ausgabeobjekts kann mehrere Eigenschaften und Komponenten umfassen und im Abschnitt “Constructors” wird festgelegt, wie diese initialisiert bzw. aus dem bei der Ausführung generierten Objekt erstellt werden.

Bei **Modellanwendungsknoten** legt der Konstruktor fest, welche Art von Objekt der Knoten zurück in den Stream liefert bzw. welche Art von Objekt er auf die Registerkarte “Modelle” ausgibt.

Bei **Modellausgabeobjekten** erfüllen Konstruktoren folgende Aufgaben:

- Festlegung des Modellanwendungsknotens, der beim Ablegen des Modellausgabeobjekts auf den Stream-Zeichenbereich erstellt wird
- Generieren eines Modellerstellungsknotens mit denselben Einstellungen, die für die Erstellung des Modellausgabeobjekts verwendet wurden

Für weitere Informationen siehe Thema Verwenden von Konstruktoren auf S. 126.

Erstellen von Modellen

Wenn Sie einen Knoten definieren, der ein Modell generieren kann (d. h., Sie definieren einen Modellerstellungsknoten), müssen Sie festlegen, wie der Knoten mit der Model Builder-Komponente von IBM® SPSS® Modeler kommuniziert. Dadurch definieren Sie den Prozess, der das Modell tatsächlich erstellt. Diese Definition legen Sie in der Spezifikationsdatei in einem `Node`-Element fest.

Sofern nicht anders angegeben, beginnt die Modellerstellung, sobald der Endbenutzer im Dialogfeld des Modellerstellungsknotens auf die Schaltfläche Ausführen klickt. Allerdings kann auch ein **interaktives Modell** definiert werden. In diesem Fall kann der Endbenutzer die Datenwerte nach dem Anklicken der Schaltfläche Ausführen noch vor der Erstellung des Modells einstellen bzw. anpassen. Für die interaktive Modellerstellung sind zusätzlich bestimmte Elemente erforderlich, die das interaktive Verhalten steuern. Für weitere Informationen siehe Thema Erstellen interaktiver Modelle auf S. 112.

Für die Definition eines Modellerstellungsknotens müssen Sie dem Node-Element folgende Elemente hinzufügen:

- Das Attribut type="modelBuilder"
- Das untergeordnete Element ModelBuilder
- Das untergeordnete Element Constructors mit einem CreateModelOutput-Element (siehe Verwenden von Konstruktoren auf S. 126)

Das Format des Node-Elements wird im Abschnitt Knoten auf S. 62 beschrieben.

Anmerkung: Die in den nachfolgenden Elementdefinitionen (Abschnitte mit der Überschrift **Format**) angegebenen Elementattribute und untergeordneten Elemente sind optional, es sei denn, es ist in Klammern angegeben, dass diese "erforderlich" sind. Die vollständige Elementsyntax ist in Anhang A*CLEF XML-Schema* auf S. 257 beschrieben.

Die Erweiterung muss für die Modellerstellung auch über ein ModelOutput-Element verfügen, das das generierte Modell beschreibt (siehe Modellausgabe auf S. 110). Das ModelOutput-Element muss ein untergeordnetes Constructors-Element mit einer CreateModelApplier-Definition enthalten. Für weitere Informationen siehe Thema CreateModelApplier (Erstellen des Modellanwenders) auf S. 129.

Modellerstellung (ModelBuilder)

Das ModelBuilder-Element legt das Verhalten eines Modellerstellungsknotens fest. Die Definition erfolgt über die Attribute des Elements und ein oder mehrere untergeordnete Elemente.

Format

```
<ModelBuilder allowNoInputs="true_false" allowNoOutputs="true_false" nullifyBlanks="true_false"
   miningFunctions="[function1 function2 ... ]" >
   <Algorithm ... />
   <ModelingFields ... />
   <ModelGeneration ... />
   <ModelFields ... />
   <AutoModeling ... />
</ModelBuilder>
```

Dabei gilt:

- allowNoInputs bzw. allowNoOutputs muss explizit angegeben werden, wenn Sie ein Modell erstellen möchten, das entweder über keine Eingabefelder oder über keine Ausgabefelder verfügt.

- Wenn nullifyBlanks auf false gesetzt ist, wird die Funktion deaktiviert, wobei leere Werte in den Daten, die an die Model Builder-Komponente von IBM® SPSS® Modeler übergeben werden, durch Nullwerte ersetzt werden (dargestellt durch $null$). Standardmäßig werden leere Werte durch Nullwerte ersetzt. Wenn Ihr Algorithmus leere Werte auf eine andere Weise behandeln muss, müssen Sie diese Funktion jedoch deaktivieren.
- miningFunctions (erforderlich) legt die Data-Mining-Funktionen (d. h. die Funktionen, die das Modell ausführt) fest.

Tabelle 5-1
Data-Mining-Funktionen

Funktion	**Beschreibung**
Klassifikation	Sagt aus Datensätzen mit bekannten Zielwerten einen diskreten Wert (d. h. einen Wert mit dem Datentyp set, flag oder orderedSet) eines unbekannten Zielattributs vorher.
approximation	Sagt aus Datensätzen mit bekannten Zielwerten einen stetigen Wert (d. h. einen Wert mit dem Datentyp range) eines unbekannten Zielattributs vorher.
Clustering	Ermittelt Gruppen mit ähnlichen Datensätzen und kennzeichnet sie entsprechend.
association	Ermittelt aus Daten verwandte Ereignisse oder Attribute.
sequence	Sucht in zeitlich strukturierten Daten nach sequenziellen Mustern.
reduction	Reduziert die Datenkomplexität — z. B. mittels abgeleiteter Felder, die den Inhalt der ursprünglichen Datenfelder zusammenfassen.
conceptExtraction	Wird in Text-Mining verwendet.
categorize	Wird in Text-Mining verwendet.
timeSeries	Sagt aus vergangenen Datenmustern zukünftige Werte vorher.
anomalyDetection	Sucht anhand von Abweichungen von den Normen der jeweiligen Clustergruppen nach ungewöhnlichen Fällen.
attributeImportance	Ermittelt die Attribute, die den größten Einfluss auf ein Zielattribut haben.
supervisedMultiTarget	Schätzt für eine von mehreren Möglichkeiten die Wahrscheinlichkeit eines Ergebnisses (*Ja* oder *Nein*).

Wenn das Modell mehrere Funktionen ausführt, werden die Namen der Funktionen innerhalb der eckigen Klammern durch Leerzeichen getrennt, wie das folgende Beispiel zeigt:

```
<ModelBuilder miningFunctions="[classification approximation]">
 ...
</ModelBuilder>
```

Untergeordnete Elemente

Das Element ModelBuilder kann die folgenden untergeordneten Elemente enthalten:

Tabelle 5-2
Untergeordnete Elemente der ModelBuilder-Deklaration

Untergeordnetes Element	Beschreibung	Siehe...
Algorithmus	(Erforderlich) gibt den Algorithmus für die Generierung des Modells an.	Algorithmus auf S. 103
ModelingFields	Gibt die Kennung an, die nachfolgend im Abschnitt “User Interface” zur Angabe der Position der Steuerelemente für die Eingabe- und Ausgabefelder des Modells verwendet wird. Die Steuerelemente selbst werden in den untergeordneten Elementen InputFields und OutputFields des Elements ModelingFields definiert.	Modellierungsfelder (ModelingFields) auf S. 103
ModelGeneration	Gibt die Kennung an, die nachfolgend im Abschnitt “User Interface” zur Angabe der Position der Steuerelemente für den Modellnamen des generierten Modells verwendet wird.	Modellerzeugung (ModelGeneration) auf S. 108
ModelFields	Gibt den Satz der Eingabe- und Ausgabefelder an, der für das Daten-Scoring in diesem Modell verwendet wird.	Modellfelder (ModelFields) auf S. 108
AutoModeling	Ermöglicht die Verwendung dieses Modells in einem Ensemble-Modellierungsknoten (z. B. in den Knoten “Automatischer Klassifizierer”, “Autom. Cluster” oder “Auto-Numerisch”).	Automatisierte Modellierung auf S. 116

Algorithmus

Das Element Algorithm legt die Details des Algorithmus fest, der für die Generierung des Modells verwendet wird.

```
<Algorithm value="model_output_id" label="display_label" labelKey="label_key"/>
```

Dabei gilt:

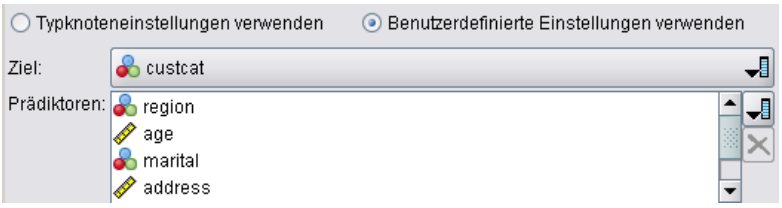
- value (erforderlich) ist der interne Name des Algorithmus. Dieser Name wird an zahlreichen anderen Stellen der Spezifikationsdatei referenziert. Für weitere Informationen siehe Thema Beispiel für den Abschnitt “Model Builder” auf S. 109.
- label (erforderlich) ist die Beschreibung des Algorithmus.
- labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke.

Modellierungsfelder (ModelingFields)

Standardmäßig werden die Eingabe- und Ausgabefelder des Modells anhand des Typknotens festgelegt. Die Benutzer legen die Feldrolle nach Bedarf mit Prädiktor oder Ziel fest. In einem Modellerstellungsknoten können Sie den Benutzern optional die Möglichkeit einräumen, die

Einstellungen in einem vorangehenden Typknoten zu überschreiben und benutzerdefinierte Einstellungen zu verwenden — z. B.:

Abbildung 5-1
Festlegen der Modelleingaben und -ausgaben



Dazu verwenden Sie das Element ModelingFields. Dieses Element legt eine Kennung fest, die nachfolgend im Abschnitt "User Interface" der Deklaration des Modellerstellungsknotens verwendet wird, um die Position der Steuerelemente der Eingabe- und Ausgabefelder des Modells anzugeben. Die Steuerelemente selbst werden in den untergeordneten Elementen InputFields und OutputFields definiert.

Format

```
<ModelingFields controlsId="control_identifier" ignoreBOTH="true_false" >
   <InputFields ... />
   <OutputFields ... />
</ModelingFields>
```

Dabei gilt:

- controlsId (erforderlich) ist die Kennung, die nachfolgend in der Deklaration des Modellerstellungsknotens im SystemControls-Element des Abschnitts "User Interface" verwendet wird. Damit legen Sie fest, auf welcher Registerkarte des Knotendialogfelds sich die Steuerelemente für die Eingabe- und Ausgabefelder des Modells befinden.
- Wenn ignoreBOTH auf true (Standardeinstellung) gesetzt ist, werden Felder, deren Feldrolle auf Beides gesetzt ist, vom Modell ignoriert.

Die Elemente InputFields und OutputFields werden ab dem Abschnitt Eingabefelder (InputFields) auf S. 105 beschrieben.

Beispiel

Dieses Beispiel veranschaulicht die Verwendung eines ModelingFields-Steuerelementsatzes auf der Registerkarte "Felder" im Dialogfeld eines Modellerstellungsknotens. Zunächst wird für den Steuerelementsatz eine Kennung festgelegt:

```
<ModelBuilder miningFunctions="[classification]">
...
   <ModelingFields controlsId="modelingFields">
      <InputFields property="inputs" onlyNumeric="true" multiple="true" label="Eingaben"
       labelKey="inputFields.LABEL"/>
      <OutputFields property="target" multiple="false" types="[set flag]" label="Target"
       labelKey="targetField.LABEL"/>
   </ModelingFields>
```

```
...
</ModelBuilder>
```

Die modelingFields-Kennung wird nachfolgend im Abschnitt “User Interface” des Knotendialogfelds an der Stelle referenziert, an der die Registerkarte “Felder” definiert wird:

```
<UserInterface ... >
   <Tabs defaultTab="1">
      <Tab label="Fields" labelKey="Fields.LABEL" helpLink="modeling_fieldstab.htm">
         <PropertiesPanel>
            <SystemControls controlsId="modelingFields">
            </SystemControls>
         </PropertiesPanel>
      </Tab>
...
</UserInterface>
```

Eingabefelder (InputFields)

Das Element InputFields legt den Feldsatz fest, aus dem die Benutzer ein oder mehrere Eingabefelder (d. h. Prädiktoren) für das Modell auswählen können.

Das Set besteht aus allen Feldern, die an diesem Knoten sichtbar sind. Wenn Felder oberhalb dieses Knotens gefiltert wurden, sind nur die Felder, die den Filter durchlaufen haben, sichtbar. Die Liste kann noch weiter eingeschränkt werden, indem angegeben wird, dass nur Felder mit bestimmten Speicher- und Datentypen zur Auswahl stehen sollen.

```
<InputFields storage="storage_types" onlyNumeric="true_false" onlySymbolic="true_false"
   onlyDatetime="true_false" types="data_types" onlyRanges="true_false" onlyDiscrete="true_false"
   property="property_name" multiple="true_false" label="Label" labelKey="label_key"/>
```

Sie können die Liste der als Eingabefelder möglichen Felder durch die Angabe von nur zwei Attributen einschränken – eines dieser Attribute muss aus folgender Liste stammen:

- storage ist eine Listeneigenschaft, die den Speichertyp der Felder angibt, die in der Liste zulässig sein sollen. So bedeutet storage="[integer real]", dass nur Felder mit diesen Speichertypen aufgelistet werden. Die Menge der möglichen Speichertypen können Sie der Tabelle unter Daten- und Speichertypen auf S. 230 entnehmen.
- onlyNumeric (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit numerischem Speichertyp aufgelistet werden sollen.
- onlySymbolic (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit symbolischem Speichertyp (also Zeichenketten) aufgelistet werden sollen.
- onlyDatetime (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit einem Speichertyp für Datum und Uhrzeit aufgelistet werden sollen.

Das zweite angegebene Attribut muss aus dieser Liste stammen:

- types ist eine Listeneigenschaft, die den Datentyp der Felder angibt, die in der Liste zulässig sein sollen. So bedeutet types="[range flag]", dass nur Felder mit diesen Speichertypen aufgelistet werden. Folgende Datentypen sind möglich:

  Bereich

  Flag

  set

  orderedSet

  Numerisch

  discrete

  typeless
- onlyRanges (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit dem Datentyp "Bereich" aufgelistet werden sollen.
- onlyDiscrete (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit diskretem Datentyp (also Flag, Set oder ohne Typ) aufgelistet werden sollen.

So stellt beispielsweise ein Steuerelement mit der Angabe storage="[integer]" und types="[flag]" sicher, dass nur ganzzahlige Felder, bei denen es sich um Flags handelt, in der Liste angezeigt werden.

Die restlichen Attribute lauten wie folgt:

- property ist die Kennung der Eigenschaft, die zum Speichern der Feldwerte verwendet werden soll.
- multiple gibt an, ob es sich bei den Feldwerten um eine Aufzählungsliste handelt (true) oder nicht (false).
- label ist der Anzeigename des Steuerelements.
- labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke.

Ausgabefelder (OutputFields)

Das Element OutputFields legt den Feldsatz fest, aus dem die Benutzer ein oder mehrere Ausgabefelder (d. h. Ziele) für das Modell auswählen können.

Das Set besteht aus allen Feldern, die an diesem Knoten sichtbar sind. Wenn Felder oberhalb dieses Knotens gefiltert wurden, sind nur die Felder, die den Filter durchlaufen haben, sichtbar. Die Liste kann noch weiter eingeschränkt werden, indem angegeben wird, dass nur Felder mit bestimmten Speicher- und Datentypen zur Auswahl stehen sollen.

```
<OutputFields storage="storage_types" onlyNumeric="true_false" onlySymbolic="true_false"
   onlyDatetime="true_false" types="data_types" onlyRanges="true_false"
   onlyDiscrete="true_false" property="property_name" multiple="true_false" label="Label"
   labelKey="label_key"/>
```

Sie können die Liste der als Ausgabefelder möglichen Felder durch die Angabe von nur zwei Attributen einschränken – eines dieser Attribute muss aus folgender Liste stammen:

- storage ist eine Listeneigenschaft, die den Speichertyp der Felder angibt, die in der Liste zulässig sein sollen. So bedeutet storage="[integer real]", dass nur Felder mit diesen Speichertypen aufgelistet werden. Die Menge der möglichen Speichertypen können Sie der Tabelle unter Daten- und Speichertypen auf S. 230 entnehmen.
- onlyNumeric (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit numerischem Speichertyp aufgelistet werden sollen.
- onlySymbolic (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit symbolischem Speichertyp (also Zeichenketten) aufgelistet werden sollen.
- onlyDatetime (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit einem Speichertyp für Datum und Uhrzeit aufgelistet werden sollen.

Das zweite angegebene Attribut muss aus dieser Liste stammen:

- types ist eine Listeneigenschaft, die den Datentyp der Felder angibt, die in der Liste zulässig sein sollen. So bedeutet types="[range flag]", dass nur Felder mit diesen Speichertypen aufgelistet werden. Folgende Datentypen sind möglich:

  Bereich

  Flag

  set

  orderedSet

  Numerisch

  discrete

  typeless
- onlyRanges (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit dem Datentyp “Bereich” aufgelistet werden sollen.
- onlyDiscrete (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit diskretem Datentyp (also Flag, Set oder ohne Typ) aufgelistet werden sollen.

So stellt beispielsweise ein Steuerelement mit der Angabe storage="[integer]" und types="[flag]" sicher, dass nur ganzzahlige Felder, bei denen es sich um Flags handelt, in der Liste angezeigt werden.

Die restlichen Attribute lauten wie folgt:

- property ist die Kennung der Eigenschaft, die zum Speichern der Feldwerte verwendet werden soll.
- multiple gibt an, ob es sich bei den Feldwerten um eine Aufzählungsliste handelt (true) oder nicht (false).
- label ist der Anzeigename des Steuerelements.
- labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke.

Modellerzeugung (ModelGeneration)

Das Element ModelGeneration legt die Kennung fest, die an anderen Stellen der Datei zur Angabe der Registerkarte des Dialogfelds des Modellerstellungsknotens verwendet werden soll, die die Steuerelemente für den Modellnamen des generierten Modells enthalten soll:

Abbildung 5-2
Steuerelemente für den Modellnamen

Format:

```
<ModelGeneration controlsId="control_identifier" />
```

Das Attribut controlsId legt die Kennung fest, die nachfolgend in der Spezifikation des Modellerstellungsknotens im SystemControls-Element des Abschnitts “User Interface” verwendet wird. Die Steuerelemente für den Modellnamen werden auf der Registerkarte eingefügt, deren Spezifikation dieses SystemControls-Element enthält.

Modellfelder (ModelFields)

Das Element ModelFields wird zur Erstellung der **Modellsignatur** verwendet. Hiermit ist der Satz der Eingabe- und Ausgabefelder gemeint, der für das Daten-Scoring in diesem Modell verwendet wird.

```
<ModelFields inputDirections="[in]" outputDirections="[out]">
   <AddField prefix="field_prefix" ... />
   ...
   <ForEach ... >
      <AddField prefix="field_prefix" ... />
   </ForEach>
   ...
</ModelFields>
```

Dabei geben inputDirections und outputDirections an, wie die Modellsignatur erstellt wird. Die Werte können in, out oder both sein.

Die Felder werden durch ein oder mehrere AddField-Elemente definiert. Das Attribut prefix gibt das Präfix an, das einem Feldnamen hinzugefügt wird, um die von dem Modell erstellten Felder zu kennzeichnen. Beispiel: Wenn der Feldname Feld1 lautet und das Präfix $S, dann erhält das generierte Feld den Namen $S-Feld1. Für weitere Informationen siehe Thema Feld hinzufügen in Kapitel 4 auf S. 82.

Iteration wird durch das Element ForEach möglich. Für weitere Informationen siehe Thema Iteration mit dem Element ‘ForEach’ in Kapitel 4 auf S. 85.

Mithilfe von **Feldgruppen** können Sie zwei oder mehr Ausgabefelder des Modells zu Iterationszwecken gruppieren. Ausgabefeldnamen erhalten ein Suffix, das die Iteration angibt, z. B. $S-field1-1 , $S-field1-2 usw. Feldgruppen lassen sich beispielsweise dafür verwenden, dieselbe Feldgruppe mehrmals in der Modellausgabe anzuzeigen. Für weitere Informationen siehe Thema Beispiel für Feldgruppe auf S. 109.

Automatische Modellierung (AutoModeling)

Das Element “AutoModeling” ermöglicht die Verwendung des Modells in einem Ensemble-Modellierungsknoten (z. B. in den Knoten “Automatischer Klassifizierer”, “Autom. Cluster” oder “Auto-Numerisch”). Für weitere Informationen siehe Thema Automatisierte Modellierung auf S. 116.

Beispiel für den Abschnitt “Model Builder”

Nachfolgend ist der vollständige Model Builder-Abschnitt der Spezifikationsdatei am Beispiel des Knotens “Interaction” dargestellt (siehe auch Modellerstellungsknoten (Interaction) auf S. 36):

```
<Node id="interaction.builder" type="modelBuilder" palette="modeling" label="Interaktion">
   <ModelBuilder miningFunctions="[classification]">
      <Algorithm value="robd" label="Roberts-Algorithmus" />
      <ModelingFields controlsId="modellingFields">
         <InputFields property="inputs" multiple="true" label="Eingaben" onlyDiscrete="true" />
         <OutputFields property="target" multiple="false" label="Ziel" onlyDiscrete="true" />
      </ModelingFields>
      <ModelFields inputDirections="[in]" outputDirections="[out]">
         <ForEach var="field" inFields="outputs">
            <AddField prefix="$I" name="${field}" fieldRef="${field}" role="predictedValue"
             targetField="${field}" />
            <AddField prefix="$IP" name="${field}" storage="real" role="probability"
             targetField="${field}">
               <Range min="0.0" max="1.0"/>
            </AddField>
         </ForEach>
      </ModelFields>
   </ModelBuilder>
...
</Node>
```

Beispiel für Feldgruppe

Dieses Beispiel entstammt dem SLRM-Knoten und fügt der Modellsignatur eine Gruppe von zwei Feldern hinzu, die die generierten Daten enthalten sollen, wenn das Modell bewertet wird. Für jeden Eingabedatensatz werden die Daten für jedes der neuen Felder mit der benutzerdefinierten Häufigkeit bewertet, was durch den Wert der Eigenschaft max_predictions festgelegt wird.

Die beiden neuen Felder sind:

- $S-*target* – enthält den vorhergesagten Wert des Zielfelds.
- $SC-*target* – enthält den Wahrscheinlichkeitswert für diese Vorhersage.

Um diese beiden Felder zu gruppieren, wird ihnen dieselbe Gruppenkennung zugewiesen wie bei ihrer Deklaration im Abschnitt ModelFields. Gruppenkennungen werden mithilfe des Attributs group des Elements AddField zugewiesen.

Die Deklaration für den Modellerstellungsknoten enthält daher:

```
<Node ... type="modelBuilder" ... >
   <ModelBuilder ... >
```

```
    ...
      <ModelFields inputDirections="[in]" outputDirections="[out]">
        <AddField prefix="$S" name="${target}" fieldRef="${target}" role="predictedValue"
          targetField="${target}" group="[1]"/>
        <AddField prefix="$SC" name="${target}" storage="real" role="probability"
          targetField="${target}" group="[1]">
          <Range min="0.0" max="1.0"/>
        </AddField>
      </ModelFields>
    </ModelBuilder>
</Node>
```

Die Deklaration für den Modellanwendungsknoten enthält:

```
<Node ... type="modelApplier" ... >
...
    <OutputDataModel mode="extend">
      <ForEach var="group" from="1" to="${max_predictions}">
        <ForEach var="field" inFields="modelOutputs" container="model">
          <AddField name="${field}" group="[${group}]" fieldRef="${field}" />
        </ForEach>
      </ForEach>
    </OutputDataModel>
</Node>
```

Das Zielfeld hat den Namen campaign und der Benutzer gibt 2 in das Feld ein, das der Eigenschaft max_predictions entspricht. Durch Ausführung des Modellerstellungsknotens werden dem Modell die folgenden Felder angehängt:

- $S-campaign-1
- $SC-campaign-1
- $S-campaign-2
- $SC-campaign-2

Modellausgabe

Das Element ModelOutput beschreibt ein Modellausgabeobjekt, d. h. ein Objekt, das nach der Ausführung eines Streams auf der Registerkarte “Modelle” des Managerfensters angezeigt wird.

Format

```
<ModelOutput id="identifier" label="display_label" labelKey="label_key" >
    <ModelProvider ... />
    <Properties>
        <Property ... />
        ...
    </Properties>
    <Containers ... />
    <UserInterface ... />
    <Constructors ... />
</ModelOutput>
```

Dabei gilt:

- id (erforderlich) ist eine eindeutige Kennung für das generierte Modell.
- label (erforderlich) ist der auf der Registerkarte “Modelle” angezeigte Name des generierten Modells.
- labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke.

Das Element ModelOutput kann die folgenden untergeordneten Elemente enthalten:

Tabelle 5-3
Untergeordnete Elemente der ModelOutput-Deklaration

Untergeordnetes Element	**Definition**	**Siehe...**
ModelProvider	Der Container der Modellausgabe; außerdem, ob die Ausgabe im PMML-Format erfolgt.	Modell-Provider (ModelProvider) auf S. 65
Eigenschaften	Die vom generierten Modell verwendeten Eigenschaften.	Eigenschaften auf S. 66
Container (Containers)	Die Container für die generierte Modellausgabe.	Container (Containers) auf S. 68
UserInterface	Die Komponenten der Benutzeroberfläche, in der die generierte Modellausgabe angezeigt wird — z. B. die Registerkarten “Modell” und “Einstellungen” eines Modellausgabeobjekts.	Benutzeroberfläche auf S. 69
Konstruktoren	Die vom generierten Modell erstellten Objekte.	Verwenden von Konstruktoren auf S. 126

Beispiel

```
<ModelOutput id="interaction.model" label="Interaction Model">
   <Properties>
   </Properties>
   <Containers>
      <Container name="model" />
   </Containers>
   <UserInterface>
      <Tabs>
         <Tab label="Model">
            <TextBrowserPanel container="model" textFormat="plainText" />
         </Tab>
      </Tabs>
   </UserInterface>
   <Constructors>
      <CreateModelApplier type="interaction.applier">
         <SetContainer target="model" source="model" />
      </CreateModelApplier>
   </Constructors>
</ModelOutput>
```

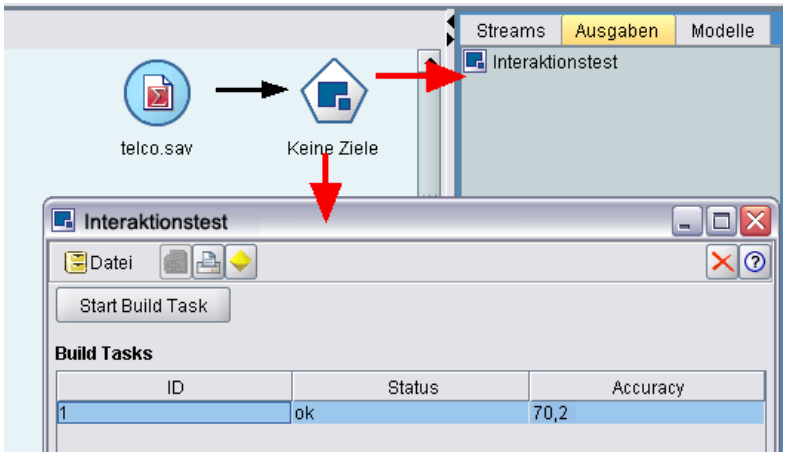
Erstellen interaktiver Modelle

Mit der interaktiven Modellierungsfunktion können Sie ein Ausgabeobjekt erstellen, mit dem der Endbenutzer interagieren kann, bevor er das Modell generiert. Das interaktive Ausgabeobjekt wird auf der Registerkarte "Ausgaben" des Managerfensters angezeigt. Es enthält ein vorläufiges Daten-Set, mit dessen Hilfe das Modell vor der Generierung verfeinert oder vereinfacht werden kann. Für die interaktive Modellierung müssen der Spezifikation eines normalen Modellerstellungsknotens einige zusätzliche Elemente hinzugefügt werden:

- Der Abschnitt Constructors der Node-Definition muss das Element CreateInteractiveModelBuilder enthalten.
- Die Erweiterung muss ein dediziertes InteractiveModelBuilder-Element enthalten.

Die Benutzerinteraktion am vorläufigen Daten-Set erfolgt im so genannten **Interaktionsfenster**, das angezeigt wird, sobald das Ausgabeobjekt erstellt ist.

Abbildung 5-3
Interaktives Ausgabeobjekt und Interaktionsfenster



Die Art der Interaktion richtet sich nach dem verwendeten Algorithmus. Die Interaktion wird durch die Erweiterung implementiert. Das Interaktionsfenster wird im Abschnitt "User Interface" des Elements InteractiveModelBuilder festgelegt. Es wird durch eines der folgenden Elemente definiert:

- Eine Frame-Klasse (siehe Abschnitt 'Benutzeroberfläche' auf S. 134), die das Fenster vollständig definiert
- Eine Panel-Klasse für jede Registerkarte des Fensters; diese wird als Attribut des Erweiterungsobjektfensters angegeben (siehe Erweiterungsobjektfenster auf S. 149)

Wenn das Interaktionsfenster geschlossen wurde, kann es mit einem Doppelklick auf den Objektnamen auf der Registerkarte "Ausgaben" wieder geöffnet werden.

Die Spezifikation des Interaktionsfensters muss Code enthalten, durch den das Modell generiert wird, sobald der Benutzer die Interaktion abgeschlossen hat. Im Beispiel in diesem Abschnitt ist dieser Code durch die Symbolleistenschaltfläche mit dem Goldnugget-Symbol implementiert, die mit einer Aktion verknüpft ist, durch die das Modell generiert wird. Der Code wird im Abschnitt InteractiveModelBuilder unter Beispiel für die interaktive Modellierung auf S. 115 gezeigt.

CreateInteractiveModelBuilder (Erstellen der interaktiven Modellerstellung)

Das Element CreateInteractiveModelBuilder beschreibt das Ausgabeobjekt, über das die Benutzerinteraktion erfolgt. Im Prinzip handelt es sich hier um nichts anderes als um eine interaktive Version des Elements CreateModelOutput.

Format

Dieses Element wird im Abschnitt “Execution” der Definition des Modellerstellungsknotens verwendet:

```
<Node ... type="modelBuilder" ... >
 ...
   <Execution>
    ...
      <Constructors>
       ...
         <CreateInteractiveModelBuilder ... >
          ...
         </CreateInteractiveModelBuilder>
      </Constructors>
   </Execution>
...
</Node>
```

Das Format des Elements selbst sieht wie folgt aus:

```
<CreateInteractiveModelBuilder type="output_object_id">
   <Condition ... />
   <And>
   <Or>
   <Not>
   <CreateModel type="model_id" target="container_id" sourceFile="container_file_id" />
   <CreateDocument type="model_id" target="container_id" sourceFile="container_file_id" />
</CreateInteractiveModelBuilder>
```

Dabei ist type (erforderlich) die Kennung des durch das Element InteractiveModelBuilder erstellten Ausgabeobjekts.

Im Abschnitt Condition können Sie eine oder mehrere Bedingungen festlegen. Für weitere Informationen siehe Thema Bedingungen in Kapitel 4 auf S. 91.

Hier können Sie auch komplexe Bedingungen mit den Operatoren And, Or und Not eingeben. Für weitere Informationen siehe Thema Logische Operatoren in Kapitel 4 auf S. 90.

In den Elementen CreateModel und CreateDocument gilt:

- type ist die Kennung für das zu definierende Modell bzw. Dokument.

- type (erforderlich) ist die Kennung des Containers für das Modell. Dieser Container wird in einem Modellausgabe-Abschnitt definiert. Für weitere Informationen siehe Thema Modellausgabe auf S. 110.
- sourceFile (erforderlich) ist die Kennung einer Ausgabedatei, die während der Knotenausführung generiert wird. Diese Datei wird in einem Abschnitt für Ausgabedateien definiert. Für weitere Informationen siehe Thema Ausgabedateien in Kapitel 4 auf S. 72.

Beispiel

```
<CreateInteractiveModelBuilder type="my.interaction">
   <Condition property="interactive" op="equals" value="true" />
</CreateInteractiveModelBuilder>
```

In diesem Beispiel wird bei der Ausführung des Modellerstellungsknotens, dessen Spezifikation dieses Element enthält, ein Ausgabeobjekt mit der Kennung my.interaction erstellt. Das Ausgabeobjekt selbst wird an einer anderen Stelle der Spezifikationsdatei durch ein InteractiveModelBuilder-Element definiert, das auf diese Kennung verweist. Beispiel:

```
<InteractiveModelBuilder id="my.interaction" label=...>
 ...
</InteractiveModelBuilder>
```

Interaktive Modellerstellung

Dieses Element definiert ein interaktives Ausgabeobjekt, mit dem der Endbenutzer ein Modell verfeinern oder vereinfachen kann, bevor es generiert wird.

Das Element InteractiveModelBuilder folgt auf die Definition eines Modellerstellungsknotens mit einem entsprechenden CreateInteractiveModelBuilder-Element. Für weitere Informationen siehe Thema CreateInteractiveModelBuilder (Erstellen der interaktiven Modellerstellung) auf S. 113.

Format

Das Format des Elements InteractiveModelBuilder sieht wie folgt aus:

```
<Node ... type="modelBuilder" ... >
 ...
   -- Create Interactive Model Builder section --
 ...
</Node>
 ...
<InteractiveModelBuilder id="identifier" label="display_label" labelKey="label_key">
   <Properties>
      <Property name=... />
       ...
   </Properties>
   <Containers>
      <Container name="container_name"/>
   </Containers>
   <UserInterface ... />
```

```
  <Constructors ... />
</InteractiveModelBuilder>
```

Dabei gilt:

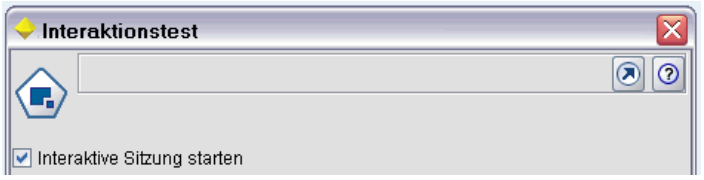
- id (erforderlich) ist eine eindeutige Kennung für das generierte Modell.
- label (erforderlich) ist der auf der Registerkarte “Modelle” angezeigte Name des generierten Modells.
- labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke.

Weitere Informationen zu den Elementen Properties, Containers, UserInterface und Constructors finden Sie in den Abschnitten Eigenschaften auf S. 66, Container (Containers) auf S. 68, Abschnitt ‘Benutzeroberfläche’ auf S. 134 und Verwenden von Konstruktoren auf S. 126.

Beispiel für die interaktive Modellierung

Dieses Beispiel zeigt, wie Sie einen Modellerstellungsknoten generieren, in dem die Benutzer anhand eines Kontrollkästchens angeben können, ob sie das Modell vor der Generierung anpassen möchten.

Abbildung 5-4
Auswahl der Interaktion



Diese Funktion können Sie an dem in diesem Release enthaltenen Beispielknoten “Interaction” testen. Für weitere Informationen siehe Thema Modellerstellungsknoten (Interaction) in Kapitel 3 auf S. 36.

Zunächst gibt der Modellerstellungsknoten eine Boole’sche Eigenschaft an:

```
<Node id="interaction.builder" type="modelBuilder" ... >
 ...
   <Properties>
    <Property name="interaktiv" valueType="boolean" />
   </Properties>
```

Im Abschnitt “User Interface” der Knotenspezifikation enthält der Abschnitt, der die Registerkarte “Modell” definiert, einen Verweis auf diese Eigenschaft:

```
<Tab label="Model">
   <PropertiesPanel>
      <CheckBoxControl property="interaktiv" label="Interaktive Sitzung starten" />
   </PropertiesPanel>
</Tab>
```

Im Abschnitt CreateInteractiveModelBuilder des gleichen Knotens wird die Einstellung der Eigenschaft untersucht. Falls diese true lautet, wird ein interaktives Ausgabeobjekt erstellt:

```
<CreateInteractiveModelBuilder type="my.interaction">
   <Condition property="interaktiv" op="equals" value="true" />
</CreateInteractiveModelBuilder>
```

Das referenzierte Ausgabeobjekt ist im Abschnitt InteractiveModelBuilder der Erweiterung definiert:

```
<InteractiveModelBuilder id="my.interaction" label="Interaktionstest">
   <Properties>
   </Properties>
   <Containers>
   </Containers>
   <UserInterface actionHandler="ui.InteractionHandler">
      <Controls>
         <ToolbarItem action="generateModel" showLabel="false" />
      </Controls>
      <Tabs>
         <Tab label="Model">
            <ExtensionObjectPanel id="model.panel" panelClass="ui.SampleInteractionPanel" />
         </Tab>
         <Tab label="Generic">
            <ExtensionObjectPanel id="generic.panel" panelClass="ui.GenericInteractionPanel" />
         </Tab>
      </Tabs>
   </UserInterface>
</InteractiveModelBuilder>
```

Die Aktion, durch die das Modell generiert wird, wird durch die im Element ToolbarItem definierte Symbolleistenschaltfläche gesteuert.

Achten Sie besonders auf die Verwendung des Attributs panelClass im Element ExtensionObjectPanel. Hiermit wird die Java-Klasse festgelegt, die die Benutzeroberfläche der einzelnen Registerkarten des Interaktionsfensters steuert.

Automatisierte Modellierung

IBM® SPSS® Modeler stellt als Standard eine Gruppe von Ensemble-Modellierungsknoten bereit, zu denen unter anderem die Knoten “Automatischer Klassifizierer”, “Autom. Cluster” und “Auto-Numerisch” gehören. Mit diesen Knoten kann die gleichzeitige Erstellung verschiedener Modelle automatisiert werden. Der Endbenutzer kann die Ergebnisse anschließend vergleichen und das beste Modell für seine Daten auswählen. Für die Festlegung, dass ein durch das ModelBuilder-Element angegebenes Modell von einem dieser Ensemble-Knoten verwendet werden soll, stellt CLEF das AutoModeling-Element bereit.

Das Format des Elements AutoModeling sieht wie folgt aus:

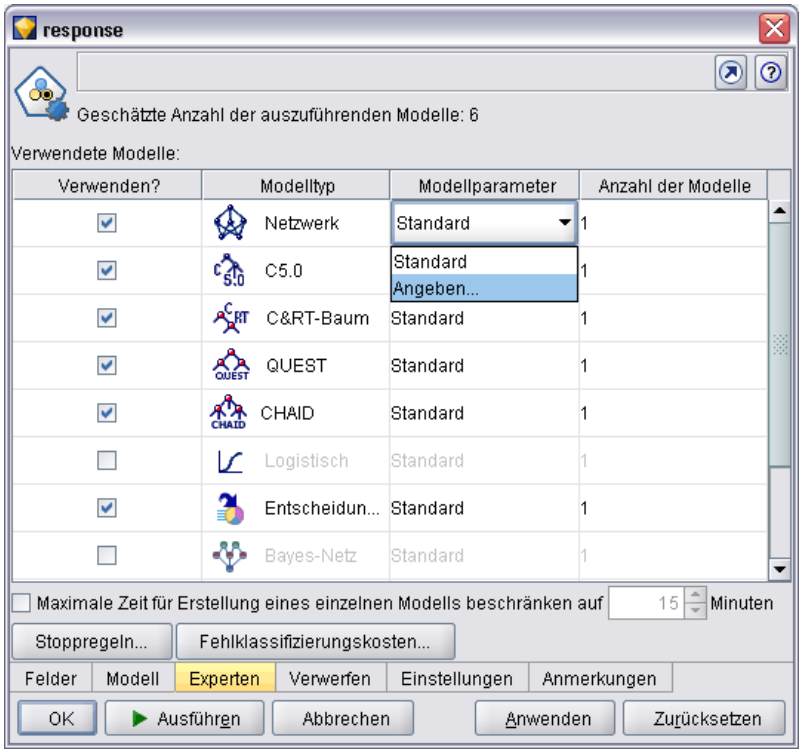
```
<AutoModeling enabledByDefault="true_false">
   <SimpleSettings ... />
   <ExpertSettings ... />
```

```
    <Constraint ... />
    <Constraint ... />
     ...
</AutoModeling>
```

Dabei gibt enabledByDefault an, ob das Modell im Modellierungsknoten standardmäßig zur Verwendung im Ensemble-Modellierungsknoten aktiviert ist (d. h. die Spalte Verwenden? ist für dieses Modell standardmäßig aktiviert, wie in folgender Abbildung gezeigt). Wenn dieses Attribut weggelassen wird, wird der Wert true angenommen.

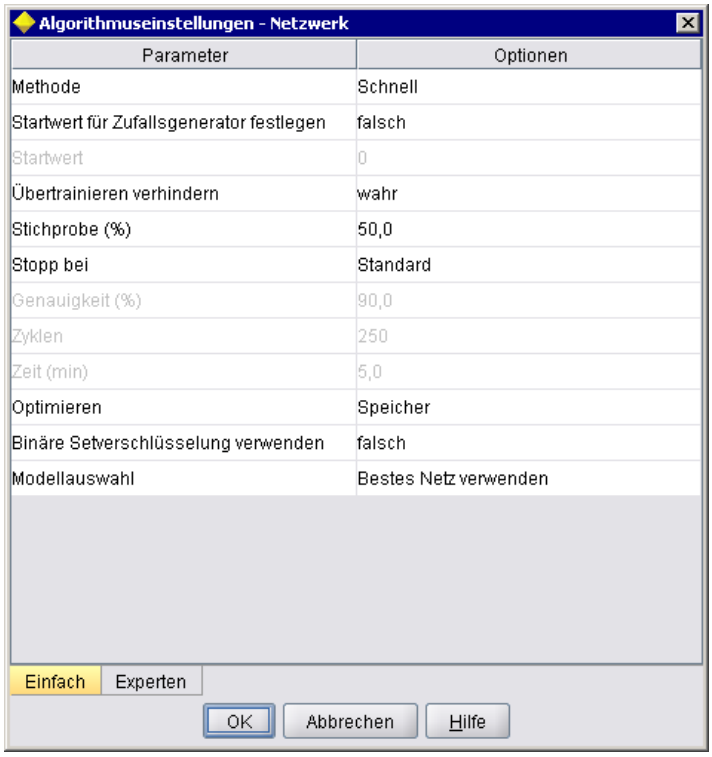
Die Registerkarte "Experten" im Dialogfeld eines Ensemble-Modellierungsknotens listet die Modelle auf, die vom Benutzer erstellt werden können. Beispiel:

Abbildung 5-5
Registerkarte "Experten" eines Ensemble-Modellierungsknotens



Wenn der Benutzer im Feld Modellparameter eines bestimmten Modells auf Angeben klickt, wird das Dialogfeld "Algorithmuseinstellungen" geöffnet, in dem er die Optionen für den Modelltyp auswählen kann.

Abbildung 5-6
Algorithmuseinstellungen für einen Ensemble-Modellierungsknoten



Das Dialogfeld “Algorithmuseinstellungen” enthält entsprechend den Ausführungsmodi des Modellierungsknotens die Registerkarten “Einfach” und “Experten”. Der Inhalt der Registerkarten “Einfach” und “Experten” wird durch die in den nachfolgenden Abschnitten beschriebenen Elemente SimpleSettings und ExpertSettings gesteuert.

Darüber hinaus ermöglichen Constraint-Elemente die Angabe von Bedingungen, die es dem Endbenutzer erlauben, Parameter im Dialogfeld “Algorithmuseinstellungen” zu bearbeiten, bzw. unter denen die Bearbeitung eingeschränkt ist. Für weitere Informationen siehe Thema Nebenbedingungen auf S. 122.

Einigen Parametern im Dialogfeld “Algorithmuseinstellungen” können mehrere Werte zugewiesen sein. In diesem Fall versucht der Ensemble-Knoten, Modelle für alle möglichen Kombinationen dieser Parameterwerte zu erstellen. Gibt der Benutzer bei einem verallgemeinerten linearen Modell z. B. zwei Verteilungen (normale und Gamma-Verteilung) und drei Linkfunktionen (Identität, Log und Exponent) an, dann versucht der Knoten “Auto-Numerisch” sechs verallgemeinerte lineare Modelle zu erstellen – eines für jede mögliche Kombination dieser Parameter.

Einfache Einstellungen (SimpleSettings)

Das Element SimpleSettings bestimmt, welche Parameter im Dialogfeld “Algorithmuseinstellungen” dieses Modells eines Ensemble-Modellierungsknotens auf der Registerkarte “Einfach” angezeigt werden. Für weitere Informationen siehe Thema Automatisierte Modellierung auf S. 116.

Format

```
<SimpleSettings>
   <PropertyGroup label="group_name" labelKey="resource_key" properties="[prop_name1
    prop_name2 ...]"/>
   <PropertyGroup ... />
</SimpleSettings>
```

In einem PropertyGroup-Element (mindestens eines ist erforderlich):

label ist eine Anzeigebeschriftung für die Eigenschaftsgruppe, das als Unterbeschriftung in das Dialogfeld oberhalb des ersten Parameters in der Gruppe eingefügt wird.

labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke. Wenn weder label noch labelKey verwendet wird, wird keine Unterbeschriftung eingefügt.

properties (erforderlich) ist eine Liste mit einer oder mehreren Eigenschaften, die auf der Registerkarte angezeigt werden sollen. Der Wert von *prop_name1*, *prop_name2* usw. ist der Wert des Attributs name des Elements Property, in dem diese Eigenschaft definiert wird. Für weitere Informationen siehe Thema Eigenschaften in Kapitel 4 auf S. 66.

Beispiel

```
<SimpleSettings>
   <PropertyGroup properties="[method]"/>
</SimpleSettings>
```

Dieses Beispiel aus dem Diskriminanzknoten legt fest, dass für dieses Modell des relevanten Ensemble-Modellierungsknotens (in diesem Fall der Knoten "Automatischer Klassifizierer") auf der Registerkarte "Einfach" des Dialogfelds "Algorithmuseinstellungen" nur der Parameter Methode angezeigt wird. Da kein label- oder labelKey-Attribut angegeben ist, enthält das Dialogfeld für den Parameter keine Unterbeschriftung.

Experteneinstellungen (ExpertSettings)

Das Element ExpertSettings bestimmt, welche Parameter im Dialogfeld "Algorithmuseinstellungen" dieses Modells eines Ensemble-Modellierungsknotens auf der Registerkarte "Experten" angezeigt werden. Für weitere Informationen siehe Thema Automatisierte Modellierung auf S. 116.

Format

```
<ExpertSettings>
   <Condition ... />
   <PropertyGroup label="group_name" labelKey="resource_key"
    properties="[property1 property2 ...]"/>
   <PropertyGroup ... />
    ...
</ExpertSettings>
```

Das Element Condition gibt eine Bedingung an, die, sofern sie true ist, bewirkt, dass die durch die nachfolgenden PropertyGroup-Elemente angegebenen Parameter aktiviert werden. Für weitere Informationen siehe Thema Bedingungen in Kapitel 4 auf S. 91.

In einem PropertyGroup-Element (mindestens eines ist erforderlich):

label ist eine Anzeigebeschriftung für die Eigenschaftsgruppe, das als Unterbeschriftung in das Dialogfeld oberhalb des ersten Parameters in der Gruppe eingefügt wird.

labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke. Wenn weder label noch labelKey verwendet wird, wird keine Unterbeschriftung eingefügt.

properties (erforderlich) ist eine Liste mit einer oder mehreren Eigenschaften, die auf der Registerkarte angezeigt werden sollen. Der Wert von *prop_name1*, *prop_name2* usw. ist der Wert des Attributs name des Elements Property, in dem diese Eigenschaft definiert wird. Für weitere Informationen siehe Thema Eigenschaften in Kapitel 4 auf S. 66.

Beispiele

Im folgenden Beispiel ist der Parameter Modus auf der Registerkarte “Experten” des Dialogfelds “Algorithmuseinstellungen” zunächst auf Einfach gesetzt:

Abbildung 5-7
Deaktivierte Experteneinstellungen

Der folgende Code legt fest, dass die übrigen Parameter der Registerkarte “Experten” nur dann aktiviert werden, wenn der Benutzer den Parameter Modus auf Experten setzt:

```
<ExpertSettings>
   <Condition property="mode" op="equals" value="Expert"/>
   <PropertyGroup properties="[mode prior_probabilities covariance_matrix]"/>
   ...
</ExpertSettings>
```

Sobald die Einstellung Modus auf Experten gesetzt wird, werden die beiden Parameter der Eigenschaftengruppe aktiviert:

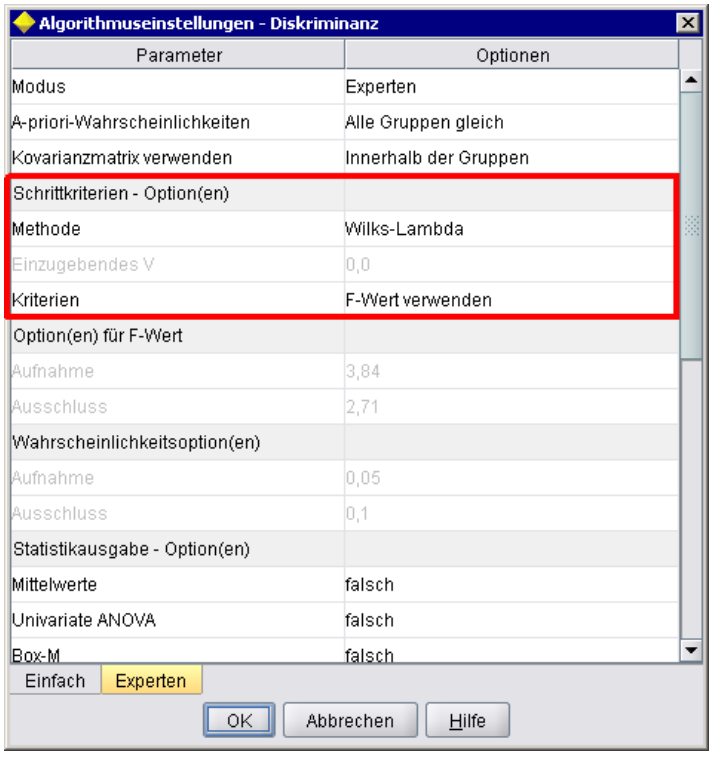
Abbildung 5-8
Aktivierte Experteneinstellungen

Das nächste Beispiel veranschaulicht die Verwendung von Beschriftungen in der Eigenschaftengruppe:

```
<ExpertSettings>
    ...
   <PropertyGroup labelKey="automodel.stepping_criteria_options"
    properties="[stepwise_method V_to_enter criteria]"/>
    ...
</ExpertSettings>
```

Hier steuert das Element PropertyGroup die in der Abbildung hervorgehobenen Parameter:

Abbildung 5-9
Deaktivierte Experteneinstellungen



Das Attribut labelKey veranlasst CLEF, den Anzeigetext für die Unterbeschriftung der Eigenschaftengruppe aus dem entsprechenden Eintrag der Eigenschaftendatei der Erweiterung abzurufen:

```
automodel.stepping_criteria_options=Stepping Criteria option(s)
```

Für weitere Informationen siehe Thema Eigenschaftendateien in Kapitel 8 auf S. 212.

Nebenbedingungen

Das Element Constraint bestimmt die Bedingungen, unter denen die Bearbeitung der Parameter im Dialogfeld “Algorithmuseinstellungen” eines Modells in einem Ensemble-Modellierungsknoten erlaubt ist bzw. unter denen die Bearbeitung eingeschränkt ist. Bestimmte Parameter können z. B. deaktiviert sein, wenn der Endbenutzer diese zurzeit nicht bearbeiten darf.

Format

```
<Constraint property="prop_name" singleSelection="true_false">
   <Condition property="prop_name" op="operator" value="Wert"/>
   <And ... />
   <Or ... />
   <Not ... />
</Constraint>
```

Dabei gilt:

- property (erforderlich) gibt den Parameter an, der bearbeitet werden darf bzw. dessen Bearbeitung eingeschränkt wird. *Eigenschaftenname* ist der Wert des Attributs name des Property-Elements, in dem die zu diesem Parameter gehörige Eigenschaft definiert ist. Für weitere Informationen siehe Thema Eigenschaften in Kapitel 4 auf S. 66.
- singleSelection legt fest, ob der Endbenutzer mehrere der für einen Parameter zur Verfügung stehenden Werte auswählen darf. Wenn diese Einstellung auf true gesetzt ist, darf nur ein Wert ausgewählt werden, selbst wenn das Feld “Optionen” dieses Parameters im Dialogfeld “Algorithmuseinstellungen” mehrere Werte enthält. Wenn diese Einstellung auf false gesetzt ist (Standardeinstellung), kann der Benutzer, wie im nachfolgenden Beispiel gezeigt, ein oder mehrere der verfügbaren Werte auswählen.

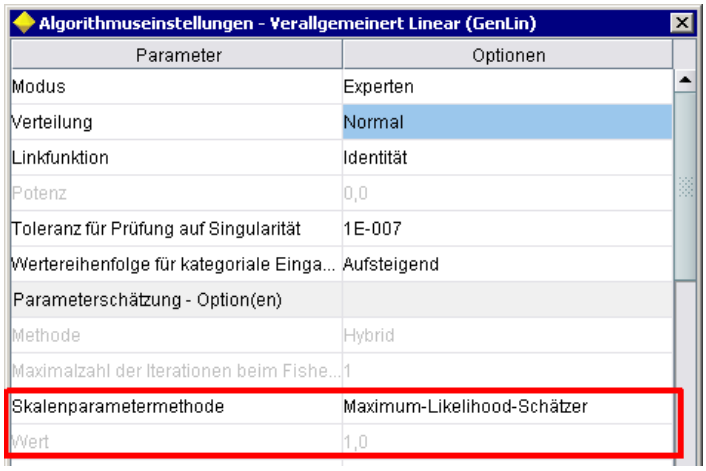
Das Element Condition legt die tatsächliche Einschränkung fest. Für weitere Informationen siehe Thema Bedingungen in Kapitel 4 auf S. 91.

Die Elemente And, Or und Not können zur Angabe zusammengesetzter Bedingungen verwendet werden. Für weitere Informationen siehe Thema Logische Operatoren in Kapitel 4 auf S. 90.

Beispiel

Dieses Beispiel stammt aus der Spezifikationsdatei des verallgemeinerten linearen Knotens. Selbst im Expertenmodus sind in der Standardeinstellung einige Parameter nicht aktiviert.

Abbildung 5-10
Auswirkung einer Einschränkung — deaktivierter Parameter

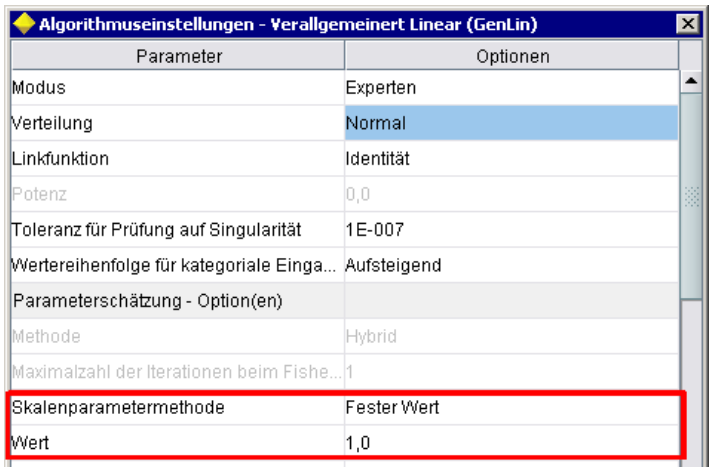


Die Einschränkung legt die Bedingung fest, unter der der Parameter “Wert” aktiviert ist:

```
<Constraint property="scale_value">
   <And>
      <Condition property="scale_method" op="equals" value="FixedValue"/>
      <Condition property="distribution" op="in" value="[IGAUSS GAMMA NORMAL]"/>
   </And>
</Constraint>
```

Damit der Parameter Wert aktiviert wird, muss der Parameter Skalenparametermethode (festgelegt durch die Eigenschaft scale_method) auf Fester Wert gesetzt sein und Verteilung muss auf Normal, Invers normal oder Gamma gesetzt sein.

Abbildung 5-11
Auswirkung einer Einschränkung — aktivierter Parameter



Anwenden von Modellen

Das Anwenden eines Modells bedeutet, ein generiertes Modell für das Scoring von Daten zu verwenden. Dabei werden die aus der Modellerstellung gewonnenen Informationen zur Erstellung von Vorhersagen für neue Datensätze verwendet. In IBM® SPSS® Modeler verwenden Sie dazu einen Modellanwendungsknoten. Für weitere Informationen siehe Thema Modellanwendungsknoten in Kapitel 2 auf S. 14.

Die Definition eines Modellanwendungsknotens in der Spezifikationsdatei bildet das Rahmenwerk für die Anwendung eines generierten Modells. In SPSS Modeler erstellen Sie eine Instanz eines Modellanwendungsknotens, indem Sie das Symbol des Modellausgabeobjekts von der Registerkarte “Modelle” des Managerfensters auf den Stream-Zeichenbereich ziehen. Ohne Definition des Modellanwendungsknotens würde die Ausführung des Modellerstellungsknotens lediglich ein nicht verfeinertes Modell ergeben, das dem Stream-Zeichenbereich nicht hinzugefügt werden kann.

Für die Definition eines Modellanwendungsknotens müssen Sie dem Node-Element folgende Elemente hinzufügen:

- Das Attribut type="modelApplier"
- Das untergeordnete Element Constructors mit einem CreateModelOutput-Element (siehe Verwenden von Konstruktoren auf S. 126)

Das Format des Node-Elements wird im Abschnitt Knoten auf S. 62 beschrieben.

Erstellen von Dokumenten

Für die Definition eines Dokumenterstellungsknotens müssen Sie dem Node-Element folgende Elemente hinzufügen:

- Das Attribut type="documentBuilder"
- Das untergeordnete Element DocumentBuilder

Die Erweiterung muss für die Dokumenterstellung auch über ein DocumentOutput-Element verfügen, das das generierte Dokument beschreibt. Für weitere Informationen siehe Thema Modellausgabe auf S. 110.

Das Format des Node-Elements wird im Abschnitt Knoten auf S. 62 beschrieben.

Dokumentersteller (DocumentBuilder)

Das DocumentBuilder-Element legt das Verhalten eines Dokumenterstellungsknotens fest. Die Definition muss das untergeordnete Element DocumentGeneration enthalten. Dieses Element legt fest, welcher Registerkarte des Dialogfelds des Dokumenterstellungsknotens die Steuerelemente für die Dokumentgenerierung hinzugefügt werden. Die Steuerelemente selbst werden im Abschnitt “User Interface” definiert (siehe Kapitel 6, *Erstellen von Benutzeroberflächen* auf S. 131).

Format

```
<DocumentBuilder>
   <DocumentGeneration controlsId="control_identifier />
</DocumentBuilder>
```

Dabei ist controlsId (erforderlich) die von den Systemsteuerelementen verwendete Kennung, die angibt, wo die Steuerelemente für die Dokumentgenerierung angezeigt werden.

Beispiel

```
<DocumentBuilder>
   <DocumentGeneration controlsId="1"/>
</DocumentBuilder>
```

Dokumentausgabe

Das Element DocumentOutput beschreibt ein Dokumentausgabeobjekt, d. h. ein Objekt, das nach der Ausführung eines Streams auf der Registerkarte “Ausgaben” des Managerfensters angezeigt wird.

Format

```
<DocumentOutput id="identifier" label="display_label" labelKey="label_key" >
   <Properties>
      <Property ... />
      ...
   </Properties>
   <Containers>
      <Container ... />
      ...
   </Containers>
   <UserInterface ... />
   <Constructors ... />
</DocumentOutput>
```

Dabei gilt:

- id (erforderlich) ist eine eindeutige Kennung für das generierte Dokument.
- label (erforderlich) ist der auf der Registerkarte “Ausgaben” angezeigte Name des generierten Dokuments.
- labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke.

Das Element DocumentOutput kann die folgenden untergeordneten Elemente enthalten:

Tabelle 5-4
Untergeordnete Elemente der DocumentOutput-Deklaration

Untergeordnetes Element	Definition	Siehe...
Eigenschaften	Die vom generierten Dokument verwendeten Eigenschaften.	Eigenschaften auf S. 66
Container (Containers)	Die Container für die generierte Dokumentausgabe.	Container (Containers) auf S. 68
UserInterface	Die Komponente der Benutzeroberfläche, in der die generierte Dokumentausgabe angezeigt wird.	Benutzeroberfläche auf S. 69
Konstruktoren	Die vom generierten Dokument erstellten Objekte.	Verwenden von Konstruktoren auf S. 126

Beispiel

```
<DocumentOutput id="webstatusreport">
   <Containers>
      <Container name="webstatusreportdata" />
   </Containers>
   <UserInterface>
      <Tabs>
         <Tab label="Advanced" labelKey="advancedTab.LABEL" >
            <TextBrowserPanel container="webstatusreportdata" textFormat="html" />
         </Tab>
      </Tabs>
   </UserInterface>
</DocumentOutput>
```

Verwenden von Konstruktoren

Constructors-Elemente können an zwei Stellen der Spezifikationsdatei eingefügt werden:

- Im Abschnitt “Execution” einer Knotendefinition (bei Modell- oder Dokumentausgabeobjekten)
- In einer ModelOutput-Definition (bei Modellanwendungsknoten)

Ein Knoten kann nur ein Ausgabeobjekt generieren. Diese Einschränkung wurde aus Konsistenzgründen mit bereits bestehenden Knoten und den Scripting- und Client-API-Schnittstellen für diese Knotentypen beibehalten.

Format

Im Abschnitt “Execution” sieht das Format des Constructors-Elements wie folgt aus:

```
<Constructors>
   <CreateModelOutput ... />
   ...
   <CreateDocumentOutput ... />
   ...
```

```
   <CreateInteractiveModelBuilder ... />
   ...
</Constructors>
```

In einer ModelOutput-Definition sieht das Format wie folgt aus:

```
<Constructors>
   <CreateModelApplier ... />
</Constructors>
```

CreateModelOutput (Erstellen der Modellausgabe)

In diesem Abschnitt wird beschrieben, wie ein Modellausgabeobjekt für die Registerkarte “Modelle” bzw. ein Dokumentausgabeobjekt für die Registerkarte “Ausgaben” erstellt wird.

Format

```
<CreateModelOutput type="output_object_id">
   <Condition ... ./>
   <And>
   <Or>
   <Not>
   <CreateModel type="model_id" target="container_id" sourceFile="container_file_id" />
   ...
   <CreateDocument type="document_id" target="container_id" sourceFile="container_file_id" />
   ...
</CreateModelOutput>
```

Im Element CreateModelOutput:

- type (erforderlich) ist die Kennung eines Modellausgabeobjekts, das in einem ModelOutput-Abschnitt definiert wird. Für weitere Informationen siehe Thema Modellausgabe in Kapitel 4 auf S. 65.

Im Abschnitt Condition können Sie eine oder mehrere Bedingungen festlegen. Für weitere Informationen siehe Thema Bedingungen in Kapitel 4 auf S. 91.

Hier können Sie auch komplexe Bedingungen mit den Operatoren And, Or und Not eingeben. Für weitere Informationen siehe Thema Logische Operatoren in Kapitel 4 auf S. 90.

In den Elementen CreateModel und CreateDocument gilt:

- type ist die Kennung für das zu definierende Modell bzw. Dokument.
- type (erforderlich) ist die Kennung des Containers für das Modell. Dieser Container wird in einem Modellausgabe-Abschnitt definiert. Für weitere Informationen siehe Thema Modellausgabe auf S. 110.
- sourceFile (erforderlich) ist die Kennung einer Ausgabedatei, die während der Knotenausführung generiert wird. Diese Datei wird in einem Abschnitt für Ausgabedateien definiert. Für weitere Informationen siehe Thema Ausgabedateien in Kapitel 4 auf S. 72.

Beispiel

```
<Constructors>
   <CreateModelOutput type="naivebayes">
      <CreateModel type="naivebayes_model" target="model" sourceFile="pmml"/>
      <CreateDocument type="html_output" target="advanced_output" sourceFile="htmloutput" />
      <CreateDocument type="zip_outputType" target="zip_output" sourceFile="zipoutput" />
   </CreateModelOutput>
</Constructors>
```

CreateDocumentOutput (Erstellen der Dokumentausgabe)

Dieses Element wird im Abschnitt “Execution” der Definition eines Dokumenterstellungsknotens verwendet. Es kennzeichnet das zu erstellende Dokumentausgabeobjekt.

Format

```
<CreateDocumentOutput type="output_object_id">
   <Condition ... ./>
   <And>
   <Or>
   <Not>
   <CreateModel type="model_id" target="container_id" sourceFile="container_file_id" />
    ...
   <CreateDocument type="document_id" target="container_id" sourceFile="container_file_id" />
    ...
</CreateModelOutput>
```

Dabei ist type (erforderlich) die Kennung eines Dokumentausgabeobjekts, das in einem DocumentOutput-Abschnitt definiert wird. Für weitere Informationen siehe Thema Dokumentausgabe in Kapitel 4 auf S. 65.

Im Abschnitt Condition können Sie eine oder mehrere Bedingungen festlegen. Für weitere Informationen siehe Thema Bedingungen in Kapitel 4 auf S. 91.

Hier können Sie auch komplexe Bedingungen mit den Operatoren And, Or und Not eingeben. Für weitere Informationen siehe Thema Logische Operatoren in Kapitel 4 auf S. 90.

In den Elementen CreateModel und CreateDocument gilt:

- type ist die Kennung für das zu definierende Modell bzw. Dokument.
- type (erforderlich) ist die Kennung des Containers für das Modell. Dieser Container wird in einem Modellausgabe-Abschnitt definiert. Für weitere Informationen siehe Thema Modellausgabe auf S. 110.
- sourceFile (erforderlich) ist die Kennung einer Ausgabedatei, die während der Knotenausführung generiert wird. Diese Datei wird in einem Abschnitt für Ausgabedateien definiert. Für weitere Informationen siehe Thema Ausgabedateien in Kapitel 4 auf S. 72.

Beispiel

```
<Constructors>
   <CreateDocumentOutput type="webstatusreport">
```

```
        <CreateDocument type="webstatusreport" target="webstatusreportdata"
            sourceFile="webstatusreport_output_file" />
    </CreateDocumentOutput>
</Constructors>
```

CreateInteractiveModelBuilder (Erstellen der interaktiven Modellerstellung)

Dieses Element wird im Abschnitt “Execution” der Definition eines interaktiven Modellerstellungsknotens verwendet. Es kennzeichnet das Ausgabeobjekt, mit dem der Benutzer interagiert. Für weitere Informationen siehe Thema CreateInteractiveModelBuilder (Erstellen der interaktiven Modellerstellung) auf S. 113.

CreateModelApplier (Erstellen des Modellanwenders)

Dieses Element wird in einem Constructors-Element im Abschnitt “ModelOutput” der Definition eines Modellerstellungsknotens verwendet (siehe Modellausgabe auf S. 110). Das Element CreateModelApplier legt fest, wie der Modellanwendungsknoten erstellt wird, wenn ein vom Modellerstellungsknoten generiertes Modellausgabeobjekt auf den Zeichenbereich gezogen wird.

Format

```
<CreateModelApplier type="apply_node_identifier">
   <Condition ... ./>
   <And>
   <Or>
   <Not>
   <CreateModel type="model_id" target="container_id" sourceFile="container_file_id" />
    ...
   <CreateDocument type="document_id" target="container_id" sourceFile="container_file_id" />
    ...
</CreateModelApplier>
```

Im Element CreateModelApplier:

- type (erforderlich) ist die Kennung des zu erstellenden Modellanwendungsknotens. Der Knoten selbst wird erst später im Element Node ... type="modelApplier" dieser Datei definiert.

Im Abschnitt Condition können Sie eine oder mehrere Bedingungen festlegen. Für weitere Informationen siehe Thema Bedingungen in Kapitel 4 auf S. 91.

Hier können Sie auch komplexe Bedingungen mit den Operatoren And, Or und Not eingeben. Für weitere Informationen siehe Thema Logische Operatoren in Kapitel 4 auf S. 90.

In den Elementen CreateModel und CreateDocument gilt:

- type ist die Kennung für das zu definierende Modell bzw. Dokument.

- type (erforderlich) ist die Kennung des Containers für das Modell. Dieser Container wird in einem Modellausgabe-Abschnitt definiert. Für weitere Informationen siehe Thema Modellausgabe auf S. 110.
- sourceFile (erforderlich) ist die Kennung einer Ausgabedatei, die während der Knotenausführung generiert wird. Diese Datei wird in einem Abschnitt für Ausgabedateien definiert. Für weitere Informationen siehe Thema Ausgabedateien in Kapitel 4 auf S. 72.

Beispiel

Im folgenden Beispiel enthält das Element CreateModelApplier einen Vorwärtsverweis auf den Modellanwendungsknoten mit dem Namen myapplier. Dieser Knoten wird im nachfolgenden Node-Element definiert.

```
<ModelOutput>
   <Constructors>
      <CreateModelApplier type="myapplier"></CreateModelApplier>
   </Constructors>
</ModelOutput>
<Node id="myapplier" type="modelApplier">
 ...
</Node>
```

Kapitel 6

Erstellen von Benutzeroberflächen

Info zu Benutzeroberflächen

Wenn Sie einen neuen CLEF-Knoten hinzufügen, müssen Sie definieren, wie der Endbenutzer mit dem Knoten und durch die Erweiterung spezifizierten Modellausgaben, Dokumentausgaben oder Anwendungsknoten interagieren soll. Die Benutzeroberfläche für die einzelnen Objekte ist im Abschnitt UserInterface der Spezifikationsdatei für die Erweiterung definiert. Dieses Kapitel bietet eine detaillierte Beschreibung des Abschnitts UserInterface.

Anmerkung: In einer einzelnen Spezifikationsdatei können mehrere UserInterface-Abschnitte vorhanden sein. Dies hängt davon ab, welche Objekttypen in der Datei definiert sind.

Die Benutzeroberfläche für ein CLEF-Objekt kann folgende Elemente enthalten:

- Menüeintrag
- Paletteneintrag
- Symbole
- Eines oder mehrere Dialogfenster
- Eines oder mehrere Ausgabefenster

Menüeinträge und **Paletteneinträge** ermöglichen den Zugriff auf das Objekt aus dem IBM® SPSS® Modeler-Menüsystem bzw. aus Knotenpaletten. **Symbole** kennzeichnen Objekte in den Menüs, im Zeichenbereich und in den verschiedenen Knotenpaletten. **Dialogfenster** enthalten Registerkarten und Steuerelemente, mit denen Benutzer verschiedene Optionen festlegen können, bevor ein Stream ausgeführt wird, und um optional eine Ausgabe festzulegen, die nach Abschluss der Ausführung erfolgt. **Ausgabefenster** werden verwendet, um Modellausgaben (wie Suchergebnisse, die aus der Anwendung eines Modells auf ein Daten-Set stammen) oder Dokumentausgaben (wie Diagramme) anzuzeigen.

CLEF bietet Ihnen die Möglichkeit, zu den mit SPSS Modeler gelieferten Standardobjekten neue Objekttypen hinzuzufügen und unterstützt einen durchgängigen Ansatz zur Definition der Benutzeroberfläche für diese neuen Objekte. Richtlinien zum Erstellen von Symbolen und Dialogfeldern in CLEF finden Sie unter Erstellen von Knotensymbolen auf S. 18 und Erstellen von Dialogfeldern auf S. 22.

Symbole sind Grafikdateien, mit denen ein bestimmter Knoten identifiziert wird und die im Rahmen der geometrischen Formen dargestellt werden, mit denen der Knotentyp identifiziert wird.

Dialogfelder und **Ausgabefenster** haben folgende Merkmale:

- Titelleiste mit Miniatursymbol und Objekttitel
- Menuleiste mit:
  - Menüs
  - Objektspezifischen Aktionsschaltflächen

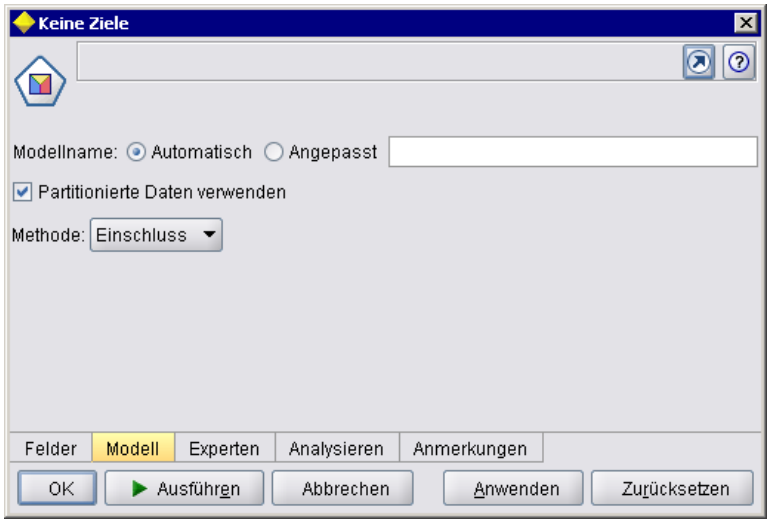
  - Allgemeinen Aktionsschaltflächen (z. B. zum Maximieren oder für die Hilfe)
- Hauptinhaltsbereich
- Mehrere Registerkarten, mit denen Komponenten der Benutzeroberfläche in logischen Gruppen angeordnet werden
- Größenanpassung

Eine Registerkarte ändert den Hauptinhaltsbereich eines Fensters auf verschiedene Weisen. Bei einem Dialogfenster zeigen unterschiedliche Registerkarten verschiedene Sätze von Steuerelementen für die Objekteigenschaften an. Die Steuerelemente können geändert werden und die Ergebnisse werden auf das zugrunde liegende Objekt angewendet, wenn der Benutzer auf die Schaltflächen Anwenden oder OK klickt.

Bei einem Ausgabefenster können Benutzer mit Registerkarten verschiedene mit der Ausgabe zusammenhängende Aktionen durchführen, wie die Anzeige einer Zusammenfassung der Ergebnisse oder das Hinzufügen von Anmerkungen.

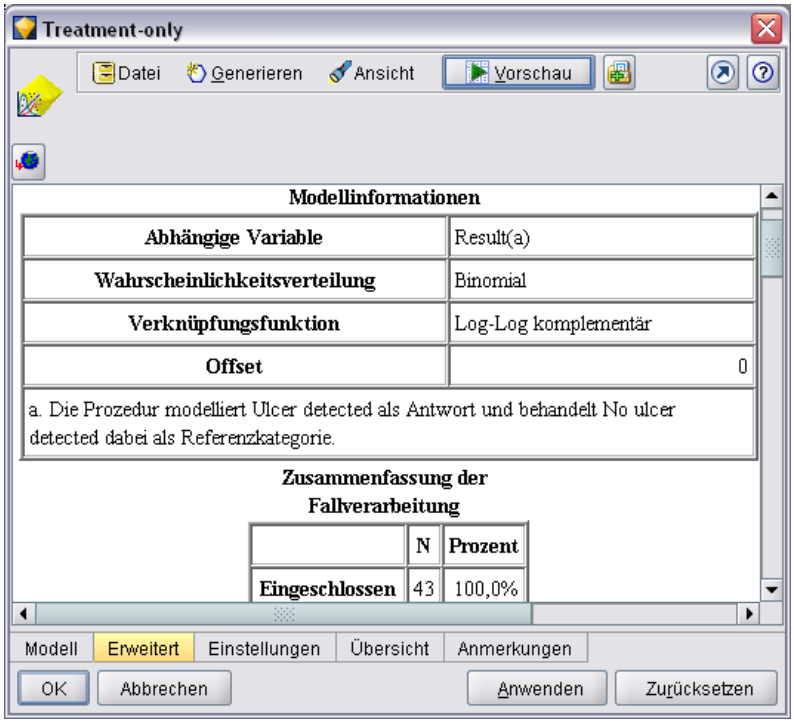
Dies ist ein Beispiel für ein einfaches Dialogfenster mit verschiedenen Registerkarten und Steuerelementen:

Abbildung 6-1
Dialogfeld mit Registerkarten



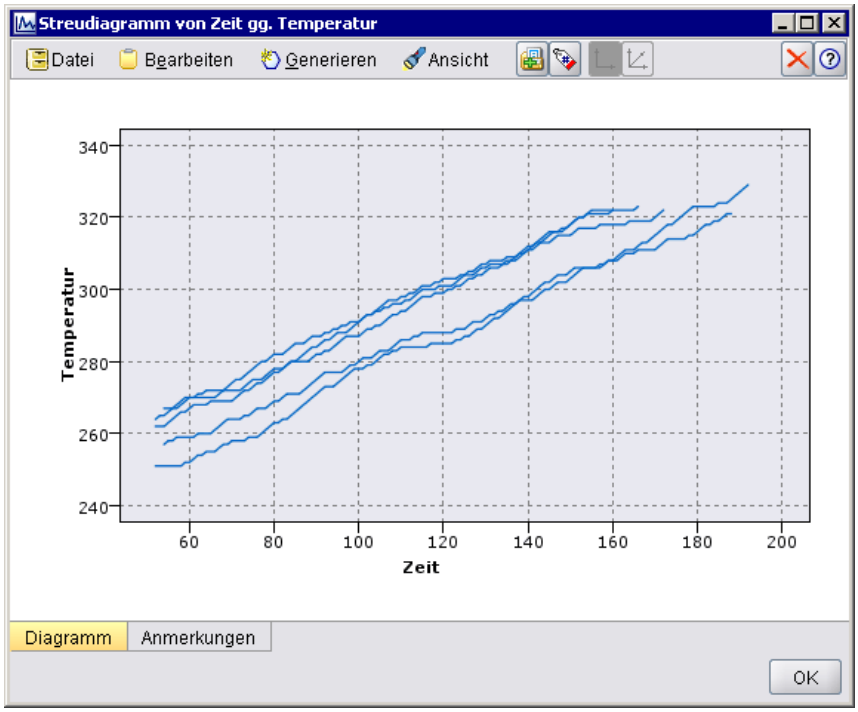
Im Folgenden finden Sie ein Beispiel für ein Modellausgabefenster:

Abbildung 6-2
Ausgabefenster eines Modells



Das letzte Beispiel zeigt ein Dokumentausgabefenster (in diesem Fall ein Diagramm):

Abbildung 6-3
Dokumentausgabefenster



Abschnitt 'Benutzeroberfläche'

Die Benutzeroberfläche für ein Objekt wird innerhalb der Objektdefinition in der Spezifikationsdatei in einem UserInterface-Element deklariert. In einer Spezifikationsdatei können mehrere UserInterface-Elemente vorhanden sein - je nachdem, wie viele Objekte (z. B. Modellierungsknoten, Modellausgabeobjekte, Modellanwendungsknoten) in der Datei definiert sind.

In jedem Benutzeroberflächenabschnitt können folgende Definitionen vorgenommen werden:

- Symbole für die Anzeige im Zeichenbereich oder in Paletten
- Steuerelemente (benutzerdefinierte Menü- und Symbolleistenelemente), die in Dialogfeldern oder Ausgabefenstern angezeigt werden
- Registerkarten, die Sätze von Eigenschaftssteuerelementen definieren (für Dialogfelder oder Ausgabefenster)

Anmerkung: In den folgenden Elementdefinitionen (normalerweise durch die Überschrift **Format** gekennzeichnet) sind Elementattribute und untergeordnete Elemente optional, sofern sie nicht als "(erforderlich)" gekennzeichnet sind. Die vollständige Syntax aller Elemente finden Sie unter Anhang A, *CLEF XML-Schema* auf S. 257.

Für jede Benutzeroberfläche wird die durchzuführende Verarbeitung festgelegt. Dies erfolgt entweder mittels eines Action Handlers oder eines Frame-Klassenattributs, wobei beide optional sind. Wenn weder ein Action Handler noch ein Frame-Klassenattribut angegeben sind, ist die durchzuführende Verarbeitung an anderer Stelle in der Datei spezifiziert.

Format

Das Basisformat des Elements UserInterface sieht wie folgt aus:

```
<UserInterface>
  <Icons>
    <Icon ... />
     ...
  </Icons>
  <Controls>
    <Menu ... />
    <MenuItem ... />
     ...
    <ToolbarItem ... />
     ...
  </Controls>
  <Tabs>
    <Tab ... />
     ...
  </Tabs>
</UserInterface>
```

Ein **Action Handler** wird verwendet, wenn benutzerdefinierte Aktionen zu einem Standard-IBM® SPSS® Modeler-Fenster hinzugefügt werden. Der Action Handler spezifiziert die Java-Klasse, die aufgerufen wird, wenn ein Benutzer in einem Knotendialogfeld, einem Modellausgabefenster oder einem Dokumentausgabefenster eine benutzerdefinierte Menüoption oder Symbolleistenschaltfläche auswählt. Es handelt sich um eine Implementierung einer ExtensionObjectFrame-Klasse oder einer ActionHandler-Klasse. In beiden Fällen werden die Standardfensterkomponenten automatisch eingeschlossen - standardmäßige Menüs, Registerkarten und Symbolleistenschaltflächen. Für weitere Informationen siehe Thema Clientseitige API-Klassen in Kapitel 9 auf S. 221.

Das Format für einen Action Handler ist mit dem Basisformat identisch, mit Ausnahme der ersten Zeile:

```
<UserInterface actionHandler="Java_class" >
   ...
</UserInterface>
```

wobei actionHandler der Name der Action Handler Java-Klasse ist.

Eine **Frame-Klasse** wird für Modellausgabe- und Dokumentausgabeobjekte verwendet, bei denen die Erweiterung ihr eigenes Fenster liefert, anstatt ein standardmäßiges SPSS Modeler-Fenster anzupassen. Eine Frame-Klasse ist eine Java-Klasse, die das gesamte Fenster und dessen Verarbeitung vollständig spezifiziert. Standardfensterkomponenten werden nicht automatisch

aufgenommen — diese müssen in der Klasse einzeln angegeben werden. Frame-Klassen können nur für Modellausgabe- und Dokumentausgabeobjekte verwendet werden und nicht für Knoten, die immer SPSS Modeler-Dialogfelder verwenden. Für weitere Informationen siehe Thema Benutzerdefinierte Ausgabefenster auf S. 204.

Das Format für eine Frame-Klasse ist einfach:

```
<UserInterface frameClass="Java_class" />
```

wobei frameClass der Name der Frame-Klasse für ein Modellausgabe- oder Dokumentausgabeobjekt ist. Alle Symbole, Steuerelemente und Registerkarten werden durch die Frame-Klasse selbst spezifiziert, daher werden diese Elemente in diesem Format nicht verwendet.

Die untergeordneten Elemente eines UserInterface-Elements werden in den folgenden Abschnitten beschrieben.

Beispiele

Das erste Beispiel zeigt die Benutzeroberfläche für einen Modellierungsknoten:

```
<UserInterface actionHandler="com.spss.myextension.MyActionHandler">
  <Icons>
    <Icon type="standardNode" imagePath="images/lg_discriminant.gif" />
    <Icon type="smallNode" imagePath="images/sm_discriminant.gif" />
  </Icons>
  <Tabs defaultTab="1">
   ...
  </Tabs>
</UserInterface>
```

Der zugehörige Abschnitt für ein Modellausgabeobjekt ist:

```
<UserInterface>
  <Icons>
    <Icon type="standardWindow" imagePath="images/browser_discriminant.gif" />
  </Icons>
  <Tabs>
    <Tab label="Erweitert" labelKey="advancedTab.LABEL"
     helpLink="discriminant_output_advancedtab.htm">
      <ExtensionObjectPanel id="DiscriminantPanel"
       panelClass="com.spss.clef.discriminant.DiscriminantPanel"/>
    </Tab>
  </Tabs>
</UserInterface>
```

Der Benutzeroberflächenabschnitt für einen Modellanwendungsknoten sieht wie folgt aus:

```
<UserInterface>
  <Icons>
    <Icon type="standardNode" imagePath="images/lg_gm_discriminant.gif" />
    <Icon type="smallNode" imagePath="images/sm_gm_discriminant.gif" />
  </Icons>
```

```
  <Tabs>
    <Tab label="Erweitert" labelKey="advancedTab.LABEL"
     helpLink="discriminant_output_advancedtab.htm">
      <ExtensionObjectPanel id="DiscriminantPanel"
       panelClass="com.spss.clef.discriminant.DiscriminantPanel"/>
    </Tab>
  </Tabs>
</UserInterface>
```

Symbole

Dieser Abschnitt definiert die dem Objekt zugeordneten Symbole.

Für Knoten sollte dieser Abschnitt zwei Icon-Elemente definieren:

- Eine große Version für die Anzeige im Zeichenbereich
- Eine kleine Version für die Anzeige auf der Palette

Für Modell- und Dokumentausgaben wird damit das Miniatursymbol definiert, das in der oberen linken Ecke des Fensterrahmens angezeigt wird.

Erstellen von Knotensymbolen auf S. 18 bietet Anleitungen zum Erstellen von Symbolen in CLEF.

Format

```
<Icons>
  <Icon type="icon_type" imagePath="image_path" />
  ...
</Icons>
```

Erläuterung:

type (erforderlich) ist einer der folgenden Symboltypen:

Tabelle 6-1
Symboltypen

Typ	Beispiel	Beschreibung
standardNode		Das große Symbol (49 x 49 Pixel), das in der Knotenform des Zeichenbereichs angezeigt wird.
smallNode		Das kleinere Symbol (38 x 38 Pixel), das in der Knotenform auf der Knotenpalette angezeigt wird.
window		Das Miniatursymbol (16 x 16 Pixel), das in einem Browser oder einem Ausgabefenster angezeigt wird.

imagePath (erforderlich) identifiziert den Speicherort des für das Symbol verwendeten Bildes. Der Speicherort wird relativ zu dem Verzeichnis angegeben, in dem die Spezifikationsdatei installiert ist.

Beispiele

```
<Icon type="standardNode" imagePath="images/lg_mynode.gif" />
<Icon type="smallNode" imagePath="images/sm_mynode.gif" />
<Icon type="window" imagePath="images/mynode16.gif" />
```

Steuerelemente

Dieser Abschnitt definiert benutzerdefinierte Menü- und Symbolleistenelemente, mit denen Aktionen aufgerufen werden, die im Abschnitt ‘Gemeinsame Objekte’ der Spezifikationsdatei deklariert sind. Für weitere Informationen siehe Thema Aktionen in Kapitel 4 auf S. 51.

Format

```
<Controls>
   <Menu ... />
    ...
   <MenuItem ... />
    ...
   <ToolbarItem ... />
    ...
</Controls>
```

Die Elemente Menu, MenuItem und ToolbarItem werden in anschließenden Abschnitten beschrieben.

Beispiel

Das folgende Beispiel fügt ein Element zum Menü ‘Generieren’ und zur Symbolleiste des Dialogfelds hinzu, in dessen Spezifizierung es enthalten ist. Beide Elemente implementieren eine zuvor definierte Aktion namens generateDerive, die einen Ableitungskonten generiert.

```
<Controls>
   <MenuItem action="generateDerive" systemMenu="generate"/>
   <ToolbarItem action="generateDerive" showLabel="false"/>
    ...
</Controls>
```

Menüs

Sie können ein benutzerdefiniertes Menü definieren, das zu einem der Standardmenüs hinzugefügt werden soll.

Format

```
<Menu id="name" label="display_label" labelKey="label_key" systemMenu="menu_name"
   showLabel="true_false" showIcon="true_false" separatorBefore="true_false" separatorAfter="true_false"
   offset="integer" mnemonic="mnemonic_char" mnemonicKey="mnemonic_key"/>
```

Erläuterung:

id (erforderlich) ist eine eindeutige ID für das Menü, das hinzugefügt wird.

label (erforderlich) ist der Anzeigename für das Menü, der auf der Benutzeroberfläche angezeigt wird (sofern showLabel auf true gesetzt ist).

labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke.

systemMenu (erforderlich) identifiziert das Standardmenü, zu dem das benutzerdefinierte Menü hinzugefügt wird. Der Wert von *menu_name* hat einen der folgenden Werte:

- file
- edit
- insert*
- view*
- tools*
- window*
- generate
- help*
- file.project
- file.outputs
- file.models
- edit.stream
- edit.node
- edit.outputs
- edit.models
- edit.project
- tools.repository
- tools.options
- tools.streamProperties

* nur gültig beim Hinzufügen zum IBM® SPSS® Modeler-Hauptfenster

showLabel gibt an, ob die Anzeigebeschriftung des Elements auf der Benutzeroberfläche verwendet werden soll.

showIcon gibt an, ob das dem Element zugeordnete Symbol auf der Benutzeroberfläche angezeigt werden soll.

separatorBefore gibt an, ob vor diesem neuen Element ein Trennzeichen (z. B. ein horizontaler Balken für Menüelemente oder ein Leerraum für Symbolleistenschaltflächen) zum Menü hinzugefügt werden soll.

separatorAfter gibt an, ob nach diesem neuen Element ein Trennzeichen zum Menü hinzugefügt werden soll.

offset ist eine nichtnegative ganze Zahl, die die Position des neuen Elements angibt. Ein Offset von 0 beispielsweise fügt das neue Element als erstes Element hinzu (laut Standardeinstellung wird es am Ende hinzugefügt).

mnemonic ist der Buchstabe, der zusammen mit der Alt-Taste gedrückt wird, um diese Steuerung zu aktivieren (wenn z. B. der Wert S angegeben ist, kann der Benutzer diese Steuerung über Alt+S aktivieren).

mnemonicKey kennzeichnet die mnemonische Taste für Lokalisierungszwecke. Wenn weder mnemonic noch mnemonicKey verwendet wird, ist keine Direktzugriffstaste für diese Aktion verfügbar. Für weitere Informationen siehe Thema Zugriffstasten und Tastenkombinationen auf S. 145.

Menüelemente

Sie können ein benutzerdefiniertes Menüelement definieren, das zu einem der Standardmenüs oder zu einem benutzerdefinierten Menü hinzugefügt werden soll.

Format

```
<MenuItem action="identifier" systemMenu="menu_name" customMenu="menu_name"
   showLabel="true_false" showIcon="true_false" separatorBefore="true_false"
   separatorAfter="true_false" offset="integer" />
```

Erläuterung:

action (erforderlich) ist die ID einer Aktion, die im Abschnitt ‘Gemeinsame Objekte’ definiert ist und durch dieses Menüelement ausgeführt wird.

systemMenu legt fest, dass das Element in dem durch *menu_name* angegebenen Standardmenü angezeigt wird, bei dem es sich um eines der folgenden Menüs handelt:

- file
- edit
- insert*
- view*
- tools*
- window*
- generate
- help*
- file.project
- file.outputs
- file.models

- edit.stream
- edit.node
- edit.outputs
- edit.models
- edit.project
- tools.repository
- tools.options
- tools.streamProperties

* nur gültig beim Hinzufügen zum IBM® SPSS® Modeler-Hauptfenster

customMenu ist eine ID von einem Menu-Element, die angibt, dass das Menüelement in dem benutzerdefinierten Menü angezeigt werden soll.

showLabel gibt an, ob die Anzeigebeschriftung des Elements auf der Benutzeroberfläche verwendet werden soll.

showIcon gibt an, ob das dem Element zugeordnete Symbol auf der Benutzeroberfläche angezeigt werden soll.

separatorBefore gibt an, ob vor diesem neuen Element ein Trennzeichen (z. B. ein horizontaler Balken für Menüelemente oder ein Leerraum für Symbolleistenschaltflächen) zum Menü hinzugefügt werden soll.

separatorAfter gibt an, ob nach diesem neuen Element ein Trennzeichen zum Menü hinzugefügt werden soll.

offset ist eine nichtnegative ganze Zahl, die die Position des neuen Elements angibt. Ein Offset von 0 beispielsweise fügt das neue Element als erstes Element hinzu (laut Standardeinstellung wird es am Ende hinzugefügt).

Beispiel

```
<MenuItem action="generateSelect" systemMenu="generate" showIcon="true"/>
```

Symbolleistenelemente

Sie können ein benutzerdefiniertes Symbolleistenelement für ein Dialog- oder Ausgabefenster definieren.

Format

```
<ToolbarItem action="action" showLabel="true_false" showIcon="true_false"
   separatorBefore="true_false" separatorAfter="true_false" offset="integer" />
```

Erläuterung:

action (erforderlich) ist die ID einer Aktion, die im Abschnitt ‘Gemeinsame Objekte’ definiert ist und durch dieses Symbolleistenelement ausgeführt wird.

showLabel gibt an, ob die Anzeigebeschriftung des Elements auf der Benutzeroberfläche verwendet werden soll.

showIcon gibt an, ob das dem Element zugeordnete Symbol auf der Benutzeroberfläche angezeigt werden soll.

separatorBefore gibt an, ob vor diesem neuen Element ein Trennzeichen (z. B. ein horizontaler Balken für Menüelemente oder ein Leerraum für Symbolleistenschaltflächen) zum Menü hinzugefügt werden soll.

separatorAfter gibt an, ob nach diesem neuen Element ein Trennzeichen zum Menü hinzugefügt werden soll.

offset ist eine nichtnegative ganze Zahl, die die Position des neuen Elements angibt. Ein Offset von 0 beispielsweise fügt das neue Element als erstes Element hinzu (laut Standardeinstellung wird es am Ende hinzugefügt).

Beispiel

```
<ToolbarItem action="generateDerive" showLabel="true"/>
```

Beispiel: Hinzufügen zum Hauptfenster

Dies ist ein Beispiel für eine Spezifikationsdatei, die dem Menü “Tools” im Hauptfenster einen neuen Eintrag hinzufügt. Es definiert keine Standardobjekte (z. B. Knoten, Modellausgabefenster oder Dokumentausgabefenster), hat aber ein UserInterface-Element in der Extension der obersten Ebene, was soviel bedeutet, wie ‘ändere das Hauptfenster’. Alle anderen UserInterface-Abschnitte müssen dann in einem der Standard-Objektdefinitionen erscheinen und wirken sich auf Dialogfelder aus, die diesen Objekten zugeordnet sind.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Extension version="1.0">
   <ExtensionDetails providerTag="example" id="main_window" label="Hauptfenster" version="1.0"
      provider="IBM Corp." copyright="(c) 2005-2006 IBM Corp."
      description="Eine Beispielerweiterung, die in das Hauptfenster integriert wird."/>
   <Resources/>
   <CommonObjects>
      <Actions>
         <Action id="customTool1" label="Custom Tool..." labelKey="customTool.LABEL"
            imagePath="images/generate.gif" description="Invokes the custom tool"
            descriptionKey="customTool.TOOLTIP"/>
         <Action id="customTool2" label="Custom Tool..." labelKey="customTool.LABEL"
            imagePath="images/generate.gif" description="Invokes the custom tool"
            descriptionKey="customTool.TOOLTIP"/>
      </Actions>
```

```
  </CommonObjects>
  <UserInterface actionHandler="com.spss.cleftest.MainWindowActionHandler">
    <Controls>
      <Menu systemMenu="tools" id ="toolsExtension" separatorBefore="true"
        label="Extension Items" offset="3"/>
      <MenuItem action="customTool2" customMenu="toolsExtension" showIcon="true"/>
      <MenuItem action="customTool1" systemMenu="file.models" showIcon="true"/>
      <ToolbarItem action="customTool1" offset="0"/>
    </Controls>
  </UserInterface>
</Extension>
```

Diese Anweisungen sorgen dafür, dass ein neues Untermenü namens Extension Items zum Menü ‘Extras’ hinzugefügt wird. Dieses neue Untermenü besitzt einen einzigen Eintrag namens Custom Tool:

Abbildung 6-4
Erweitern der Standardmenüs

Sie können dieses Beispiel ausprobieren, indem Sie den XML-Code in einer Datei mit dem Namen *extension.xml* speichern und dann CLEF die Erweiterung hinzufügen. Für weitere Informationen siehe Thema Testen einer CLEF-Erweiterung in Kapitel 10 auf S. 251.

Registerkarten

Im Abschnitt Tabs werden die Registerkarten definiert, die an folgenden Stellen erscheinen können:

- Dem Dialogfeld, das angezeigt wird, wenn der Benutzer im Zeichenbereich einen Knoten öffnet
- Einem neuen Modellausgabefenster
- Einem Dokumentausgabefenster

Jeder Tabs-Abschnitt kann mehrere Tab-Elemente enthalten, die jeweils eine anzuzeigende benutzerdefinierte Registerkarte deklarieren:

```
<Tabs defaultTab="integer">
  <Tab ... />
  <Tab ... />
   ...
</Tabs>
```

Wobei defaultTab eine positive ganze Zahl ist, die angibt, welche Registerkarte angezeigt wird, wenn das Knotendialogfeld oder das Fenster zum ersten Mal in einem Stream geöffnet wird. Wenn der Benutzer eine andere Registerkarte auswählt, wird beim Schließen und erneuten Öffnen des Dialogfelds oder Fensters während der Stream aktiv ist, statt der Standardregisterkarte die zuletzt angezeigte Registerkarte angezeigt. Die Nummerierung der Registerkarten beginnt bei 0.

Beachten Sie, dass andere Registerkarten automatisch in das Dialogfeld oder Fenster aufgenommen werden können — z. B. enthalten alle Dialogfelder und Ausgabefenster eine Registerkarte Anmerkungen und alle Dialogfelder für Datenquellen enthalten die Registerkarten Filter und Typen.

In einem Tab-Element muss die Registerkartenbeschriftung deklariert sein (der Text, der auf der Registerkarte angezeigt wird). Es sollte außerdem einen Beschriftungsschlüssel enthalten, nach dem gesucht wird, wenn die Registerkartenbeschriftung übersetzt werden soll.

Innerhalb aller Tab-Elemente befindet sich eine Fensterspezifizierung, die definiert, wie der Hauptinhaltsbereich der Registerkarte aufgebaut ist. Die Fensterspezifizierung kann einen der folgenden Typen besitzen: Textbrowser, Erweiterungsobjekt oder Eigenschaften. Für weitere Informationen siehe Thema Fensterspezifizierungen auf S. 147.

Das Format eines einzelnen Tab-Elements lautet:

```
<Tab id="identifier" label="
display_label" labelKey="label_key
" helpLink="help_ID"
      mnemonic="mnemonic_char" mnemonicKey="
mnemonic_key" >
   <TextBrowserPanel ... />
   <ExtensionObjectPanel ... />
   <PropertiesPanel ... />
   <ModelViewerPanel ... />
</Tab>
```

Erläuterung:

id ist eine eindeutige ID, über die die Registerkarte in Java-Code referenziert werden kann.

label (erforderlich) ist der Anzeigename für die Registerkarte, mit dem sie auf der Benutzeroberfläche angezeigt wird.

labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke.

helpLink ist die Kennung für ein Hilfethema, das angezeigt wird, wenn der Benutzer das Hilfesystem aufruft, sofern vorhanden. Das Format der Kennung hängt vom Typ des Hilfesystems ab (siehe Festlegen von Speicherort und Typ des Hilfesystems auf S. 208):

HTML-Hilfe: URL des Hilfethemas
JavaHelp: ID des Themas

mnemonic ist der Buchstabe, der zusammen mit der Alt-Taste gedrückt wird, um diese Steuerung zu aktivieren (wenn z. B. der Wert S angegeben ist, kann der Benutzer diese Steuerung über Alt+S aktivieren).

mnemonicKey kennzeichnet die mnemonische Taste für Lokalisierungszwecke. Wenn weder mnemonic noch mnemonicKey verwendet wird, ist keine Direktzugriffstaste für diese Aktion verfügbar. Für weitere Informationen siehe Thema Zugriffstasten und Tastenkombinationen auf S. 145.

Beispiele

Beispiele zu Tab-Elementen finden Sie in den Abschnitten über die verschiedenen Fensterspezifizierungen, die folgende Inhalte besitzen können: Textbrowserfenster auf S. 147, Erweiterungsobjektfenster auf S. 149, Eigenschaftsfenster auf S. 150 und Modell-Viewer-Fenster auf S. 152.

Zugriffstasten und Tastenkombinationen

Als Alternative für den Mauszugriff auf Funktionen der Benutzeroberfläche können Sie verschiedene Tastenkombinationen angeben, damit Benutzer über die Tastatur auf Funktionen zugreifen können.

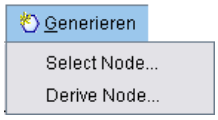
Zugriffstasten

Zugriffstasten sind Tasten, die zusammen mit der ALT-Taste verwendet werden. Für Menüeinträge, Registerkarten und viele andere Dialogsteuerungen können Sie Zugriffstasten durch Verwendung des Attributs mnemonic der folgenden Elemente festlegen.

Feature (Funktion)	Element	Siehe...
Bildschirmaktion (z. B. für einen Menüeintrag)	action	Aktionen auf S. 51
Menü	Menü	Menüs auf S. 138
Registerkarte	tab	Registerkarten auf S. 143
Controller	Verschiedene	Controller auf S. 161

Für die Angabe von Zugriffstasten für Elemente im folgenden Menü:

Abbildung 6-5
Menüelemente



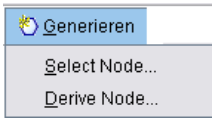
würden Sie folgenden Code verwenden:

```
<Actions>
   <Action id="generateSelect" label="Knoten auswählen..." labelKey="generate.selectNode.LABEL"
      imagePath="images/generate.gif" description="Erzeugt einen Auswahlknoten"
      descriptionKey="generate.selectNode.TOOLTIP" mnemonic="S" />
   <Action id="generateDerive" label="Derive Node..." labelKey="generate.deriveNode.LABEL"
      imagePath="images/generate.gif" description="Erzeugt einen Geräteknoten"
      descriptionKey="generate.deriveNode.TOOLTIP"  mnemonic="D" />
  ...
```

```
</Actions>
```

Damit erhalten Sie die Unterstrich-Zeichen in den folgenden Menüelementen:

Abbildung 6-6
Verwenden von Zugriffstasten mit Menüeinträgen



Benutzer können nun mithilfe von Alt+S und Alt+D und per Mausklick auf die Menüeinträge zugreifen.

Direktzugriffstasten

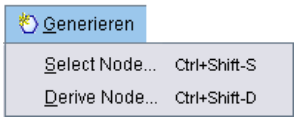
Direktzugriffstasten sind Tastenkombinationen, mit deren Hilfe Benutzer in der Benutzeroberfläche navigieren können. IBM® SPSS® Modeler bietet eine Reihe von Direktzugriffstasten als Standard. In CLEF können Sie Direktzugriffstasten nur für benutzerdefinierte Menüelemente, die Sie hinzugefügt haben, definieren.

Für die Angabe von Direktzugriffstasten für Elemente im Beispiel für Zugriffstasten würden Sie folgenden Code verwenden:

```
<Actions>
   <Action id="generateSelect" label="Knoten auswählen..." labelKey="generate.selectNode.LABEL"
      imagePath="images/generate.gif" description="Erzeugt einen Auswahlknoten"
      descriptionKey="generate.selectNode.TOOLTIP" mnemonic="S" shortcut="CTRL+SHIFT+S" />
   <Action id="generateDerive" label="Derive Node..." labelKey="generate.deriveNode.LABEL"
      imagePath="images/generate.gif" description="Erzeugt einen Geräteknoten"
      descriptionKey="generate.deriveNode.TOOLTIP"  mnemonic="D"  shortcut="CTRL+SHIFT+D" />
   ...
</Actions>
```

Die Tastenkombinationen werden nun den Menüelementen hinzugefügt:

Abbildung 6-7
Verwenden von Direktzugriffstasten mit Menüeinträgen



Benutzer können nun mithilfe der Direktzugriffstasten und per Mausklick auf die Menüeinträge zugreifen. Sie können Direktzugriffstasten wie im Beispiel zusammen mit Tastenkombinationen angeben.

Achten Sie sorgfältig darauf, dass Sie keine Zugriffstasten verwenden, die bereits im selben Dialogfeld angegeben sind oder zu den SPSS Modeler-Standardkombinationen gehören. Für weitere Informationen siehe Thema Tastatureingabehilfen in Anhang A in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Fensterspezifizierungen

Jedes Tab-Element kann die Spezifizierung für ein einzelnes Fenster enthalten, das einen der folgenden Typen besitzen kann:

Tabelle 6-2
Typen von Fensterspezifizierungen

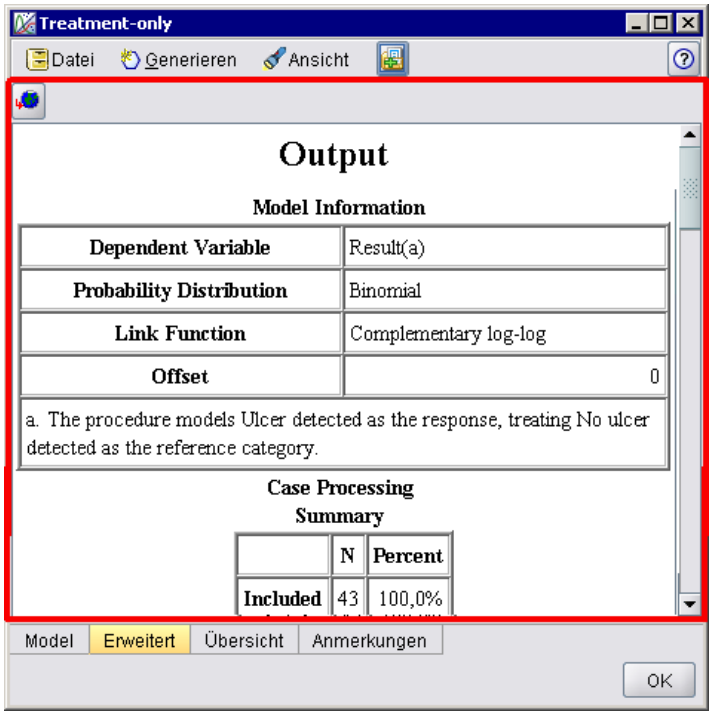
Fenster	Anzeigen...	Siehe...
Textbrowserfenster	Textinhalt eines angegebenen Containers.	Textbrowserfenster auf S. 147
Erweiterungsobjekt-fenster	Durch eine angegeben Java-Klasse definierter Inhalt.	Erweiterungsobjektfenster auf S. 149
Eigenschaftsfenster	Eigenschaftssteuerelemente (z. B. Schaltflächen, Kontrollkästchen, Eingabefelder).	Eigenschaftsfenster auf S. 150
Modell-Viewer-Fenster	Modellausgabe im PMML-Format aus einem festgelegten Container.	Modell-Viewer-Fenster auf S. 152

Textbrowserfenster

Ein Textbrowserfenster zeigt den Textinhalt eines in der Erweiterung angegeben Containers. Unterstützte Formate (UTF-8-Kodierung) sind reiner Text, HTML und Rich Text Format (RTF).

Das folgende Beispiel für ein Modellausgabefenster enthält ein Textbrowserfenster im HTML-Format:

Abbildung 6-8
Modellausgabefenster mit hervorgehobenem Textbrowserfenster



Format

```
<TextBrowserPanel container="name" textFormat="text_format" rows="integer"
   columns="integer" wrapLines="true_false" >
   -- advanced custom layout options --
</TextBrowserPanel>
```

Erläuterung:

container (erforderlich) ist der Name des Containers, aus dem die Inhalte des Fensters erhalten werden.

textFormat (erforderlich) gibt das Format des im Fenster angezeigten Texts an, das folgende Werte besitzen kann:

- plainText
- html
- rtf

rows und columns geben die Anzahl der Textzeilen und -spalten an, die angezeigt werden, wenn das Fenster geöffnet wird.

wrapLines gibt an, ob lange Zeilen umgebrochen werden (true) oder ob ein horizontales Blättern erforderlich ist, um lange Textzeilen zu lesen (false). Die Standardeinstellung ist false.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Das folgende Beispiel zeigt den Registerkartenabschnitt, der das zuvor angezeigte Textbrowserfenster definiert:

```
<Tab label="Advanced" labelKey="advancedTab.LABEL" helpLink="genlin_output_advancedtab.htm">
   <TextBrowserPanel container="advanced_output" textFormat="html" />
</Tab>
```

Die Modellausgabe wird an einen Container gesendet, der in demselben ModelOutput-Abschnitt definiert ist, wie die Registerkartenspezifikation:

```
<ModelOutput id="generalizedlinear" label="Generalized Linear">
   <Containers>
   ...
      <Container name="advanced_output"/>
   </Containers>
   <UserInterface>
   ...
      <Tabs>
         <Tab label="Advanced" ... >
            <TextBrowserPanel container="advanced_output" ... />
```

```
        </Tab>
      </Tabs>
    </UserInterface>
</ModelOutput>
```

Der Container wird in einem CreateDocument-Element im Ausführungsabschnitt für den zuständigen Erstellerknoten referenziert:

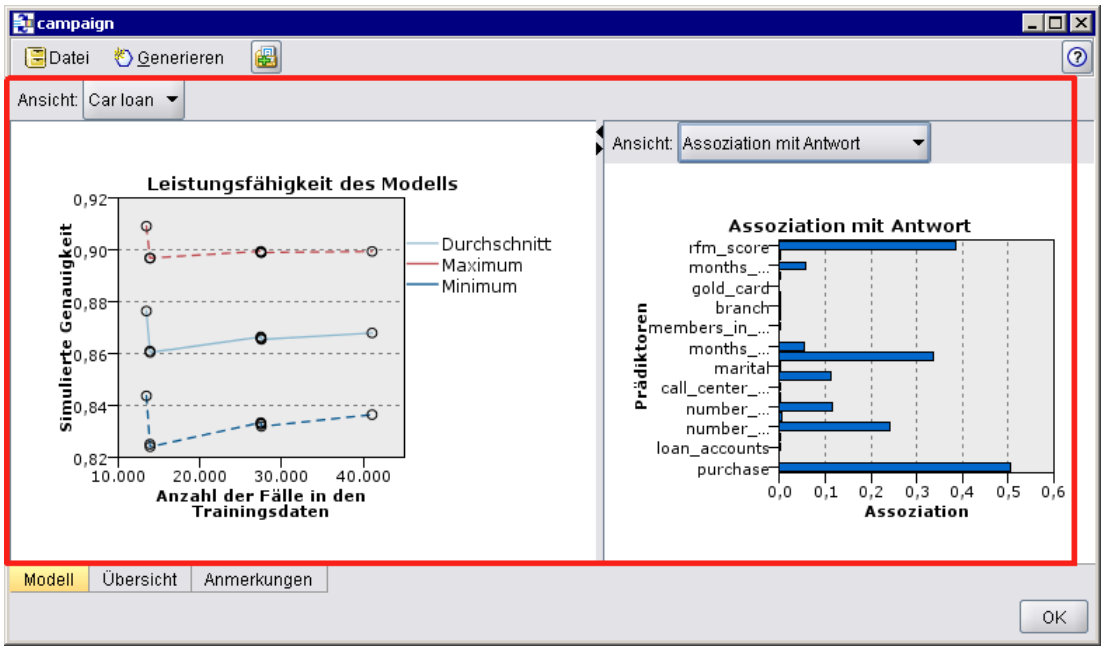
```
<Execution>
 ...
   <Constructors>
      <CreateModelOutput type="generalizedlinear">
         <CreateModel type="generalizedlinear_model" target="model" sourceFile="pmml"/>
         <CreateDocument type="html_output" target="advanced_output" sourceFile="htmloutput"/>
      </CreateModelOutput>
   </Constructors>
</Execution>
```

Erweiterungsobjektfenster

Ein Erweiterungsobjektfenster funktioniert ähnlich wie ein Textbrowserfenster, anstelle der Textinhalte eines Containers erstellt es jedoch eine Instanz der angegebenen Java-Klasse, die die von der CLEF-Java-API definierte ExtensionObjectPanel-Schnittstelle implementiert.

Hier sehen Sie ein Beispiel für das Dialogfeld eines Modellanwendungsknotens, das ein Erweiterungsobjektfenster enthält:

Abbildung 6-9
Modellausgabefenster mit hervorgehobenem Erweiterungsobjektfenster



Format

```
<ExtensionObjectPanel id="identifier" panelClass="Java_class" >
   -- advanced custom layout options --
</ExtensionObjectPanel>
```

Erläuterung:

id ist eine eindeutige ID, über die das Fenster im Java-Code referenziert werden kann.

panelClass (erforderlich) ist der Name der Java-Klasse, die das Fenster implementiert.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Das folgende Beispiel zeigt den Registerkartenabschnitt, der das zuvor angezeigte Erweiterungsobjektfenster definiert:

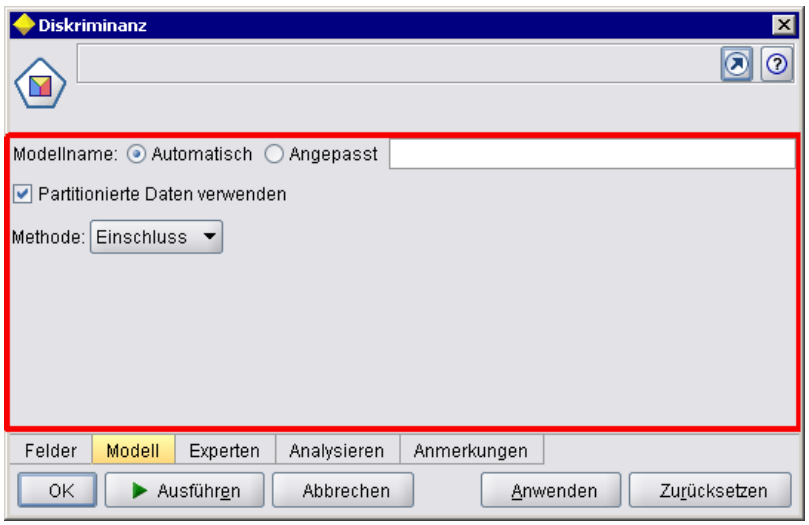
```
<Tab label="Model" labelKey="Model.LABEL" helpLink="selflearnnode_output.htm">
   <ExtensionObjectPanel id="SelfLearningPanel"
      panelClass="com.spss.clef.selflearning.SelfLearningPanel"/>
</Tab>
```

Eigenschaftsfenster

In einem Eigenschaftsfenster können eine Registerkarte oder ein Unterfenster für Eigenschaften angezeigt werden (siehe Eigenschaftsunterfenster auf S. 158), um **Eigenschaftssteuerelemente** anzuzeigen, bei denen es sich um Bildschirmkomponenten handelt (wie Schaltflächen, Kontrollkästchen und Eingabefelder), mit denen die Eigenschaften eines am Bildschirm angezeigten Objekts geändert werden können. Das Eigenschaftsfenster wendet die über diese Steuerelemente vorgenommenen Änderungen automatisch an, wenn der Benutzer auf OK oder auf Anwenden klickt. Wenn der Benutzer auf Abbrechen oder Zurücksetzen klickt, verwirft das Fenster alle Änderungen, die seit dem letzten Anwendungsvorgang durchgeführt wurden.

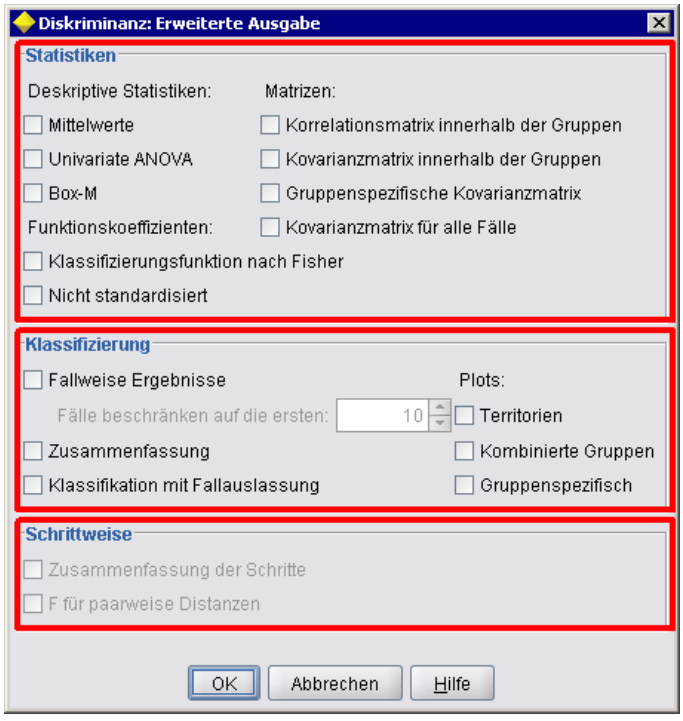
Das folgende Beispiel zeigt ein Knotendialogfeld, das ein Eigenschaftsfenster enthält:

Abbildung 6-10
Knotendialogfeld mit hervorgehobenem Eigenschaftsfenster



Das nächste Beispiel zeigt ein Eigenschaftsunterfenster, das drei Eigenschaftsfenster enthält:

Abbildung 6-11
Eigenschaftsunterfenster mit hervorgehobenen Eigenschaftsfenstern



Format

```
<PropertiesPanel id="identifier" label="display_label" labelKey="label_key">
   -- advanced custom layout options --
   -- property control specifications --
</PropertiesPanel>
```

Erläuterung:

id ist eine eindeutige ID, über die das Fenster im Java-Code referenziert werden kann.

label ist die Anzeigeüberschrift für eine Gruppe von Eigenschaftssteuerelementen (z. B. Statistiken, Klassifizierung und Schrittweise im letzten Beispiel).

labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Eine Beschreibung der Spezifikationen der einzelnen Eigenschaftssteuerelemente finden Sie unter Spezifikationen von Eigenschaftssteuerelementen auf S. 154.

Beispiel

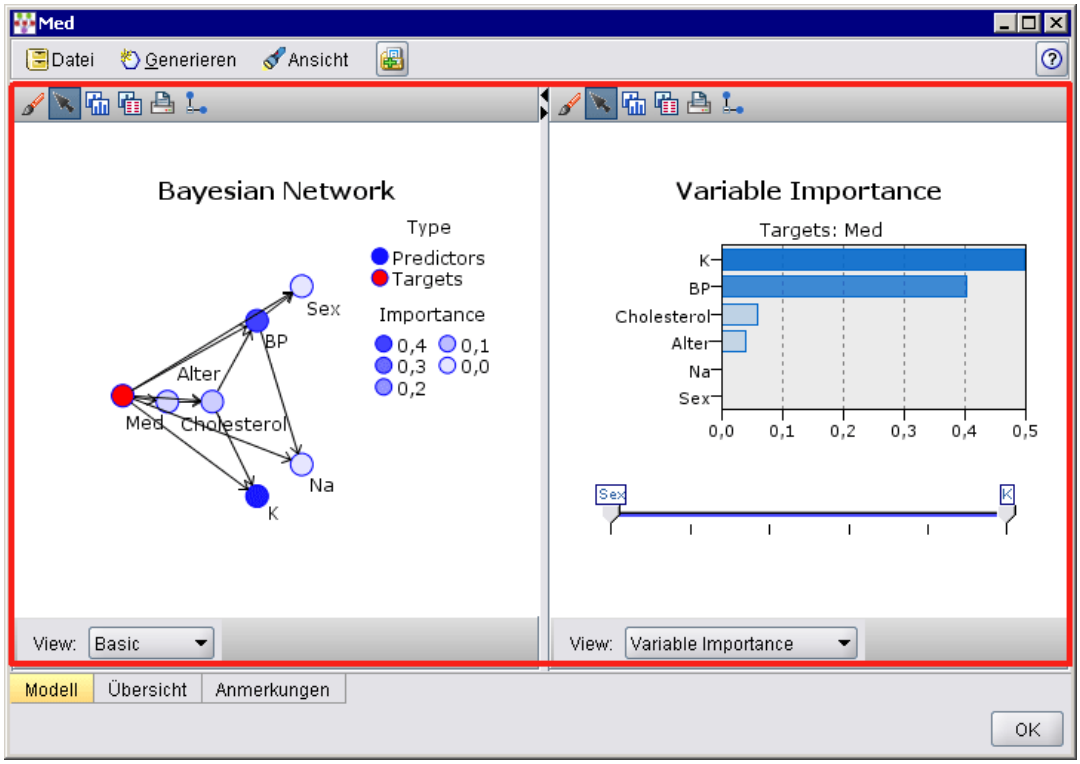
```
<Tab label="Modell" labelKey="Model.LABEL" helpLink="discriminant_node_model.htm">
   <PropertiesPanel>
      <SystemControls controlsId="ModelGeneration" />
      <ComboBoxControl property="method">
         <Layout fill="none" />
      </ComboBoxControl>
   </PropertiesPanel>
</Tab>
```

Modell-Viewer-Fenster

Ein Modell-Viewer-Fenster enthält alle Modellausgaben im PMML-Format eines in der Erweiterung angegebenen Containers.

Das folgende Beispiel zeigt das Fenster eines Modellanwenderknotens, das ein Modell-Viewer-Fenster enthält.

Abbildung 6-12
Ausgabefenster mit hervorgehobenem Modell-Viewer-Fenster



Format

```
<ModelViewerPanel container="container_name">
   -- advanced custom layout options --
</ModelViewerPanel>
```

Dabei ist container (erforderlich) der Name des Containers, dem die Modellausgabe zugewiesen wird.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Dieses Beispiel zeigt die Verwendung eines Modell-Viewer-Fensters in einer Modellanwenderspezifikation. Die Modellausgabe wurde zuvor dem Container namens model zugewiesen. Hier nimmt die Modellanwenderspezifikation den Container und ordnet ihn dem Modell-Viewer-Fenster zu:

```
<Node id="applyBN" type="modelApplier">
   <ModelProvider container="model" isPMML="true" />
    ...
```

```
    <Containers>
      <Container name="model" />
    </Containers>
    <UserInterface>
     ...
    <Tabs>
      <Tab label="Modell" labelKey="modelTab.LABEL" helpLink="BN_output_modeltab.htm">
        <ModelViewerPanel container="model
"/>
      </Tab>
     ...
    </Tabs>
    </UserInterface>
     ...
</Node>
```

Spezifikationen von Eigenschaftssteuerelementen

Eigenschaftssteuerelemente sind Bildschirmkomponenten, wie Schaltflächen, Kontrollkästchen und Eingabefelder, mit deren Hilfe die Eigenschaften eines auf dem Bildschirm angezeigten Objekts geändert werden können. Das Format der Spezifikation eines Eigenschaftssteuerelements hängt vom Typ des Eigenschaftsteuerelements ab, der folgende Ausprägungen haben kann:

- Benutzeroberflächenkomponente
- Eigenschaftsfenster
- Controller

Steuerelemente vom Typ **Benutzeroberflächenkomponente** sind Aktionsschaltflächen, statischer Text in der Anzeige und Systemsteuerelemente (ein Satz von Steuerelementen, die Eigenschaften behandeln, die in allen Dialogfeldern gleich sind).

Steuerelemente vom Typ **Eigenschaftsfenster** sind einzelnen Fenster innerhalb der Spezifikation des Eigenschaftsfensters.

Controller bilden die größte Gruppe der Eigenschaftssteuerelemente. Hierzu gehören Elemente wie Kontrollkästchen, Kombinationsfelder und Drehfelder.

Steuerelemente der Benutzeroberflächenkomponente

Steuerelemente der Benutzeroberflächenkomponente:

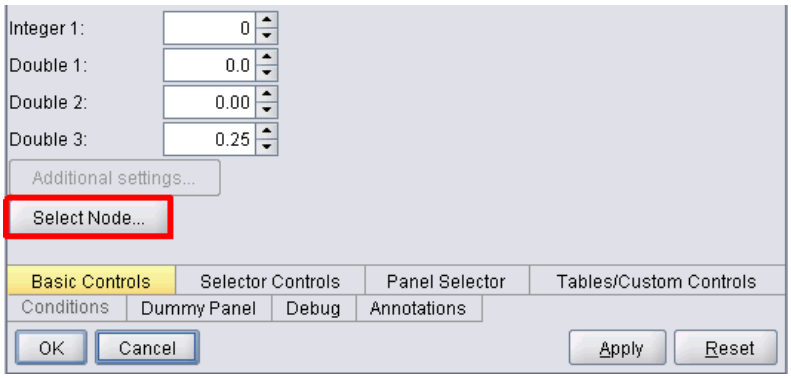
Tabelle 6-3
Steuerelemente der Benutzeroberflächenkomponente

Steuerelement	Beschreibung
ActionButton	Eine Bildschirmschaltfläche, die beim Anklicken eine vordefinierte Aktion ausführt.
StaticText	Eine nicht variierbare Textzeichenfolge, die auf dem Bildschirm angezeigt wird.
SystemControls	Standardsätze von Steuerelementen, die Eigenschaften behandeln, die alle Modelle gemeinsam besitzen.

Aktionsschaltfläche

Definiert eine Dialogfeld- oder Symbolleistenschaltfläche, die eine Aktion ausführt, die im Abschnitt ‘Gemeinsame Objekte’ definiert ist. Die Aktion (z. B. die Anzeige eines neuen Bildschirms) wird durchgeführt, wenn der Benutzer auf diese Schaltfläche klickt.

Abbildung 6-13
Dialogfeld mit hervorgehobener Aktionsschaltfläche



Format

```
<ActionButton action="action" showLabel="true_false" showIcon="true_false" >
   -- advanced custom layout options --
</ActionButton>
```

Erläuterung:

action (erforderlich) ist die ID für die durchzuführende Aktion.

showLabel gibt an, ob die Beschriftung der Schaltfläche angezeigt (true) oder ausgeblendet (false) wird (in einer Symbolleiste wird z. B. anstelle einer Beschriftung eher ein reines Symbol angezeigt). Die Standardeinstellung lautet true.

showIcon gibt an, ob ein der Schaltfläche zugeordnetes Symbol angezeigt (true) oder ausgeblendet (false) werden soll. Die Standardeinstellung ist false.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Der Code für die oben dargestellte Aktionsschaltfläche lautet:

```
<ActionButton action="generateSelect"/>
```

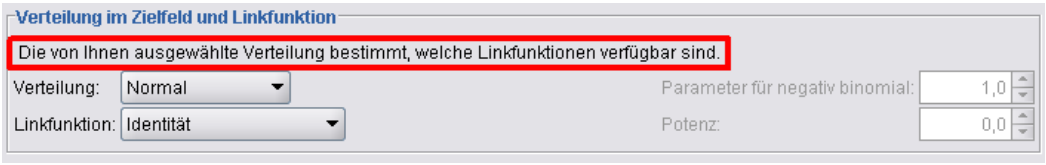
Die Aktion wird wie folgt im Abschnitt ‘Gemeinsame Objekte’ definiert (beachten Sie, dass hier auch die Schaltflächenbeschriftung definiert wird):

```
<CommonObjects extensionListenerClass="com.spss.cleftest.TestExtensionListener">
   ...
  <Actions>
   <Action id="generateSelect" label="Knoten auswählen..." labelKey="generate.selectNode.LABEL"
      imagePath="images/generate.gif" description="Erzeugt einen Auswahlknoten"
      descriptionKey="generate.selectNode.TOOLTIP"/>
    ...
  </Actions>
</CommonObjects>
```

Statischer Text

Dieses Element bietet die Möglichkeit, eine nicht variierbare Textzeichenfolge in ein Dialogfeld oder ein Ausgabefenster einzufügen. Das folgende Beispiel zeigt ein Eigenschaftsfenster, das statischen Text enthält:

Abbildung 6-14
Eigenschaftsfenster mit hervorgehobenem statischen Text



Format

```
<StaticText text="static_text" textKey="text_key" >
   -- advanced custom layout options --
</StaticText>
```

Erläuterung:

text ist die zu verwendende Textzeichenfolge.

textKey kennzeichnet den statischen Text für Lokalisierungszwecke.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

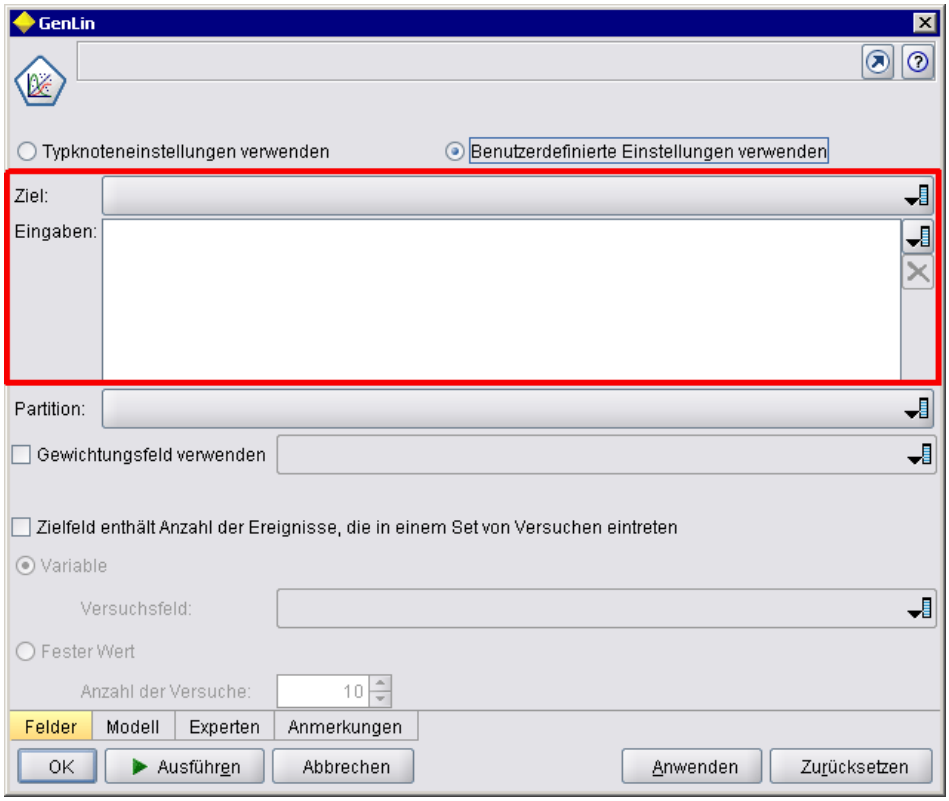
Das folgende Beispiel zeigt die Deklaration, die für den oben gezeigten statischen Text verwendet wird:

```
<StaticText text="The distribution that you choose determines which link functions are available."
   textKey="Genlin_staticText1"/>
```

Systemsteuerelemente

Einige Eigenschaften sind für alle Modelle identisch. In einem Modellierungsknoten handelt es sich bei den Systemsteuerelementen um Standardsätze von Steuerelementen, die diese Eigenschaften behandeln.

Abbildung 6-15
Beispiel für ein Dialogfeld mit hervorgehobenen Systemsteuerelementen



Format

```
<SystemControls controlsID="identifier" >
   -- advanced custom layout options --
</SystemControls>
```

wobei controlsID die ID des Steuerelementsatzes ist. Diese ID muss identisch sein mit der, die im Attribut controlsID eines ModelingFields-Elements in einer Modellierungsdeklaration festgelegt ist (siehe Modellerstellung (ModelBuilder) auf S. 65).

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Dieses Beispiel zeigt die für die Systemsteuerelemente der letzten Abbildung verwendete Deklaration.

Im Modellierungsabschnitt der Knotenspezifikation definieren die folgenden Anweisungen einen Systemsteuerungssatz, zu denen in diesem Fall die Feldauswahllisten für Ein- und Ausgabefelder des Modells gehören:

```
<ModelBuilder ... >
   <ModelingFields controlsId="modelingFields">
      <InputFields property="inputs" multiple="true" label="Inputs" types="[range set orderedSet flag]"
labelKey="inputFields.LABEL"/>
      <OutputFields property="target" multiple="false" types="[range flag]"
label="Target" labelKey="targetField.LABEL"/>
   </ModelingFields>
    ...
</ModelBuilder>
```

Im weiteren Verlauf der Datei wird dieser Steuerelementsatz in der Definition für die Registerkarte des Modellierungsknotendialogfelds referenziert, auf der er angezeigt wird:

```
<Tab label="Fields" labelKey="Fields.LABEL" helpLink="genlin_node_fieldstab.htm">
   <PropertiesPanel>
      <SystemControls controlsId="modelingFields">
      </SystemControls>
    ...
   </PropertiesPanel>
</Tab>
```

Steuerelemente des Eigenschaftsfensters

Steuerelemente der Eigenschaftsfenster:

Tabelle 6-4
Steuerelemente des Eigenschaftsfensters

Steuerelement	Beschreibung
PropertiesSubPanel	Separates Dialogfeld, das angezeigt wird, wenn der Benutzer in einem Eigenschaftsfenster auf eine Schaltfläche klickt.
PropertiesPanel	Eigenschaftsfenster, das in die Deklaration eines Eigenschaftsunterfensters oder in eine Eigenschaftsfensterdeklaration auf oberster Ebene eingebettet ist.

Eigenschaftsunterfenster

Definiert ein separates Dialogfeld, das angezeigt wird, wenn der Benutzer in einem Eigenschaftsfenster auf eine Schaltfläche klickt. Die Deklaration des Eigenschaftsunterfensters erfolgt im Rahmen der Spezifikation des Haupteigenschaftsfensters für eine Registerkarte.

Format

```
<PropertiesSubPanel buttonLabel="display_label" buttonLabelKey="label_key"
   dialogTitle="display_title" dialogTitleKey="title_key" helplink="help_ID"
   mnemonic="mnemonic_char" mnemonicKey="mnemonic_key" >
   -- advanced custom layout options --
   -- property control specifications --
</PropertiesSubPanel>
```

Erläuterung:

buttonLabel ist die Beschriftung der Schaltfläche, die Zugriff auf das Unterfenster gewährt.

buttonLabelKey kennzeichnet die Schaltflächenbeschriftung für Lokalisierungszwecke.

dialogTitle ist der Text, der in der Titelleiste des Unterfensterdialogfelds angezeigt wird.

dialogTitleKey identifiziert den Titel des Dialogfelds des Unterfensters für Lokalisierungszwecke.

helpLink ist die Kennung für ein Hilfethema, das angezeigt wird, wenn der Benutzer das Hilfesystem aufruft, sofern vorhanden. Das Format der Kennung hängt vom Typ des Hilfesystems ab (siehe Festlegen von Speicherort und Typ des Hilfesystems auf S. 208):

HTML-Hilfe: URL des Hilfethemas
JavaHelp: ID des Themas

mnemonic ist der Buchstabe, der zusammen mit der Alt-Taste gedrückt wird, um diese Steuerung zu aktivieren (wenn z. B. der Wert S angegeben ist, kann der Benutzer diese Steuerung über Alt+S aktivieren).

mnemonicKey kennzeichnet die mnemonische Taste für Lokalisierungszwecke. Wenn weder mnemonic noch mnemonicKey verwendet wird, ist keine Direktzugriffstaste für diese Aktion verfügbar. Für weitere Informationen siehe Thema Zugriffstasten und Tastenkombinationen auf S. 145.

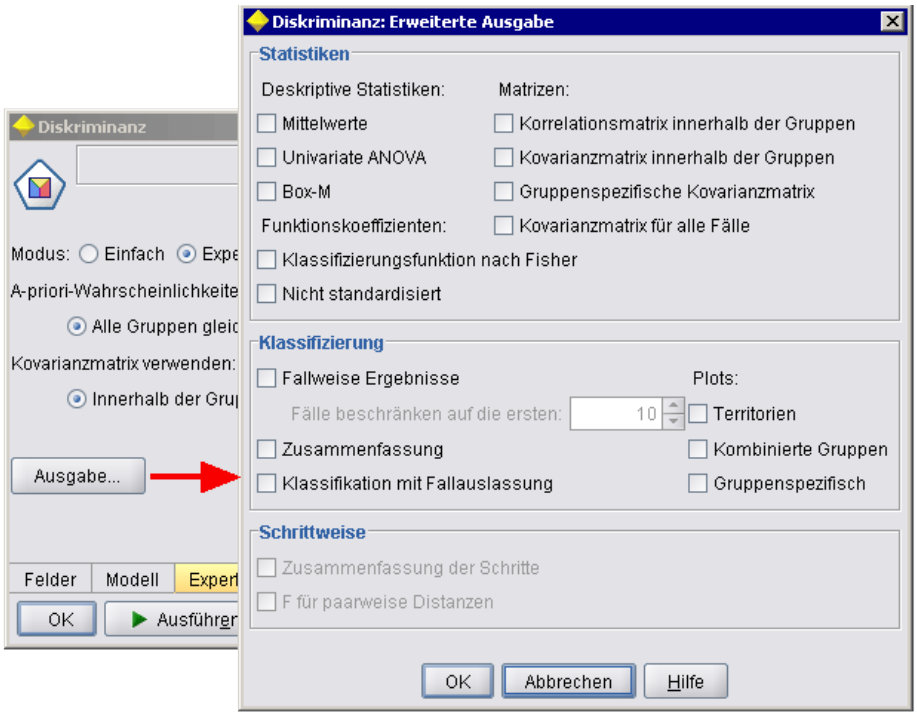
Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Eine Beschreibung der Spezifikationen der einzelnen Eigenschaftssteuerelemente finden Sie unter Spezifikationen von Eigenschaftssteuerelementen auf S. 154.

Beispiel

Das folgende Beispiel zeigt ein Eigenschaftsunterfenster, das angezeigt wird, wenn der Benutzer im Haupteigenschaftsbereich einer Registerkarte auf die Schaltfläche Ausgabe klickt.

Abbildung 6-16
Eigenschaftsunterfenster



Der folgende Code zeigt die entscheidenden Teile der Deklaration, die für das Eigenschaftsunterfenster verwendet werden. Beachten Sie, dass innerhalb der Unterfensterdeklaration jede Feldgruppe (Statistik, Klassifizierung und Schrittweise) ihre eigene Spezifikation des Eigenschaftsfensters hat:

```
<PropertiesSubPanel buttonLabel="Ausgabe..." buttonLabelKey="OutputSubPanel.LABEL"
   dialogTitle="Diskriminanz: Erweiterte Ausgabe" dialogTitleKey="AdvancedOutputSubDialog.LABEL"
   helpLink="discriminant_node_outputdlg.htm">
   ...
   <PropertiesPanel>
      <PropertiesPanel label="Statistik" ... >
       ...
      </PropertiesPanel>
      <PropertiesPanel label="Classification" ... >
       ...
      </PropertiesPanel>
      <PropertiesPanel label="Stepwise" ... >
       ...
      </PropertiesPanel>
   </PropertiesPanel>
</PropertiesSubPanel>
```

Eigenschaftsfenster (verschachtelt)

Sie können eine Eigenschaftsfensterspezifikation in einer Eigenschaftsunterfensterdeklaration verschachteln, um die Inhalte des vom Unterfenster angezeigten Dialogfelds zu definieren. Für weitere Informationen siehe Thema Eigenschaftsunterfenster auf S. 158.

Sie können auch eine Eigenschaftsfensterspezifikation in der Eigenschaftsfensterdeklaration der obersten Ebene verschachteln. Dies bietet sich z. B. dann an, wenn der Inhalt einer gesamten Registerkarte, die aus mehreren Eigenschaftsfenstern besteht, abhängig davon aktiviert oder deaktiviert werden soll, ob eine bestimmte Schaltfläche auf der Registerkarte ausgewählt wird. In diesem Fall sieht die Registerkartenspezifikation etwa wie folgt aus:

```
<Tab .... >
   <PropertiesPanel>
    --- button specification ---
      <PropertiesPanel>
         <Enabled>
          --- condition involving button value ---
         </Enabled>
       ...
      </PropertiesPanel>
      <PropertiesPanel>
         <Enabled>
          --- condition involving button value ---
         </Enabled>
       ...
      </PropertiesPanel>
       ...
   </PropertiesPanel>
</Tab>
```

Das Format einer verschachtelten Eigenschaftsfensterspezifikation ist dasselbe, wie das des Elements auf der obersten Ebene. Für weitere Informationen siehe Thema Eigenschaftsfenster auf S. 150.

Controller

Controller bilden die größte Gruppe der Eigenschaftssteuerelemente:

Tabelle 6-5
Controller

Steuerelement	Beschreibung
CheckBoxControl	Kontrollkästchen.
CheckBoxGroupControl	Satz von Kontrollkästchen, ein Kontrollkästchen pro enum-Wert.
ClientDirectoryChooserControl	Einzeiliges Textfeld und zugehörige Schaltfläche mit der der Benutzer ein Verzeichnis auf dem Client auswählen kann.
ClientFileChooserControl	Einzeiliges Textfeld und zugehörige Schaltfläche mit der der Benutzer eine Datei auf dem Client auswählen kann.

Steuerelement	Beschreibung
ComboBoxControl	Dropdown-Liste eines Kombinationsfelds, die die enum-Werte enthält.
DBConnectionChooserControl	Ermöglicht dem Benutzer die Auswahl einer Datenquelle und das Herstellen einer Verbindung zur Datenbank.
DBTableChooserControl	Ermöglicht dem Benutzer die Auswahl einer Datenbanktabelle, nachdem erfolgreich eine Datenbankverbindung hergestellt wurde.
MultiFieldChooserControl	(Nur Knoten) Liste mit Feldnamen, aus der der Benutzer eines oder mehrere Felder auswählen kann.
MultiItemChooserControl	Ermöglicht dem Benutzer, eines oder mehrere Elemente aus einer Werteliste auszuwählen.
PasswordBoxControl	Einzeilige Textzeile, in der die eingegebenen Zeichen ausgeblendet werden.
PropertyControl	Ein benutzerdefinierbares Steuerelement für eine Eigenschaft.
RadioButtonGroupControl	Satz von Optionsschaltflächen, die gleichzeitig ausgewählt werden können. Für enum-Eigenschaften gibt es eine Optionsschaltfläche pro enum-Wert; für boolesche Eigenschaften werden zwei Optionsschaltflächen angezeigt.
ServerDirectoryChooserControl	Einzeiliges Textfeld und zugehörige Schaltfläche mit der der Benutzer ein Verzeichnis auf dem Server auswählen kann.
ServerFileChooserControl	Einzeiliges Textfeld und zugehörige Schaltfläche mit der der Benutzer eine Datei auf dem Server auswählen kann.
SingleFieldChooserControl	(Nur Knoten) Liste mit Feldnamen, aus der der Benutzer ein einzelnes Feld auswählen kann.
SingleItemChooserControl	Ermöglicht dem Benutzer, ein einzelnes Element aus einer Werteliste auszuwählen.
SpinnerControl	Drehfeld (numerisches Feld mit aufwärts und abwärts weisenden Pfeilen, mit denen der Wert geändert wird).
TableControl	Fügt eine Tabelle zu einem Dialogfeld oder Fenster hinzu.
TextAreaControl	Mehrzeiliges Textfeld.
TextBoxControl	Einzeiliges Textfeld.

Controller-Attribute

Controller-Spezifikationen können folgende Attribute enthalten:

property="*value*" showLabel="*true_false*" label="*display_label*" labelKey="*label_key*" labelWidth="*label_width*" labelAbove="*true_false*" description="*description*" descriptionKey="*description_key*" mnemonic="*mnemonic_char*" mnemonicKey="*mnemonic_key*"

Erläuterung:

property (erforderlich) ist die eindeutige ID des Eigenschaftssteuerelements.

showLabel gibt an, ob die Anzeigebeschriftung des Eigenschaftssteuerelements angezeigt (true) oder ausgeblendet (false) werden soll. Die Standardeinstellung lautet true.

label ist der Anzeigename für das Eigenschaftssteuerelement, der auf der Benutzeroberfläche angezeigt wird. Dieser Wert wird auch als kurze aufrufbare Beschreibung des Steuerelements verwendet. Für weitere Informationen siehe Thema Zugriffsmöglichkeiten in Kapitel 8 auf S. 218.

labelKey kennzeichnet die Beschriftung für Lokalisierungszwecke.

labelWidth ist die Anzahl der anzuzeigenden Rasterspalten, über die sich die Beschriftung erstreckt. Die Standardeinstellung lautet 1.

labelAbove legt fest, ob die Beschriftung für das Steuerelement über (true) oder neben (false) dem Steuerelement angezeigt wird. Die Standardeinstellung ist false.

description ist der Text der QuickInfo, die angezeigt wird, wenn der Mauszeiger auf das Steuerelement bewegt wird. Dieser Wert wird auch als lange aufrufbare Beschreibung des Eigenschaftssteuerelements verwendet. Für weitere Informationen siehe Thema Zugriffsmöglichkeiten in Kapitel 8 auf S. 218.

descriptionKey kennzeichnet die Beschreibung für Lokalisierungszwecke.

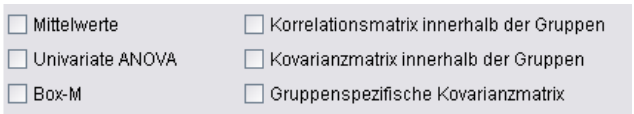
mnemonic ist der Buchstabe, der zusammen mit der Alt-Taste gedrückt wird, um diese Steuerung zu aktivieren (wenn z. B. der Wert S angegeben ist, kann der Benutzer diese Steuerung über Alt+S aktivieren).

mnemonicKey kennzeichnet die mnemonische Taste für Lokalisierungszwecke. Wenn weder mnemonic noch mnemonicKey verwendet wird, ist keine Direktzugriffstaste für diese Aktion verfügbar. Für weitere Informationen siehe Thema Zugriffstasten und Tastenkombinationen auf S. 145.

Kontrollkästchensteuerelement

Definiert ein Kontrollkästchen.

Abbildung 6-17
Kontrollkästchen



Format

```
<CheckBoxControl controller_attributes invert="true_false" >
   -- advanced custom layout options --
</CheckBoxControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

invert wird selten verwendet. Wenn es auf true gesetzt ist, kehrt es den Effekt des Aktivierens bzw. Deaktivierens des Kontrollkästchens um. Die Standardeinstellung ist false.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Das folgende Beispiel zeigt den Code, der verwendet wird, um die oben dargestellten Kontrollkästchen zu gestalten (die Kontrollkästchenbeschriftungen werden an anderer Stelle der Spezifikationsdatei definiert). Eine Beschreibung des Layout-Elements finden Sie unter Erweitertes benutzerdefiniertes Layout auf S. 190.

```
<CheckBoxControl property="means">
   <Layout rowIncrement="0" gridWidth="1"/>
</CheckBoxControl>
<CheckBoxControl property="within_groups_correlation">
   <Layout gridColumn="2" />
</CheckBoxControl>
<CheckBoxControl property="univariate_anovas">
   <Layout gridWidth="1" rowIncrement="0" />
</CheckBoxControl>
<CheckBoxControl property="within_group_covariance">
   <Layout gridColumn="2" />
</CheckBoxControl>
<CheckBoxControl property="box_m">
   <Layout gridWidth="1" rowIncrement="0" />
</CheckBoxControl>
<CheckBoxControl property="separate_groups_covariance">
   <Layout gridColumn="2" />
</CheckBoxControl>
```

Gruppensteuerelement für Kontrollkästchen

Definiert einen Satz von Kontrollkästchen, die gruppiert und als eine Einheit behandelt werden. Diese Option kann nur zusammen mit einer aufgezählten Listeneigenschaft verwendet werden, die die Mitglieder der Gruppe definiert.

Abbildung 6-18
Kontrollkästchengruppe

Format

```
<CheckBoxGroupControl controller_attributes rows="integer" layoutByRow="true_false"
   useSubPanel="true_false" >
   -- advanced custom layout options --
</CheckBoxGroupControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

rows ist eine positive ganze Zahl, die die Anzahl der Zeilen auf dem Bildschirm angibt, die von der Kontrollkästchengruppe belegt werden. Die Standardeinstellung lautet 1.

layoutByRow gibt an, ob die Kontrollkästchen zuerst innerhalb der Zeile (true) oder in der Spalte (false) angeordnet werden. Die Standardeinstellung lautet true. Informationen zur Verwendung von layoutByRow mit einer Optionsschaltflächengruppe finden Sie unter Ändern der Reihenfolge der Steuerelemente auf S. 190.

useSubPanel gibt an, ob die Kontrollkästchen als Unterfenster angeordnet (true) werden oder nicht (false). Die Standardeinstellung lautet true.

Kontrollkästchengruppen werden normalerweise als Unterfenster angeordnet, in denen alle Kästchen der Gruppe enthalten sind. Dies kann jedoch zu Ausrichtungsproblemen führen, wenn das Kontrollkästchen zu einem daneben liegenden Textfeld gehört. Wenn useSubPanel auf false gesetzt ist, wird dieses Problem vermieden.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Der Code für die oben dargestellte Kontrollkästchengruppe lautet:

```
<CheckBoxGroupControl property="enum3" label="Enum 3" labelKey="enum3.LABEL"/>
```

Die den einzelnen Kontrollkästchen zugeordneten Beschriftungen und Werte sind im Abschnitt ‘Eigenschaften’ des entsprechenden Knotens definiert:

```
<Property name="enum3" valueType="enum" isList="true" defaultValue="[value1 value3]">
   <Enumeration>
      <Enum value="value1" label="Wert 3.1" labelKey="enum3.value1.LABEL"/>
      <Enum value="value2" label="Wert 3.2" labelKey="enum3.value2.LABEL"/>
      <Enum value="value3" label="Wert 3.3" labelKey="enum3.value3.LABEL"/>
      <Enum value="value4" label="Wert 3.4" labelKey="enum3.value4.LABEL"/>
      <Enum value="value5" label="Wert 3.5" labelKey="enum3.value5.LABEL"/>
   </Enumeration>
</Property>
```

Auswahlsteuerelement für das Client-Verzeichnis

Definiert ein einzeiliges Textfeld und die zugehörige Schaltfläche, mit der der Benutzer ein Verzeichnis auf dem Client auswählen kann. Das Verzeichnis muss bereits vorhanden sein. Benutzer können in diesem Verzeichnis entweder eine Datei öffnen oder speichern. Dies hängt vom eingestellten Modus ab.

Abbildung 6-19
Steuerelement zur Client-Verzeichnisauswahl

Der Benutzer kann entweder den Verzeichnispfad und den Namen direkt in das Textfeld eingeben oder auf die daneben liegende Schaltfläche klicken, um ein Dialogfeld zu öffnen, in dem er ein Verzeichnis auswählen kann.

Format

```
<ClientDirectoryChooserControl controller_attributes mode="chooser_mode" >
   -- advanced custom layout options --
</ClientDirectoryChooserControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

mode legt die Schaltfläche fest, die im Dialogfeld angezeigt wird, in dem der Benutzer ein Verzeichnis auswählt. Folgende Optionen sind verfügbar:

- open (Standard) zeigt die Schaltfläche Öffnen an.
- save zeigt die Schaltfläche Speichern an.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

```
<ClientDirectoryChooserControl property="directory2" label="Client-Verzeichnis"
   labelKey="directory2.LABEL"/>
```

Auswahlsteuerelement für Client-Dateien

Definiert ein einzeiliges Textfeld und die zugehörige Schaltfläche, mit der der Benutzer eine Datei auf dem Client auswählen kann. Die Datei muss bereits vorhanden sein. Benutzer können die Datei entweder öffnen oder speichern. Dies hängt vom festgelegten Modus ab.

Abbildung 6-20
Element für die Client-Dateiauswahl

Der Benutzer kann entweder den Dateipfad und den Namen direkt in das Textfeld eingeben oder auf die daneben liegende Schaltfläche klicken, um ein Dialogfeld zu öffnen, in dem er eine Datei auswählen kann.

Format

<ClientFileChooserControl *controller_attributes* mode="*chooser_mode*" >
  -- *advanced custom layout options* --
</ClientFileChooserControl>

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

mode legt die Schaltfläche fest, die im Dialogfeld angezeigt wird, in dem der Benutzer eine Datei auswählt. Folgende Optionen sind verfügbar:

- open (Standard) zeigt die Schaltfläche Öffnen an.
- save zeigt die Schaltfläche Speichern an.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

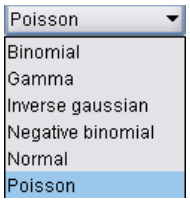
Beispiel

```
<ClientFileChooserControl property="file2" label="Client File" labelKey="file2.LABEL"/>
```

Kombinationsfeldsteuerelement

Definiert eine Dropdown-Liste für ein Kombinationsfeld.

Abbildung 6-21
Kombinationsfeld



Format

<ComboBoxControl *controller_attributes* >
  -- *advanced custom layout options* --
</ComboBoxControl>

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Das folgende Beispiel zeigt den Code, der für die Anordnung der in der letzten Abbildung gezeigten Dropdown-Liste des Kombinationsfelds verwendet wird:

```
<ComboBoxControl property="distribution" >
   <Layout rowIncrement="0" gridWidth="1" fill="none"/>
</ComboBoxControl>
```

Eine Beschreibung des Layout-Elements finden Sie unter Erweitertes benutzerdefiniertes Layout auf S. 190

Anmerkung: Die eigentlichen Listeneinträge werden im Abschnitt ‘Eigenschaften’ des entsprechenden Knotens definiert; in diesem Fall als aufgezählte Liste in der Deklaration für die distribution-Eigenschaft:

```
<Property name="distribution" valueType="enum" label="Verteilung" labelKey="distribution.LABEL"
defaultValue="NORMAL">
   <Enumeration>
      <Enum value="BINOMIAL" label="Binomial" labelKey="distribution.BINOMIAL.LABEL"/>
      <Enum value="GAMMA" label="Gamma" labelKey="distribution.GAMMA.LABEL"/>
      <Enum value="IGAUSS" label="Inverse gaussian" labelKey="distribution.IGAUSS.LABEL"/>
      <Enum value="NEGBIN" label="Negative binomial" labelKey="distribution.NEGBIN.LABEL"/>
      <Enum value="NORMAL" label="Normal" labelKey="distribution.NORMAL.LABEL"/>
      <Enum value="POISSON" label="Poisson" labelKey="distribution.POISSON.LABEL"/>
   </Enumeration>
</Property>
```

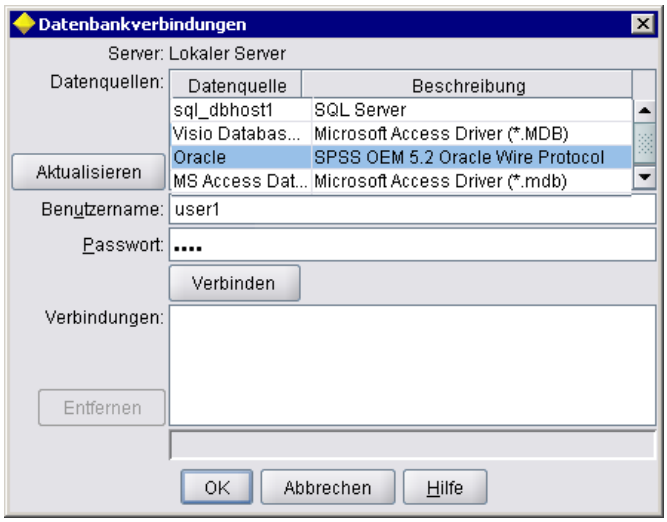
Steuerelement für die Auswahl der Datenbankverbindung

Definiert ein Steuerelement, das es dem Benutzer ermöglicht, eine Datenquelle auszuwählen und eine Verbindung zur Datenbank herzustellen.

Abbildung 6-22
Steuerelement für die Auswahl der Datenbankverbindung

Benutzer können keinen Text in das Textfeld eingeben — sie müssen stattdessen auf die Schaltfläche klicken, damit das standardmäßige IBM® SPSS® Modeler-Dialogfeld für Datenbankverbindungen angezeigt wird:

Abbildung 6-23
Datenbankverbindungen (Dialogfeld)



Bei einer erfolgreichen Verbindung werden die Details der Verbindung im Textfeld des Steuerelements für die Datenbankauswahl angezeigt.

Format

```
<DBConnectionChooserControl controller_attributes >
   -- advanced custom layout options --
</DBConnectionChooserControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Das folgende Beispiel illustriert, wie das Steuerelement die Definition einer Zeichenketteneigenschaft benötigt, die für die Verbindungszeichenfolge verwendet werden kann.

```
<Node ... >
   <Properties>
   ...
      <Property name="dbconnect" valueType="databaseConnection" />
   </Properties>
   ...
```

```
    <UserInterface>
     ...
       <Tabs>
          <Tab label="Database">
             <PropertiesPanel>
                <DBConnectionChooserControl property="dbconnect" label="Datenbankverbindung"/>
              ...
             </PropertiesPanel>
              ...
    </UserInterface>
</Node>
```

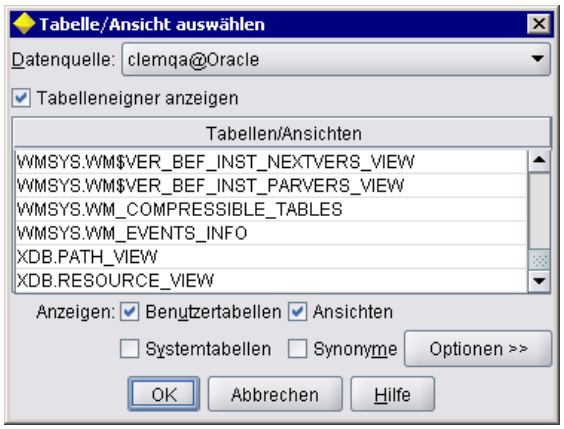
Steuerelement für die Auswahl der Datenbanktabelle

Definiert ein Textfeld und die zugehörige Schaltfläche, mit der ein Benutzer im Anschluss an eine erfolgreiche Datenbankverbindung eine Datenbanktabelle auswählen kann.

Abbildung 6-24
Steuerelement für die Auswahl der Datenbanktabelle

Benutzer können den Tabellennamen entweder direkt in das Textfeld eingeben oder auf die Schaltfläche klicken und ihn aus einer Liste auswählen.

Abbildung 6-25
Datenbanktabellenliste



Format

```
<DBTableChooserControl connectionProperty="DB_connection_property" controller_attributes >
   -- advanced custom layout options --
</DBTableChooserControl>
```

Erläuterung:

connectionProperty ist der Name einer bereits definierten Datenbankverbindungseigenschaft. Dies ist der Wert des property-Attributs des DBConnectionChooserControl-Elements, das zuvor für den Knoten definiert wurde.

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Dieses Beispiel schließt an das für DBConnectionChooserControl an und zeigt wie Sie außerdem ein DBTableChooserControl-Element aufnehmen können, um nach dem erfolgreichen Herstellen einer Datenbankverbindung eine Datenbanktabelle auszuwählen.

```
<Node ... >
   <Properties>
    ...
      <Property name="dbconnect" valueType="databaseConnection" />
      <Property name="dbtable" valueType="string" />
   </Properties>
    ...
   <UserInterface>
    ...
      <Tabs>
         <Tab label="Database">
            <PropertiesPanel>
               <DBConnectionChooserControl property="dbconnect" label="Database connection"/>
               <DBTableChooserControl property="dbtable" connectionProperty="dbconnect"
                  label="Datenbanktabelle" />
             ...
            </PropertiesPanel>
    ...
   </UserInterface>
</Node>
```

Steuerelement für die Auswahl mehrerer Felder

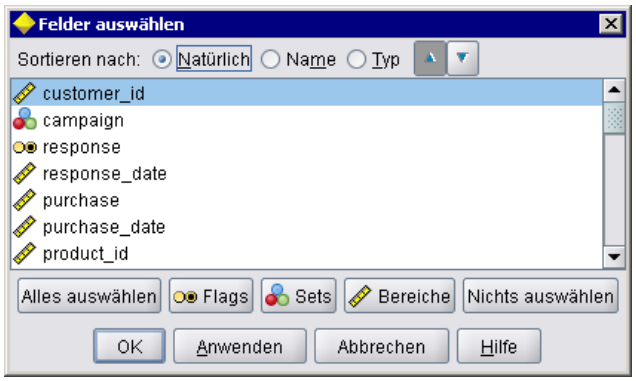
Definiert ein Steuerelement, mit dem der Benutzer einen oder mehrere Feldnamen aus einer Liste auswählen kann.

Abbildung 6-26
Steuerelement für die Auswahl mehrerer Felder

Wenn der Benutzer auf dieses Steuerelement klickt, wird eine Liste mit Feldern angezeigt, die der Benutzer auswählen kann.

Das Set besteht aus allen Feldern, die an diesem Knoten sichtbar sind. Wenn Felder oberhalb dieses Knotens gefiltert wurden, sind nur die Felder, die den Filter durchlaufen haben, sichtbar. Die Liste kann noch weiter eingeschränkt werden, indem angegeben wird, dass nur Felder mit bestimmten Speicher- und Datentypen zur Auswahl stehen sollen.

Abbildung 6-27
Liste mit mehreren Feldern



Alle Steuerelemente für die Auswahl mehrerer Felder legen ein Eigenschaftsattribut fest, das an anderer Stelle in der Datei deklariert ist und definiert wie die Liste im Knotendialogfeld angezeigt wird.

Format

<MultiFieldChooserControl *controller_attributes* storage="*storage_types*" onlyNumeric="*true_false*"
onlySymbolic="*true_false*" onlyDatetime="*true_false*" types="*data_types*"
onlyRanges="*true_false*" onlyDiscrete="*true_false*" >
-- *advanced custom layout options* --
</MultiFieldChooserControl>

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

Sie können die Feldliste weiter einschränken, indem Sie außerdem zwei weitere Attribute angeben, von denen eines aus der folgenden Liste stammen muss:

- storage ist eine Listeneigenschaft, die den Speichertyp der Felder angibt, die in der Liste zulässig sein sollen. So bedeutet storage="[integer real]", dass nur Felder mit diesen Speichertypen aufgelistet werden. Die Menge der möglichen Speichertypen können Sie der Tabelle unter Daten- und Speichertypen auf S. 230 entnehmen.
- onlyNumeric (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit numerischem Speichertyp aufgelistet werden sollen.
- onlySymbolic (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit symbolischem Speichertyp (also Zeichenketten) aufgelistet werden sollen.
- onlyDatetime (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit einem Speichertyp für Datum und Uhrzeit aufgelistet werden sollen.

Das zweite angegebene Attribut muss aus dieser Liste stammen:

- types ist eine Listeneigenschaft, die den Datentyp der Felder angibt, die in der Liste zulässig sein sollen. So bedeutet types="[range flag]", dass nur Felder mit diesen Speichertypen aufgelistet werden. Folgende Datentypen sind möglich:

  Bereich

  Flag

  set

  orderedSet

  Numerisch

  discrete

  typeless
- onlyRanges (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit dem Datentyp "Bereich" aufgelistet werden sollen.
- onlyDiscrete (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit diskretem Datentyp (also Flag, Set oder ohne Typ) aufgelistet werden sollen.

So stellt beispielsweise ein Steuerelement mit der Angabe storage="[integer]" und types="[flag]" sicher, dass nur ganzzahlige Felder, bei denen es sich um Flags handelt, in der Liste angezeigt werden.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Anmerkung: Dieses Steuerelement wird nur in Knotenelementdefinitionen verwendet. Verwenden Sie folgendes Format, wenn Sie ein Auswahlelement für mehrere Felder ohne Ausgabedatenmodelldefinition festlegen möchten:

```
<OutputDataModel mode="mode">
 ...
   <ForEach var="field" inProperty="prop_name">
      <AddField name="${field_name}_NEW" fieldRef="${field_name}" />
   </ForEach>
 ...
</OutputDataModel>
```

Für weitere Informationen siehe Thema Ausgabedatenmodell (OutputDataModel) in Kapitel 4 auf S. 75. Eine Beschreibung des ForEach-Elements finden Sie unter Iteration mit dem Element 'ForEach' auf S. 85. Eine Beschreibung zu AddField finden Sie unter Feld hinzufügen auf S. 82.

Beispiel

Das folgende Beispiel zeigt den Code, mit dem das Auswahlelement für mehrere Felder aus der letzten Abbildung spezifiziert wird.

```
<MultiFieldChooserControl property="inputs" >
   <Enabled>
      <Condition control="custom_fields" op="equals" value="true"/>
   </Enabled>
</MultiFieldChooserControl>
```

Der Abschnitt Enabled sorgt dafür, dass das Steuerelement nur dann aktiviert ist, wenn das Steuerelement custom_fields ausgewählt ist.

Anmerkung: Der Inhalt dieser Liste wird durch die Deklaration für die Eigenschaft inputs im Abschnitt ‘Eigenschaften’ des entsprechenden Knotens bestimmt:

```
<Property name="inputs" valueType="string" isList="true" label="Inputs" labelKey="inputs.LABEL"/>
```

Steuerelement für die Auswahl mehrerer Elemente

Definiert ein Steuerelement, mit dem der Benutzer eines oder mehrere Elemente aus einer Werteliste auswählen kann. Es verknüpft eine Eigenschaft mit einem Katalog, der eine Werteliste enthält. Für weitere Informationen siehe Thema Kataloge in Kapitel 4 auf S. 52.

Abbildung 6-28
Steuerelement für die Auswahl mehrerer Elemente

Format

```
<MultiItemChooserControl controller_attributes catalog="catalog_name" >
   -- advanced custom layout options --
</MultiItemChooserControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

catalog (erforderlich) ist der Name für den zu verknüpfenden Katalog. Die Bibliothek, aus der der Katalog bezogen wird, wird im Element Module des Abschnitts “Execution” angegeben. Für weitere Informationen siehe Thema Module in Kapitel 4 auf S. 72.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

```
<MultiItemChooserControl property="selection2" catalog="cat2" />
```

Die vom Attribut property (selection2 in diesem Fall) referenzierte Eigenschaft muss über ein Attribut isList="true" verfügen. Eine Erklärung und ein Beispiel für die Verwendung von MultiItemChooserControl finden Sie unter Kataloge auf S. 52.

Steuerelement für das Passwortfeld

Definiert ein einzeiliges Textfeld, in dem Zeichen bei der Eingabe ausgeblendet werden.

Abbildung 6-29
Steuerelement für das Passwortfeld

Format

```
<PasswordBoxControl controller_attributes columns="integer" >
   -- advanced custom layout options --
</PasswordBoxControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

columns ist eine positive ganze Zahl, die die Anzahl der Zeichenspalten definiert, die das Passwortfeld belegt. Die Standardeinstellung lautet 20.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

```
<PasswordBoxControl property="encrypted_string1" label="Encrypted string
1" labelKey="encryptedString1.LABEL"/>
```

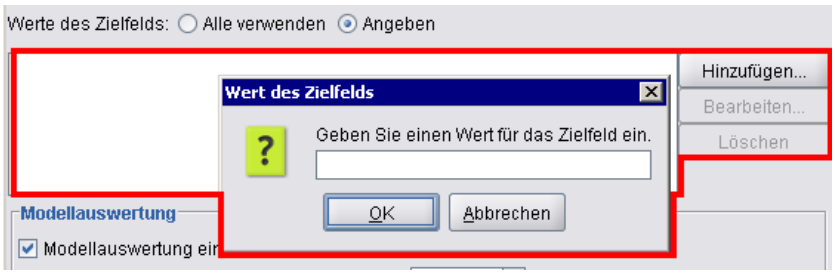
Das Textfeld wird verschlüsselt, indem es einer Eigenschaft zugeordnet wird, die im Abschnitt ‘Eigenschaften’ des zugehörigen Knotens als verschlüsselte Zeichenkette definiert ist:

```
<Property name="encrypted_string1" valueType="encryptedString"/>
```

Eigenschaftssteuerelement

Ein Eigenschaftssteuerelement ist ein vollständig benutzerdefinierbares Steuerelement, mit dem Benutzer Eigenschaften für den Knoten eingeben können. Die Verarbeitung wird durch eine vom Benutzer geschriebene Java-Klasse durchgeführt. Beispiel für ein Eigenschaftssteuerelement:

Abbildung 6-30
Dialogfeldabschnitt mit hervorgehobenem Beispiel für ein Eigenschaftssteuerelement



Format

```
<PropertyControl controller_attributes controlClass="Java_class" >
   -- advanced custom layout options --
</PropertyControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

controlClass (erforderlich) ist der Pfad innerhalb einer *.jar*-Datei zu der Java-Klasse, die das Eigenschaftssteuerelement implementiert. (*Anmerkung*: Die *.jar*-Datei ist in einem JarFile-Element im Abschnitt 'Ressourcen' deklariert. Für weitere Informationen siehe Thema Jar-Dateien in Kapitel 4 auf S. 44.)

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

```
<PropertyControl property="target_field_values_specify" labelAbove="true"
   controlClass="com.spss.clef.selflearning.propertycontrols.list.CustomListControl" label=""
   labelKey="target_field_values_specify.LABEL">
   <Enabled>
      <Condition control="target_field_values" op="equals" value="specify"/>
   </Enabled>
   <Layout rowIncrement="2" />
</PropertyControl>
```

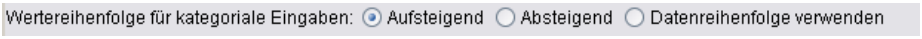
Das Eigenschaftssteuerelement ist einer Eigenschaft zugeordnet, die im Abschnitt 'Eigenschaften' für den entsprechenden Knoten definiert ist:

```
<Property name="target_field_values_specify" valueType="string" isList="true" label=""
   labelKey="target_field_values_specify.LABEL"/>
```

Steuerelement für eine Optionsschaltflächengruppe

Definiert einen Satz von Optionsschaltflächen, die gleichzeitig ausgewählt werden können.

Abbildung 6-31
Steuerelement für eine Optionsschaltflächengruppe



Jedes Steuerelement für eine Optionsschaltflächengruppe hat ein Eigenschaftsattribut, das die Gruppe einer bestimmten Eigenschaft zuordnet. Diese Eigenschaft wird an anderer Stelle in der Datei definiert und spezifiziert die Schaltflächen, aus denen die Gruppe besteht.

Die zugeordnete Eigenschaft kann entweder eine aufgezählte Liste oder eine boolesche Eigenschaft sein. Für aufgezählte Listen (mit dem Eigenschaftsattribut valueType="enum"), wird für jeden enum-Wert eine Optionsschaltfläche angezeigt. Für boolesche Eigenschaften (mit valueType="boolean") werden immer zwei Optionsschaltflächen angezeigt.

Format

```
<RadioButtonGroupControl controller_attributes
   rows="integer" layoutByRow="true_false" useSubPanel="true_false"
   falseLabel="button_label" falseLabelKey="label_key" trueLabel="?button_label"
   trueLabelKey="label_key" trueFirst="true_false" >
   -- advanced custom layout options --
</RadioButtonGroupControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

rows ist eine positive ganze Zahl, die die Anzahl der Bildschirmzeilen festlegt, in denen die Gruppe angezeigt wird. Die Standardeinstellung lautet 1.

layoutByRow gibt an, ob die Optionsschaltflächen zuerst innerhalb der Zeile (true) oder in der Spalte (false) angeordnet werden. Die Standardeinstellung lautet true. Ein Beispiel zur Verwendung von layoutByRow mit einer Optionsschaltflächengruppe finden Sie unter Ändern der Reihenfolge der Steuerelemente auf S. 190.

useSubPanel gibt an, ob die Optionsschaltflächen als Unterfenster angeordnet (true) werden oder nicht (false). Die Standardeinstellung lautet true.

Optionsschaltflächengruppen werden normalerweise als Unterfenster angeordnet, in denen alle Schaltflächen der Gruppe enthalten sind. Dies kann jedoch zu Ausrichtungsproblemen führen, wenn die Optionsschaltfläche zu einem daneben liegenden Textfeld gehört. Wenn useSubPanel auf false gesetzt ist, wird dieses Problem vermieden.

falseLabel ist die Beschriftung für den Wert “false” einer booleschen Eigenschaft (siehe unten das zweite Beispiel). Wird nur mit booleschen Eigenschaften verwendet, sofern es benötigt wird.

falseLabelKey kennzeichnet die Beschriftung für “false” für Lokalisierungszwecke.

trueLabel ist die Beschriftung für den Wert “true” einer booleschen Eigenschaft (siehe unten das zweite Beispiel). Wird nur mit booleschen Eigenschaften verwendet, sofern es benötigt wird.

trueLabelKey kennzeichnet die Beschriftung für “true” für Lokalisierungszwecke.

trueFirst sorgt in der Einstellung true dafür, dass die Anzeigereihenfolge der Schaltflächen für eine boolesche Eigenschaft umgekehrt wird, sodass die Schaltfläche für den Wert “true” zuerst angezeigt wird. Die Standardeinstellung ist false, was bedeutet, dass die Schaltfläche für den “false”-Wert zuerst angezeigt wird.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiele

Das erste Beispiel zeigt den Code für die oben dargestellten Optionsschaltflächengruppe.

```
<RadioButtonGroupControl property="value_order" labelWidth="2">
   <Layout gridWidth="4"/>
</RadioButtonGroupControl>
```

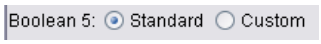
Eine Beschreibung des Layout-Elements finden Sie unter Erweitertes benutzerdefiniertes Layout auf S. 190

Anmerkung: Die Anzahl der Schaltflächen und deren Beschriftungen werden im Abschnitt ‘Eigenschaften’ des entsprechenden Knotens definiert; in diesem Fall als aufgezählte Liste in der Deklaration für die Eigenschaft value_order: Diese Deklaration enthält außerdem die Beschriftung für die Gruppe als solche:

```
<Property name="value_order" valueType="enum" label="Wertereihenfolge für kategoriale
  Eingaben" labelKey="value_order.LABEL">
   <Enumeration>
      <Enum value="Aufsteigend" label="Aufsteigend" labelKey="value_order.Ascending.LABEL"/>
      <Enum value="Absteigend" label="Absteigend" labelKey="value_order.Descending.LABEL"/>
      <Enum value="Datenreihenfolge" label="Datenreihenfolge verwenden" labelKey="value_order.UseDataOrder.LABEL"/>
   </Enumeration>
</Property>
```

Das zweite Beispiel zeigt die Verwendung von falseLabel und trueLabel für eine Optionsschaltflächengruppe, die eine boolesche Eigenschaft steuert, wie etwa, ob Standard- oder benutzerdefinierte Einstellungen aktiviert sind:

Abbildung 6-32
Eine Optionsschaltflächengruppe, die eine boolesche Eigenschaft steuert



Der Code, um dies zu erreichen:

```
<RadioButtonGroupControl property="boolean5" label="Boolean 5" labelKey="boolean5.LABEL"
   falseLabel="Standard" falseLabelKey="boolean5.false.LABEL" trueLabel="Custom"
   trueLabelKey="boolean5.true.LABEL" />
```

In diesem Fall werden die Schaltflächen- und Gruppenbeschriftungen im Element RadioButtonGroupControl selbst definiert. Die Eigenschaft, mit der die Gruppe verknüpft ist, wird im Abschnitt “Properties” für den Knoten definiert.

```
<Property name="boolean5" valueType="boolean" defaultValue="false"/>
```

Auswahlsteuerelement für das Server-Verzeichnis

Definiert ein einzeiliges Textfeld und die zugehörige Schaltfläche, mit der der Benutzer ein Verzeichnis auf dem Server auswählen kann. Das Verzeichnis muss bereits vorhanden sein. Benutzer können in diesem Verzeichnis entweder eine Datei öffnen oder speichern. Dies hängt vom eingestellten Modus ab.

Abbildung 6-33
Steuerelement zur Server-Verzeichnisauswahl

Der Benutzer kann entweder den Verzeichnispfad und den Namen direkt in das Textfeld eingeben oder auf die daneben liegende Schaltfläche klicken, um ein Dialogfeld zu öffnen, in dem er ein Verzeichnis auswählen kann.

Format

```
<ServerDirectoryChooserControl controller_attributes mode="chooser_mode" >
   -- advanced custom layout options --
</ServerDirectoryChooserControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

mode legt die Schaltfläche fest, die im Dialogfeld angezeigt wird, in dem der Benutzer ein Verzeichnis auswählt. Folgende Optionen sind verfügbar:

- open (Standard) zeigt die Schaltfläche Öffnen an.
- save zeigt die Schaltfläche Speichern an.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

```
<ServerDirectoryChooserControl property="directory1" label="Server-Verzeichnis"
   labelKey="directory1.LABEL"/>
```

Auswahlsteuerelement für die Serverdatei

Definiert ein einzeiliges Textfeld und die zugehörige Schaltfläche, mit der der Benutzer eine Datei auf dem Server auswählen kann. Die Datei muss bereits vorhanden sein. Benutzer können die Datei entweder öffnen oder speichern. Dies hängt vom festgelegten Modus ab.

Abbildung 6-34
Element für die Server-Dateiauswahl

Der Benutzer kann entweder den Dateipfad und den Namen direkt in das Textfeld eingeben oder auf die daneben liegende Schaltfläche klicken, um ein Dialogfeld zu öffnen, in dem er eine Datei auswählen kann.

Format

```
<ServerFileChooserControl controller_attributes mode="chooser_mode" >
   -- advanced custom layout options --
</ServerFileChooserControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

mode legt die Schaltfläche fest, die im Dialogfeld angezeigt wird, in dem der Benutzer eine Datei auswählt. Folgende Optionen sind verfügbar:

- open (Standard) zeigt die Schaltfläche Öffnen an.
- save zeigt die Schaltfläche Speichern an.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

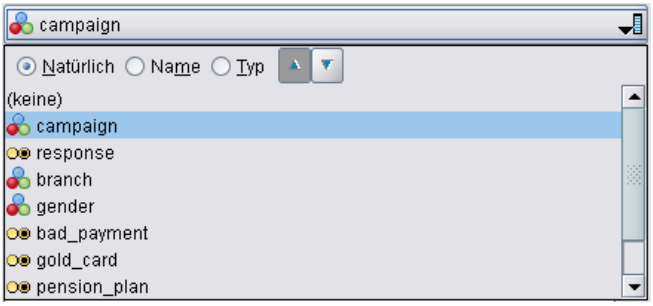
Beispiel

```
<ServerFileChooserControl property="file1" label="Server File" labelKey="file1.LABEL"/>
```

Steuerelement für die Auswahl eines einzelnen Felds

Definiert ein Steuerelement, mit dem der Benutzer ein einzelnes Feld aus einer Liste auswählen kann.

Abbildung 6-35
Steuerelement für die Auswahl eines einzelnen Felds



Wenn der Benutzer auf dieses Steuerelement klickt, wird eine Feldliste angezeigt, aus der ein einzelnes Feld ausgewählt werden kann.

Das Set besteht aus allen Feldern, die an diesem Knoten sichtbar sind. Wenn Felder oberhalb dieses Knotens gefiltert wurden, sind nur die Felder, die den Filter durchlaufen haben, sichtbar. Die Liste kann noch weiter eingeschränkt werden, indem angegeben wird, dass nur Felder mit bestimmten Speicher- und Datentypen zur Auswahl stehen sollen.

Format

<SingleFieldChooserControl *controller_attributes* storage="*storage_types*" onlyNumeric="*true_false*"
onlySymbolic="*true_false*" onlyDatetime="*true_false*" types="*data_types*"
onlyRanges="*true_false*" onlyDiscrete="*true_false*" >
-- *advanced custom layout options* --
</SingleFieldChooserControl>

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

Sie können die Feldliste weiter einschränken, indem Sie außerdem zwei weitere Attribute angeben, von denen eines aus der folgenden Liste stammen muss:

- storage ist eine Listeneigenschaft, die den Speichertyp der Felder angibt, die in der Liste zulässig sein sollen. So bedeutet storage="[integer real]", dass nur Felder mit diesen Speichertypen aufgelistet werden. Die Menge der möglichen Speichertypen können Sie der Tabelle unter Daten- und Speichertypen auf S. 230 entnehmen.
- onlyNumeric (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit numerischem Speichertyp aufgelistet werden sollen.
- onlySymbolic (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit symbolischem Speichertyp (also Zeichenketten) aufgelistet werden sollen.
- onlyDatetime (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit einem Speichertyp für Datum und Uhrzeit aufgelistet werden sollen.

Das zweite angegebene Attribut muss aus dieser Liste stammen:

- types ist eine Listeneigenschaft, die den Datentyp der Felder angibt, die in der Liste zulässig sein sollen. So bedeutet types="[range flag]", dass nur Felder mit diesen Speichertypen aufgelistet werden. Folgende Datentypen sind möglich:

  Bereich

  Flag

  set

  orderedSet

  Numerisch

  discrete

  typeless
- onlyRanges (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit dem Datentyp “Bereich” aufgelistet werden sollen.
- onlyDiscrete (sofern auf true (wahr) gesetzt) gibt an, dass nur Felder mit diskretem Datentyp (also Flag, Set oder ohne Typ) aufgelistet werden sollen.

So stellt beispielsweise ein Steuerelement mit der Angabe storage="[integer]" und types="[flag]" sicher, dass nur ganzzahlige Felder, bei denen es sich um Flags handelt, in der Liste angezeigt werden.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Anmerkung: Dieses Steuerelement wird nur in Knotendefinitionen verwendet. Verwenden Sie folgendes Format, wenn Sie ein Auswahlelement für mehrere Felder ohne Ausgabedatenmodelldefinition festlegen möchten:

```
<OutputDataModel mode="mode">
 ...
   <ForEach var="field" from="1" to="${integer}">
      <AddField name="${string}_${field}" fieldRef="${field_ref}" />
   </ForEach>
 ...
</OutputDataModel>
```

Für weitere Informationen siehe Thema Ausgabedatenmodell (OutputDataModel) in Kapitel 4 auf S. 75. Eine Beschreibung des ForEach-Elements finden Sie unter Iteration mit dem Element ‘ForEach’ auf S. 85. Eine Beschreibung zu AddField finden Sie unter Feld hinzufügen auf S. 82.

Beispiel

Das folgende Beispiel zeigt den Code, mit dem das Auswahlelement für ein einzelnes Feld aus der letzten Abbildung spezifiziert wird.

```
<SingleFieldChooserControl property="target"  storage="string" onlyDiscrete="true"/>
```

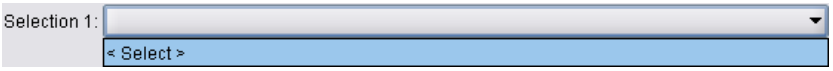
Anmerkung: Der Inhalt der Liste wird im Eigenschaftsabschnitt des entsprechenden Knotens definiert; in diesem Fall in der Deklaration für die Eigenschaft target:

```
<Property name="target" valueType="string" label="Zielfeld" labelKey="target.LABEL"/>
```

Steuerelement für die Auswahl eines einzelnen Elements

Definiert ein Steuerelement, mit dem der Benutzer ein einzelnes Element aus einer Werteliste auswählen kann. Es verknüpft eine Eigenschaft mit einem Katalog, der eine Werteliste enthält. Für weitere Informationen siehe Thema Kataloge in Kapitel 4 auf S. 52.

Abbildung 6-36
Steuerelement für die Auswahl eines einzelnen Elements



Format

```
<SingleItemChooserControl controller_attributes catalog="catalog_name" >
   -- advanced custom layout options --
</MultiItemChooserControl
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

catalog (erforderlich) ist der Name für den zu verknüpfenden Katalog. Die Bibliothek, aus der der Katalog bezogen wird, wird im Element Module des Abschnitts “Execution” angegeben. Für weitere Informationen siehe Thema Module in Kapitel 4 auf S. 72.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

```
<SingleItemChooserControl property="selection1" catalog="cat1" />
```

Eine Erklärung und ein Beispiel für die Verwendung dieser Steuerung finden Sie unter Kataloge auf S. 52.

Drehfeldsteuerelement

Definiert ein Drehfeld (ein numerisches Feld mit aufwärts und abwärts weisenden Pfeilen, mit denen der Feldwert geändert wird).

Abbildung 6-37
Drehfeld

Format

```
<SpinnerControl controller_attributes columns="integer" stepSize="increment"
   minDecimalDigits="number" maxDecimalDigits="number" >
   -- advanced custom layout options --
</SpinnerControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

columns ist eine positive ganze Zahl, die die Anzahl der Zeichenkettenspalten festlegt, über die sich das Steuerelement erstreckt. Die Standardeinstellung lautet 5.

stepSize ist eine Dezimalzahl, die angibt, um welchen Wert der Feldwert verändert wird, wenn der Benutzer eine der Pfeiltasten anklickt. Die Standardeinstellung lautet 1,0.

minDecimalDigits ist der Mindestwert, der für den Feldwert angezeigten Dezimalstellen. Die Standardeinstellung lautet 1.

maxDecimalDigits ist der Höchstwert, der für den Feldwert angezeigten Dezimalstellen.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Das folgende Beispiel zeigt den Code, mit dem das Steuerelement für das Drehfeld aus der letzten Abbildung spezifiziert wird.

```
<SpinnerControl property="double1" label="Double 1" labelKey="double1.LABEL"/>
```

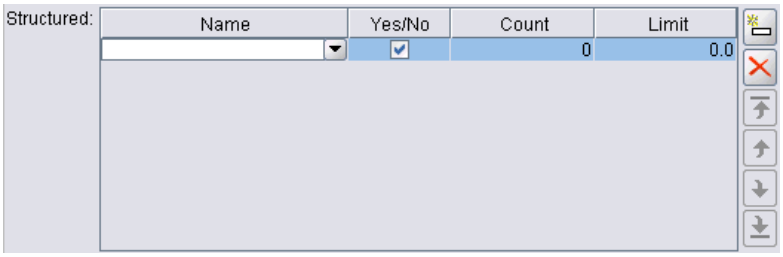
Die Präzision und der Gültigkeitsbereich für die numerischen Feldinhalte werden im Eigenschaftsabschnitt des entsprechenden Knotens definiert; in diesem Fall in der Deklaration für die Eigenschaft double1:

```
<Property name="double1" valueType="double" min="0" max="100"/>
```

Tabellensteuerelement

Definiert ein Tabellenlayoutelement, das in einem Knotendialogfeld oder einem Knotenausgabefenster angezeigt wird.

Abbildung 6-38
Tabellensteuerelement



Format

```
<TableControl controller_attributes rows="integer" columns="integer" columnWidths="list" >
   -- advanced custom layout options --
</TableControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

rows ist eine positive ganze Zahl, die die Anzahl der Tabellenzeilen angibt, die auf dem Bildschirm angezeigt werden. Die Standardeinstellung lautet 8.

columns ist eine positive ganze Zahl, die die Anzahl der Zeichenkettenspalten festlegt, über die sich die Tabelle erstreckt. Die Standardeinstellung lautet 20.

columnwidths ist eine Liste von Werten, mit der die relative Breite der Spalten angegeben wird. Beispiel: Die Werte [30 5 10] geben an, dass Spalte 1 drei Mal so breit ist wie Spalte 3.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Der Code, mit dem das in der letzten Abbildung gezeigte Steuerelement angegeben wird, lautet:

```
<TableControl property="structure1" allowReorder="true" label="Structured"
 labelKey="structure1.LABEL" columnWidths="[20 6 10 10]">
   <ColumnControl column="0" editor="fieldValue" fieldControl="field1"/>
</TableControl>
```

Die Struktur des Tabellensteuerelements wird im Abschnitt für gemeinsame Objekte der Spezifikationsdatei als Eigenschaftstyp festgelegt:

```
<PropertyType id="shared_structure1" valueType="structure" isList="true">
   <Structure>
      <Attribute name="id" valueType="string" label="Name" labelKey="structure1.id.LABEL"/>
      <Attribute name="yesno" valueType="boolean" label="Yes/No" labelKey="structure1.yesno.LABEL"
       defaultValue="true"/>
      <Attribute name="count" valueType="integer" label="Count" labelKey="structure1.count.LABEL"
       defaultValue="0"/>
      <Attribute name="limit" valueType="double" label="Limit" labelKey="structure1.limit.LABEL"
       defaultValue="0.0"/>
   </Structure>
</PropertyType>
```

In der Knotenspezifikation wird dann die ID dieses Eigenschaftstyps mittels einer Eigenschaftsdeklaration der ID des Tabellensteuerelements zugeordnet:

```
<Property name="structure1" type="shared_structure1"/>
```

Wenn in einem Skript Bezug auf den Knoten genommen wird, können in eckigen Klammern [] die Werte der Eigenschaft für die Liste und in geschweiften Klammern {} für die Struktur angegeben werden. Beispiel: So legen Sie ein Raster aus zwei Strukturen für die Eigenschaft structure1 fest:

```
set :node_ID.structure1 = [{"hello" true 4 0.21} {"bye" false 5 0.95}]
```

Beachten Sie, dass die Reihenfolge der Werte mit der Reihenfolge übereinstimmen muss, in der die Attribute-Definitionen vorgenommen werden.

Textbereichssteuerelement

Definiert ein mehrzeiliges Texteingabefeld.

Abbildung 6-39
Textbereichssteuerelement

Format

```
<TextAreaControl controller_attributes rows="integer" columns="integer" wrapLines="true_false" >
   -- advanced custom layout options --
</TextAreaControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

rows ist eine positive ganze Zahl, die die Anzahl der Bildschirmzeilen angibt, die der Textbereich belegt. Die Standardeinstellung lautet 8.

columns ist eine positive ganze Zahl, die die Anzahl der Zeichenkettenspalten festlegt, über die sich der Textbereich erstreckt. Die Standardeinstellung lautet 20.

wrapLines gibt an, ob lange Zeilen umgebrochen werden (true) oder ob ein horizontales Blättern erforderlich ist, um lange Textzeilen zu lesen (false). Die Standardeinstellung lautet true.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Der Code für das oben dargestellte Beispiel lautet:

```
<TextAreaControl property="string2" label="String 2" labelKey="string2.LABEL"/>
```

In diesem Fall wird die Beschriftung für den Textbereich in der Deklaration für die Textbereichsteuerung definiert, während der Eingabedatentyp im Abschnitt “Properties” für den relevanten Knoten definiert wird; in der Deklaration für die Eigenschaft string2:

```
<Property name="string2" valueType="string" />
```

Textfeldsteuerelement

Definiert ein einzeiliges Texteingabefeld.

Abbildung 6-40
Textfeldsteuerelement

Format

```
<TextBoxControl controller_attributes columns="integer" >
   -- advanced custom layout options --
</TextBoxControl>
```

Erläuterung:

controller_attributes werden unter Controller-Attribute auf S. 162 beschrieben

columns ist eine positive ganze Zahl, die die Anzahl der Zeichenkettenspalten festlegt, über die sich das Textfeld erstreckt. Die Standardeinstellung lautet 20.

Die erweiterten Optionen für das benutzerdefinierte Layout ermöglichen eine Feinsteuerung für Positionierung und Anzeige von Bildschirmkomponenten. Für weitere Informationen siehe Thema Erweitertes benutzerdefiniertes Layout auf S. 190.

Beispiel

Der Code für das oben dargestellte Textfeld lautet:

```
<TextBoxControl property="string1" label="String 1" labelKey="string1.LABEL"/>
```

Der Eingabedatentyp für das Textfeld wird im Abschnitt “Properties” des entsprechenden Knotens definiert; in diesem Fall in der Deklaration für die Eigenschaft string1:

```
<Property name="string1" valueType="string" />
```

Layouts für Eigenschaftssteuerelemente

In diesem Abschnitt werden die Methoden für Standardlayouts beschrieben, die für Dialogfenster und Fenster verwendet werden, sowie Verfahren, mit denen diese verändert und eigene benutzerdefinierte Layouts definiert werden können.

Standardsteuerelementlayout

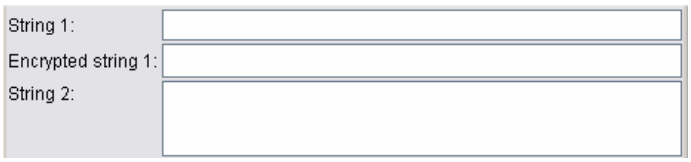
Ein Eigenschaftsfenster kann als zweidimensionales Raster von Zellen betrachtet werden. Jede Zeile kann eine andere Höhe besitzen und jede Spalte eine andere Breite. Die Komponenten der Benutzeroberfläche können mehreren zusammenhängenden Zellen zugewiesen werden, auch wenn Komponenten der Benutzeroberfläche normalerweise nur einer Zelle zugewiesen werden.

Standardmäßig wird einem Eigenschaftssteuerelement eine Zeile zugeordnet und alle Steuerelemente belegen bis zu zwei Zeilen: Eine für die Beschriftung und eine für die Steuerelementkomponenten. Die Spalte, die die Beschriftung enthält, wird an die Breite der breitesten Beschriftung angepasst. Wenn in der Spezifikationsdatei z. B. folgende Elemente angegeben sind:

```
<TextBoxControl property="string1" label="Zeichenkette 1"/>
<PasswordBoxControl property="encryptedString1" label="Verschlüsselte Zeichenkette 1"/>
<TextAreaControl property="string2" label="Zeichenkette 2"/>
```

Sieht der entsprechende Bereich wie folgt aus:

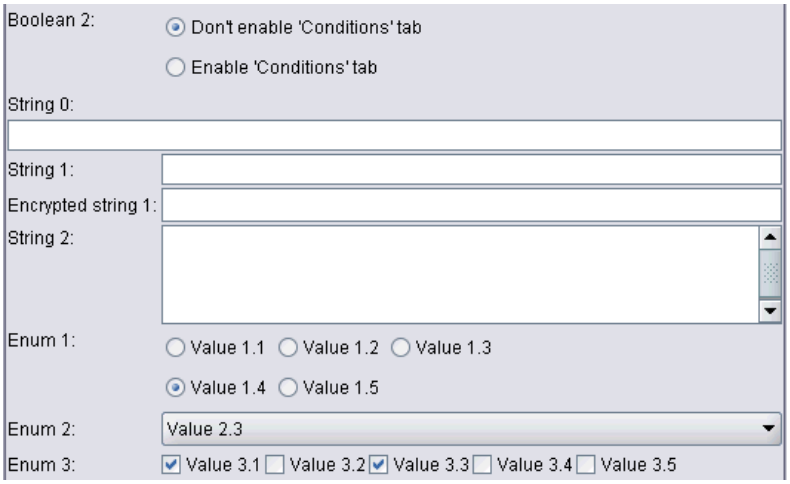
Abbildung 6-41
Einfacher Eigenschaftsbereich



Beachten Sie, dass der Doppelpunkt ":" am Ende der Beschriftung automatisch hinzugefügt wird.

Ein Eigenschaftssteuerelement, das mehrere Komponenten der Benutzeroberfläche enthält, erstellt seinen eigenen sichtbaren rechteckigen Bereich, in dem diese Komponenten angeordnet werden. Die Elemente RadioButtonGroupControl und CheckBoxGroupControl sind Beispiele für solche Steuerelemente. In der Bildschirmabbildung illustrieren die Steuerungen mit der Beschriftung Boolean2, Enum1 und Enum3 das Folgende:

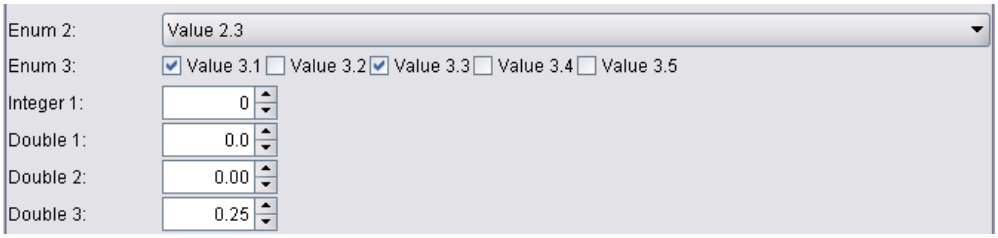
Abbildung 6-42
Eigenschaftenbereich mit verschiedenen Mehrkomponenten-Steuerelementen



Beachten Sie, dass die Form des rechteckigen Bereichs, in dem die Komponenten angeordnet sind, sich abhängig vom Eigenschaftensteuerelement ändern kann. Daher ist das Layout bei verschiedenen Steuerelementen nicht immer genau ausgerichtet. Vergleichen Sie in der vorherigen Abbildung Enum 1 mit Enum 3.

Einige Eigenschaftensteuerelemente umfassen Komponenten, die die Komponentenspalte komplett füllen und sich horizontal anpassen, wenn die Fensterbreite vergrößert oder verkleinert wird. Beispiele dafür sind die Steuerelemente, die von den Elementen TextBoxControl, PasswordBoxControl und TextAreaControl festgelegt werden, wie durch die Steuerelemente mit den Beschriftungen String 1, Encrypted string 1 und String 2 in den vorherigen Abbildungen gezeigt. Dies gilt jedoch nicht für alle Komponenten. Kontrollkästchen und Drehfelder nehmen z. B. einen fest definierten horizontalen Raum ein, selbst wenn sich die Fensterbreite ändert:

Abbildung 6-43
Eigenschaftenbereich mit Kontrollkästchen und Drehfeldsteuerelementen



Benutzerdefiniertes Steuerelementlayout

Das Standardlayout für Steuerelemente kann nach verschiedenen Verfahren geändert werden, von denen einige einfach und andere komplexer sind.

Einfaches benutzerdefiniertes Layout

Drei einfache Verfahren zur Anpassung des Steuerelementlayouts:

- Positionieren einer Beschriftung über der zugehörigen Komponente
- Änderung der Anzahl der Zeilen, über die hinweg die Steuerelemente angeordnet sind
- Änderung der Reihenfolge, in der die Steuerelemente angeordnet sind

Positionieren einer Beschriftung über der zugehörigen Komponente

Sie können eine Beschriftung in einer separaten Zeile über der zugehörigen Komponente positionieren, indem Sie das labelAbove-Attribut des Steuerelements auf true setzen. Beispiel:

```
<TextBoxControl property="string0" label="String 0" labelAbove="true"/>
<TextBoxControl property="string1" label="Zeichenkette 1"/>
<PasswordBoxControl property="encryptedString1" label="Verschlüsselte Zeichenkette 1"/>
```

Neben der Positionierung der Beschriftung über der Komponente werden die Benutzeroberflächenkomponenten der Beschriftungsspalte der Anzeige zugeordnet. Dies ergibt folgendes Fenster mit der Beschriftung String 0, das über dem zugehörigen Feld angezeigt wird:

Abbildung 6-44
Fenster mit Feldbeschriftung in separater Zeile

Ändern der Zeilenzahl

Standardmäßig werden Optionsschaltflächen- und Kontrollkästchengruppen in einer Reihe angeordnet, und die Breite des Dialogfelds wird entsprechend angepasst. Wenn eine Optionsschaltflächen- und Kontrollkästchengruppe eine große Anzahl an Optionen umfasst, kann dies zu einem sehr breiten Dialogfeld führen. Sie können die vermeiden, indem Sie die Anzahl der Zeilen für das Layout des Steuerelements ändern. Dies ist möglich, indem Sie das Attribut rows der Steuerelementdefinition auf den gewünschten Wert einstellen. Beispiel:

```
<RadioButtonGroupControl property="enum1" label="Enum 1" rows="2"/>
```

Dies führt dazu, dass die Optionsschaltflächengruppe im Fenster über zwei Zeilen ausgegeben wird:

Abbildung 6-45
Fenster mit über zwei Zeilen angeordneter Optionsschaltflächengruppe

Ändern der Reihenfolge der Steuerelemente

Für Optionsschaltflächen- und Kontrollkästchengruppen können Sie auch die Reihenfolge ändern, in der die Steuerelemente für die einzelnen enum-Werte zum Fenster hinzugefügt werden.

Standardmäßig werden die Steuerelemente in der Zeilenreihenfolge hinzugefügt, wie im vorherigen Beispiel, wo der erste, der zweite und der dritte Wert in die erste Zeile und der vierte und fünfte Wert in die zweite Zeile eingefügt werden. Sie können die Steuerelemente stattdessen innerhalb der angegebenen Zeilenzahl in der Spaltenreihenfolge einfügen, indem Sie für layoutByRow den Wert false festlegen. Beispiel:

```
<RadioButtonGroupControl property="enum1" label="Enum 1" rows="2" layoutByRow="false"/>
```

Die Werte werden weiterhin in zwei Zeilen angezeigt, der erste und zweite Wert werden jetzt jedoch in der ersten Spalte angezeigt, der dritte und vierte in der zweiten und der fünfte in der dritten:

Abbildung 6-46
Fenster mit Optionsschaltflächengruppe, nach Spaltenreihenfolge angeordnet

Boolesche Eigenschaften, die als zwei Optionsschaltflächen angezeigt werden, werden standardmäßig so angeordnet, dass die Schaltfläche für “False” vor der Schaltfläche für “True” angezeigt wird. Sie können diese Reihenfolge umkehren, indem Sie das Attribut trueFirst auf true einstellen.

Sie können auch verhindern, dass Optionsschaltflächen- und Kontrollkästchengruppen ein Unterfenster verwenden, indem Sie das Attribut useSubPanel auf false einstellen. Dies kann jedoch zu einem unerwünschten Layoutverhalten führen, wenn diese Option nicht zusammen mit dem Element Layout verwendet wird (siehe Festlegen präziser Steuerelementpositionen mit dem Element “Layout” auf S. 191).

Erweitertes benutzerdefiniertes Layout

Innerhalb der einzelnen Steuerelementdeklarationen können Sie komplexe Steuerelementlayouts festlegen, die verschiedene Elemente verwenden. Sie haben folgende Möglichkeiten:

- Festlegen präziser Steuerelementpositionen auf dem Bildschirm mit dem Element Layout
- Anzeigeeigenschaften des Steuerelements mit dem Element Enabled festlegen
- Sichtbarkeit des Steuerelements auf dem Bildschirm mit dem Element Visible steuern

Festlegen präziser Steuerelementpositionen mit dem Element "Layout"

Eine präzise Layoutpositionierung kann durch die explizite Angabe eines Layout-Elements erreicht werden, das dem Steuerelement zugeordnet wird:

Format

```
<property_control ... >
   <Layout attributes
    --- cell specification ---
      ...
</property_control>
```

Erläuterung:

property_control ist eines der Eigenschaftssteuerelemente (siehe Spezifikationen von Eigenschaftssteuerelementen auf S. 154).

attributes ist eine beliebige Anzahl der folgenden Attribute:

Tabelle 6-6
Layoutattribute

Attribut	Werte	Beschreibung
anchor	north northeast east southeast south southwest west northwest Zentrum	Definiert den Verankerungspunkt für das Steuerelement.
columnWeight	0.0–1.0	Definiert, wie sich die Änderung der horizontalen Größe des Fensters auf die Breite des Steuerelements auswirkt. Die Summe aller columnWeight-Attribute in einem Fenster sollten den Wert 1,0 nicht überschreiten.
fill	none horizontal vertical both	Definiert, ob und wie das Steuerelement die ihm zugeordneten Zellen ausfüllt.
gridColumn	integer >= 0	Definiert die erste Spalte, in der das Layout des Steuerelements beginnt.
gridHeight	integer	Definiert die Anzahl der Zeilen, über die sich das Layout des Steuerelements erstreckt. Der Wert 0 (Standard) ordnet dem Steuerelement alle verbleibenden Zeilen zu.
gridRow	integer >= 0	Definiert die erste Zeile, in das Layout des Steuerelements verwendet wird. Standardmäßig wird der Rasterzeilenindex automatisch inkrementiert.
gridWidth	integer	Definiert die Anzahl der Spalten, über die sich das Layout des Steuerelements erstreckt. Der Wert 0 (Standard) ordnet dem Steuerelement alle verbleibenden Spalten zu.

Attribut	Werte	Beschreibung
leftIndent	integer	Definiert die Anzahl Pixel, um die das Steuerelement in Bezug auf seine Standardposition eingerückt wird.
rowWeight	0.0–1.0	Definiert, wie sich die Änderung der vertikalen Größe des Fensters auf die Höhe des Steuerelements auswirkt. Die Summe aller rowWeight-Attribute in einem Fenster sollten den Wert 1,0 nicht überschreiten.

Mit einer **Zellenspezifikation** können Sie die exakte Position eines Steuerelements auf dem Bildschirm definieren. Das Format lautet wie folgt:

<Cell row="*integer*" column="*integer*" width="*integer*" />

Erläuterung:

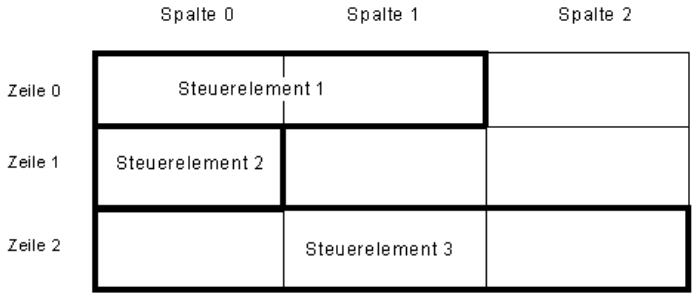
row (erforderlich) ist eine nicht negative ganze Zahl, die die Zeilenposition für den Start des Steuerzeichens angibt.

column (erforderlich) ist eine nicht negative ganze Zahl, die die Spaltenposition für den Start des Steuerzeichens angibt.

width (erforderlich) ist eine nicht negative ganze Zahl, die die Anzahl der Bildschirmrasterspalten angibt, die das Steuerelement belegt.

Wenn ein Bildschirmraster z. B. drei Spalten und drei Zeilen enthält, ist ein benutzerdefiniertes Layout im folgenden Format möglich:

Abbildung 6-47
Beispiel für Steuerelementlayout, das Zellen verwendet



benötigt ein Layout-Element mit folgenden Zellspezifikationen:

```
<Layout ... >
   <Cell row="0" column="0" width="2">
   <Cell row="1" column="0" width="1">
   <Cell row="2" column="0" width="3">
</Layout>
```

Es folgen einige detailliertere Beispiele, die die Verwendung des Layout-Elements weiter darstellen.

Beispiel: Kontrollkästchen, das ein Textfeld aktiviert

Dieses Beispiel zeigt die Verwendung eines Kontrollkästchens, um ein Textfeld in derselben Zeile der Anzeige zu aktivieren.

Wenn ein Kontrollkästchen verwendet wird, um ein anderes Steuerelement in derselben Zeile zu aktivieren, wird ein einfaches Layout-Element benötigt, damit die Steuerelemente richtig angezeigt werden. (*Anmerkung*: Eine Beschreibung des Mechanismus zur Aktivierung bzw. Deaktivierung von Steuerelementen finden Sie unter Steuerung der Anzeigeeigenschaften mit dem Element ‘Enabled’ auf S. 199).

Angenommen, wir möchten folgende Anzeige erhalten:

Abbildung 6-48
Kontrollkästchen, das ein Textfeld aktiviert

Es gibt zwei Steuerelemente:

- Ein Kontrollkästchen mit einer Beschriftung, die als Beschriftung eines Textfelds verwendet wird
- Das Textfeld als solches

Ausgangspunkt ist eine normale Deklaration der beiden Steuerelemente:

```
<CheckBoxControl property="boolean3" label="Kontrollkästchen 3"/>
<TextBoxControl property="string3" label="Zeichenkette 3"/>
```

Dies erzeugt ein wie folgt aufgebautes Fenster:

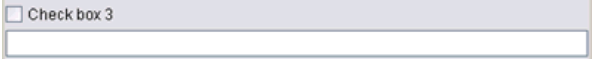
Abbildung 6-49
Kontrollkästchen und Textfeld in separaten Zeilen

Zuerst soll die Anzeige der Textfeldbeschriftung String 3 verhindert werden. Dies wird erreicht, indem das Attribut showLabel des Textfeldsteuerelements auf false gesetzt wird:

```
<CheckBoxControl property="boolean3" label="Check box 3"/>
<TextBoxControl property="string3" label="String 3" showLabel="false"/>
```

Das Textfeld wird vergrößert und füllt jetzt den zuvor von der Beschriftung belegten Raum aus:

Abbildung 6-50
Kontrollkästchen und Textfeld ohne Beschriftung



Jetzt möchten wir erreichen, dass das Textfeld in derselben Zeile angezeigt wird, wie das Kontrollkästchen. Hierzu fügen wir ein Layout-Element innerhalb des CheckBoxControl-Elements hinzu, um die Inkrementierung der Zeile auf 0 zu setzen (standardmäßig wird die Zeile für jedes Steuerelement um 1 inkrementiert):

```
<CheckBoxControl property="boolean3" label="Check box 3">
   <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="Zeichenkette 3" showLabel="false"/>
```

Die daraus folgende Anzeige sieht jedoch wie folgt aus:

Abbildung 6-51
Textfeld überschreibt Kontrollkästchen

Das Textfeld wurde um eine Zeile nach oben verschoben, es belegt aber immer noch die gesamte Zeile und überschreibt daher das Kontrollkästchen.

Anmerkung: Wenn die Anzeige wie folgt aussieht:

Abbildung 6-52
Kontrollkästchen überschreibt Textfeld

Das Kontrollkästchen wurde nach dem Textfeld angelegt, weshalb die ersten Zeichen des Textfelds verdeckt werden.

Unabhängig davon, welches Objekt zuerst angelegt wird, verursacht die Zuweisung mehrerer Benutzeroberflächenkomponenten zu derselben Zelle ein unerwünschtes bzw. undefiniertes Verhalten und sollte vermieden werden. Um dieses Problem zu beseitigen, müssen wir ein zweites Layout-Element hinzufügen, das jetzt innerhalb des TextBoxControl-Elements angegeben wird, um zu erzwingen, dass das Textfeld in der zweiten Spalte der Anzeige beginnt:

```
<CheckBoxControl property="boolean3" label="Check box 3">
   <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="Zeichenkette 3" showLabel="false">
   <Layout gridColumn="1"/>
</TextBoxControl>
```

Dies ist jedoch nur eine Teillösung, da das Ergebnis wie folgt aussieht:

Abbildung 6-53
Richtig platziertes aber sehr kurzes Textfeld

Beide Steuerelemente sind richtig platziert, das Textfeld ist aber zu kurz. Das Problem ist, dass die Zuordnung eines benutzerdefinierten Layouts zu einem Steuerelement dazu führt, dass die jedem Steuerelementtyp zugeordneten Standardwerte überschrieben werden. In diesem Fall ist das Standardverhalten des Layout-Elements hinsichtlich des Auffüllens (d. h., wie die Komponente die verfügbaren Zellen auffüllt) so festgelegt, dass die verfügbaren Zellen belegt werden und so

wenig Bildschirmplatz wie möglich eingenommen wird. Damit sich dies ändert, weisen wir das Textfeld an, den horizontalen Raum zu füllen:

```
<CheckBoxControl property="boolean3" label="Check box 3">
   <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false">
   <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Es ist erforderlich, einen kleinen Wert für columnWeight hinzuzufügen, damit Java den ausgefüllten Bereich richtig zuordnet.

So erhalten wir das gewünschte Layout:

Abbildung 6-54
Kontrollkästchen, das ein Textfeld aktiviert

Dies sieht schon gut aus, es gibt aber immer noch ein Problem. Das Kontrollkästchen versucht jetzt, die gesamte Zeile zu besetzen, obwohl wir jetzt ein weiteres Steuerelement in derselben Zeile platzieren. Das Problem ist zurzeit nicht sichtbar, weil die Beschriftung des Kontrollkästchens relativ kurz ist und weil andere Beschriftungen im Fenster (hier nicht dargestellt) die zweite Anzeigespalte vergrößert haben, weswegen keine Überlappung vorliegt. Das Problem wird sichtbar, wenn die Beschriftung des Kontrollkästchens länger wird:

```
<CheckBoxControl property="boolean3" label="Check box 3 with a much longer label than we had">
   <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false">
   <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Dies liefert folgendes Ergebnis:

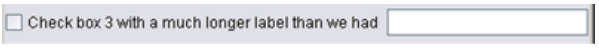
Abbildung 6-55
Lange Kontrollkästchenbeschriftung durch Textfeld überschrieben

Wir müssen das Kontrollkästchen anweisen, seine Breite auf eine einzige Spalte einzuschränken:

```
<CheckBoxControl property="boolean3" label="Check box 3 with a much longer label than we had">
   <Layout rowIncrement="0" gridWidth="1"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false">
   <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

So erhalten wir das gewünschte Ergebnis:

Abbildung 6-56
Lange Kontrollkästchenbeschriftung wird vollständig angezeigt



Beispiel: Optionsschaltflächengruppe und Textfelder

Dieses Beispiel zeigt ein Verfahren, mit dem jeder neuen Schaltfläche einer Optionsschaltflächengruppe ein eigenes Textfeld zugeordnet wird.

Wir möchten ein Fenster definieren, das wie folgt aussieht:

Abbildung 6-57
Optionsschaltflächengruppe mit Textfeldern

Dieses Mal haben wir vier Steuerelemente:

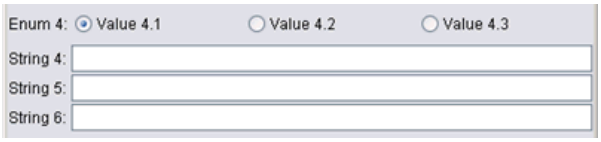
- Eine Optionsschaltflächengruppe für eine Aufzählungsliste mit drei Werten
- Drei Textfelder, für jeden Wert eines

Wie im vorherigen Beispiel beginnen wir mit einer einfachen Deklaration der Steuerelemente:

```
<RadioButtonGroupControl property="enum4" label="Aufz. 4"/>
<TextBoxControl property="string4" label="Zeichenkette 4"/>
<TextBoxControl property="string5" label="Zeichenkette 5"/>
<TextBoxControl property="string6" label="Zeichenkette 6"/>
```

Dies ergibt folgende Anzeige:

Abbildung 6-58
Optionsschaltflächengruppe mit Textfeldern und Beschriftungen



Wir möchten die Optionsschaltflächenbeschriftungen verwenden, um die Textfelder zu identifizieren, daher besteht unsere erste Aufgabe darin, die Optionsschaltflächen in einer einzigen von drei Spalten anzuordnen und die Textfeldbeschriftungen auszublenden:

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3"/>
<TextBoxControl property="string4" label="String 4" showLabel="false"/>
<TextBoxControl property="string5" label="String 5" showLabel="false"/>
<TextBoxControl property="string6" label="String 6" showLabel="false"/>
```

Die daraus resultierende Anzeige sieht wie folgt aus:

Abbildung 6-59
Optionsschaltflächen in einer einzigen Spalte und Textfelder

Wir können hier bereits ein kleines Problem erkennen — die Beschriftung der Optionsschaltflächengruppe ist nicht mit der ersten Optionsschaltfläche ausgerichtet. Wir werden uns später darum kümmern — jetzt werden wir zuerst unsere Textfelder in Bezug auf die zugehörigen Optionsschaltflächen ausrichten.

Das Verfahren ähnelt dem, das wir im 1. Beispiel angewendet haben. Wir müssen:

- Den Inkrementwert der Optionsschaltflächengruppe auf 0 setzen.
- Die Rasterbreite so einschränken, dass sich die nächsten Textfelder und Optionsschaltflächen nicht überlappen.
- Alle Textfelder in derselben Zeile anordnen wie die zugehörige Optionsschaltfläche.

Hierzu fügen wir einige Layout-Elemente hinzu, genau so wie im vorherigen Beispiel. In diesem Fall ändern wir die Spezifikationsdatei wie folgt:

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3">
   <Layout rowIncrement="0" gridWidth="1"/>
</RadioButtonGroupControl>
<TextBoxControl property="string4" label="String 4" showLabel="false">
   <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string5" label="String 5" showLabel="false">
   <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string6" label="String 6" showLabel="false">
   <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Leider sieht unsere Anzeige jetzt wie folgt aus:

Abbildung 6-60
Textfelder überschreiben die Optionsschaltflächen

Wir haben genau dieselben Layout-Elemente verwendet, wie im 1. Beispiel. Was ist passiert?

Die Antwort lautet, dass die Optionsschaltflächengruppe (wie die meisten Steuerelemente), anders als das Kontrollkästchen im vorherigen Beispiel, eine separate Beschriftung und das tatsächliche Steuerelement besitzt. Dies bedeutet, dass für die Optionsschaltflächengruppe eine

gesonderte Spalte erforderlich ist, weswegen die Textfelder eine Spalte später beginnen müssen, d. h. in Spalte 2 anstatt in Spalte 1. Daher setzen wir in den Layout-Elementen für die Textfelder die gridColumn-Werte auf 2:

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3">
   <Layout rowIncrement="0" gridWidth="1"/>
</RadioButtonGroupControl>
<TextBoxControl property="string4" label="String 4" showLabel="false">
   <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string5" label="String 5" showLabel="false">
   <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string6" label="String 6" showLabel="false">
   <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Beachten Sie, dass die Rasterbreite für die Optionsschaltflächengruppe unverändert bei 1 bleibt, obwohl wir die Textfeldrasterspalte mit 2 inkrementieren. Dies hat den Grund, dass sich für die Eigenschaftssteuerelemente die meisten Layout-Attribute nur auf die Benutzeroberflächenkomponenten auswirken, die den editierbaren Bereich des Steuerelements ausmachen, und nicht die Beschriftung.

Wir haben nun:

Abbildung 6-61
Textfelder überschreiben die Optionsschaltflächen nicht mehr

Dies sieht schon viel eher wie das gewünschten Ergebnis aus. Es gibt aber immer noch ein paar Ausrichtungsprobleme zwischen den Optionsschaltflächen und den Textfeldern.

Das Hauptproblem ist, dass die Optionsschaltflächen in separaten Unterfenstern angeordnet werden, weswegen keine wirkliche Beziehung zwischen einer Optionsschaltfläche und ihrem Textfeld besteht. Wir müssen also lediglich dafür sorgen, dass die Optionsschaltflächengruppe kein Unterfenster verwendet:

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3" useSubPanel="false">
   <Layout rowIncrement="0" gridWidth="1"/>
</RadioButtonGroupControl>
<TextBoxControl property="string4" label="String 4" showLabel="false">
   <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string5" label="String 5" showLabel="false">
   <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string6" label="String 6" showLabel="false">
   <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Nun haben wir das gewünschte Layout:

Abbildung 6-62
Optionsschaltflächengruppe mit Textfeldern

Steuerung der Anzeigeeigenschaften mit dem Element 'Enabled'

Sie können mit dem Element Enabled dafür sorgen, dass ein Steuerelement aktiviert bzw. deaktiviert wird, in der Regel unabhängig von einer bestimmten Bedingung.

Fenster- und Eigenschaftssteuerelemente können Bedingungen zugeordnet werden, über die verschiedene Anzeigeeigenschaften festgelegt werden. Beispiel: Ein Kontrollkästchen kann verwendet werden, um ein zugehöriges Textfeld zu aktivieren. Oder eine Optionsschaltfläche kann dafür sorgen, dass eine Gruppe ausgeblendeter Felder angezeigt wird.

Bedingungen für die Benutzeroberfläche basieren in der Regel auf dem Wert eines anderen Steuerelements, anstatt auf einer Eigenschaft. Bedingungen, die auf Eigenschaften basieren, treten nur in Kraft, wenn Änderungen am zugrunde liegenden Objekt durchgeführt wurden (z. B. ein Knoten, eine Modellausgabe oder eine Dokumentausgabe). Auf der Benutzeroberfläche müssen Steuerelemente aktiviert werden, sobald ein zugehöriges Steuerelement geändert wurde.

Format

```
<Enabled>
   <Condition .../>
   <And ... />
   <Or ... />
   <Not ... />
</Enabled>
```

Das Element Condition gibt eine Bedingung an, die getestet wird, um zu ermitteln, ob das Steuerelement aktiviert ist.

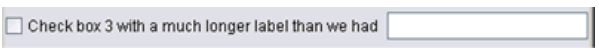
Mit den Elementen And, Or und Not können zusammengesetzte Bedingungen festgelegt werden.

Für weitere Informationen siehe Thema Bedingungen in Kapitel 4 auf S. 91.

Beispiel: Aktivieren von Steuerelementen mit einer einfachen Bedingung

In Beispiel: Kontrollkästchen, das ein Textfeld aktiviert auf S. 193 haben wir ein Kontrollkästchen entworfen, das ein Textfeld aktiviert, wenn es ausgewählt wird:

Abbildung 6-63
Lange Kontrollkästchenbeschriftung wird vollständig angezeigt



Wir möchten, dass das Textfeld aktiviert wird, sobald das Kontrollkästchen markiert wird und nicht erst, wenn die Eigenschaft des zugrunde liegenden Objekts geändert wird. Hierzu müssen wir eine Bedingung für Enabled hinzufügen:

```
<CheckBoxControl property="boolean3" label="Check box 3 with a much longer label than we had">
   <Layout rowIncrement="0" gridWidth="1"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="Zeichenkette 3" showLabel="false">
   <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
      <Enabled>
         <Condition control="boolean3" op="equals" value="true"/>
       </Enabled>
</TextBoxControl>
```

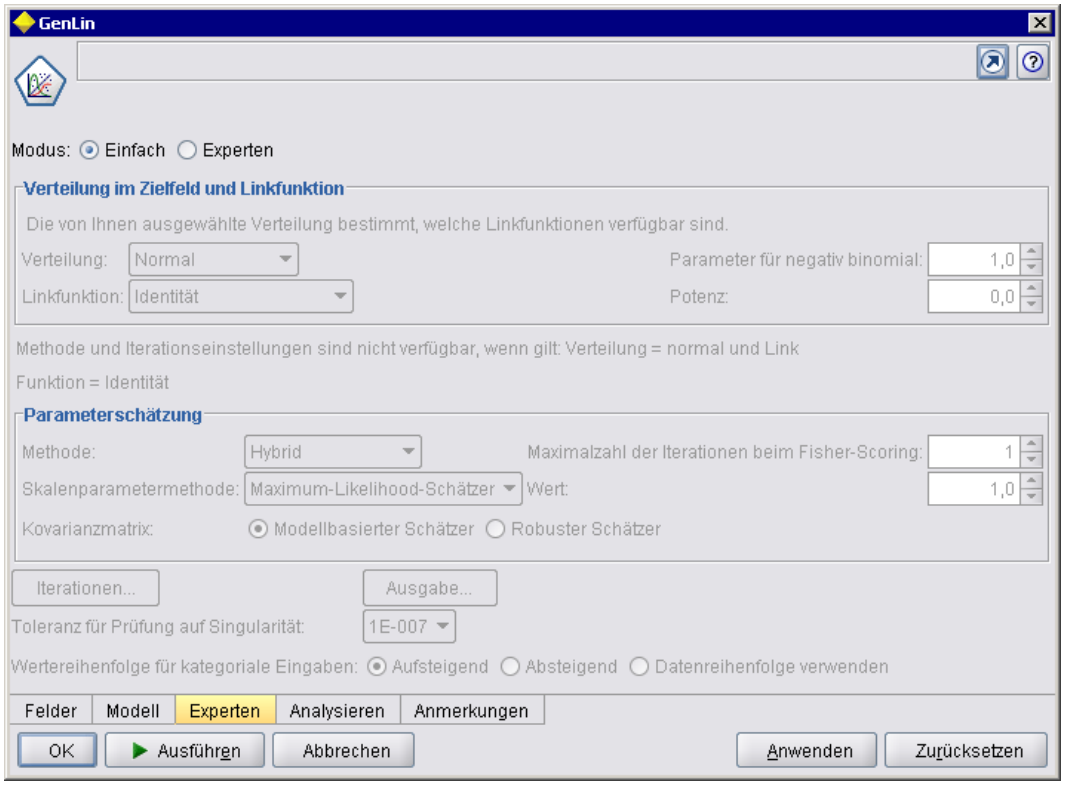
Diese Anweisungen sorgen dafür, dass das Textfeld nur dann aktiviert wird, wenn der dem Textfeld zugeordnete boolesche Wert wahr ist.

Beispiel: Aktivieren von Steuerelementen mit einer komplexen Bedingung

Um die Kodierung komplexer Bedingungen zu veranschaulichen, werden wir uns eine der Dialogfeldregisterkarten für den generalisierten linearen Knoten ansehen, der mit CLEF entwickelt wurde.

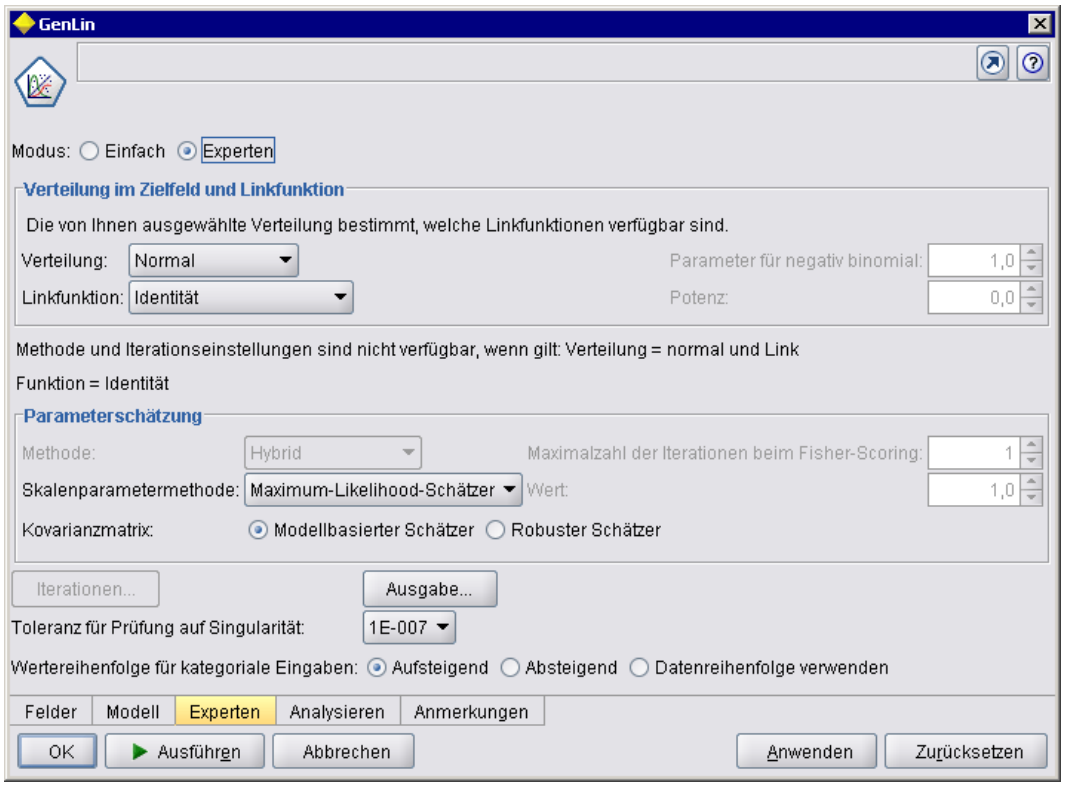
Das Knotendialogfeld besitzt eine Registerkarte ‘Experten’, die Optionen enthält, die für Benutzer mit detaillierten Kenntnissen solcher Modelle gedacht sind. Alle auf der Registerkarte vorhandenen Optionen sind ursprünglich deaktiviert:

Abbildung 6-64
Registerkarte ‘Experten’ mit deaktivierten Optionen



Wenn für das Kontrollkästchen Modus die Option Experten aktiviert wird, werden eine Reihe der Optionen verfügbar:

Abbildung 6-65
Registerkarte 'Experten' im Expertenmodus

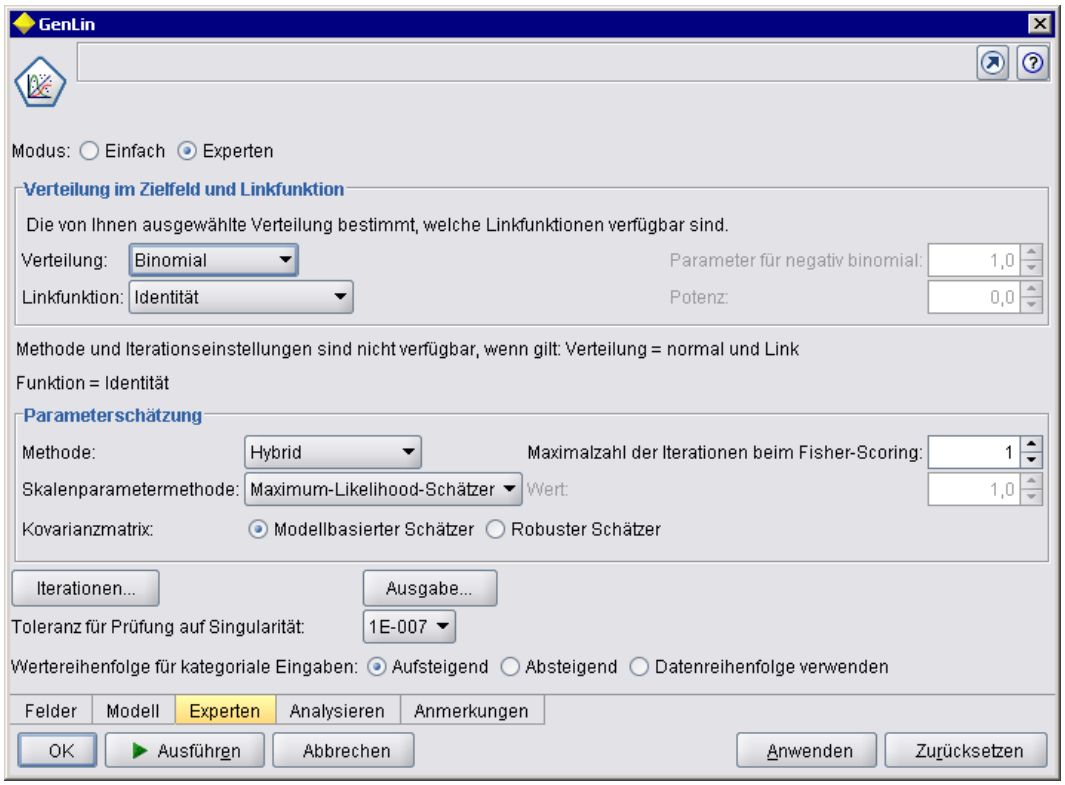


Einige sind jedoch immer noch deaktiviert, wie z. B. das Steuerelement Iterationen am unteren Rand des Dialogfelds. Dieses Steuerelement ist nur dann deaktiviert, wenn die **beiden** folgenden Bedingungen wahr sind:

- Verteilung ist auf Normal festgelegt
- Linkfunktion ist auf Identität festgelegt

Diese Kombination entspricht der Standardeinstellung für diese Registerkarte im Expertenmodus. Eine Änderung der Einstellung eines dieser Kombinationsfelder sorgt dafür, dass Iterationen aktiviert wird:

Abbildung 6-66
Änderung der Verteilungseinstellung aktiviert die Iterationsschaltfläche



Der entsprechende Code befindet sich in einer PropertiesSubPanel-Deklaration für die Schaltfläche Iterationen:

```
<PropertiesSubPanel buttonLabel="Iterations..." buttonLabelKey= ...
   <Enabled>
      <And>
         <Condition control="mode" op="equals" value="Expert"/>
            <Not>
               <And>
                  <Condition control="distribution" op="equals" value="NORMAL"/>
                  <Condition control="link_function" op="equals" value="IDENTITY"/>
               </And>
            </Not>
      </And>
   </Enabled>
    ...
</PropertiesSubPanel>
```

Das Element Condition im äußeren And-Abschnitt legt fest, dass der Modus auf Experten eingestellt sein muss, damit eine Änderung erfolgt. Wenn diese Bedingung wahr ist, ist im Abschnitt Not festgelegt, dass die Schaltfläche nicht aktiviert ist (d. h. deaktiviert ist), außer wenn *beide* Bedingungen des inneren And-Abschnitts erfüllt sind. Das heißt, im Expertenmodus wird Iterationen aktiviert, wenn entweder Verteilung oder Linkfunktion einen anderen Wert besitzen, als den jeweiligen Standardwert.

Steuerung der Anzeigeeigenschaften mit dem Element 'Visible'

Sie können auch Bedingungen verwenden, um dafür zu sorgen, dass Steuerelemente unter bestimmten Umständen angezeigt oder ausgeblendet werden. Hierzu wird das Element Visible verwendet.

Format

```
<Visible>
   <Condition .../>
   <And ... />
   <Or ... />
   <Not ... />
</Visible>
```

Das Element Condition gibt eine Bedingung an, die getestet wird, um zu ermitteln, ob das Steuerelement sichtbar ist.

Mit den Elementen And, Or und Not können zusammengesetzte Bedingungen festgelegt werden.

Für weitere Informationen siehe Thema Bedingungen in Kapitel 4 auf S. 91.

Beispiel

Das folgende Beispiel sorgt dafür, dass das angegebene Eigenschaftsfenster nur angezeigt wird, wenn die source_language-Bedingung erfüllt ist:

```
<PropertiesPanel>
   <Visible>
      <Condition control="source_language" op="equals" value="eng" />
   </Visible>
    ...
</PropertiesPanel>
```

Benutzerdefinierte Ausgabefenster

Für Modellausgabe-, Dokumentausgabe- und interaktive Ausgabeobjekte (nicht aber für Knoten) ist es möglich, dass eine Erweiterung das Standardausgabefenster vollständig durch ein benutzerdefiniertes Fenster ersetzt. Dies ist als standardmäßige java.awt.Frame-Klasse implementiert.

Um ein benutzerdefiniertes Fenster zu erstellen, müssen Sie eine Java-Klasse als frameClass-Attribut des UserInterface-Elements angeben:

```
<DocumentOutput id="my_modelling_node" type="modelBuilder" ...>
   <Properties>
      <Property name="use_custom_type" valueType="boolean" …/>
       ...
   </Eigenschaften (Properties)>
   <UserInterface frameClass="com.myextension.MyOutputFrame"/>
   ...
 </DocumentOutput>
```

Die angegebene Klasse muss die ExtensionObjectFrame-Schnittstelle implementieren, die durch die clientseitige API von CLEF definiert ist. Während der Lebensdauer des Fensters:

- Zugriff auf den zugrunde liegenden java.awt.Frame
- Fensterinitialisierung, einschließlich Zugriff auf das Ausgabeobjekt und die Sitzung
- Synchronisierung des Fensters und des zugrunde liegenden Objekts, wenn das Objekt gespeichert oder gelöscht wird
- Fensteranordnung

Für weitere Informationen siehe Thema Clientseitige API-Klassen in Kapitel 9 auf S. 221.

Kapitel
7

Hinzufügen eines Hilfesystems

Hilfesystemtypen

Wenn Sie eine CLEF-Erweiterung entwickeln, möchten Sie vermutlich auch ein Online-Hilfesystem hinzufügen, in der die Verwendung der Erweiterung beschrieben wird. CLEF unterstützt die folgenden Hilfesystemtypen:

- HTML Help
- JavaHelp

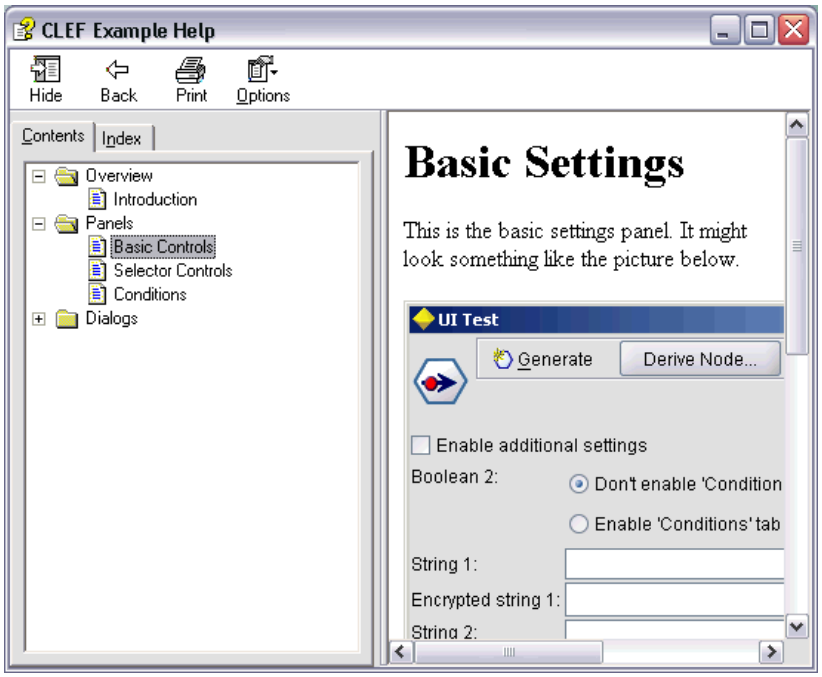
HTML Help

HTML Help ist ein proprietäres Hilfeformat von Microsoft, das nur auf Windows-Plattformen ausgeführt werden kann. Ein HTML Help-System besteht aus mehreren .htm- oder .html-Dateien, die in eine komprimierte Datei mit der Erweiterung .chm kompiliert werden. Auch das Hilfesystem von IBM® SPSS® Modeler wird im HTML Help-Format bereitgestellt.

HTML Help unterstützt Inhaltsverzeichnis, Index, Volltextsuche sowie Glossarbegriffe in Pop-up-Fenstern. Die .htm- oder .html-Quellendateien mit den einzelnen Themen können in einem HTML-Editor oder mit einem professionellen Hilfe-Authoringtool erstellt werden. Die .chm-Datei kann mit HTML Help Workshop kompiliert werden. Dieses Tool erhalten Sie als kostenloses Download auf der Microsoft Download Centre-Website (Informationen zur Erstellung von .chm-Dateien finden Sie im Hilfesystem von HTML Help Workshop). Alternativ können Sie zur Kompilierung Ihrer Themendateien und der zugehörigen Grafikdateien in eine .chm-Datei jedes Hilfe-Authoringtool verwenden, das das HTML Help-Format unterstützt.

Der HTML Help-Browser sieht wie folgt aus:

Abbildung 7-1
HTML Help-Browser



JavaHelp

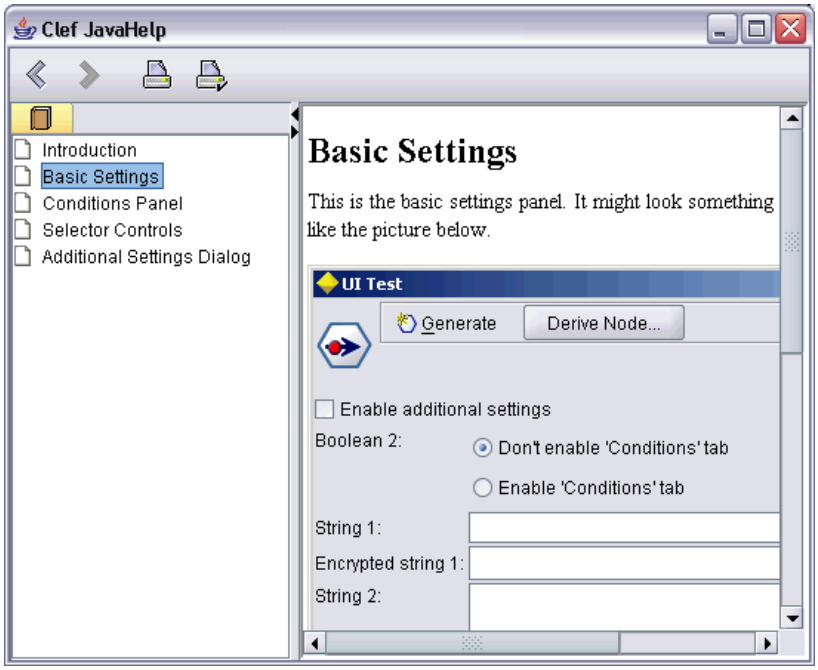
JavaHelp ist ein von Sun Microsystems entwickeltes Open Source-Hilfeformat, das auf jeder Plattform ausgeführt werden kann, die Java unterstützt. Ein JavaHelp-System besteht aus den folgenden Dateien:

- Die .htm- oder .html-Quellendateien mit den einzelnen Themen
- Die in den Themen verwendeten Grafikdateien
- Eine Helpset-Datei (mit der Erweiterung .hs), die das Hilfesystem steuert
- Die Datei map.xml, die die Themen-IDs und die Themendateien einander zuordnet und das Fenster für die Anzeige der Hilfethemen definiert
- Die Datei index.xml, die die Indexeinträge enthält
- Die Datei toc.xml, die die Einträge des Inhaltsverzeichnisses enthält

JavaHelp unterstützt Inhaltsverzeichnis, Index, Volltextsuche und Glossar. Die .htm- oder .html-Quellendateien können in einem HTML-Editor oder mit einem professionellen Hilfe-Authoringtool erstellt werden. Zudem benötigen Sie die JavaHelp-Software, die Sie als kostenloses Download auf der Sun Developer Network-Website erhalten (weitere Informationen finden Sie im Benutzerhandbuch *JavaHelp System User's Guide*, das Sie ebenfalls von dieser Website herunterladen können).

Der JavaHelp-Browser sieht wie folgt aus:

Abbildung 7-2
JavaHelp-Browser



Implementieren eines Hilfesystems

In diesem Abschnitt wird beschrieben, wie die relevanten Komponenten des Hilfesystems in der Spezifikationsdatei festgelegt werden.

Festlegen von Speicherort und Typ des Hilfesystems

Sofern einer Erweiterung ein Hilfesystem hinzugefügt wird, wird dessen Typ im Abschnitt “Resources” der Spezifikationsdatei der Erweiterung im Element HelpInfo angegeben.

Format

```
<Resources>
 ...
   <HelpInfo id="Name" type="Hilfetyp" path="Hilfepfad" helpset="Helpset_Speicherort"
    default="Standard_Themen_ID" />
 ...
</Resources>
```

Dabei gilt:

id (erforderlich) ist die Kennung für die Hilfeinformationen dieser Erweiterung.

type (erforderlich) gibt einen der folgenden Hilfetypen an:

- htmlhelp – das Hilfesystem hat das Format HTML Help und ist in der durch das Attribut path angegebenen .chm-Datei enthalten.
- javahelp – das Hilfesystem hat das Format JavaHelp und verwendet die durch das Attribut helpset angegebene Helpset-Datei (.hs) sowie die Quellen- und zugehörigen Dateien der Hilfe.

Beim Hilfetyp htmlhelp ist das folgende zusätzliche Attribut erforderlich:

- path – der Speicherort (relativ zur Spezifikationsdatei) und der Name der .chm-Datei mit dem Hilfesystem.

Beim Hilfetyp javahelp sind die folgenden zusätzlichen Attribute erforderlich:

- helpset – der Speicherort (relativ zur Spezifikationsdatei) und der Name der zu verwendenden .hs-Datei mit dem Helpset.
- default – die Kennung des Standardthemas, das angezeigt wird, wenn für eine Registerkarte kein bestimmtes Thema angegeben wurde.

Wenn kein HelpInfo-Element angegeben ist, wird der Erweiterung keine Hilfe hinzugefügt.

Beispiele

Das erste Beispiel zeigt ein HelpInfo-Element für HTML Help:

```
<HelpInfo id="help" type="htmlhelp" path="help/mynode.chm" />
```

Für ein JavaHelp-System würden die gleichen Angaben wie folgt aussehen:

```
<HelpInfo id="help" type="javahelp" helpset="help/mynode.hs"/>
```

Bei einem JavaHelp-System müssen sich die zugehörigen Dateien (Bilder sowie die Dateien map.xml, index.xml und toc.xml) im gleichen Ordner befinden wie die .hs-Helpset-Datei.

Festlegen des anzuzeigenden Hilfethemas

Für Knotendialogfelder, Registerkarten und untergeordnete Fenster der Eigenschaftenseiten können Sie festlegen, welches Hilfethema angezeigt wird, wenn der Benutzer im jeweiligen Bereich Hilfe aufruft. Dazu verwenden Sie in der Spezifikation für den Knoten, die Registerkarte oder den Eigenschaftenbereich das Attribut helpLink.

Wenn kein helpLink-Attribut angegeben ist, wird das Standardthema des Hilfesystems angezeigt, sobald der Benutzer die Hilfe aufruft.

Weitere Informationen zum Attribut helpLink finden Sie in den Abschnitten Knoten auf S. 62, Registerkarten auf S. 143 und Eigenschaftsunterfenster auf S. 158.

Beispiel

Dieses Beispiel geht von einem HTML Help-System aus. Es zeigt, wie in Abhängigkeit des Fensterfokus beim Aufrufen der Hilfe verschiedene kontextsensitive Themen angezeigt werden.

```
<Resources>
    ...
   <HelpInfo id="help" type="htmlhelp" path="help/mynode.chm"/>
    ...
</Resources>
 ...
<Node id="mynode" scriptName="my_node" type="dataTransformer" palette="recordOp"
 label="Sorter" description="Sortiert eine Datendatei" >
 ...
   <Tabs defaultTab="1">
      <Tab label="Grundlegende Steuerelemente" labelKey="basicControlsTab.LABEL"
       helpLink="basic_controls.htm">
         <PropertiesPanel>
          ...
            <PropertiesSubPanel buttonLabel="Zusätzliche Einstellungen..."
             buttonLabelKey="AdditionalOptions.LABEL" dialogTitle="Zusätzliche Einstellungen"
             dialogTitleKey="AdditionalOptionsDialog.LABEL" helpLink="addsettingsdlg.htm">
 ...
      </Tab>
      <Tab label="Selector Controls" labelKey="selectorControlsTab.LABEL"
       helpLink="selector_controls.htm">
 ...
      </Tab>
 ...
</Node>
```

In diesem Beispiel ist Folgendes festgelegt: Wenn sich der Fokus beim Aufrufen der Hilfe auf der Registerkarte “Basic Controls” befindet, wird das Hilfethema basic_controls.htm der Hilfedatei mynode.chm angezeigt. Wenn der Benutzer danach auf die Schaltfläche Additional settings klickt, um das Dialogfeld “Additional Settings” zu öffnen, und dort ebenfalls Hilfe aufruft, wird das Thema addsettingsdlg.htm angezeigt. Schließt der Benutzer daraufhin das Dialogfeld “Additional Settings”, klickt dafür aber auf die Registerkarte “Selector Controls” und ruft auch dort die Hilfe auf, wird das Thema selector_controls.htm angezeigt.

Bei einem JavaHelp-System muss der Wert des Attributs helpLink mit dem Wert des Attributs target der Datei map.xml übereinstimmen. Wenn die Datei map.xml beispielsweise Folgendes enthält:

```
<Map-Version="1.0">
 ...
   <mapID target="basic_controls" url="basic_controls.htm"/>
 ...
</map>
```

muss das entsprechende helpLink-Attribut den folgenden Wert haben:

```
helpLink="basic_controls"
```

Diese Zuordnung ist wichtig, da JavaHelp beim Aufrufen der Hilfe den Wert des Attributs target liest und ihm den zugehörigen url-Wert zuordnet, um die anzuzeigende Hilfedatei zu finden.

Kapitel 8

Lokalisierung und Eingabehilfen

Einführung

Mit **Lokalisierung** bezeichnet man die Anpassung von Software, Hilfe und Dokumentation an die Sprache und die Eigenheiten eines bestimmten Landes. Sie umfasst die Übersetzung der Benutzeroberfläche, der Hilfe und der Dokumentation sowie das Testen des Systems mit der entsprechenden Ländereinstellung. Wenn Sie Ihre Erweiterung für ein internationales Publikum entwickeln, können Sie lokalisierte Versionen der Erweiterung bereitstellen.

Eingabehilfen sind Funktionen der Benutzeroberfläche, die den Zugriff auf das System für Personen mit gewissen körperlichen Beeinträchtigungen, beispielsweise mit Sehschwächen oder eingeschränkten feinmotorischen Fähigkeiten, erleichtern.

Lokalisierung

IBM® SPSS® Modeler wurde für verschiedene Länder und Sprachen lokalisiert. Wenn der Benutzer die Windows-Ländereinstellung (Regions- und Sprachoptionen) auf seine eigene Sprache eingestellt und diese Sprache unterstützt wird, werden die Standardkomponenten der Benutzeroberfläche von IBM® SPSS® Modeler in dieser Sprache angezeigt. Zu den lokalisierten Komponenten zählen unter anderem:

- Systemmenüs und Menüelemente
- Systemschaltflächen (Generieren, OK, Ausführen, Abbrechen, Anwenden, Zurücksetzen)
- Registerkarten in Standarddialogfeldern (Anmerkungen und Debuggen, sofern verwendet)
- Fehler- und Systemmeldungen (z. B. “Dieses Objekt wurde nicht gespeichert.”)

Diese Standardkomponenten von SPSS Modeler werden auch in Ihrer Erweiterung automatisch in der ausgewählten Sprache angezeigt, sofern diese von SPSS Modeler unterstützt wird.

Für alle anderen Komponenten Ihrer Erweiterung stellt CLEF eine Lokalisierungsfunktion bereit. Folgende Komponenten können lokalisiert werden:

- Knotennamen (auf Palette und Zeichenbereich)
- Modellnamen (auf der Registerkarte “Modelle” des Managerfensters)
- Dokumentnamen (auf der Registerkarte “Ausgaben” des Managerfensters)
- Speicherort des Symbolbilds, das mit einer Aktion verknüpft ist
- QuickInfo-Text
- Hilfesysteme
- Knotendialogfelder:
  - Titelleistentext
  - Benutzerdefinierte Menüs und Menüelemente

  - Beschriftungen von Feldern, Eigenschaften, Schaltflächen und Registerkarten
  - Statischer Text
- System- und Fehlermeldungen

Textstrings sollten relativ kurz gehalten werden, damit auch bei Übersetzungen genügend Platz vorhanden ist.

System- und Fehlermeldungen können durch eine Kombination aus Spezifikationsdatei, Eigenschaftendateien und dem Server-seitigen API lokalisiert werden. Für weitere Informationen siehe Thema Statusdetaildokument (StatusDetail-Dokument) in Kapitel 9 auf S. 243.

Eigenschaftendateien

Die Textstrings der lokalisierbaren Komponenten werden in sogenannten **Eigenschaftendateien** gespeichert. Diese verwenden zum Speichern der Lokalisierungsressourcenbündel ein Java-Standardformat. Jede Eigenschaftendatei enthält jeweils einen Datensatz für jede lokalisierte Komponente der Erweiterung. Da in jedem Datensatz ein Feld dem labelKey-Attribut der Spezifikationsdatei entspricht, kann CLEF den jeweils passenden lokalisierten Textstring aus der Eigenschaftendatei einlesen und ihn an der richtigen Stelle auf dem Bildschirm anzeigen.

Eigenschaftendateien müssen die Dateinamenerweiterung .properties haben und im gleichen Verzeichnis gespeichert sein, in dem sich auch die Spezifikationsdatei des Knotens befindet, zu der die Datei gehört. IBM® SPSS® Modeler sucht zunächst nach der Standardeigenschaftendatei, die folgenden Namen hat:

path.properties

Dabei ist *Pfad* der Wert des path-Attributs des Bundle-Elements (im Abschnitt “Resources”), das das Eigenschaftenbündel definiert. Beispiel:

```
<Bundle id="bundle" path="my_resources"/>
```

Wenn keine Standardeigenschaftendatei vorliegt, ruft SPSS Modeler die Textstrings aus den Definitionen der Spezifikationsdatei ab.

Für jede von der Lokalisierung unterstützte Sprache muss eine eigene Eigenschaftendatei vorliegen. Die Dateien für alle Sprachen, mit Ausnahme der Standardsprache, werden durch ein Suffix im Dateinamen unterschieden. Beispiel:

```
my_ressources.properties
my_ressources_de.properties
my_ressources_fr.properties
```

Die Suffixe entsprechen den zweistelligen ISO 639-1-Standardsprachcodes.

Jeder Datensatz einer Eigenschaftendatei hat das folgende Format:

id=Textstring

Dabei gilt:

id ist die Kennung eines buttonLabelKey-, descriptionKey-, dialogTitleKey-, falseLabelKey-, imagePathKey-, labelKey-, messageKey-, textKey- oder trueLabelKey-Attributs in der Spezifikationsdatei. Diese Kennung hat in der Regel das Suffix .LABEL, damit sie leichter erkannt wird. Sie kann aber auch ein beliebiges Suffix oder gar kein Suffix aufweisen, je nachdem, wie sie in der Spezifikationsdatei angegeben ist.

Textstring ist der Text für die Komponente.

Beispiel: Lokalisierung einer Registerkarte

In diesem Beispiel, das die Lokalisierung einer Registerkarte eines Knotendialogfelds zeigt, werden zwei Eigenschaftendateien verwendet: die englische Standardversion und die französische Version. Die Lokalisierung sieht wie folgt aus:

*Erweiterungsordner***my_resources**.properties
*Erweiterungsordner***my_resources**_fr.properties

Dabei ist *Erweiterungsordner* der Ordner, in dem sich die Spezifikationsdatei der Erweiterung befindet.

In der Spezifikationsdatei werden die Eigenschaftendateien durch ein Bundle-Element im Abschnitt Resources referenziert:

```
<Resources>
   <Bundle id="bundle" type="properties" path="my_resources"/>
</Resources>
```

Das path-Attribut darf weder Spracherweiterungen noch das Suffix .properties enthalten.

Die restlichen relevanten Abschnitte der Spezifikationsdateien sehen wie folgt aus:

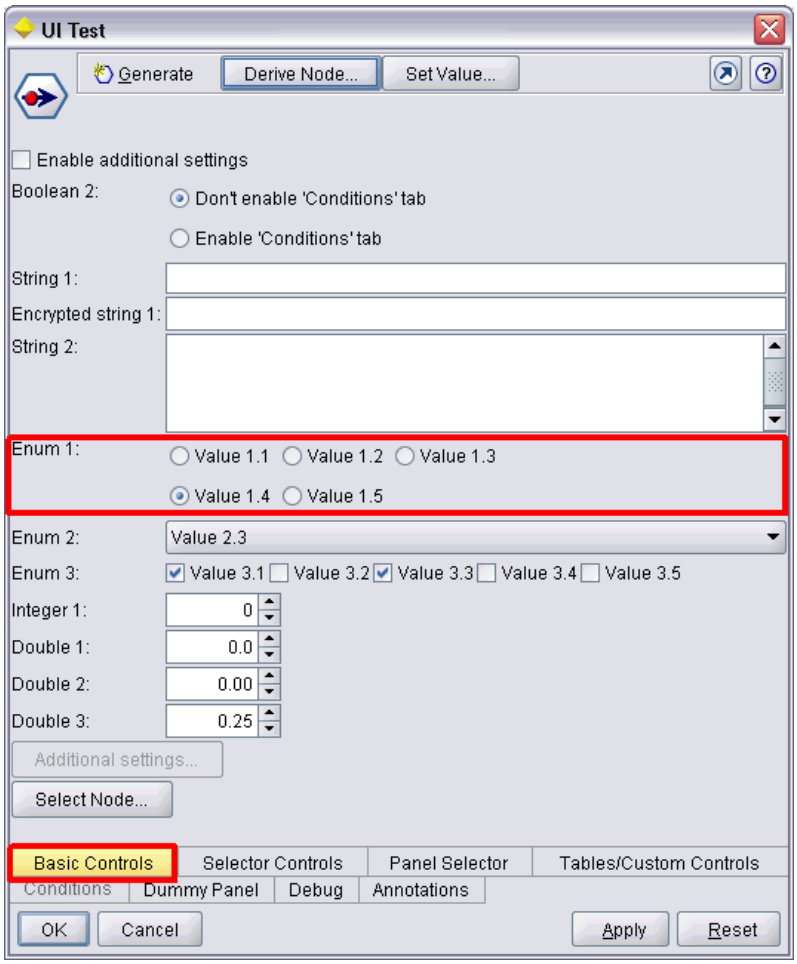
```
<Node id="uiTest" scriptName="ui_test" type="dataTransformer" palette="recordOp" label="UI Test" ... >
   <Properties>
      <Property name="enum1" valueType="enum" defaultValue="value4">
         <Enumeration>
            <Enum value="value1" label="Value 1.1" labelKey="enum1.value1.LABEL"/>
            <Enum value="value2" label="Value 1.2" labelKey="enum1.value2.LABEL"/>
            <Enum value="value3" label="Value 1.3" labelKey="enum1.value3.LABEL"/>
            <Enum value="value4" label="Value 1.4" labelKey="enum1.value4.LABEL"/>
            <Enum value="value5" label="Value 1.5" labelKey="enum1.value5.LABEL"/>
         </Enumeration>
      </Property>
   </Properties>
...
   <UserInterface ... >
      <Tabs defaultTab="1">
         <Tab label="Basic Controls" labelKey="basicControlsTab.LABEL" ... >
    ...
   </UserInterface>
...
</Node>
```

Die englische Version der Eigenschaftendatei enthält die folgenden Datensätze:

```
basicControlsTab.LABEL=Basic Controls
enum1.value1.LABEL=Value 1.1
enum1.value2.LABEL=Value 1.2
enum1.value3.LABEL=Value 1.3
enum1.value4.LABEL=Value 1.4
enum1.value5.LABEL=Value 1.5
```

In der folgenden Abbildung sind die Teile des Dialogfelds hervorgehoben, auf die sich diese Datensätze auswirken:

Abbildung 8-1
Nicht lokalisierte Registerkarte



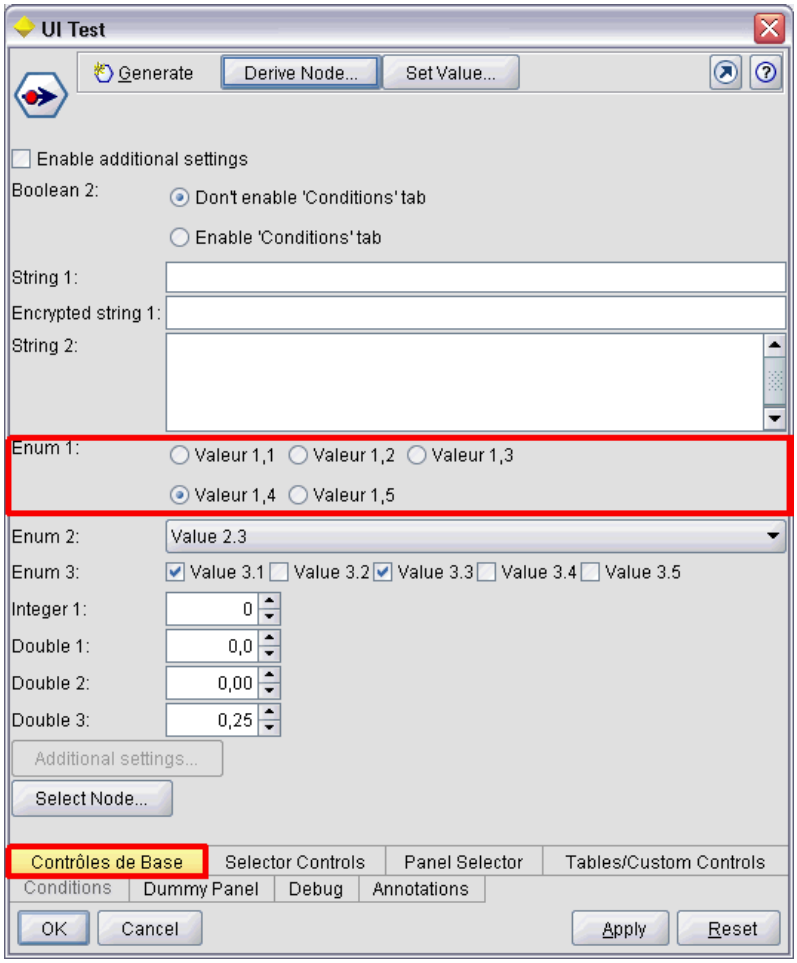
Der entsprechende Abschnitt der französischen Version der Eigenschaftendatei (my_resources_fr.properties) sieht wie folgt aus:

```
basicControlsTab.LABEL=Contrôles de Base
enum1.value1.LABEL=Valeur 1,1
enum1.value2.LABEL=Valeur 1,2
enum1.value3.LABEL=Valeur 1,3
```

```
enum1.value4.LABEL=Valeur 1,4
enum1.value5.LABEL=Valeur 1,5
```

Diese Datensätze bewirken, dass die entsprechenden Bildschirmkomponenten französisch angezeigt werden:

Abbildung 8-2
Lokalisierte Registerkarte



Die vier Schaltflächen am unteren Rand des Bildschirms werden durch die Standardfunktion von IBM® SPSS® Modeler und nicht durch CLEF lokalisiert.

Beispiel: Verwendung von Sonderzeichen

In der Eigenschaftendatei müssen Sie für Sonderzeichen in Standard-ASCII-Textstrings Unicode-Escape-Sequenzen verwenden. Nachfolgend sehen Sie einen Ausschnitt einer Eigenschaftendatei, die in die französische Sprache lokalisiert wurde:

```
Genlinnode.LABEL=Lin\u00e9aire g\u00e9n\u00e9ralis\u00e9
```

```
Fields.LABEL=Champs
Model.LABEL=Mod\u00e8le
Expert.LABEL=Expert

inputFields.LABEL=Entr\u00e9es
targetField.LABEL=Cible
 ...
```

Für Sprachen wie Japanisch und Chinesisch, die keine lateinischen Zeichen verwenden, dürfen Sie für die Textstrings nur Unicode-Escape-Sequenzen verwenden. Nachfolgend sehen Sie die gleichen Datensätze in der japanischen Lokalisierung:

```
Genlinnode.LABEL=\u4e00\u822c\u5316\u7dda\u578b

Fields.LABEL=\u30d5\u30a3\u30fc\u30eb\u30c9
Model.LABEL=\u30e2\u30c7\u30eb
Expert.LABEL=\u30a8\u30ad\u30b9\u30d1\u30fc\u30c8

inputFields.LABEL=\u5165\u529b
targetField.LABEL=\u5bfe\u8c61
 ...
```

Hilfedateien

Wenn Sie eine Erweiterung lokalisieren, die über ein Hilfesystem verfügt, sollten Sie auch eine lokalisierte Version des Hilfesystems bereitstellen. Normalerweise wird das Hilfesystem in jede Sprache lokalisiert, in der die Erweiterung zur Verfügung steht.

Lokalisieren eines HTML Help-Systems

Wenn die Erweiterung, die Sie lokalisieren, über eine HTML Help-Datei verfügt (eine Datei mit dem Suffix .chm), ersetzen Sie einfach die Standard-.chm-Datei durch die lokalisierte Version. Weitere Informationen zu HTML Help-Systemen finden Sie im Abschnitt HTML Help auf S. 206.

So erstellen Sie eine lokalisierte .chm-Datei:

- Übersetzen Sie die einzelnen Hilfethemen (.htm- oder .html-Dateien), aus denen das Hilfesystem besteht. Die Dateinamen dürfen Sie nicht ändern.
- Lokalisieren Sie gegebenenfalls auch die im Hilfesystem enthaltenen Grafiken (erstellen Sie z. B. lokalisierte Screenshots).
- Kompilieren Sie die übersetzten Dateien mit Microsoft HTML Help Workshop oder einem anderen Hilfe-Authoringtool in eine lokalisierte .chm-Datei.
- Testen Sie das Hilfesystem im lokalisierten Knoten. Für weitere Informationen siehe Thema Testen eines lokalisierten CLEF-Knotens auf S. 217.

Lokalisieren eines JavaHelp-Systems

Wenn die Erweiterung, die Sie lokalisieren, über ein JavaHelp-System verfügt, müssen Sie für jede unterstützte Sprache eine lokalisierte Version der Quellendateien der Hilfe bereitstellen. Die Anzeige der richtigen lokalisierten Version erfolgt bei JavaHelp automatisch, sofern eine Lokalisierung vorliegt. Für weitere Informationen siehe Thema JavaHelp in Kapitel 7 auf S. 207.

So erstellen Sie ein lokalisiertes JavaHelp-System:

- Übersetzen Sie die einzelnen Hilfethemen (.htm- oder .html-Dateien), aus denen das Hilfesystem besteht. Die Dateinamen dürfen Sie nicht ändern.
- Lokalisieren Sie gegebenenfalls auch die im Hilfesystem enthaltenen Grafiken (erstellen Sie z. B. lokalisierte Screenshots).
- Generieren Sie die Helpset-Datei und andere erforderliche Dateien (map.xml, index.xml und toc.xml).
- Testen Sie das Hilfesystem im lokalisierten Knoten. Für weitere Informationen siehe Thema Testen eines lokalisierten CLEF-Knotens auf S. 217.

Testen eines lokalisierten CLEF-Knotens

So testen Sie einen lokalisierten Knoten und das zugehörige Hilfesystem:

- Gehen Sie zum Abschnitt “Resources” der Spezifikationsdatei des lokalisierten Knotens und setzen Sie dort das path-Attribut des HelpInfo-Elements auf die lokalisierte .chm- bzw. .hs-Datei. Beispiel für HTML Help:

```
<Resources>
 ...
   <HelpInfo id="help" type="HTMLHelp" path="help/mynode_fr.chm "/>
</Resources>
```

Beispiel für JavaHelp:

```
<Resources>
 ...
   <HelpInfo id="help" type="javahelp" helpset="help/mynode_fr.hs "/>
</Resources>
```

- Kopieren Sie die lokalisierte .chm- oder .jar-Datei in das im path-Attribut angegebene Verzeichnis.
- Legen Sie die Windows-Region für die gewünschte Ländereinstellung fest:
  Systemsteuerung > Regions- und Sprachoptionen > Regionale Einstellungen > Standards und Formate > <language>
- Starten Sie IBM® SPSS® Modeler und vergewissern Sie sich, dass es in der gewünschten Sprache angezeigt wird.
- Fügen Sie den lokalisierten Knoten zu SPSS Modeler hinzu. Für weitere Informationen siehe Thema Testen einer CLEF-Erweiterung in Kapitel 10 auf S. 251.

- Ziehen Sie eine Kopie des Knotens in den Zeichenbereich.

  Öffnen Sie das Knotendialogfeld und vergewissern Sie sich, dass es korrekt in der gewünschten Sprache angezeigt wird.

- Klicken Sie auf die Hilfeschaltfläche des Dialogfelds und vergewissern Sie sich, dass das richtige Hilfethema in der gewünschten Sprache angezeigt wird.

Zugriffsmöglichkeiten

CLEF stellt sämtliche Standardeingabehilfen von IBM® SPSS® Modeler wie Direktzugriffstasten für Mausaktionen und Unterstützung für Bildschirm-Lesesysteme bereit. Für weitere Informationen siehe Thema Übersicht über die Eingabehilfen in IBM SPSS Modeler in Anhang A in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Darüber hinaus können Sie für einen CLEF-Knoten als Eingabehilfe benutzerdefinierte QuickInfos bereitstellen.

Sie können auch Tastenkombinationen angeben, um Endbenutzern einen alternativen Zugriff auf verschiedene Funktionen der Benutzeroberfläche zu bieten, die Sie in CLEF hinzugefügt haben. Für weitere Informationen siehe Thema Zugriffstasten und Tastenkombinationen in Kapitel 6 auf S. 145.

Für Aktionsschaltflächen und für alle Bildschirmkomponenten, die als Steuerelemente fungieren (z. B. Kontrollkästchen und Optionsschaltflächen), können Sie folgende Texte definieren:

- Label
- Beschreibung

Die Beschriftung (**Label**) ist der Bildschirmtext, der als Name einer Komponente angezeigt wird und von Bildschirm-Lesesystemen gelesen werden kann. Für Benutzer mit Sehschwächen kann die Größe der Beschriftung auf der Registerkarte “Anzeige” des Dialogfelds “Benutzeroptionen” geändert werden. Dieses Dialogfeld öffnen Sie mit folgender Befehlsfolge:

Werkzeuge > Benutzeroptionen

Die Beschreibung (**Description**) ist der QuickInfo-Text, der angezeigt wird, wenn Sie mit dem Mauszeiger auf eine Bildschirmkomponente zeigen. QuickInfos bieten Raum für weitere Informationen und beschreiben eine Bildschirmkomponente ausführlicher als die Beschriftung. Auch QuickInfos können von Bildschirm-Lesesystemen gelesen werden, sofern diese entsprechend konfiguriert sind.

Beschriftungen und Beschreibungen werden in dem Element, das die Komponente in der Spezifikationsdatei definiert, mit den Attributen `label` und `description` festgelegt. Beide können mittels der Attribute `labelKey` bzw. `descriptionKey` lokalisiert werden.

Beispiel

Nachfolgend wird die Verwendung der Beschriftungs- und Beschreibungsfunktion am Beispiel einer Aktionsschaltfläche veranschaulicht.

```
<Action id="setValue" label="Wert festlegen..." labelKey="setValue.LABEL"
    description="Zum Festlegen eines Werts" descriptionKey="setValue.TOOLTIP"/>
```

Kapitel
9

Programmierung

Informationen zur Programmierung von CLEF-Knoten

Ein Knoten kann auch Verarbeitungsschritte durchführen, die sich nicht in der Spezifikationsdatei festlegen lassen. Hierfür stellt CLEF die folgenden APIs (Anwendungsprogrammierschnittstellen) zur Verfügung, die aus Ihren Programmen aufgerufen werden können:

- **Clientseitige API**: Eine Java-API, die von Erweiterungen verwendet werden kann, für die Steuerelemente, Benutzeroberflächenkomponenten oder Interaktivitätselemente erforderlich sind, die nicht in der Spezifikationsdatei bereitgestellt werden können.
- **Predictive Server-API (PSAPI)**: Eine Java-API, die Funktionalität von IBM® SPSS® Modeler bereitstellt für Anwendungen, die Data-Mining-Funktionen und Mechanismen für die Vorhersageanalyse benötigen. Der PSAPI und der Data-Mining-Workbench von SPSS Modeler liegt die gleiche Technologie zugrunde.
- **Serverseitige API**: Eine C-basierte API, die Aspekte wie die Festlegung und das Abrufen von Ausführungseinstellungen, die Persistenz dieser Einstellungen, Ausführungsrückmeldungen, die Auftragssteuerung (z. B. Unterbrechung der Ausführung), die SQL-Generierung und die zurückgegebenen Objekte definiert.

Dokumentation zu den CLEF-APIs

Die nachfolgenden Abschnitte bieten einen Überblick über die client- und serverseitigen APIs. Eine ausführlichere Dokumentation zu den APIs ist jeder IBM® SPSS® Modeler-Installation in einer .zip-Datei beigefügt, die vor der Verwendung extrahiert werden muss.

So extrahieren Sie die API-Dokumentation:

▶ Suchen Sie auf den DVD des Produkts im Ordner *\Documentation\en* nach der Datei *clef_apidoc.zip*.

▶ Extrahieren Sie den Inhalt der .zip-Datei mit WinZip oder einem ähnlichen Dekomprimierungstool in ein beliebiges Verzeichnis. Durch die Extraktion wird in diesem Verzeichnis der Unterordner *clef_apidoc* mit der gesamten API-Dokumentation erstellt.

So zeigen Sie die API-Dokumentation an:

▶ Wechseln Sie in den Unterordner *clef_apidoc* und öffnen Sie dort die Datei *clef_apidoc.htm*.

▶ Wählen Sie die gewünschte API-Dokumentation aus (clientseitige, serverseitige oder PSAPI-API).

Clientseitige API

CLEF enthält eine Reihe von Java-Klassen mit Methoden, die Sie für die clientseitige Verarbeitung verwenden können. Die Klasse DataModelProvider ermöglicht beispielsweise die Berechnung eines Ausgabedatenmodells, wenn Änderungen am Eingabedatenmodell zu komplex für die durch die Spezifikationsdatei bereitgestellten Funktionen sind.

Clientseitige API-Klassen

Im Folgenden sind die Client-seitigen Klassen aufgeführt.

Tabelle 9-1
Client-seitige API-Klassen

Class	Beschreibung
ActionHandler	Ermöglicht der Erweiterung die Verwaltung von Aktionen, die vom Benutzer über Menüoptionen und Symbolleistenschaltflächen angefordert werden
DataModelProvider	Ermöglicht Knoten, die komplexe Änderungen am Datenmodell vornehmen, die Verwendung von Java zur Berechnung des Ausgabedatenmodells
ExtensionObjectFrame	Legt die Funktionalität von Fenstern fest, die Modell- oder Dokumentausgaben enthalten
ExtensionObjectPanel	Legt die Funktionalität von Erweiterungsobjektfenstern fest
PropertyControl	Legt die Funktionalität benutzerdefinierter Systemsteuerungen für Eigenschaften in einem Eigenschaftenfenster fest

Ausführliche Informationen zu diesen Klassen erhalten Sie in der Dokumentation zur clientseitigen API. Für weitere Informationen siehe Thema Dokumentation zu den CLEF-APIs auf S. 220.

Verwenden der clientseitigen API

So fügen Sie einem CLEF-Knoten clientseitige Funktionsaufrufe hinzu:

- Erstellen Sie *.java*-Quellendateien mit den gewünschten Funktionsaufrufen.
- Kompilieren Sie die Quellendateien in *.class*-Dateien.
- Sie können mehrere *.class*-Dateien in einer *.jar*-Datei zusammenfassen und in der Spezifikationsdatei auf diese *.jar*-Datei verweisen. Beispiel:

```
<Resources>
 ...
   <JarFile id="selfjar" path="selflearning.jar"/>
 ...
</Resources>
```

In einigen Elementen von CLEF können Sie direkt auf eine Klasse verweisen. Beispielsweise können Sie im Attribut controlClass eines PropertyControl-Elements der Spezifikationsdatei, wie nachfolgend gezeigt, einen Verweis auf eine Klasse einfügen:

```
<PropertyControl property="target_field_values_specify" ...
   controlClass="com.spss.clef.selflearning.propertycontrols.list.CustomListControl" ... />
```

Dabei bezeichnet CustomListControl den Namen der Klasse, die die Systemsteuerung für Eigenschaften implementiert, und com.spss.clef.selflearning.propertycontrols.list bezeichnet den Pfad zu dieser Klasse innerhalb der *.jar*-Datei, die im JarFile-Element deklariert ist.

Für weitere Informationen siehe Thema Abschnitt ‘Ressourcen’ in Kapitel 4 auf S. 43.

Sehen Sie sich hierzu auch den Quellcode der in dieser Version bereitgestellten Beispielknoten an. Sie finden dort eventuell einige nützliche Anwendungsbeispiele. Für weitere Informationen siehe Thema Untersuchen des Quellcodes in Kapitel 3 auf S. 38.

Predictive Server-API (PSAPI)

Die PSAPI ist eine Programmierschnittstelle für die diesem Produkt zugrunde liegende Predictive Server-Technologie. Die wichtigsten Elemente der PSAPI sind als Java-Schnittstellen definiert. Die meisten dieser Schnittstellen sind mittels interner Klassen implementiert, die von der PSAPI bereitgestellt werden, aber nicht Bestandteil der PSAPI-Spezifikation sind. Der PSAPI-Benutzer soll dadurch vor Änderungen an der Predictive Server-Technologie geschützt werden (z. B. vor Architekturänderungen oder vor Änderungen am privaten Client-/Serverprotokoll).

Ausführliche Informationen zu diesen Klassen erhalten Sie in der PSAPI-Dokumentation. Für weitere Informationen siehe Thema Dokumentation zu den CLEF-APIs auf S. 220.

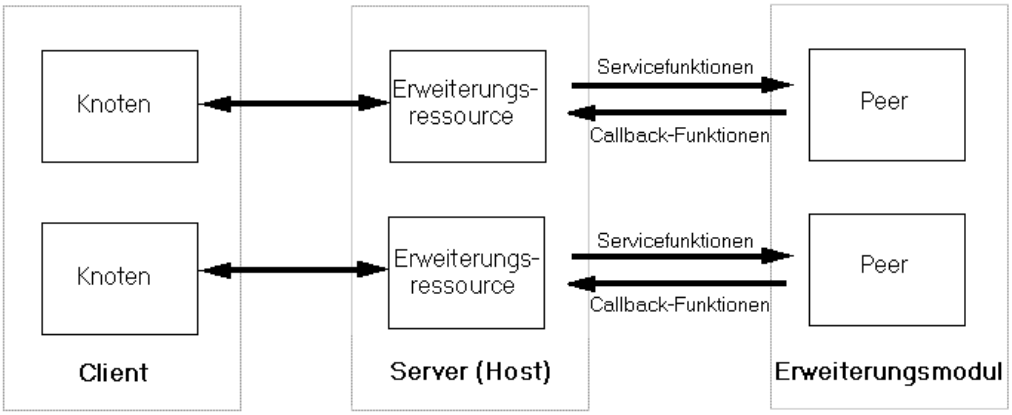
Serverseitige API

Die serverseitige API ist in der Programmiersprache C definiert, unterstützt aber auch Implementierungen in der Programmiersprache C++. Erweiterungsmodule können in C oder C++ direkt für die C-basierte API programmiert werden. Darüber hinaus können auch andere Programmiersprachen verwendet werden, sofern der Entwickler über eine Methode der Bindung an die CAPI verfügt. CLEF bietet zudem verschiedene C++-Helper-Quellendateien, die als Wrapper für einen Teil der CAPI fungieren können.

Architektur

Der **Knoten** einer Erweiterung auf dem Client wird auf dem Server durch einen **Peer** der Erweiterung ergänzt. Ein Peer wird durch ein **Erweiterungsmodul** definiert, das als gemeinsame Bibliothek auf dem Server implementiert wird. Die Kommunikation zwischen einem Knoten und seinem Peer wird über eine vom Server verwaltete Erweiterungs**ressource** vermittelt. Eine Ressource ruft zur Erstellung und Manipulation seines Peers die im Erweiterungsmodul definierten **Servicefunktionen** auf, während der Peer **Callback-Funktionen** verwendet, um Informationen und Services von seinem Host anzufordern.

Abbildung 9-1
CLEF API-Architektur



Servicefunktionen

Servicefunktionen werden durch das Erweiterungsmodul implementiert. Ein Erweiterungsmodul muss alle Funktionen implementieren, die als erforderlich gekennzeichnet sind. Die als nicht erforderlich gekennzeichneten Funktionen kann, muss es aber nicht implementieren.

Es gibt zwei Arten von Servicefunktionen:

- Modulfunktionen
- Peer-Funktionen

Die nachfolgenden Abschnitte bieten einen Überblick über die Servicefunktionen. Ausführlichere Beschreibungen dieser Funktionen finden Sie in der Dokumentation der serverseitigen API, die Sie wie folgt aufrufen:

▶ Wählen Sie im API-Dokumentationsfenster von CLEF die Option Server-side API overview (Überblick über die serverseitige API) aus.

▶ Klicken Sie auf die Registerkarte Module.

▶ Wählen Sie API Service Functions to be implemented by the extension module (Vom Erweiterungsmodul zu implementierende API-Servicefunktionen) aus.

Informationen zum Zugriff auf die API-Dokumentation von CLEF finden Sie im Abschnitt Dokumentation zu den CLEF-APIs auf S. 220.

Modulfunktionen

Modulfunktionen werden von nur einem Thread aufgerufen.

Funktion	Erforderlich?	Beschreibung
clemext_initialise	Ja	Initialisiert ein Erweiterungsmodul
clemext_cleanup	Ja	Gibt die durch ein Erweiterungsmodul reservierten Ressourcen wieder frei
clemext_getModuleInformation	Nein	Ruft Informationen über ein Erweiterungsmodul ab
clemext_create_peer	Ja	Erstellt eine Peer-Instanz für ein Erweiterungsmodul
clemext_destroy_peer	Ja	Entfernt eine Peer-Instanz

Peer-Funktionen

Peer-Funktionen werden auf das Handle einer Peer-Instanz angewendet, das von einem früheren clemext_create_peer-Aufruf zurückgegeben wurde. Peer-Funktionen können nur dann gleichzeitig von verschiedenen Threads aufgerufen werden, wenn sich die Peer-Handles der Funktionen unterscheiden. Die einzige Ausnahme hiervon ist die Funktion clemext_peer_cancelExecution. Diese kann, sofern sie definiert ist, jederzeit von jedem Thread aufgerufen werden, um die Ausführung eines anderen Threads, die zu lange dauert, abzubrechen.

Funktion	Erforderlich?	Beschreibung
clemext_peer_configure	Ja	Weist eine Peer-Instanz an, ihre Parameter und ihr Datenmodell auszuführen
clemext_peer_getDataModel	Ja	Ruft das Ausgabedatenmodell einer Peer-Instanz ab
clemext_peer_getCatalogueInformation	Nein	Ruft die Kataloginformationen eines Moduls ab
clemext_peer_getExecutionRequirements	Nein	Ruft die Ausführungsanforderungen einer Peer-Instanz ab
clemext_peer_beginExecution	Ja	Startet die Ausführung einer Peer-Instanz
clemext_peer_nextRecord	Ja	Geht zum nächsten Datensatz im Ergebnis-Set eines Peers
clemext_peer_getRecordValue	Ja	Gibt den Wert eines bestimmten Felds aus dem zuletzt abgerufenen Ergebnisdatensatz zurück
clemext_peer_endExecution	Ja	Zeigt einer Peer-Instanz an, dass die Ausführung abgeschlossen ist
clemext_peer_cancelExecution	Nein	Wird von einem anderen Thread aufgerufen, um einen beginExecution- oder nextRecord-Funktionsaufruf, der zu lange dauert, abzubrechen.

Funktion	Erforderlich?	Beschreibung
clemext_peer_getSQLGeneration	Nein	Generiert aus einer Peer-Instanz SQL-Anweisungen
clemext_peer_getErrorDetail	Ja	Ruft zusätzliche Informationen über den letzten modulspezifischen Fehler eines Peers ab

Die folgenden Peer-Funktionen sind für die interaktive Modellerstellung vorgesehen:

Funktion	Erforderlich?	Beschreibung
clemext_peer_beginInteraction	Nein	Beginnt die Interaktion mit einer Peer-Instanz
clemext_peer_request	Nein	Führt eine interaktive Anforderung an einem Peer aus
clemext_peer_getRequestReply	Nein	Ruft die Antwort der letzten, erfolgreich ausgeführten Anforderung ab
clemext_peer_endInteraction	Nein	Zeigt einer Peer-Instanz an, dass die Interaktion abgeschlossen ist

Callback-Funktionen

Wenn ein Erweiterungsmodul Informationen oder Services vom Hostprozess benötigt, muss es diese über ein **Callback** anfordern. Ein Callback wird auf ein **Handle** angewendet, also auf einen Zeiger, der das Ziel der Anforderung angibt.

Der Aufruf eines Callbacks erfolgt durch Übergabe des Handles desjenigen IBM® SPSS® Modeler-Objekts, an das der Aufruf gerichtet ist. Die Übergabe eines Handles in ein Erweiterungsmodul erfolgt in Form eines Parameters in einer Servicefunktion.

Falls eine Callback-Funktion fehlschlägt, sollte sie im zugehörigen modulspezifischen Fehlercode (gekennzeichnet durch CLEMEXTErrorCode) Informationen über den Fehler zurückgeben. Das Erweiterungsmodul kann seinerseits einen Callback-Fehler zurückgeben, indem es diese Informationen weiterleitet, damit diese vom Host untersucht werden können.

Folgende Typen von Callback-Funktionen stehen zur Verfügung:

- Hostfunktionen
- Knotenfunktionen
- Iteratorfunktionen
- Fortschrittsfunktionen
- Kanalfunktionen (nur für interaktive Modelle)

Die nachfolgenden Abschnitte bieten einen Überblick über die Callback-Funktionen. Ausführlichere Beschreibungen dieser Funktionen finden Sie in der Dokumentation der serverseitigen API, die Sie wie folgt aufrufen:

▶ Wählen Sie im API-Dokumentationsfenster von CLEF die Option Server-side API overview (Überblick über die serverseitige API) aus.

▶ Klicken Sie auf die Registerkarte Module.

▶ Wählen Sie General callbacks (Allgemeine Callbacks) aus.

Informationen zum Zugriff auf die API-Dokumentation von CLEF finden Sie im Abschnitt Dokumentation zu den CLEF-APIs auf S. 220.

Hostfunktionen

Hostfunktionen werden für ein Host-Handle definiert, das mittels clemext_initialise übergeben wird.

Funktion	Beschreibung
clemext_host_getHostInformation	Ruft statische Informationen zur Hostumgebung ab

Knotenfunktionen

Knotenfunktionen werden für ein Knoten-Handle definiert, das mittels clemext_create_peer übergeben wird.

Funktion	Beschreibung
clemext_node_getNodeInformation	Ruft statische Informationen zu einem Knoten ab
clemext_node_getParameters	Ruft die Parameter eines Knotens ab
clemext_node_getDataModel	Ruft das Eingabedatenmodell eines Knotens ab
clemext_node_getOutputDataModel	Ruft das Ausgabedatenmodell eines Knotens ab
clemext_node_getSQLGeneration	Ruft die bisherigen SQL-Generierungsinformationen eines Knotens ab
clemext_node_getPassword	Ruft die Klartextversion eines verschlüsselten Passworts ab
clemext_node_getFilePath	Ruft den Pfad einer Datei ab, die während der Ausführung zwischen Client und Server ausgetauscht wird

Iteratorfunktionen

Iteratorfunktionen werden für ein Iterator-Handle definiert, das mittels clemext_peer_beginExecution übergeben wird. Ein Iterator stellt einem Erweiterungsmodul ein Eingabe-Daten-Set bereit.

Funktion	Beschreibung
clemext_iterator_nextRecord	Geht zum nächsten Datensatz im Eingabe-Daten-Set
clemext_iterator_getRecordValue	Gibt den Wert eines bestimmten Felds aus dem zuletzt abgerufenen Eingabedatensatz zurück
clemext_iterator_rewind	Setzt den Status des Eingabe-Daten-Sets zurück, damit der nächste nextRecord-Aufruf beim ersten Datensatz des Eingabe-Daten-Sets beginnt

Fortschrittsfunktionen

Fortschrittsfunktionen werden für ein Fortschritts-Handle definiert, das mittels clemext_peer_beginExecution übergeben wird.

Funktion	Beschreibung
clemext_progress_report	Meldet den Fortschritt an den Host

Kanalfunktionen

Kanalfunktionen werden nur bei interaktiven Modellen verwendet. Sie werden für ein Kanal-Handle definiert, das mittels clemext_peer_beginInteraction übergeben wird.

Funktion	Beschreibung
clemext_channel_send	Sendet eine asynchrone Nachricht an den Client, über die ein Peer-Thread Fortschritts- und Statusinformationen meldet

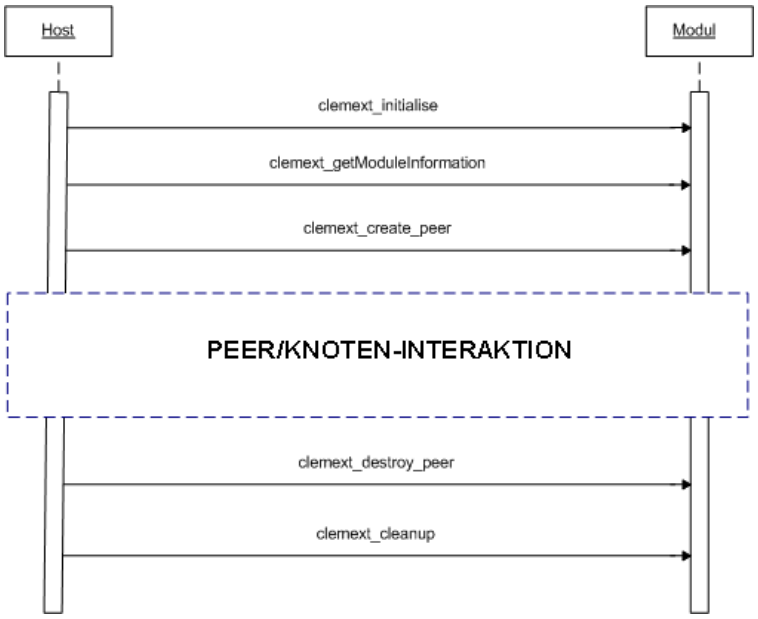
Prozessfluss

Ein Erweiterungsmodul ruft zur Ausführung seiner Verarbeitungsaufgaben verschiedene Service- und Callback-Funktionen auf. Welche Funktionen aufgerufen werden, richtet sich nach den Verarbeitungsschritten, die das Modul ausführen muss.

Beispiel

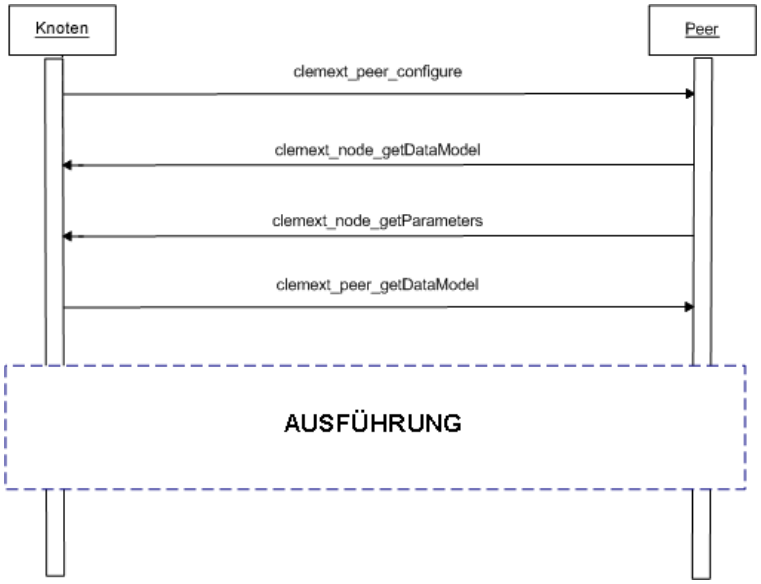
Das Flussdiagramm einer typischen Modulausführung sieht wie folgt aus:

Abbildung 9-2
Typischer Prozessfluss



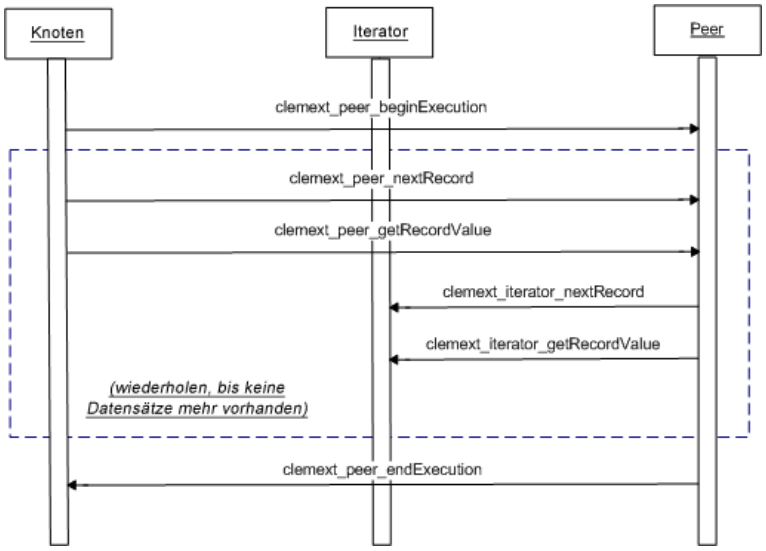
Der Interaktionsblock zwischen Peer und Knoten sieht so aus:

Abbildung 9-3
Typischer Peer/Knoten-Interaktionsblock



Ein typischer Ausführungsblock sieht so aus:

Abbildung 9-4
Typischer Ausführungsblock



Anmerkungen:

- Ein Erweiterungsmodul kann bereits beim Starten des Servers in den Serverprozess von IBM® SPSS® Modeler geladen werden. Es kann aber auch erst später geladen werden, wenn seine Services zum ersten Mal benötigt werden.
- Nach dem Laden des Moduls ruft der Host einmal die Servicefunktion clemext_initialise auf.

- Nach dem Laden und der Initialisierung des Moduls kann der Host das Modul anhand der Servicefunktion clemext_getModuleInformation abfragen.
- Nachdem das Modul geladen wurde, werden seine Services anhand der vom Modul bereitgestellten Peer-Objekte aufgerufen. Im Modul selbst wird das Peer-Objekt durch die Servicefunktion clemext_create_peer als Gegenstück zum Knotenobjekt des Hosts erstellt. Über dieses Peer-Objekt wird die Ausführung einer Task nach Angabe der Hostanwendung verwaltet. Es können mehrere Peer-Objekte des gleichen Typs vorhanden und gleichzeitig innerhalb eines Prozesses ausgeführt werden.
- Nach der Erstellung eines Peer-Objekts kann dieses durch die Servicefunktion clemext_peer_configure konfiguriert werden.
- Nach der Konfiguration kann der Peer Callback-Funktionen ausführen (z. B. clemext_node_getDataModel und clemext_node_getParameters), um Informationen vom Client abzurufen.
- SPSS Modeler ruft das Ausgabedatenmodell einer Peer-Instanz anhand der Servicefunktion clemext_peer_getDataModel ab.
- Die Ausführung einer Peer-Instanz beginnt mit dem Aufruf der Servicefunktion clemext_peer_beginExecution.
- Die Servicefunktion clemext_peer_nextRecord verschiebt den Fokus zum nächsten Datensatz im Ergebnis-Set des Peers (bzw. zum ersten Datensatz, wenn die Funktion zum ersten Mal aufgerufen wird). Danach wird die Servicefunktion clemext_peer_getRecordValue ausgeführt, die den Wert eines bestimmten Felds aus dem aktuellen Datensatz abruft.
- Die Iterator-Callback-Funktionen clemext_iterator_nextRecord und clemext_iterator_getRecordValue können vom CLEF-Modul aufgerufen werden, um die einzelnen Eingabedatensätze abzufragen und die Werte bestimmter Felder zurückzugeben.
- Die Ausführung einer Peer-Instanz endet mit dem Aufruf der Servicefunktion clemext_peer_endExecution.
- Danach wird die Peer-Instanz durch einen clemext_destroy_peer-Aufruf entfernt.
- Vor dem Entladen des Moduls ruft der Host die Servicefunktion clemext_cleanup auf.
- Das Entladen eines Moduls erfolgt entweder beim Beenden des Serverprozesses oder bereits früher, wenn die Services des Moduls nicht mehr benötigt werden.

Serverseitige API-Funktionen

In diesem Abschnitt werden einige Konzepte der serverseitigen API beschrieben:

- Informationen zum Knotentyp
- Datentypen für verschiedene Datenspeichertypen
- Serverseitige gemeinsam verwendete Bibliotheken
- Filespaces und temporäre Dateien
- SQL-Pushback zur Ausführung von SQL-Anweisungen in der Datenbank
- Austausch von Datenmodellinformationen zwischen IBM® SPSS® Modeler und der Erweiterung

- Ausgabedokumente
- C++-Helper

Knotentypen

In der Spezifikationsdatei hat eine Knotendefinition folgendes Format:

```
<Node id="identifier" type="node_type" .../>
```

Das Attribut id ist eine Zeichenfolge, die den Knoten eindeutig kennzeichnet.

Das Attribut type gibt den Typ des Knotens an. Folgende Typen sind möglich:

- Data reader (Datenleser)
- Data writer (Datenschreiber)
- Data transformer (Datentransformer)
- Model builder (Modellersteller)
- Model applier (Modellanwender)
- Document builder (Dokumentersteller)

Für weitere Informationen siehe Thema Überblick über die Knoten in Kapitel 2 auf S. 10.

Die Funktion clemext_create_peer beinhaltet sowohl den Wert des Attributs id als auch des Attributs type des Node-Elements als Argumente.

Ein Erweiterungsmodul kann mehrere Knoten verschiedener Typen implementieren, von denen jeder unterschiedliche Funktionen ausführt. Von einem Modul können beispielsweise die folgenden Typen implementiert werden:

- Ein Datenleser- und ein Datenschreiberknoten für eine Datenquelle
- Modellerstellungs- und Modellanwendungsknoten für verschiedene Modellierungsalgorithmen
- Dokumenterstellungsknoten für verschiedene Diagrammtypen

Daten- und Speichertypen

Eine Peer-Instanz ruft Eingabedaten mittels eines clemext_iterator_getRecordValue-Aufrufs mit dem Iterator ab, der der Instanz zu Beginn der Ausführung bereitgestellt wird. Die Ausgabedaten stellt die Peer-Instanz als Antwort auf eine clemext_peer_getRecordValue-Anforderung des Hosts bereit. Die Daten werden in binärem Format im Speicher übertragen, wobei sich der Peer und der Host über den Datentyp abstimmen müssen.

Der binäre Datentyp wird vom Datenmodell bestimmt. Er richtet sich nach dem Speicherattribut eines Felds.

In folgender Tabelle sind die möglichen Speichertypen sowie die Datentypen aufgelistet, die zu deren Darstellung verwendet werden:

Tabelle 9-2
Speichertypen

Speichertyp	Dargestellt durch	Anmerkungen
string	char *	Zeichenfolgen sind immer UTF-8-kodiert
real	CLEMEXTReal	Gleitkommawert mit zwei Stellen nach dem Komma

Speichertyp	Dargestellt durch	Anmerkungen
integer	CLEMEXTInteger	Ganze Zahl mit Vorzeichen (64 Bit)
Datum	CLEMEXTDate	Ganze Zahl mit Vorzeichen (64 Bit), die die Anzahl der Tage seit dem 1. Januar 1970 angibt
time	CLEMEXTTime	Ganze Zahl mit Vorzeichen (64 Bit), die die Anzahl der Sekunden seit Mitternacht angibt
timestamp	CLEMEXTTimestamp	Ganze Zahl mit Vorzeichen (64 Bit), die die Anzahl der Sekunden seit dem 1. Januar 1970 (Mitternacht) angibt
unknown	-	Unbekannter Datentyp; die Werte können nicht dargestellt werden

Bibliotheken

Zur Unterstützung der Knotenausführung kann in der Spezifikationsdatei eine serverseitige gemeinsam verwendete Bibliothek deklariert werden. Der Pfad der gemeinsam verwendeten Bibliothek wird zur Lokalisierung der Bibliothek benötigt, um sie dynamisch in den Hostprozess laden zu können. Die gemeinsam verwendete Bibliothek muss alle erforderlichen API-Funktionen definieren. Für weitere Informationen siehe Thema Freigegebene Bibliotheken in Kapitel 4 auf S. 45.

Wenn in der Spezifikationsdatei ein Modulname angegeben ist (im Abschnitt Execution der Knotendefinition), wird dieser Name an den Parameter nodeId der Servicefunktion clemext_create_peer zur Erstellung des Peer-Objekts übergeben. Auf diese Weise kann die Erweiterung das benötigte Peer-Modul erstellen. Der Wert des Parameters nodeType bestimmt darüber hinaus den Typ des erstellten Peers. Der Modulname kann auch leer bleiben, da eine gemeinsam verwendete Bibliothek jeweils nur ein Modul eines Typs implementieren kann.

Eventuell sind für eine gemeinsam verwendete Bibliothek, die ein Erweiterungsmodul implementiert, abhängige Bibliotheken erforderlich. Diese sollten sich im gleichen Verzeichnis befinden wie die gemeinsam verwendete Bibliothek der Erweiterung.

Temporäre Dateien

In der Client-Spezifikationsdatei und dem Server-Erweiterungsmodul können Pfadnamen relativ zu einem **Filespace** angegeben werden. Es handelt sich hier um einen temporären, privaten Speicherbereich, in dem ein Peer während seiner Ausführung Dateien erstellen kann. Ein Filespace ist ein Unterverzeichnis des für einen Peer erstellten temporären Verzeichnisses auf dem Server. Er wird bei Bedarf erstellt und wieder gelöscht, wenn der Peer entfernt wird.

Solange der Filespace existiert, hat der Peer vollständige Kontrolle über diesen Speicherbereich. Der vollständige Pfadname des Filespace ist in einem **Knoteninformationsdokument** angegeben. Dieses Dokument enthält Informationen im XML-Format, die als Ergebnis der Ausführung einer clemext_node_getNodeInformation-Callback-Funktion zurückgegeben werden. Für weitere Informationen siehe Thema Knoteninformationsdokument (NodeInformation-Dokument) auf S. 240.

SQL-Pushback

Ein IBM® SPSS® Modeler-Stream liest die Daten einer SQL-Datenbank ein und verarbeitet die Daten anschließend. Erfahrene Benutzer können diesen Vorgang anhand eines Pushbacks der SQL-Anweisungen effizienter gestalten, wodurch die Anweisungen direkt in der Datenbank ausgeführt werden.

SQL-Pushback wird von einigen SPSS Modeler-Standardknoten unterstützt und die serverseitige API enthält Funktionsaufrufe, mit der ein SQL-Pushback auch in CLEF-Knoten implementiert werden kann.

Die Servicefunktion clemext_peer_getSQLGeneration generiert aus einer Peer-Instanz SQL-Anweisungen und verlegt die SQL-Ausführung in die Datenbank. Bei einem Datenleserknoten müssen die generierten SQL-Anweisungen in sich abgeschlossen sein, damit das Peer-Ergebnis-Set erstellt werden kann. Bei allen anderen Knotentypen hängen die generierten SQL-Anweisungen in der Regel von den für frühere Knoten generierten SQL-Anweisungen ab, die die Eingabedaten für den Peer bereitstellen. Ein Peer kann die SQL der früheren Knoten durch einen Aufruf der clemext_node_getSQLGeneration-Callback-Funktion mit dem ihm zugeordneten Knoten-Handle abrufen.

Behandlung des Datenmodells

Einige serverseitige API-Aufrufe bewirken den Austausch von Datenmodellinformationen zwischen IBM® SPSS® Modeler und dem Erweiterungsmodul:

- clemext_node_getDataModel ruft das Eingabedatenmodell eines Knotens ab.
- clemext_peer_getDataModel ruft das Ausgabedatenmodell einer Peer-Instanz ab.
- clemext_node_getOutputDataModel ruft das Ausgabedatenmodell eines Knotens ab.

Andere Aufrufe beziehen sich auf die Methoden zur Übertragung der Daten in und aus dem Modul. Das Datenmodell bestimmt den Index, der für das Look-up der Feldwerte in den folgenden Funktionen verwendet wird (diese Funktionen geben den Wert eines bestimmten Felds aus dem zuletzt abgerufenen Eingabe-Datensatz zurück):

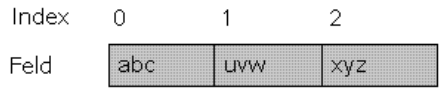
- clemext_peer_getRecordValue
- clemext_iterator_getRecordValue

SPSS Modeler ruft clemext_node_getDataModel auf, um Informationen über die Felder des Eingabedatenmodells abzurufen. Die Informationen werden im XML-Format zurückgegeben. Beispiel:

```
<DataModel>
  <Fields>
    <Field name="abc" storage="string" type="set" />
    <Field name="uvw" storage="integer" type="range" />
    <Field name="xyz" storage="real" type="range" />
  <Fields>
</DataModel>
```

Anhand dieser Informationen kann ein Modul den Feldindex bereitstellen, wenn es mittels der clemext_iterator_getRecordValue-Funktion Werte aus einem Eingabe-Datensatz abruft. Beispiel:

Abbildung 9-5
Beispiel eines Eingabedatenmodells



Auf welche Weise das Modul das Eingabedatenmodell modifiziert, wird durch den Wert des mode-Attributs des Elements OutputDataModel der Spezifikationsdatei bestimmt. Das Modul kann folgende Modifizierungen durchführen:

- Es kann das Modell durch Hinzufügen neuer Felder erweitern.
- Es kann das Modell durch Entfernen oder Umbenennen vorhandener Felder ändern.
- Es kann das vorhandene Modell durch neue Felder ersetzen.
- Es kann das Modell unverändert beibehalten.

Die folgenden Beispiele zeigen, wie ein Eingabedatenmodell erweitert bzw. ersetzt wird.

Beispiel: Erweitern des Eingabedatenmodells

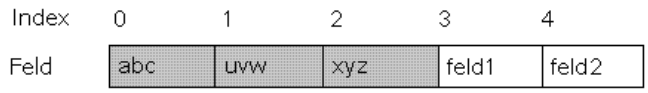
Dies ist der einfachste Fall: Ein Modul wird so eingerichtet, dass es neue Felder hinzufügt und deren Werte einstellt, jedoch die Werte vorhandener Felder unverändert lässt.

In diesem Beispiel wird davon ausgegangen, dass die Knotendefinition der Spezifikationsdatei die folgenden Anweisungen enthält:

```
<OutputDataModel mode="extend">
 <AddField name="field1" storage="string" ... />
   <AddField name="field2" storage="real" ... />
   ...
</OutputDataModel>
```

Die Definition des Ausgabedatenmodells legt in diesem Fall fest, dass das Modell alle Felder des Eingabedatenmodells sowie zusätzlich die beiden im Element OutputDataModel angegebenen Felder enthalten soll. Das Ausgabedatenmodell besteht also aus fünf Feldern.

Abbildung 9-6
Beispiel für ein Ausgabedatenmodell, das das Eingabedatenmodell erweitert



Die Funktion clemext_peer_getDataModel gibt nur die Informationen der hinzugefügten Felder zurück. Beispiel:

```
<DataModel>
 <Fields>
```

```
<Field name="field1" storage="string" ... />
      <Field name="field2" storage="real" ... />
   <Fields>
</DataModel>
```

Die zurückgegebenen Informationen müssen mit den "type"- und "number"-Werten (jedoch nicht mit dem "name"-Wert) des Elements <AddField> der Spezifikationsdatei übereinstimmen.

Ein Modul kann mit der Callback-Funktion clemext_node_getOutputDataModel die Informationen der Felder abrufen, von denen IBM® SPSS® Modeler erwartet, dass sie hinzugefügt werden. Diese Informationen können als Antwort auf einen clemext_peer_getDataModel-Aufruf direkt an SPSS Modeler zurückgegeben werden. Diese Vorgehensweise empfiehlt sich vor allem dann, wenn die Logik der Spezifikationsdatei für die Erstellung und Benennung der Ausgabefelder kompliziert ist.

Das Modul stellt die neuen Werte für jeden Ausgabedatensatz bereit, wenn SPSS Modeler die Funktion clemext_peer_getRecordValue aufruft. Die Feldindizes der neuen Felder beginnen nach dem letzten Index der Eingabefelder. In diesem Beispiel enthält das Eingabedatenmodell drei Felder (mit den Indexpositionen 0, 1 und 2). Den beiden neuen Ausgabefeldern werden daher die Feldindizes 3 und 4 zugewiesen. SPSS Modeler ruft die Funktion clemext_peer_getRecordValue nicht mit den Feldindizes der Eingabefelder auf, da diese Felder vom Modul nicht geändert werden können.

Beispiel: Ersetzen des Eingabedatenmodells (1)

In diesem Beispiel entfernt das Erweiterungsmodul in seiner Ausgabe sämtliche Felder des Eingabedatenmodells und ersetzt sie durch neue Felder.

Die Spezifikationsdatei enthält folgende Definition:

```
<OutputDataModel mode="modify">
 <AddField name="key" storage="integer" ... />
   <AddField name="field1" storage="real" ... />
   <AddField name="field2" storage="real" ... />
   ...
</OutputDataModel>
```

In diesem Fall beschreiben die von einem clemext_peer_getDataModel-Aufruf zurückgegebenen XML-Daten alle Felder des Ausgabedatenmodells:

```
<DataModel>
 <Fields>
 <Field name="key" storage="integer" ... />
      <Field name="field1" storage="real" ... />
      <Field name="field2" storage="real" ... />
   <Fields>
</DataModel>
```

Das Ausgabedatenmodell hat folgende Struktur:

Abbildung 9-7
Beispiel für ein Ausgabedatenmodell, das das Eingabedatenmodell ersetzt

Index	0	1	2
Feld	key	feld1	feld2

Das erste Ausgabefeld (key) eines clemext_peer_getRecordValue-Aufrufs erhält den Feldindex 0, das zweite Ausgabefeld (field1) erhält den Feldindex 1 usw.

Beispiel: Ersetzen des Eingabedatenmodells (2)

Auch in diesem Beispiel wird das Eingabedatenmodell durch das von der Erweiterung bereitgestellte Ausgabedatenmodell ersetzt. Im Gegensatz zum vorherigen Beispiel ist das Ausgabedatenmodell jedoch nicht in der Spezifikationsdatei definiert, sondern es wird während der Laufzeit vom Erweiterungsmodul auf dem Server berechnet. Die Spezifikationsdatei enthält folgende Definition:

```
<OutputDataModel mode="modify" method="sharedLibrary" libraryId="myLibraryId" />
```

Zur Berechnung des Ausgabedatenmodells ruft IBM® SPSS® Modeler zunächst clemext_peer_configure und danach clemext_peer_getDataModel auf. Wie im vorherigen Beispiel wird keines der Felder des Eingabedatenmodells automatisch auch im Ausgabedatenmodell übernommen. Dieses wird vollständig durch die vom clemext_peer_getDataModel-Aufruf zurückgegebene Antwort definiert.

Hinweis: Wenn das Erweiterungsmodul das Ausgabedatenmodell wie in diesem Beispiel auf dem Server definiert, kann das Modul das Ausgabedatenmodell nicht gleichzeitig mit der Funktion clemext_node_getOutputDataModel abrufen, da dies einen Fehler aufgrund einer ungültigen Operation herbeiführen würde.

XML-Ausgabedokumente

Einige Service- und Callback-Funktionen verwenden XML-Ausgabedokumente zur Übergabe der Informationen zwischen dem Host und dem Erweiterungsmodul. Hierzu stehen eine Reihe verschiedener Dokumente zur Verfügung.

XML-Ausgabe-dokument	Anmerkungen	Zurückgegeben von Aufruf
Katalog-dokument	Enthält die Liste der Werte für eine Steuerung, die mit einem Katalog verknüpft ist.	clemext_peer_getCatalogInformation
Datenmodell-dokument	Beschreibt den ein- bzw. ausgehenden Feldsatz eines Knotens	clemext_peer_getDataModel clemext_node_getDataModel clemext_node_getOutputDataModel
Fehlerdetail-dokument	Enthält Informationen zu einem Fehler oder einer anderen Bedingung	clemext_peer_getErrorDetail

XML-Ausgabe-dokument	Anmerkungen	Zurückgegeben von Aufruf
Ausführungsanforderungs-dokument	Beschreibt die für eine Peer-Instanz erforderlichen Ausführungsvoraussetzungen, beispielsweise einen Daten-Cache oder erforderliche Eingabefelder	clemext_peer_getExecutionRequire-ments
Hostinformations-dokument	Enthält Informationen zur Hostumgebung wie die Kennung, die Beschreibung, die Version und den Hersteller des Produkts sowie Copyright- und Plattformangaben	clemext_host_getHostInformation
Modulinformations-dokument	Enthält Informationen zum Erweiterungsmodul wie die Kennung, die Beschreibung, die Version und den Hersteller des Moduls sowie Copyright- und Lizenzangaben	clemext_getModuleInformation
Knoteninformations-dokument	Enthält Informationen zu dem mit einer Peer-Instanz verbundenen Knoten wie die Kennung, den Typ und den Filespace des Knotens	clemext_node_getNodeInformation
Parameter-dokument	Enthält die Konfigurationsparameter des Clientknotens; der Inhalt wird durch die Erweiterung bestimmt	clemext_node_getParameters
SQL-Generierungs-dokument	Beschreibt, wie die Ausführung eines Peers in SQL-Anweisungen umgesetzt wird	clemext_peer_getSQLGeneration clemext_node_getSQLGeneration
Statusdetail-dokument	Enthält ergänzende Informationen über den Fortschritt, über Warnungen oder über andere Bedingungen, die während der Ausführung auftreten	clemext_progress_report

Katalogdokument

Ein Katalogdokument beschreibt den Inhalt eines Katalogs, der eine Liste von Werten enthält, die über ein Steuerelement der Benutzeroberfläche angezeigt werden kann.

Das Modul CLEF implementiert wie folgt einen Aufruf an getCatalogInformation:

```
CLEMEXTStatus
getCatalogInformation(
      const char *catalogId,
        char* buffer,
        size_t buffer_size,
        size_t* data_size,
        CLEMEXTErrorCode* errorCode) {
   …
```

}

Dabei ist catalogId die ID eines bestimmten Katalogs, wie im Element Catalog in der Spezifikationsdatei definiert.

Diese Funktion gibt das Katalogdokument zurück.

Beispiel

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<CatalogInformation>
   <CatalogEntry>
      <CatalogValue>apples</CatalogValue>
      <CatalogValue>0</CatalogValue>
   </CatalogEntry>
   <CatalogEntry>
      <CatalogValue>oranges</CatalogValue>
      <CatalogValue>1</CatalogValue>
   </CatalogEntry>
   <CatalogEntry>
      <CatalogValue>bananas</CatalogValue>
      <CatalogValue>2</CatalogValue>
   </CatalogEntry>
</CatalogInformation>
```

Datenmodelldokument (DataModel-Dokument)

Ein Datenmodelldokument beschreibt das **Datenmodell**, d. h. den Satz der in einen Knoten ein- oder ausgehenden Felder mit ihren Namen, Typen und zugehörigen Informationen. Es verkapselt die in einem Typknoten verfügbaren Informationen.

Ein Peer, der keine Eingabe annimmt (ein Quellenknoten), verfügt über ein leeres Eingabemodell, während ein Peer, der keine Ausgabe generiert (ein Endknoten), über ein leeres Ausgabemodell verfügt. Ein Peer, der Eingaben annimmt und Ausgaben generiert (ein Prozessknoten), muss wissen, wie er aus der Eingabe sein Ausgabemodell berechnet.

Ein Peer ruft sein Eingabedatenmodell ab, indem er clemext_node_getDataModel mit dem zugehörigen Knoten-Handle aufruft. Sein Ausgabedatenmodell stellt er als Antwort auf eine clemext_peer_getDataModel-Anforderung des Hosts bereit.

Jedes Datenmodell kann als Datenwörterbuch dargestellt werden, das alle Felder des Datenmodells mit ihren Eigenschaften auflistet. Ein Knoten stellt das Eingabedatenmodell für einen Peer immer in dieser Form bereit. Das von einem Peer generierte Ausgabedatenmodell kann in der Form eines Datenwörterbuchs übergeben oder als Operationssequenz dargestellt werden (Feld hinzufügen, Feld entfernen, Feld ändern), die auf das Eingabemodell angewendet werden muss. Bei einigen Knoten vereinfacht sich dadurch das Ausgabemodell erheblich.

Die Reihenfolge der Felder in einem Datenmodelldokument ist wichtig. Sie legt die Reihenfolge fest, in der die Daten des entsprechenden Eingabe- oder Ausgabe-Daten-Sets übergeben werden.

Ein Datenmodell kann unvollständig sein und nur eine Teilspezifikation der Daten bereitstellen. Ein Eingabemodell, das alle erforderlichen Spezifikationen für den Ausführungsplan des Peers enthält, wird für diesen Peer als **ausführbar** bezeichnet. Ein ausführbares Datenmodell muss

immer den binären Typ jedes Felds enthalten, damit die Eingabe- und Ausgabedaten korrekt zusammengestellt werden können.

Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<DataModel>
 <Fields>
  <Field name="Age" type="range" storage="integer" direction="in">
   <Range minValue="15" maxValue="74"/>
  </Field>
  <Field name="Sex" type="flag" storage="string">
   <Values>
    <Value value="F" flagValue="false" displayLabel="Female"/>
    <Value value="M" flagValue="true" displayLabel="Male"/>
   </Values>
  </Field>
  <Field name="BP" type="orderedSet" storage="integer">
   <Values>
    <Value value="-1" />
    <Value value="0" />
    <Value value="1" />
   </Values>
  </Field>
  <Field name="Cholesterol" type="flag" storage="string">
   <Values>
    <Value value="NORMAL" flagValue="false"/>
    <Value value="HIGH" flagValue="true"/>
   </Values>
  </Field>
  <Field name="Na" type="range" storage="real" displayLabel="Blood sodium">
   <Range minValue="0.500517" maxValue="0.899774"/>
  </Field>
  <Field name="K" type="range" storage="real" displayLabel="Potassium concentration">
   <Range minValue="0.020152" maxValue="0.079925"/>
  </Field>
  <Field name="Drug" type="set" storage="string" direction="out">
   <Values>
    <Value value="drugA"/>
    <Value value="drugB"/>
    <Value value="drugC"/>
    <Value value="drugX"/>
    <Value value="drugY"/>
   </Values>
  </Field>
 </Fields>
</DataModel>
```

Fehlerdetaildokument (ErrorDetail-Dokument)

In einem Fehlerdetaildokument werden Meldungen (Fehler, Warnungen, Informationen) zurück an IBM® SPSS® Modeler gesendet. Es enthält Informationen über einen Fehler oder eine andere Bedingung. Ein Erweiterungsmodul kann als Antwort auf eine clemext_peer_getErrorDetail-Anforderung des Hosts ein Fehlerdetaildokument mit Informationen zu einem modulspezifischen Fehler bereitstellen.

Die Fehlerinformationen umfassen ein oder mehrere Diagnostic-Elemente, wobei jedes dieser Elemente mindestens einen Fehlercode, eine Meldung sowie ein oder mehrere Parameter mit weiteren Informationen für die Meldung enthält. Die Fehlercodes entsprechen den Werten der StatusCode-Elemente der Spezifikationsdatei.

Die Meldung kann in verschiedenen Sprachvarianten vorliegen oder der Client kann die lokalisierte Meldung mithilfe des Fehlercodes aus einem Ressourcenbündel abrufen. Eine Abfolge mehrerer Diagnostic-Elemente beschreibt eine kausale Fehlerkette.

Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<ErrorDetail>
 <Diagnostic code="123" severity="error">
  <Message>You can't do that ({0})</Message>
  <Parameter>Permission denied</Parameter>
 </Diagnostic>
 <Diagnostic code="456" severity="warning">
  <Message>That was silly!</Message>
  <Message lang="fr">Quel idiot!</Message>
 </Diagnostic>
</ErrorDetail>
```

Ausführungsanforderungsdokument (ExecutionRequirements-Dokument)

Ein Ausführungsanforderungsdokument beschreibt die für eine Peer-Instanz erforderlichen Ausführungsvoraussetzungen. Eine Peer-Instanz kann ein Ausführungsanforderungsdokument als Antwort auf eine clemext_peer_getExecutionRequirements-Anforderung vom Host bereitstellen. Vor einem clemext_peer_beginExecution-Aufruf an den Peer konsultiert der Host das Ausführungsanforderungsdokument, um die erforderliche Ausführungsumgebung bereitzustellen.

Der Host kann einen Daten-Cache-Service bereitstellen, um sicherzustellen, dass das Modul die Eingabedaten mittels der clemext_iterator_rewind-Funktion mehrmals übergeben kann.

Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<ExecutionRequirements>
 <Cache/><!-- stellt sicher, dass das CLEF-Modul mehrere Durchläufe über die Eingabedaten ausführen kann -->
</ExecutionRequirements>
```

Hostinformationsdokument (HostInformation-Dokument)

Ein Hostinformationsdokument beschreibt die Hostumgebung. Ein Erweiterungsmodul kann mittels eines clemext_host_getHostInformation-Aufrufs mit dem entsprechenden Host-Handle Informationen über einen Host abrufen.

Die zurückgegebenen Informationen umfassen die Kennung, die Beschreibung, die Version und den Hersteller des Produkts sowie Copyright- und Plattformangaben.

Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<HostInformation>
 <Host name="clemlocal" externalEncoding="cp1252" language="english_us"
     locale="English_United Kingdom.1252" provider="IBM Corp." version="15" platform="Windows XP SP2"
     copyright="Copyright 1995-2011 IBM Corp. All rights reserved.">
  <VersionDetail major="12" minor="0"/>
  <PlatformDetail osType="windows" osName="WindowsNT" osMajor="5" osMinor="1"/>
  <LibraryDetail path="C:\Program Files\IBM\SPSS\Modeler\15\ext\bin\my.module\myModule.dll"/>
 </Host>
</HostInformation>
```

Modulinformationsdokument (ModuleInformation-Dokument)

Ein Modulinformationsdokument beschreibt ein Erweiterungsmodul. Ein Erweiterungsmodul muss als Antwort auf eine clemext_getModuleInformation-Anforderung des Hosts ein Modulinformationsdokument bereitstellen.

Die zurückgegebenen Informationen umfassen die Kennung, die Beschreibung, die Version und den Hersteller des Moduls sowie Copyright- und Lizenzangaben.

Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<ModuleInformation>
 <Module name="MyModule" provider="My Company Inc." version="10.1.0.329"
      copyright="Copyright 2006 My Company Inc. All rights reserved.">
  <VersionDetail major="10" minor="1" release="0" build="329"/>
  <Licence code="1234" type="mandatory"/>
  <Description>Gründlicher Test der neuen Erweiterungsstruktur.</Description>
 </Module>
</ModuleInformation>
```

Knoteninformationsdokument (NodeInformation-Dokument)

Ein Knoteninformationsdokument beschreibt den mit einer Peer-Instanz verbundenen Knoten. Eine Peer-Instanz kann mittels eines clemext_node_getNodeInformation-Aufrufs mit dem entsprechenden Knoten-Handle Informationen über einen Knoten abrufen. Knoteninformationen umfassen die Kennung, den Typ und den Filespace des Knotens.

Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<NodeInformation>
 <Node name="databaseImport" type="dataReader">
  <FileSpace path="C:\Program Files\IBM SPSS Modeler Server15\tmp\ext-8005-6711-01"/>
 </Node>
</NodeInformation>
```

Parameterdokument (Parameters-Dokument)

Ein Parameterdokument enthält Informationen über alle Property-Elemente einer Spezifikationsdatei. Die Informationen werden in Form von Konfigurationsparametern zurückgegeben, die ein Peer mittels eines clemext_node_getParameters-Aufrufs mit dem entsprechenden Knoten-Handle abrufen kann.

Ein Parameter hat einen Namen und einen Wert, wobei der Wert Folgendes sein kann:

- Einfacher Wert (String)
- Verschlüsselter Wert (Schlüssel und Wert)
- Strukturierter Wert (Satz mit benannten Werten)
- Werteliste

Der Inhalt eines Parameterdokuments wird durch das Erweiterungspaket bestimmt. Die Parameter sind in der clientseitigen Spezifikationsdatei definiert und werden vom Erweiterungsmodul auf dem Server interpretiert.

Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<Parameters>
 <Parameter name="linesToScan" value="50"/>
 <Parameter name="useCaption" value="true"/>
 <Parameter name="caption" value="My Caption"/>
 <Parameter name="captionPosition" value="north"/>
 <Parameter name="defaultAggregation">
  <ListValue>
   <Value value="min"/>
   <Value value="max"/>
   <Value value="mean"/>
   <Value value="stddev"/>
  </ListValue>
 </Parameter>
</Parameters>
```

SQL-Generierungsdokument (SqlGeneration-Dokument)

Ein SQL-Generierungsdokument beschreibt, wie die Ausführung eines Peers in SQL-Anweisungen umgesetzt wird.

Ein Peer kann ein SQL-Generierungsdokument als Antwort auf eine clemext_peer_getSQLGeneration-Anforderung vom Host bereitstellen. Wenn dem Host ein SQL-Generierungsdokument vorliegt, zieht er die Ausführung der SQL-Anweisungen der internen Ausführung des Peers vor.

Ein Peer, der Eingabedaten annimmt, kann seine Eingabe-SQL-Anweisungen abrufen, indem er clemext_node_getSQLGeneration mit dem zugehörigen Knoten-Handle aufruft.

Der wichtigste Teil eines SQL-Generierungsdokuments ist eine SQL-Anweisung, die das Ausführungsverhalten eines Knotens oder eines Stream-Fragments repliziert. Bei einem Knoten, der Daten generiert, also einem Datenleser- oder Datentransformationsknoten, muss es sich um eine SELECT-Anweisung handeln. Dieser muss ein Wörterbuch beigefügt sein, das die Feldnamen des Datenmodells den Spaltennamen der SELECT-Anweisung zuordnet.

Darüber hinaus kann ein SQL-Generierungsdokument auch die Eigenschaften der Datenbankverbindung enthalten, über die die Anweisung ausgeführt wird (z. B. den Datenquellen- und den Produktnamen). Diese Eigenschaften helfen dem Peer bei der Ermittlung der zu generierenden SQL-Anweisungen.

Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<SqlGeneration>
 <Properties
  datasourceName="SQL Server"
  databaseName="DataMining"
  serverName="GB1-RDUNCAN1"
  passwordKey="PW0"
  userName="fred"
  dbmsName="Microsoft SQL Server"
  dbmsVersion="09.00.1399"/>
 <Statement>
  <Bindings>
   <Binding columnName="C0" fieldName="ID"/>
   <Binding columnName="C1" fieldName="START_DATE"/>
  </Bindings>
  <TableParameters>
  <TableParameter name="${TABLE26}" value="dbo.DRUG4N"/>
  </TableParameters>
  <Sql>
   SELECT
   T0.ID AS C0,T0."START_DATE" AS C1
   FROM ${TABLE26} T0
   WHERE (T0."START_DATE" > '2003-01-01')
   ORDER BY 2 ASC
  </Sql>
 </Statement>
</SqlGeneration>
```

Statusdetaildokument (StatusDetail-Dokument)

Ein Statusdetaildokument stellt Informationen über den Fortschritt sowie Informationen über unkritische Warnungen und andere Bedingungen bereit, die während der Ausführung auftreten. Statusdetaildokumente können vom Erweiterungsmodul mithilfe der Callback-Funktion clemext_progress_report asynchron versendet werden.

Ein Statusdetaildokument enthält eines oder mehrere Diagnostic-Elemente, wobei jedes dieser Elemente mindestens einen Bedingungscode, eine Meldung (außer diese ist in einer Eigenschaftendatei angegeben) sowie einen oder mehrere Parameter mit weiteren Informationen für die Meldung enthält. Das Element StatusDetail kann ebenfalls ein optionales destination-Attribut enthalten, um die Meldung an eine der folgenden Stellen weiterzugeben:

- Eine lokale, von IBM® SPSS® Modeler verwaltete Trace-Datei
- Der Client (für Meldungen an den Benutzer)
- Alle (die Meldung wird an alle möglichen Ziele gesendet)

Das Format des Elements Diagnostic sieht wie folgt aus:

```
<Diagnostic code="integer" severity="severity_level">
   <Meldung>message_text</Message>
   <Parameter>value</Parameter>
</Diagnostic>
```

Dabei gilt:

code (erforderlich) ist eine ganze Zahl, die den Bedingungscode angibt.

severity gibt den Schweregrad der Bedingung an und ist auf einen der folgenden Werte eingestellt: unknown, information, warning, error oder fatal.

Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<StatusDetail destination="client">
 <Diagnostic code="654" severity="information">
  <Message>Processed {0} records</Message>
  <Parameter>10000</Parameter>
 </Diagnostic>
</StatusDetail>
```

Verwenden lokalisierter Meldungen

Wenn Sie lokalisierte Meldungen aus einer Eigenschaftendatei benutzen möchten, lassen Sie das Element Message aus dem Statusdetaildokument weg und verwenden in der Spezifikationsdatei Meldungsschlüssel wie im folgenden Beispiel:

```
...
<Execution ...>
...
   <StatusCodes>
      ...
      <StatusCode code="21" status="warning" messageKey="fieldIgnoredMsg.LABEL"/>
```

```
      ...
    </StatusCodes>
  </Execution>
  ...
```

Die Datei “messages.properties” würde dann Folgendes enthalten:

fieldIgnoredMsg.LABEL=Field "{0}" cannot be used for model building and was ignored

Im Statusdetaildokument können Sie dann einen Parameter (wie einen Feldnamen) verwenden, der in der lokalisierten Meldung ersetzt werden soll, z. B.:

```
<?xml version="1.0" encoding="utf-8"?>
<StatusDetail>
  <Diagnostic code="21">
    <Parameter>BP</Parameter>
  </Diagnostic>
</StatusDetail>
```

C++-Helper

Einige der Beispielknoten von CLEF enthalten vordefinierte C++-Quellendateien. Diese Dateien werden auch als **Helper** bezeichnet. C++-Helper-Dateien fungieren als Wrapper für einen Teil der C-basierten, serverseitigen API und lassen sich problemlos in ein C++ CLEF kompilieren.

Tabelle 9-3
C++-Helper

Helper	Beschreibung
BufferHelper	Verwaltet die in der CAPI verwendeten, skalierbaren Speicherpuffer
DataHelper	Hilft bei der Aufnahme von Datenlese- und Datenschreiboperationen
HostHelper	Nimmt das CLEMEXT HostHandle-Objekt auf
XMLHelper	Nimmt die XML-Verarbeitung auf

Ein Helper ist immer ein Dateienpaar bestehend aus einer *.cpp*- und einer *.h*-Datei (z. B. *BufferHelper.cpp* und *BufferHelper.h*).

Informationen zum Anzeigen dieser Helper-Dateien finden Sie im Abschnitt Untersuchen des Quellcodes auf S. 38.

Ausführlichere Beschreibungen dieser Dateien finden Sie in der Dokumentation der serverseitigen API, die Sie wie folgt aufrufen:

- Wählen Sie im API-Dokumentationsfenster von CLEF die Option Server-side API overview (Überblick über die serverseitige API) aus.
- Klicken Sie auf die Registerkarte Dateien.
- Klicken Sie auf den Namen der .h-Datei des Helpers, dessen Informationen Sie anzeigen möchten.
- Klicken Sie unter Data Structures (Datenstrukturen) auf den zugehörigen Klassennamen, um die Dokumentation anzuzeigen.

Informationen zum Zugriff auf die API-Dokumentation von CLEF finden Sie im Abschnitt Dokumentation zu den CLEF-APIs auf S. 220.

Fehlerbehandlung

Jeder API-Funktionsaufruf gibt einen Statuscode (CLEMEXTStatus) und optional einen modulspezifischen Fehlercode (CLEMEXTErrorCode) zurück. Der Statuscode kann success (kein Fehler) lauten oder einen der für die API-Funktion definierten Fehlercodes enthalten (bei diesen handelt es sich fast immer um modulspezifische Fehler). Wenn es sich nicht um einen modulspezifischen Fehler handelt, lautet der modulspezifische Fehlercode 0.

Die Statuscode-Meldungen werden von IBM® SPSS® Modeler bereitgestellt. Beschreibungen der allgemeinen Statuscodes finden Sie in der Dokumentation der serverseitigen API, die Sie wie folgt aufrufen:

- Wählen Sie im API-Dokumentationsfenster von CLEF die Option Server-side API overview (Überblick über die serverseitige API) aus.
- Klicken Sie auf die Registerkarte Module.
- Wählen Sie Common Status Codes (Allgemeine Statuscodes) aus.

Informationen zum Zugriff auf die API-Dokumentation von CLEF finden Sie im Abschnitt Dokumentation zu den CLEF-APIs auf S. 220.

Modulspezifische Meldungen können wie folgt bereitgestellt werden:

- In der Spezifikationsdatei (im Element StatusCodes des Abschnitts “Module”)
- In einem Ressourcenbündel, auf das die Spezifikationsdatei verweist
- Durch das Erweiterungsmodul

Für einen modulspezifischen Fehlercode kann das Modul zusätzliche Fehlerinformationen bereitstellen. Diese bestehen aus einer Standardfehlermeldung (bei Fehlern, die nicht in der Spezifikationsdatei oder im Ressourcenbündel definiert sind) und Parametern, die in die Meldung eingefügt werden. Kausale Fehlerketten können in mehreren Fehlermeldungen beschrieben werden.

Ein Fehlerbericht für den Client hat folgendes Format:

Knotenbeschriftung:message

Dabei gilt:

- *Knotenbeschriftung* ist der Wert des label-Attributs des Node-Elements, in dem das Modul definiert ist.
- *Meldung* ist der Meldungstext. Dieser kann entweder vom Server bereitgestellt oder in der Spezifikationsdatei definiert werden. Bei lokalisierten Modulen wird der Meldungstext in einer *.properties*-Datei bereitgestellt.

XML Parsing-API

IBM® SPSS® Modeler enthält einen Xerces-C XML-Parser von Apache, der verschiedene Callbacks bereitstellt, mit denen ein Modul XML-Daten lesen und schreiben kann. Sie können statt dieses Parsers auch Ihren eigenen XML-Parser verwenden.

Verwenden der serverseitigen API

So fügen Sie einem Knoten serverseitige Funktionsaufrufe hinzu:

- Erstellen Sie in C++ die *.cpp*- und *.h*-Quellendateien mit den gewünschten Funktionsaufrufen.
- Kompilieren Sie die Quellendateien in eine dynamische Verknüpfungsbibliothek (*.dll*-Datei).
- Fügen Sie in der Spezifikationsdatei einen Verweis auf die *.dll*-Datei ein. Beispiel:

```
<Resources>
.
   <SharedLibrary id="mylib1" path="mycorp.mynode/mylib" />
.
</Resources>
```

Für weitere Informationen siehe Thema Freigegebene Bibliotheken in Kapitel 4 auf S. 45.

Sehen Sie sich hierzu auch den Quellcode der in dieser Version bereitgestellten Beispielknoten an. Sie finden dort eventuell einige nützliche Anwendungsbeispiele. Für weitere Informationen siehe Thema Untersuchen des Quellcodes in Kapitel 3 auf S. 38.

Richtlinien für die Server-seitige Programmierung

Der Server-seitige DLL-Teil (Dynamic Link Library) eines CLEF-Moduls muss eine Reihe von Richtlinien einhalten, damit das Modul korrekt arbeitet und der Betrieb von IBM® SPSS® Modeler nicht beeinträchtigt wird. Ein CLEF-Modul muss:

- selbstständige Peer-Ausführung sicherstellen.
- mehrere Peer-Instanzen in einem einzigen Prozess unterstützen.
- Thread-Sicherheit garantieren.
- Ändern der Thread- oder Prozessumgebung vermeiden.
- Thread-Nutzung im Modul beschränken.
- Anforderungen zum Abbrechen der Ausführung korrekt behandeln.
- unterbrochene Systemaufrufe neu starten (UNIX).
- Aufruf von CoInitialize oder CoUninitialize (Windows) sorgfältig abwickeln.
- Annahmen über den Zeitpunkt vermeiden, an dem das Modul entladen wird.
- beim Start von Unterprozessen sorgfältig vorgehen.
- Schreiben an Standardausgabe oder Standardfehler vermeiden.

Die folgenden Abschnitte behandeln jedes dieser Themen ausführlicher.

Selbstständige Peer-Ausführung sicherstellen

Peer-Instanzen sollten das Vorhandensein anderer Peer-Instanzen innerhalb des IBM® SPSS® Modeler-Serverprozesses nicht voraussetzen. SPSS Modeler kann die Ausführung so planen, dass Peer-Instanzen, die direkt benachbarten Knoten in einem Stream entsprechen, in anderen Phasen ausgeführt werden und sich Existenz und Ausführung der Instanzen nicht überlagern.

Peer-Instanzen sollten daher eigenständig sein und nicht versuchen, direkt mit anderen Peer-Instanzen zu kommunizieren, etwa durch Pipes oder Sockets. Sämtliche Kommunikation zwischen verschiedenen Peer-Instanzen muss direkt durch Lesen und Schreiben von Daten in den Stream oder indirekt durch einen externen Agenten erfolgen (z. B. durch einen Datenbankserver, der gemeinsam benutzte Daten zwischen Peers verwaltet).

Mehrere Peer-Instanzen in einem einzigen Prozess unterstützen

Endbenutzer können beim Ausführen eines Streams mehrere Peer-Instanzen eines bestimmten CLEF-Moduls (d. h. mehrere Knoten desselben Typs) in einem Serverprozess erstellen. So werden alle statischen Daten im CLEF-Modul von mehreren Peer-Instanzen gemeinsam benutzt und sollten nicht zum Speichern von privaten Daten eines Peer-Objekts verwendet werden. Beispiele für statische Daten sind statische Mitglieder von C++-Klassen sowie globale oder statische Variablen in C-Kompilierungseinheiten.

API-Funktionen des CLEF-Moduls müssen ablaufinvariant sein und Systemaufrufe vermeiden, die nicht ablaufinvariant sind. Beispiel: Wenn eine Peer-Instanz Eingabedaten mithilfe von clemext_iterator_nextRecord von seinem Eingabe-Iterator abruft, kann dieser wiederum clemext_peer_nextRecord von einer zweiten Peer-Instanz abrufen, die aufwärts vom ersten Peer liegt und die Daten erzeugt, die der erste Peer schließlich verwendet.

Systemaufrufe wie strtok sind nicht ablaufinvariant und dürfen nicht benutzt werden. Einzelheiten zu Alternativen, die ablaufinvariant sind, können Sie der Betriebssystemdokumentation für Ihre verwendete Plattform entnehmen.

Thread-Sicherheit garantieren

IBM® SPSS® Modeler kann die Ausführung von mehreren Peer-Instanzen von verschiedenen Ausführungs-Threads verschachteln. Zugriff auf alle Ressourcen, die zwischen Peer-Objekten gemeinsam genutzt werden, müssen daher geschützt werden, z. B. durch Synchronsisierung mit Mutexes (Mutual Exclusion Objects - sich gegenseitig ausschließende Objekte) oder ähnliche Threading-Bibliotheksdienste.

CLEF-Module müssen Systemaufrufe vermeiden, die nicht Thread-sicher sind. Konsultieren Sie für weitere Informationen die Dokumentation zu Ihrem Betriebssystem oder UNIX-man-Seiten.

Ändern der Thread- oder Prozessumgebung vermeiden

Vermeiden Sie Systemaufrufe, mit denen die Umgebung des Aufruf-Threads oder -Prozesses geändert werden könnte.

Nachfolgend sind einige solche Aufrufe aufgeführt, die Liste ist jedoch bei Weitem nicht vollständig:

- setlocale, wenn zum Ändern und nicht zum einfachen Lesen einer Ländereinstellung benutzt
- SetCurrentDirectory (Windows) oder chdir (UNIX)
- LogonUser (Windows) oder seteuid (UNIX)
- putenv
- exit
- signal

Anmerkung: Ein Aufruf unter Windows, der die Umgebung eines Threads ändert, aber erforderlich sein kann, ist CoInitialize. Für weitere Informationen siehe Thema Aufruf von CoInitialize oder CoUninitialize (Windows) sorgfältig abwickeln auf S. 249.

Thread-Nutzung im Modul beschränken

Im Allgemeinen können Module intern beliebige Threads verwenden. Jedoch sollten Callback-Funktionen nur für den Thread aufgerufen werden, den IBM® SPSS® Modeler für den Aufruf der Funktionen des CLEF-Moduls verwendet hat (mit Ausnahme von clemext_peer_cancelExecution).

Die folgenden Callback-Funktionen können asynchron von einem beliebigen innerhalb des Moduls ausgeführten Thread aufgerufen werden:

- clemext_progress_report
- clemext_channel_send

Eine Peer-Instanz muss sicherstellen, dass mehrere Threads keinen dieser Aufrufe simultan auslösen.

Anforderungen zum Abbrechen der Ausführung korrekt behandeln

Wenn ein Endbenutzer den Abbruch der Ausführung einer Peer-Instanz anfordert, führt IBM® SPSS® Modeler einen asynchronen Aufruf an die Funktion clemext_peer_cancelExecution des Moduls durch. Entwickler sollten versuchen, diesen Aufruf zu implementieren. Beachten Sie, dass er für diese Funktion als asynchroner Aufruf vorgesehen ist und aufgerufen wird, während ein anderer CLEF-API-Funktionsaufruf vom Modul ausgeführt wird.

Unterbrochene Systemaufrufe neu starten (UNIX)

Unter UNIX verwendet die IBM® SPSS® Modeler-Anwendung Signale und Signal-Handler. Einige UNIX-Systemaufrufe können den Code EINTR zurückgeben, falls der Prozess bei der Ausführung des Aufrufs ein Signal empfängt. Prüfen Sie die man-Seiten für den Systemaufruf auf Ihrer verwendeten UNIX-Plattform.

Wenn dieses Ereignis eintritt, sollte der aufrufende Code das Vorhandensein des Rückgabecodes EINTR prüfen und den Aufruf neu starten. Eine Möglichkeit dafür ist, eine einfache Wrapper-Funktion zu erstellen (open_safe) und diesen Wrapper von Ihrer Anwendung aufrufen zu lassen:

```
int
open_safe(const char* path, int oflag, mode_t mode) {
 int res;
 while ((res = ::open(path, oflag, mode)) == -1
                          && errno == EINTR) {
 }
 return res;
}
```

Aufruf von CoInitialize oder CoUninitialize (Windows) sorgfältig abwickeln

Unter Windows sollten Threads, die Windows COM-Bibliotheksdienste (Component Object Model) verwenden müssen, die System-API-Funktion CoInitialize vor dem Verwenden von COM-Diensten und am Ende CoUninitialize aufrufen. Die Threads, aus denen IBM® SPSS® Modeler das CLEF-API für ein Modul aufruft, hat eventuell bereits CoInitialize aufgerufen.

Ein CLEF-Modul, das COM-Dienste aus diesen Threads nutzen möchte, sollte CoInitialize (gewöhnlich in einer clemext_create_peer- oder clemext_peer_beginExecution-Funktion) aufrufen. Falls dieser Aufruf erfolgreich ist, muss das Modul später ebenfalls CoUninitialize aufrufen, wenn der Thread die Ausführung beendet, gewöhnlich in clemext_destroy_peer bzw. clemext_peer_endExecution.

Weitere Informationen zum Aufruf CoInitiaize finden Sie in der Dokumentation auf der Website Microsoft Developer Network (MSDN) unter http://msdn.microsoft.com.

Annahmen über den Zeitpunkt vermeiden, an dem das Modul entladen wird

Derzeit bleibt ein CLEF-Modul bis zum Sitzungsende geladen (d. h., Module können nicht nach Anforderung entladen und neu geladen werden). Die Funktion clemext_cleanup wird nie aufgerufen, nicht einmal beim Beenden des IBM® SPSS® Modeler-Serverprozesses, in den das Modul geladen wurde. Entwickler sollten also zu keinem Zeitpunkt davon ausgehen, dass ein Modul entladen wird und seine Ressourcen freigegeben werden.

Beim Start von Unterprozessen sorgfältig vorgehen

Das Starten eines Unterprozesses durch CreateProcess (Windows) bzw. fork (UNIX) kann eine Reihe von Komplikationen bei der Interaktion zwischen übergeordneten und untergeordneten Prozessen verursachen sowie dabei, wie untergeordnete Prozesse Ressourcen erben, die im übergeordneten Prozess geöffnet sind.

Wenn ein CLEF-Modul “out-of-process”-Ausführung aufrufen muss, erwägen Sie die Verwendung einer geeigneten alternativen Architektur. Beispielsweise könnte das CLEF-Modul die Dienste nutzen, die von einem Anwendungsserver angeboten werden, um die entsprechende Aufgabe zu erledigen.

Insbesondere sollten Windows-Prozesse keine Unterprozesse mithilfe der Funktion CreateProcess mit dem Parameter bInheritHandles auf TRUE starten. Denn damit erbt der Unterprozess alle Dateideskriptoren, die im übergeordneten Prozess (IBM® SPSS® Modeler-Server) geöffnet sind.

Schreiben an Standardausgabe oder Standardfehler vermeiden

Wenn ein CLEF-Modul an die Standardausgabe oder den Standard-Fehlerstream eines Prozesses schreibt (z. B. für Debugging-Zwecke), ist dies im Allgemeinen für den Endbenutzer nicht sichtbar. Wenn jedoch ein Stream, der CLEF-Knoten enthält, mit IBM® SPSS® Modeler Solution Publisher bereitgestellt und von einer Befehlszeilen-Shell (Windows oder UNIX) ausgeführt wird, wird diese Ausgabe sichtbar und kann Benutzer verwirren.

CLEF-Module können stattdessen einen Verfolgungsdienst starten, indem Sie die Host-Callback-Funktion clemext_host_trace aufrufen und die anzuzeigende Meldung als Textstring angeben. Die Verfolgung muss ebenfalls in der IBM® SPSS® Modeler-Installation über die folgende Einstellung in der IBM® SPSS® Modeler Server-Konfigurationsoptionendatei (*/config/options.cfg* im SPSS Modeler-Installationsverzeichnis) aktiviert werden:

```
trace_extension, 1
```

Verfolgte Meldungen werden in die Datei */log/trace-<process_ID>-<process_ID>.log* im SPSS Modeler-Installationsverzeichnis ausgegeben, wobei *process_ID* die Kennung des SPSS Modeler Server-Prozesses ist. Vermeiden Sie die gleichzeitige Verfolgung von mehreren Sitzungen, da sie alle dieselbe Protokolldatei verwenden.

Kapitel
10

Test und Verteilung

Testen von CLEF-Erweiterungen

Eine neue Erweiterung sollte unbedingt getestet werden, bevor sie an andere Benutzer verteilt wird.

Nach dem Anlegen einer Spezifikationsdatei und zugehörigen Ressourcen-Bundles, .jar-Dateien, freigegebenen Bibliotheken und Benutzerhilfedateien können Sie die Erweiterung testen, indem Sie die Dateien in der erforderlichen Dateistruktur anordnen und in Ihre lokale IBM® SPSS® Modeler-Installation kopieren. Beim nächsten Start von SPSS Modeler sollten Sie die neue Erweiterung an der SPSS Modeler-Benutzeroberfläche sehen.

Testen einer CLEF-Erweiterung

- Schließen Sie IBM® SPSS® Modeler, falls die Software geöffnet ist.
- Wenn die Erweiterung einen CLEF-Knoten oder eine Ausgabe definiert, wird empfohlen, dass Sie die Registerkarte “Fehlersuche” im Dialogfeld “Erweiterung” aktivieren, bis Sie sicher sind, dass die Erweiterung korrekt funktioniert. Für weitere Informationen siehe Thema Verwenden der Registerkarte “Fehlersuche” auf S. 253.
- Ordnen Sie die Client-seitigen und Server-seitigen Dateien in der erforderlichen Struktur an. Vergewissern Sie sich, dass die Spezifikationsdatei und alle für den Knoten erforderlichen Ressourcen (z. B. .jar- oder .dll-Dateien) in die richtigen Verzeichnisse kopiert wurden. Für weitere Informationen siehe Thema Dateistruktur in Kapitel 1 auf S. 6.
- Kopieren Sie das Client-seitige Verzeichnis in den Ordner *\ext\lib* im Installationsverzeichnis von SPSS Modeler.
- Kopieren Sie das Server-seitige Verzeichnis in den Ordner *\ext\bin* im Installationsverzeichnis von SPSS Modeler.
- Starten Sie SPSS Modeler.
- Wenn die Erweiterung ein Menü oder ein Menüelement definiert, überprüfen Sie, ob dieses korrekt im Menüsystem angezeigt wird. Wenn die Erweiterung einen neuen Knoten definiert, stellen Sie sicher, dass der Knoten an der gewünschten Position auf der korrekten Knotenpalette erscheint, wie in der Spezifikationsdatei festgelegt.
- Testen Sie die Erweiterung gründlich.

  Stellen Sie z. B. Folgendes sicher:

  - Dass die Knotenleistung nicht nachlässt, auch wenn weitere Felder und Datensätze hinzukommen

- Dass Nullwerte konsistent behandelt werden
- Dass gegebenenfalls verschiedene Ländereinstellungen unterstützt werden (z. B. Europa, Ostasien)

▶ Nachdem Sie eine Erweiterung hinzugefügt haben, können Sie immer noch Änderungen an ihrer Spezifikationsdatei vornehmen. Jedoch werden diese Änderungen erst nach einem Neustart von SPSS Modeler wirksam.

Fehlersuche in einer CLEF-Erweiterung

Die folgenden Funktionen von CLEF helfen Ihnen bei der Fehlersuche in einer Erweiterung:

- XML-Syntaxfehlermeldungen
- Externe Ausführung
- Registerkarte “Fehlersuche”

XML-Syntaxfehler

Bei einem XML-Syntaxfehler in der Spezifikationsdatei gibt der XML-Parser eine Fehlermeldung zurück. Beispiel:

Abbildung 10-1
XML-Syntaxfehler

Die Meldung versucht, die Zeilennummer des Fehlers sowie die Fehlerursache möglichst genau zu bestimmen.

Führen Sie in diesem Fall die folgenden Schritte aus:

▶ Korrigieren Sie den Fehler in der Spezifikationsdatei.

▶ Testen Sie die Datei erneut, indem Sie das unter Testen einer CLEF-Erweiterung auf S. 251 beschriebene Testverfahren nochmals ausführen.

▶ Wiederholen Sie dieses Verfahren so oft, bis in der Spezifikationsdatei keine Syntaxfehler mehr gefunden werden.

Externe Ausführung

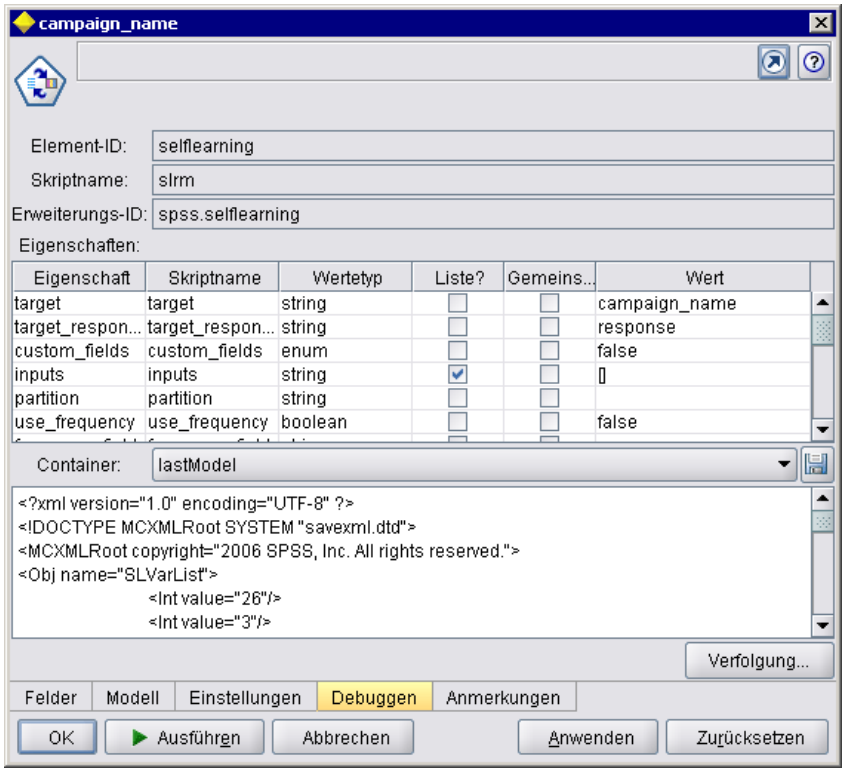
Eine benutzerdefinierte CLEF-Erweiterung wird normalerweise in einem eigenen Prozess, getrennt vom IBM® SPSS® Modeler-Prozess, ausgeführt. Diese Trennung kann bei der Fehlersuche hilfreich sein. Dadurch fällt bei einem Fehler im Erweiterungsprozess nicht der gesamte IBM® SPSS® Modeler Server-Prozess aus.

Anmerkung: Diese Standardeinstellung kann überschrieben werden. Für weitere Informationen siehe Thema Ändern der Ausführungsoptionen auf S. 254.

Verwenden der Registerkarte "Fehlersuche"

Sie können für jedes Dialogfeld (bzw. jeden Rahmen), das einem CLEF-Knoten oder einer CLEF-Ausgabe zugeordnet ist, die Registerkarte "Fehlersuche" aktivieren. Auf dieser Registerkarte können Sie die Eigenschafteneinstellungen des Objekts untersuchen. Auch die Inhalte der in der Erweiterung definierten Container können Sie anzeigen und für eine nähere Untersuchung in eine Datei speichern. Für weitere Informationen siehe Thema Container (Containers) in Kapitel 4 auf S. 68.

Abbildung 10-2
Registerkarte "Fehlersuche" in Dialogfeldern



Die Registerkarte "Fehlersuche" aktivieren Sie, indem Sie den Wert des Attributs debug des Elements Extension der Spezifikationsdatei auf true setzen. Für weitere Informationen siehe Thema Erweiterungselement in Kapitel 4 auf S. 42.

Die Registerkarte enthält folgende Felder:

Element-ID: Die eindeutige Kennung der Erweiterung (der Wert des Attributs id des Elements ExtensionDetails der Spezifikationsdatei).

Skriptname: Die eindeutige Kennung des Knotens in einem Skript (der Wert des Attributs scriptName des Elements Node).

Erweiterungs-ID: Der Name des Erweiterungsordners, in dem sich die Datei- und Verzeichnisressourcen der Erweiterung befinden. Der Wert ergibt sich aus der Verkettung der Attribute providerTag und id (getrennt durch einen Punkt (.)) des Elements ExtensionDetails. Die für die interne Verwendung reservierte Zeichenfolge spss darf im providerTag-Teil der Kennung nicht enthalten sein.

Eigenschaften: Diese Tabelle enthält ausgewählte Details der Property-Deklarationen des Knotens:

- **Eigenschaft**: Die eindeutige Kennung der Eigenschaft (der Wert des Felds name des Elements Property).
- **Skriptname**: Die eindeutige Kennung der Eigenschaft in einem Skript (der Wert des Attributs scriptName des Elements Property).
- **Wertetyp**: Der Wertetyp, den diese Eigenschaft annehmen kann (der Wert des Attributs valueType des Elements Property).
- **Liste?** Gibt an, ob die Eigenschaft eine Liste von Werten mit dem angegebenen Wertetyp ist (der Wert des Attributs isList des Elements Property).
- **Gemeinsam?** Wenn aktiviert, wird diese Eigenschaft an mehreren Stellen der Erweiterung verwendet (z. B. vom Modellerstellungs-, Modellausgabe- und Modellanwendungsknoten).
- **Wert**: Der Standardwert der Eigenschaft, sofern festgelegt.

Container: Zeigt den Inhalt des ausgewählten Containers an (z. B. Modelldaten). Wenn Sie auf dieses Feld klicken, wird eine Liste aller anderen, für die Erweiterung definierten Container angezeigt. Sie können dort den gewünschten Container auswählen, um dessen Inhalt anzuzeigen. Mit der nebenstehenden Schaltfläche Container speichern können Sie den Inhalt des ausgewählten Containers zur näheren Untersuchung in einer XML-Datei speichern.

Verfolgung: Öffnet bei der Ausführung des Knotens ein Dialogfeld mit der Trace-Ausgabe.

Ändern der Ausführungsoptionen

Ein benutzerdefiniertes CLEF-Erweiterungsmodul wird standardmäßig in einem eigenen Prozess, getrennt vom IBM® SPSS® Modeler-Hauptprozess, ausgeführt. Ein Fehler im Erweiterungsprozess führt auf diese Weise nicht zu einem Ausfall des SPSS Modeler-Prozesses. Die von IBM Corp. bereitgestellten Module werden hingegen standardmäßig im Hauptprozess ausgeführt.

Diese Standardeinstellung kann der Systemadministrator anhand zweier Konfigurationsoptionen für ein oder mehrere benannte Module in beide Richtungen ändern. Für beide Optionen werden die Modulkennungen der von der Änderung betroffenen Module in einer kommagetrennten Liste angegeben.

Anmerkung: Änderungen an diesen Optionen sollten nur auf ausdrückliche Anweisung eines Kundendienstmitarbeiters vorgenommen werden.

Es handelt sich hier um die folgenden Optionen:

Ausführungsoption "inprocess_execution" (Ausführung im Prozess)

Diese Option bewirkt, dass Erweiterungsmodule, die normalerweise in einem externen Prozess geladen werden (also in der Regel benutzerdefinierte Module), direkt in SPSS Modeler geladen werden. Format:

clef_inprocess_execution, "
Modul_ID1[,
Modul_ID2[,...
Modul_IDn]]"

Dabei ist *Modul_ID* der Wert des id-Attributs im ExtensionDetails-Element der relevanten Spezifikationsdatei. Beispiel:

clef_inprocess_execution, "test.example_filereader"

Ausführungsoption "external_execution" (externe Ausführung)

Diese Option bewirkt, dass Erweiterungsmodule, die normalerweise direkt in SPSS Modeler geladen werden (also in der Regel die von IBM Corp. bereitgestellten Module), in einem externen Prozess geladen werden. Format:

clef_external_execution, "
Modul_ID1[,
Modul_ID2[,...
Modul_IDn]]"

Modul_ID ist dabei ebenso wie bei der Option clef_inprocess_execution die Modulkennung. Hier ein fiktives Beispiel:

clef_external_execution, "spss.naivebayes,spss.terminator"

Hinzufügen oder Ändern einer Ausführungsoption

Informationen zum Hinzufügen oder Ändern einer Ausführungsoption finden Sie in der Anleitung des Abschnitts "Verwenden der Datei options.cfg" im *SPSS Modeler 15 Server-Verwaltungs- und Leistungshandbuch*.

Verteilen von CLEF-Erweiterungen

Wenn die neue Erweiterung gründlich getestet und bereit zur Verteilung ist, führen Sie die folgenden Schritte aus:

- Deaktivieren Sie die Registerkarte "Fehlersuche", falls diese aktiviert wurde. Für weitere Informationen siehe Thema Verwenden der Registerkarte "Fehlersuche" auf S. 253.
- Erstellen Sie eine Dateistruktur, die exakt der Struktur entspricht, in der die Erweiterungsdateien installiert werden sollen. Für weitere Informationen siehe Thema Dateistruktur in Kapitel 1 auf S. 6.

- Komprimieren Sie die Dateistruktur in eine .zip-Datei. Es ist einfacher, wenn Sie für die Client-seitige und die Server-seitige Installation jeweils separate .zip-Dateien anlegen.
- Verteilen Sie die .zip-Dateien an die Endbenutzer.

Installieren von CLEF-Erweiterungen

So installieren Sie eine CLEF-Erweiterung:

- Wenn Sie eine .zip-Datei mit der Dateistruktur einer Erweiterung erhalten, extrahieren Sie die clientseitigen Dateien in den Ordner \ext\lib im Installationsverzeichnis von IBM® SPSS® Modeler.
- Extrahieren Sie die serverseitigen Dateien in den Ordner \ext\bin im Installationsverzeichnis von SPSS Modeler (bzw. im Installationsverzeichnis von IBM® SPSS® Modeler Server, wenn Sie SPSS Modeler Server verwenden).
- Starten Sie IBM® SPSS® Modeler und überprüfen Sie, ob die neuen Knoten an den erwarteten Stellen der Knotenpalette angezeigt werden.

Deinstallieren von CLEF-Erweiterungen

So deinstallieren Sie eine CLEF-Erweiterung:

- Suchen Sie den Erweiterungsordner im Ordner \ext\lib im Installationsverzeichnis von IBM® SPSS® Modeler.

  Falls durch die Erweiterung auch ein serverseitiger Erweiterungsordner installiert wurde, suchen Sie diesen Ordner im Ordner \ext\bin im Installationsverzeichnis von SPSS Modeler bzw. IBM® SPSS® Modeler Server.
- Löschen Sie den bzw. die Erweiterungsordner.

  Die Änderung wird wirksam, sobald Sie IBM® SPSS® Modeler das nächste Mal starten.

Anhang

A

CLEF XML-Schema

CLEF-Elementreferenz

Dieser Abschnitt enthält eine Referenz für alle Elemente von CLEF.

Jedes Thema listet die gültigen Attribute eines Elements sowie seine über- und untergeordneten Elemente auf. Das Diagramm zeigt alle untergeordneten Elemente des Elements an. Die Pfeile im Diagramm verweisen auf Elemente, die mit anderen Elementen gemeinsam verwendet werden können. Diese Elemente werden im Inhaltsverzeichnis direkt unter diesem Thema ("CLEF-Elementreferenz"), nicht unter dem übergeordneten Element aufgeführt.

Action Element

Tabelle A-1
Attribute für Action

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
id	**erforderlich**		*string*
imagePath	optional		*string*
imagePathKey	optional		*string*
label	**erforderlich**		*string*
labelKey	optional		*string*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
shortcut	optional		*string*
shortcutKey	optional		*string*

XML-Darstellung

```
<xs:element name="Action">
    <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="imagePath" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="imagePathKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="shortcut" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="shortcutKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Actions Element

ActionButton Element

Tabelle A-2
Attribute für ActionButton

Attribut	Verwenden	Beschreibung	Gültige Werte
action	**erforderlich**		*string*
showIcon	optional		*boolean*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="ActionButton">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="action" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
   <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

ComboBoxControl Element, ExtensionObjectPanel Element, ModelViewerPanel Element, SelectorPanel Element, StaticText Element, SystemControls Element, TabbedPanel Element, TextBrowserPanel Element

Actions Element

XML-Darstellung

```
<xs:element name="Actions">
   <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
         <xs:element ref="Action"></xs:element>
      </xs:choice>
   </xs:sequence>
```

</xs:element>

Übergeordnet Elemente

CommonObjects Element

Untergeordnet Elemente

Action Element

AddField Element

Tabelle A-3
Attribute für AddField

Attribut	Verwenden	Beschreibung	Gültige Werte
direction	optional		**in** **out** **both** **none** **partition**
directionRef	optional		
fieldRef	optional		
group	optional		*string*
label	optional		*string*
missingValuesRef	optional		
name	**erforderlich**		
prefix	optional		*string*
role	optional		**unknown** **predictedValue** **predictedDisplayValue** **probability** **residual** **standardError** **entityId** **entityAffinity** **upperConfidenceLimit** **lowerConfidenceLimit** **propensity** **value** **supplementary**
storage	optional		**unknown** **integer** **real** **string** **date** **time** **timestamp**
storageRef	optional		
tag	optional		*string*
targetField	optional		*string*

Attribut	Verwenden	Beschreibung	Gültige Werte
type	optional		**auto** **range** **discrete** **set** **orderedSet** **flag** **typeless**
typeRef	optional		
value	optional		*string*

XML-Darstellung

```
<xs:element name="AddField">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Range" minOccurs="0"></xs:element>
         <xs:element ref="Values" minOccurs="0"></xs:element>
         <xs:element ref="NumericInfo" minOccurs="0"></xs:element>
         <xs:element name="MissingValues" minOccurs="0">
            <xs:sequence>
               <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"></xs:element>
               <xs:element ref="Range" minOccurs="0"></xs:element>
            </xs:sequence>
         </xs:element>
         <xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="name" type="FIELD-NAME" use="required"></xs:attribute>
   <xs:attribute name="storage" type="FIELD-STORAGE">
      <xs:enumeration value="unknown"></xs:enumeration>
      <xs:enumeration value="integer"></xs:enumeration>
      <xs:enumeration value="real"></xs:enumeration>
      <xs:enumeration value="string"></xs:enumeration>
      <xs:enumeration value="date"></xs:enumeration>
      <xs:enumeration value="time"></xs:enumeration>
      <xs:enumeration value="timestamp"></xs:enumeration>
   </xs:attribute>
   <xs:attribute name="type" type="FIELD-TYPE">
      <xs:enumeration value="auto"></xs:enumeration>
      <xs:enumeration value="range"></xs:enumeration>
      <xs:enumeration value="discrete"></xs:enumeration>
      <xs:enumeration value="set"></xs:enumeration>
      <xs:enumeration value="orderedSet"></xs:enumeration>
      <xs:enumeration value="flag"></xs:enumeration>
      <xs:enumeration value="typeless"></xs:enumeration>
   </xs:attribute>
   <xs:attribute name="direction" type="FIELD-DIRECTION">
      <xs:enumeration value="in"></xs:enumeration>
      <xs:enumeration value="out"></xs:enumeration>
      <xs:enumeration value="both"></xs:enumeration>
      <xs:enumeration value="none"></xs:enumeration>
      <xs:enumeration value="partition"></xs:enumeration>
   </xs:attribute>
   <xs:attribute name="label" type="xs:string"></xs:attribute>
   <xs:attribute name="fieldRef" type="EVALUATED-STRING" use="optional"></xs:attribute>
```

```
    <xs:attribute name="storageRef" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="typeRef" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="directionRef" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="missingValuesRef" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="role" type="MODEL-FIELD-ROLE" use="optional">
        <xs:enumeration value="unknown"></xs:enumeration>
        <xs:enumeration value="predictedValue"></xs:enumeration>
        <xs:enumeration value="predictedDisplayValue"></xs:enumeration>
        <xs:enumeration value="probability"></xs:enumeration>
        <xs:enumeration value="residual"></xs:enumeration>
        <xs:enumeration value="standardError"></xs:enumeration>
        <xs:enumeration value="entityId"></xs:enumeration>
        <xs:enumeration value="entityAffinity"></xs:enumeration>
        <xs:enumeration value="upperConfidenceLimit"></xs:enumeration>
        <xs:enumeration value="lowerConfidenceLimit"></xs:enumeration>
        <xs:enumeration value="propensity"></xs:enumeration>
        <xs:enumeration value="value"></xs:enumeration>
        <xs:enumeration value="supplementary"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="targetField" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="value" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="group" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="tag" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="prefix" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ForEach Element, ModelFields Element

Untergeordnet Elemente

MissingValues Element, ModelField Element, NumericInfo Element, Range Element, Values Element

Verwandt Elemente

ChangeField Element

MissingValues Element

Tabelle A-4
Attribute für MissingValues

Attribut	Verwenden	Beschreibung	Gültige Werte
treatNullAsMissing	optional		*boolean*
treatWhitespaceAsMissing	optional		*boolean*

XML-Darstellung

```
<xs:element name="MissingValues" minOccurs="0">
    <xs:sequence>
        <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"></xs:element>
        <xs:element ref="Range" minOccurs="0"></xs:element>
    </xs:sequence>
```

```
    <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" use="optional"
    default="true"></xs:attribute>
    <xs:attribute name="treatNullAsMissing" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

AddField Element

Untergeordnet Elemente

Range Element, Values Element

ModelField Element

Tabelle A-5
Attribute für ModelField

Attribut	Verwenden	Beschreibung	Gültige Werte
group	optional		
role	**erforderlich**		**unknown** **predictedValue** **predictedDisplayValue** **probability** **residual** **standardError** **entityId** **entityAffinity** **upperConfidenceLimit** **lowerConfidenceLimit** **propensity** **value** **supplementary**
tag	optional		*string*
targetField	optional		*string*
value	optional		*string*

XML-Darstellung

```
<xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
    <xs:attribute name="role" type="MODEL-FIELD-ROLE" use="required">
        <xs:enumeration value="unknown"></xs:enumeration>
        <xs:enumeration value="predictedValue"></xs:enumeration>
        <xs:enumeration value="predictedDisplayValue"></xs:enumeration>
        <xs:enumeration value="probability"></xs:enumeration>
        <xs:enumeration value="residual"></xs:enumeration>
        <xs:enumeration value="standardError"></xs:enumeration>
        <xs:enumeration value="entityId"></xs:enumeration>
        <xs:enumeration value="entityAffinity"></xs:enumeration>
        <xs:enumeration value="upperConfidenceLimit"></xs:enumeration>
        <xs:enumeration value="lowerConfidenceLimit"></xs:enumeration>
        <xs:enumeration value="propensity"></xs:enumeration>
        <xs:enumeration value="value"></xs:enumeration>
        <xs:enumeration value="supplementary"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="targetField" type="xs:string"></xs:attribute>
```

```
    <xs:attribute name="value" type="xs:string"></xs:attribute>
    <xs:attribute name="group" type="MODEL-FIELD-GROUP"></xs:attribute>
    <xs:attribute name="tag" type="xs:string"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

AddField Element

And Element

XML-Darstellung

```
<xs:element name="And">
    <xs:sequence minOccurs="2" maxOccurs="unbounded">
        <xs:group ref="CONDITION-EXPRESSION">
            <xs:choice>
                <xs:element ref="Condition"></xs:element>
                <xs:element ref="And"></xs:element>
                <xs:element ref="Or"></xs:element>
                <xs:element ref="Not"></xs:element>
            </xs:choice>
        </xs:group>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

Command Element, Constraint Element, CreateDocument Element, CreateDocumentOutput Element, CreateInteractiveDocumentBuilder Element, CreateInteractiveModelBuilder Element, CreateModel Element, CreateModelApplier Element, CreateModelOutput Element, Enabled Element, Not Element, Option Element, Or Element, Run Element, Visible Element

Untergeordnet Elemente

And Element, Condition Element, Not Element, Or Element

Attribute Element

Tabelle A-6
Attribute für Attribute

Attribut	Verwenden	Beschreibung	Gültige Werte
defaultValue	optional		
description	optional		*string*
descriptionKey	optional		*string*
isList	optional		*boolean*
label	**erforderlich**		*string*
labelKey	optional		*string*

Attribut	Verwenden	Beschreibung	Gültige Werte
name	**erforderlich**		*string*
valueType	optional		**string** **encryptedString** **integer** **double** **boolean** **date** **enum**

XML-Darstellung

```
<xs:element name="Attribute">
    <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="valueType" type="ATTRIBUTE-VALUE-TYPE">
        <xs:enumeration value="string"></xs:enumeration>
        <xs:enumeration value="encryptedString"></xs:enumeration>
        <xs:enumeration value="integer"></xs:enumeration>
        <xs:enumeration value="double"></xs:enumeration>
        <xs:enumeration value="boolean"></xs:enumeration>
        <xs:enumeration value="date"></xs:enumeration>
        <xs:enumeration value="enum"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="defaultValue" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="isList" type="xs:boolean" use="optional" default="false"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Catalog Element, Structure Element

BinaryFormat Element

XML-Darstellung

```
<xs:element name="BinaryFormat"></xs:element>
```

Übergeordnet Elemente

FileFormatType Element

Catalog Element

Tabelle A-7
Attribute für Catalog

Attribut	Verwenden	Beschreibung	Gültige Werte
id	**erforderlich**		*string*
valueColumn	**erforderlich**		*integer*

XML-Darstellung

```
<xs:element name="Catalog">
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
        <xs:element ref="Attribute"></xs:element>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="valueColumn" type="xs:integer" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Catalogs Element

Untergeordnet Elemente

Attribute Element

Catalogs Element

XML-Darstellung

```
<xs:element name="Catalogs">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:choice>
            <xs:element ref="Catalog"></xs:element>
        </xs:choice>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

CommonObjects Element

Untergeordnet Elemente

Catalog Element

ChangeField Element

Tabelle A-8
Attribute für ChangeField

Attribut	Verwenden	Beschreibung	Gültige Werte
direction	optional		**in** **out** **both** **none** **partition**
directionRef	optional		
fieldRef	**erforderlich**		
label	optional		*string*
missingValuesRef	optional		

Attribut	Verwenden	Beschreibung	Gültige Werte
name	**erforderlich**		
storage	optional		**unknown** **integer** **real** **string** **date** **time** **timestamp**
storageRef	optional		
type	optional		**auto** **range** **discrete** **set** **orderedSet** **flag** **typeless**
typeRef	optional		

XML-Darstellung

```
<xs:element name="ChangeField">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Range" minOccurs="0"></xs:element>
         <xs:element ref="Values" minOccurs="0"></xs:element>
         <xs:element ref="NumericInfo" minOccurs="0"></xs:element>
         <xs:element name="MissingValues" minOccurs="0">
            <xs:sequence>
               <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"></xs:element>
               <xs:element ref="Range" minOccurs="0"></xs:element>
            </xs:sequence>
         </xs:element>
         <xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="name" type="FIELD-NAME" use="required"></xs:attribute>
   <xs:attribute name="storage" type="FIELD-STORAGE">
      <xs:enumeration value="unknown"></xs:enumeration>
      <xs:enumeration value="integer"></xs:enumeration>
      <xs:enumeration value="real"></xs:enumeration>
      <xs:enumeration value="string"></xs:enumeration>
      <xs:enumeration value="date"></xs:enumeration>
      <xs:enumeration value="time"></xs:enumeration>
      <xs:enumeration value="timestamp"></xs:enumeration>
   </xs:attribute>
   <xs:attribute name="type" type="FIELD-TYPE">
      <xs:enumeration value="auto"></xs:enumeration>
      <xs:enumeration value="range"></xs:enumeration>
      <xs:enumeration value="discrete"></xs:enumeration>
      <xs:enumeration value="set"></xs:enumeration>
      <xs:enumeration value="orderedSet"></xs:enumeration>
      <xs:enumeration value="flag"></xs:enumeration>
      <xs:enumeration value="typeless"></xs:enumeration>
   </xs:attribute>
   <xs:attribute name="direction" type="FIELD-DIRECTION">
```

```
        <xs:enumeration value="in"></xs:enumeration>
        <xs:enumeration value="out"></xs:enumeration>
        <xs:enumeration value="both"></xs:enumeration>
        <xs:enumeration value="none"></xs:enumeration>
        <xs:enumeration value="partition"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="label" type="xs:string"></xs:attribute>
    <xs:attribute name="fieldRef" type="EVALUATED-STRING" use="required"></xs:attribute>
    <xs:attribute name="storageRef" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="typeRef" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="directionRef" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="missingValuesRef" type="EVALUATED-STRING" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ForEach Element, ModelFields Element

Untergeordnet Elemente

MissingValues Element, ModelField Element, NumericInfo Element, Range Element, Values Element

Verwandt Elemente

AddField Element

MissingValues Element

Tabelle A-9
Attribute für MissingValues

Attribut	Verwenden	Beschreibung	Gültige Werte
treatNullAsMissing	optional		*boolean*
treatWhitespaceAsMissing	optional		*boolean*

XML-Darstellung

```
<xs:element name="MissingValues" minOccurs="0">
    <xs:sequence>
        <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"></xs:element>
        <xs:element ref="Range" minOccurs="0"></xs:element>
    </xs:sequence>
    <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" use="optional"
    default="true"></xs:attribute>
    <xs:attribute name="treatNullAsMissing" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ChangeField Element

Untergeordnet Elemente

Range Element, Values Element

ModelField Element

Tabelle A-10
Attribute für ModelField

Attribut	Verwenden	Beschreibung	Gültige Werte
group	optional		
role	**erforderlich**		**unknown** **predictedValue** **predictedDisplayValue** **probability** **residual** **standardError** **entityId** **entityAffinity** **upperConfidenceLimit** **lowerConfidenceLimit** **propensity** **value** **supplementary**
tag	optional		*string*
targetField	optional		*string*
value	optional		*string*

XML-Darstellung

```
<xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
    <xs:attribute name="role" type="MODEL-FIELD-ROLE" use="required">
        <xs:enumeration value="unknown"></xs:enumeration>
        <xs:enumeration value="predictedValue"></xs:enumeration>
        <xs:enumeration value="predictedDisplayValue"></xs:enumeration>
        <xs:enumeration value="probability"></xs:enumeration>
        <xs:enumeration value="residual"></xs:enumeration>
        <xs:enumeration value="standardError"></xs:enumeration>
        <xs:enumeration value="entityId"></xs:enumeration>
        <xs:enumeration value="entityAffinity"></xs:enumeration>
        <xs:enumeration value="upperConfidenceLimit"></xs:enumeration>
        <xs:enumeration value="lowerConfidenceLimit"></xs:enumeration>
        <xs:enumeration value="propensity"></xs:enumeration>
        <xs:enumeration value="value"></xs:enumeration>
        <xs:enumeration value="supplementary"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="targetField" type="xs:string"></xs:attribute>
    <xs:attribute name="value" type="xs:string"></xs:attribute>
    <xs:attribute name="group" type="MODEL-FIELD-GROUP"></xs:attribute>
    <xs:attribute name="tag" type="xs:string"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ChangeField Element

CheckBoxControl Element

Tabelle A-11
Attribute für CheckBoxControl

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
invert	optional		*boolean*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="CheckBoxControl">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="invert" type="xs:boolean" use="optional" default="false"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl

Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

CheckBoxGroupControl Element

Tabelle A-12
Attribute für CheckBoxGroupControl

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
layoutByRow	optional		*boolean*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
rows	optional		*positiveInteger*
showLabel	optional		*boolean*
useSubPanel	optional		*boolean*

XML-Darstellung

```
<xs:element name="CheckBoxGroupControl">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
   <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
   <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
   <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
   <xs:attribute name="layoutByRow" type="xs:boolean" use="optional" default="true"></xs:attribute>
   <xs:attribute name="useSubPanel" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

ClientDirectoryChooserControl Element

Tabelle A-13
Attribute für ClientDirectoryChooserControl

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
mode	**erforderlich**		**open** **save** **import** **export**
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="ClientDirectoryChooserControl">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
   <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
```

```
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
        <xs:enumeration value="open"></xs:enumeration>
        <xs:enumeration value="save"></xs:enumeration>
        <xs:enumeration value="import"></xs:enumeration>
        <xs:enumeration value="export"></xs:enumeration>
    </xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

ClientFileChooserControl Element

Tabelle A-14
Attribute für ClientFileChooserControl

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
mode	**erforderlich**		**open** **save** **import** **export**
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="ClientFileChooserControl">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
        <xs:enumeration value="open"></xs:enumeration>
        <xs:enumeration value="save"></xs:enumeration>
        <xs:enumeration value="import"></xs:enumeration>
        <xs:enumeration value="export"></xs:enumeration>
    </xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

ComboBoxControl Element

Tabelle A-15
Attribute für ComboBoxControl

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*

Attribut	Verwenden	Beschreibung	Gültige Werte
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="ComboBoxControl" type="CONTROLLER">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
   <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
   <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
   <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Tabelle A-16
Erweitert Typen

Geben Sie	Beschreibung
ItemChooserControl	

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

ActionButton Element, ExtensionObjectPanel Element, ModelViewerPanel Element, SelectorPanel Element, StaticText Element, SystemControls Element, TabbedPanel Element, TextBrowserPanel Element

Command Element

Tabelle A-17
Attribute für Command

Attribut	Verwenden	Beschreibung	Gültige Werte
path	**erforderlich**		

XML-Darstellung

```
<xs:element name="Command">
   <xs:sequence>
      <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
         <xs:choice>
            <xs:element ref="Condition"></xs:element>
            <xs:element ref="And"></xs:element>
            <xs:element ref="Or"></xs:element>
            <xs:element ref="Not"></xs:element>
         </xs:choice>
      </xs:group>
   </xs:sequence>
   <xs:attribute name="path" type="EVALUATED-STRING" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Run Element

Untergeordnet Elemente

And Element, Condition Element, Not Element, Or Element

CommonObjects Element

Provides a location for definitions that are global to the extension

Tabelle A-18
Attribute für CommonObjects

Attribut	Verwenden	Beschreibung	Gültige Werte
extensionListenerClass	optional		*string*

XML-Darstellung

```
<xs:element name="CommonObjects">
   <xs:all>
      <xs:element ref="PropertyTypes" minOccurs="0"></xs:element>
      <xs:element ref="PropertySets" minOccurs="0"></xs:element>
      <xs:element ref="FileFormatTypes" minOccurs="0"></xs:element>
      <xs:element ref="ContainerTypes" minOccurs="0"></xs:element>
      <xs:element ref="Actions" minOccurs="0"></xs:element>
      <xs:element ref="Catalogs" minOccurs="0"></xs:element>
   </xs:all>
   <xs:attribute name="extensionListenerClass" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Extension Element

Untergeordnet Elemente

Actions Element, Catalogs Element, ContainerTypes Element, FileFormatTypes Element, PropertySets Element, PropertyTypes Element

Condition Element

Tabelle A-19
Attribute für Condition

Attribut	Verwenden	Beschreibung	Gültige Werte
container	optional		*string*
control	optional		*string*
expression	optional		
listMode	optional		**all** **any**
op	**erforderlich**		**equals** **notEquals** **isEmpty** **isNotEmpty** **lessThan** **lessOrEquals** **greaterThan** **greaterOrEquals** **equalsIgnoreCase** **isSubstring** **startsWith** **endsWith** **startsWithIgnoreCase** **endsWithIgnoreCase** **isSubstring** **hasSubstring** **isSubstringIgnoreCase** **hasSubstringIgnoreCase** **in** **countEquals** **countLessThan** **countLessOrEquals** **countGreaterThan** **countGreaterOrEquals** **contains** **storageEquals** **typeEquals** **directionEquals** **isMeasureDiscrete** **isMeasureContinuous** **isMeasureTypeless** **isMeasureUnknown** **isStorageString** **isStorageNumeric** **isStorageDatetime** **isStorageUnknown** **isModelOutput** **modelOutputRoleEquals**

Attribut	Verwenden	Beschreibung	Gültige Werte
			modelOutputTargetField-Equals **modelOutputHasValue** **modelOutputTagEquals** **enumRestriction**
property	optional		*string*
value	optional		

XML-Darstellung

```
<xs:element name="Condition">
    <xs:attribute name="expression" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="control" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="property" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="container" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="op" type="CONDITION-TEST" use="required">
        <xs:enumeration value="equals"></xs:enumeration>
        <xs:enumeration value="notEquals"></xs:enumeration>
        <xs:enumeration value="isEmpty"></xs:enumeration>
        <xs:enumeration value="isNotEmpty"></xs:enumeration>
        <xs:enumeration value="lessThan"></xs:enumeration>
        <xs:enumeration value="lessOrEquals"></xs:enumeration>
        <xs:enumeration value="greaterThan"></xs:enumeration>
        <xs:enumeration value="greaterOrEquals"></xs:enumeration>
        <xs:enumeration value="equalsIgnoreCase"></xs:enumeration>
        <xs:enumeration value="isSubstring"></xs:enumeration>
        <xs:enumeration value="startsWith"></xs:enumeration>
        <xs:enumeration value="endsWith"></xs:enumeration>
        <xs:enumeration value="startsWithIgnoreCase"></xs:enumeration>
        <xs:enumeration value="endsWithIgnoreCase"></xs:enumeration>
        <xs:enumeration value="isSubstring"></xs:enumeration>
        <xs:enumeration value="hasSubstring"></xs:enumeration>
        <xs:enumeration value="isSubstringIgnoreCase"></xs:enumeration>
        <xs:enumeration value="hasSubstringIgnoreCase"></xs:enumeration>
        <xs:enumeration value="in"></xs:enumeration>
        <xs:enumeration value="countEquals"></xs:enumeration>
        <xs:enumeration value="countLessThan"></xs:enumeration>
        <xs:enumeration value="countLessOrEquals"></xs:enumeration>
        <xs:enumeration value="countGreaterThan"></xs:enumeration>
        <xs:enumeration value="countGreaterOrEquals"></xs:enumeration>
        <xs:enumeration value="contains"></xs:enumeration>
        <xs:enumeration value="storageEquals"></xs:enumeration>
        <xs:enumeration value="typeEquals"></xs:enumeration>
        <xs:enumeration value="directionEquals"></xs:enumeration>
        <xs:enumeration value="isMeasureDiscrete"></xs:enumeration>
        <xs:enumeration value="isMeasureContinuous"></xs:enumeration>
        <xs:enumeration value="isMeasureTypeless"></xs:enumeration>
        <xs:enumeration value="isMeasureUnknown"></xs:enumeration>
        <xs:enumeration value="isStorageString"></xs:enumeration>
        <xs:enumeration value="isStorageNumeric"></xs:enumeration>
        <xs:enumeration value="isStorageDatetime"></xs:enumeration>
        <xs:enumeration value="isStorageUnknown"></xs:enumeration>
        <xs:enumeration value="isModelOutput"></xs:enumeration>
        <xs:enumeration value="modelOutputRoleEquals"></xs:enumeration>
        <xs:enumeration value="modelOutputTargetFieldEquals"></xs:enumeration>
```

```
        <xs:enumeration value="modelOutputHasValue"></xs:enumeration>
        <xs:enumeration value="modelOutputTagEquals"></xs:enumeration>
        <xs:enumeration value="enumRestriction"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="value" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="listMode" use="optional" default="all">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="all"></xs:enumeration>
                <xs:enumeration value="any"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:element>
```

Übergeordnet Elemente

And Element, Command Element, Constraint Element, CreateDocument Element, CreateDocumentOutput Element, CreateInteractiveDocumentBuilder Element, CreateInteractiveModelBuilder Element, CreateModel Element, CreateModelApplier Element, CreateModelOutput Element, Enabled Element, ExpertSettings Element, Not Element, Option Element, Or Element, Run Element, Visible Element

Constraint Element

Tabelle A-20
Attribute für Constraint

Attribut	Verwenden	Beschreibung	Gültige Werte
property	**erforderlich**		*string*
singleSelection	optional		*boolean*

XML-Darstellung

```
<xs:element name="Constraint">
    <xs:sequence>
        <xs:group ref="CONDITION-EXPRESSION">
            <xs:choice>
                <xs:element ref="Condition"></xs:element>
                <xs:element ref="And"></xs:element>
                <xs:element ref="Or"></xs:element>
                <xs:element ref="Not"></xs:element>
            </xs:choice>
        </xs:group>
    </xs:sequence>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="singleSelection" type="xs:boolean" use="optional" default="false"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

AutoModeling Element

Untergeordnet Elemente

And Element, Condition Element, Not Element, Or Element

Constructors Element

XML-Darstellung

```
<xs:element name="Constructors">
   <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
         <xs:element ref="CreateModelOutput"></xs:element>
         <xs:element ref="CreateDocumentOutput"></xs:element>
         <xs:element ref="CreateInteractiveModelBuilder"></xs:element>
         <xs:element ref="CreateInteractiveDocumentBuilder"></xs:element>
         <xs:element ref="CreateModelApplier"></xs:element>
      </xs:choice>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

DocumentOutput Element, Execution Element, InteractiveDocumentBuilder Element, InteractiveModelBuilder Element, ModelOutput Element, Node Element

Untergeordnet Elemente

CreateDocumentOutput Element, CreateInteractiveDocumentBuilder Element, CreateInteractiveModelBuilder Element, CreateModelApplier Element, CreateModelOutput Element

Container Element

Tabelle A-21
Attribute für Container

Attribut	Verwenden	Beschreibung	Gültige Werte
name	**erforderlich**		*string*
type	optional		*string*

XML-Darstellung

```
<xs:element name="Container">
   <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="type" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Containers Element, Containers Element, Containers Element, Containers Element, Containers Element

ContainerFile Element

Tabelle A-22
Attribute für ContainerFile

Attribut	Verwenden	Beschreibung	Gültige Werte
container	optional		
containerType	optional		
path	**erforderlich**		

XML-Darstellung

```
<xs:element name="ContainerFile" type="SERVER-CONTAINER-FILE">
    <xs:attribute name="path" type="EVALUATED-STRING" use="required"></xs:attribute>
    <xs:attribute name="container" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="containerType" type="EVALUATED-STRING" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

InputFiles Element, OutputFiles Element

ContainerTypes Element

XML-Darstellung

```
<xs:element name="ContainerTypes">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:choice>
            <xs:element ref="DocumentType"></xs:element>
            <xs:element ref="ModelType"></xs:element>
        </xs:choice>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

CommonObjects Element

Untergeordnet Elemente

DocumentType Element, ModelType Element

Controls Element

XML-Darstellung

```
<xs:element name="Controls">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:choice>
            <xs:element ref="Menu"></xs:element>
            <xs:element ref="MenuItem"></xs:element>
            <xs:element ref="ToolbarItem"></xs:element>
```

```
        </xs:choice>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

UserInterface Element

Untergeordnet Elemente

Menu Element, MenuItem Element, ToolbarItem Element

CreateDocument Element

Tabelle A-23
Attribute für CreateDocument

Attribut	Verwenden	Beschreibung	Gültige Werte
sourceFile	**erforderlich**		*string*
target	**erforderlich**		*string*
type	optional		*string*

XML-Darstellung

```
<xs:element name="CreateDocument">
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
        <xs:choice>
            <xs:element ref="Condition"></xs:element>
            <xs:element ref="And"></xs:element>
            <xs:element ref="Or"></xs:element>
            <xs:element ref="Not"></xs:element>
        </xs:choice>
    </xs:group>
    <xs:attribute name="sourceFile" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="target" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="type" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

CreateDocumentOutput Element, CreateInteractiveDocumentBuilder Element, CreateInteractiveModelBuilder Element, CreateModelApplier Element, CreateModelOutput Element

Untergeordnet Elemente

And Element, Condition Element, Not Element, Or Element

Verwandt Elemente

CreateModel Element

CreateDocumentOutput Element

Tabelle A-24
Attribute für CreateDocumentOutput

Attribut	Verwenden	Beschreibung	Gültige Werte
type	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="CreateDocumentOutput">
   <xs:sequence>
      <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
         <xs:choice>
            <xs:element ref="Condition"></xs:element>
            <xs:element ref="And"></xs:element>
            <xs:element ref="Or"></xs:element>
            <xs:element ref="Not"></xs:element>
         </xs:choice>
      </xs:group>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:choice>
            <xs:element ref="SetProperty"></xs:element>
            <xs:element ref="SetContainer"></xs:element>
            <xs:element ref="CreateModel"></xs:element>
            <xs:element ref="CreateDocument"></xs:element>
         </xs:choice>
      </xs:sequence>
   </xs:sequence>
   <xs:attribute name="type" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Constructors Element

Untergeordnet Elemente

And Element, Condition Element, CreateDocument Element, CreateModel Element, Not Element, Or Element, SetContainer Element, SetProperty Element

Verwandt Elemente

CreateInteractiveDocumentBuilder Element, CreateInteractiveModelBuilder Element, CreateModelApplier Element, CreateModelOutput Element

CreateInteractiveDocumentBuilder Element

Tabelle A-25
Attribute für CreateInteractiveDocumentBuilder

Attribut	Verwenden	Beschreibung	Gültige Werte
type	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="CreateInteractiveDocumentBuilder">
   <xs:sequence>
      <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
         <xs:choice>
            <xs:element ref="Condition"></xs:element>
            <xs:element ref="And"></xs:element>
            <xs:element ref="Or"></xs:element>
            <xs:element ref="Not"></xs:element>
         </xs:choice>
      </xs:group>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:choice>
            <xs:element ref="SetProperty"></xs:element>
            <xs:element ref="SetContainer"></xs:element>
            <xs:element ref="CreateModel"></xs:element>
            <xs:element ref="CreateDocument"></xs:element>
         </xs:choice>
      </xs:sequence>
   </xs:sequence>
   <xs:attribute name="type" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Constructors Element

Untergeordnet Elemente

And Element, Condition Element, CreateDocument Element, CreateModel Element, Not Element, Or Element, SetContainer Element, SetProperty Element

Verwandt Elemente

CreateDocumentOutput Element, CreateInteractiveModelBuilder Element, CreateModelApplier Element, CreateModelOutput Element

CreateInteractiveModelBuilder Element

Tabelle A-26
Attribute für CreateInteractiveModelBuilder

Attribut	Verwenden	Beschreibung	Gültige Werte
type	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="CreateInteractiveModelBuilder">
   <xs:sequence>
      <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
         <xs:choice>
            <xs:element ref="Condition"></xs:element>
            <xs:element ref="And"></xs:element>
            <xs:element ref="Or"></xs:element>
```

```
            <xs:element ref="Not"></xs:element>
        </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:choice>
            <xs:element ref="SetProperty"></xs:element>
            <xs:element ref="SetContainer"></xs:element>
            <xs:element ref="CreateModel"></xs:element>
            <xs:element ref="CreateDocument"></xs:element>
        </xs:choice>
    </xs:sequence>
</xs:sequence>
<xs:attribute name="type" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Constructors Element

Untergeordnet Elemente

And Element, Condition Element, CreateDocument Element, CreateModel Element, Not Element, Or Element, SetContainer Element, SetProperty Element

Verwandt Elemente

CreateDocumentOutput Element, CreateInteractiveDocumentBuilder Element, CreateModelApplier Element, CreateModelOutput Element

CreateModel Element

Tabelle A-27
Attribute für CreateModel

Attribut	Verwenden	Beschreibung	Gültige Werte
signatureFile	optional		*string*
sourceFile	**erforderlich**		*string*
target	**erforderlich**		*string*
type	optional		*string*

XML-Darstellung

```
<xs:element name="CreateModel">
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
        <xs:choice>
            <xs:element ref="Condition"></xs:element>
            <xs:element ref="And"></xs:element>
            <xs:element ref="Or"></xs:element>
            <xs:element ref="Not"></xs:element>
        </xs:choice>
    </xs:group>
    <xs:attribute name="sourceFile" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="target" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="type" type="xs:string" use="optional"></xs:attribute>
```

```
    <xs:sequence>
        <xs:element name="ModelDetail" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
    <xs:attribute name="signatureFile" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

CreateDocumentOutput Element, CreateInteractiveDocumentBuilder Element, CreateInteractiveModelBuilder Element, CreateModelApplier Element, CreateModelOutput Element

Untergeordnet Elemente

And Element, Condition Element, ModelDetail Element, Not Element, Or Element

Verwandt Elemente

CreateDocument Element

ModelDetail Element

Tabelle A-28
Attribute für ModelDetail

Attribut	Verwenden	Beschreibung	Gültige Werte
algorithm	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="ModelDetail" maxOccurs="unbounded">
    <xs:attribute name="algorithm" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

CreateModel Element

CreateModelApplier Element

Tabelle A-29
Attribute für CreateModelApplier

Attribut	Verwenden	Beschreibung	Gültige Werte
type	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="CreateModelApplier">
    <xs:sequence>
        <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
            <xs:choice>
                <xs:element ref="Condition"></xs:element>
                <xs:element ref="And"></xs:element>
```

```
            <xs:element ref="Or"></xs:element>
            <xs:element ref="Not"></xs:element>
        </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:choice>
            <xs:element ref="SetProperty"></xs:element>
            <xs:element ref="SetContainer"></xs:element>
            <xs:element ref="CreateModel"></xs:element>
            <xs:element ref="CreateDocument"></xs:element>
        </xs:choice>
    </xs:sequence>
</xs:sequence>
<xs:attribute name="type" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Constructors Element

Untergeordnet Elemente

And Element, Condition Element, CreateDocument Element, CreateModel Element, Not Element, Or Element, SetContainer Element, SetProperty Element

Verwandt Elemente

CreateDocumentOutput Element, CreateInteractiveDocumentBuilder Element, CreateInteractiveModelBuilder Element, CreateModelOutput Element

CreateModelOutput Element

Tabelle A-30
Attribute für CreateModelOutput

Attribut	Verwenden	Beschreibung	Gültige Werte
type	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="CreateModelOutput">
    <xs:sequence>
        <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
            <xs:choice>
                <xs:element ref="Condition"></xs:element>
                <xs:element ref="And"></xs:element>
                <xs:element ref="Or"></xs:element>
                <xs:element ref="Not"></xs:element>
            </xs:choice>
        </xs:group>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:choice>
                <xs:element ref="SetProperty"></xs:element>
                <xs:element ref="SetContainer"></xs:element>
                <xs:element ref="CreateModel"></xs:element>
```

```
            <xs:element ref="CreateDocument"></xs:element>
          </xs:choice>
        </xs:sequence>
    </xs:sequence>
    <xs:attribute name="type" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Constructors Element

Untergeordnet Elemente

And Element, Condition Element, CreateDocument Element, CreateModel Element, Not Element, Or Element, SetContainer Element, SetProperty Element

Verwandt Elemente

CreateDocumentOutput Element, CreateInteractiveDocumentBuilder Element, CreateInteractiveModelBuilder Element, CreateModelApplier Element

DatabaseConnectionValue Element

A value specifying the details of a database connection.

Tabelle A-31
Attribute für DatabaseConnectionValue

Attribut	Verwenden	Beschreibung	Gültige Werte
connectionType	**erforderlich**		*string*
datasourceName	**erforderlich**		*string*
password	**erforderlich**		*string*
userName	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="DatabaseConnectionValue" type="DATABASE-CONNECTION-VALUE">
    <xs:attribute name="connectionType" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="datasourceName" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="userName" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="password" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ListValue Element, Attribute Element, ListValue Element, Parameter Element, Attribute Element, ListValue Element

DataFile Element

Tabelle A-32
Attribute für DataFile

Attribut	Verwenden	Beschreibung	Gültige Werte
path	**erforderlich**		

XML-Darstellung

```
<xs:element name="DataFile" type="SERVER-DATA-FILE">
    <xs:attribute name="path" type="EVALUATED-STRING" use="required"></xs:attribute>
    <xs:choice>
        <xs:element ref="DelimitedDataFormat"></xs:element>
    </xs:choice>
</xs:element>
```

Übergeordnet Elemente

InputFiles Element, OutputFiles Element

Untergeordnet Elemente

DelimitedDataFormat Element

DataFormat Element

XML-Darstellung

```
<xs:element name="DataFormat">
    <xs:group ref="DATA-FORMAT-TYPE">
        <xs:choice>
            <xs:element ref="DelimitedDataFormat"></xs:element>
            <xs:element ref="SPSSDataFormat"></xs:element>
        </xs:choice>
    </xs:group>
</xs:element>
```

Übergeordnet Elemente

FileFormatType Element

Untergeordnet Elemente

DelimitedDataFormat Element, SPSSDataFormat Element

DataModel Element

The data model coming into or out of a node. An input/Output data model is a set of Fieldsl.

XML-Darstellung

```
<xs:element name="DataModel" type="DATA-MODEL">
```

```
    <xs:sequence>
        <xs:element name="FieldFormats" type="FIELD-FORMATS" minOccurs="0">
            <xs:sequence>
                <xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0"
                maxOccurs="unbounded"></xs:element>
            </xs:sequence>
        </xs:element>
        <xs:element name="FieldGroups" type="FIELD-GROUPS" minOccurs="0">
            <xs:sequence>
                <xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0"
                maxOccurs="unbounded">
                    <xs:sequence>
                        <xs:element name="FieldName"></xs:element>
                    </xs:sequence>
                </xs:element>
            </xs:sequence>
        </xs:element>
        <xs:element name="Fields" type="FIELDS">
            <xs:sequence>
                <xs:element name="Field" type="FIELD" minOccurs="0" maxOccurs="unbounded">
                    <xs:group ref="FIELD-CONTENT">
                        <xs:sequence>
                            <xs:element ref="DisplayLabel"></xs:element>
                            <xs:choice minOccurs="0">
                                <xs:element ref="Range"></xs:element>
                                <xs:element ref="Values"></xs:element>
                            </xs:choice>
                            <xs:element ref="MissingValues"></xs:element>
                        </xs:sequence>
                    </xs:group>
                </xs:element>
            </xs:sequence>
        </xs:element>
    </xs:sequence>
</xs:element>
```

Untergeordnet Elemente

FieldFormats Element, FieldGroups Element, Fields Element

FieldFormats Element

Defines the default field formats. Field formats are used when displaying values in output such as the general format (standard number, scientific or currency formats), number of decimal places to display, decimal separator etc. Currently field formats are only used for numeric fields although this may change in future versions.

Tabelle A-33
Attribute für FieldFormats

Attribut	Verwenden	Beschreibung	Gültige Werte
count	optional		*nonNegativeInteger*
defaultNumberFormat	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="FieldFormats" type="FIELD-FORMATS" minOccurs="0">
    <xs:sequence>
        <xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0"
        maxOccurs="unbounded"></xs:element>
    </xs:sequence>
    <xs:attribute name="defaultNumberFormat" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

DataModel Element

Untergeordnet Elemente

NumberFormat Element

NumberFormat Element

Defines format information for a numeric field.

Tabelle A-34
Attribute für NumberFormat

Attribut	Verwenden	Beschreibung	Gültige Werte
decimalPlaces	**erforderlich**		*nonNegativeInteger*
decimalSymbol	**erforderlich**		**period** **comma**
formatType	**erforderlich**		**standard** **scientific** **currency**
groupingSymbol	**erforderlich**		**none** **period** **comma** **space**
name	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0"
maxOccurs="unbounded">
    <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="formatType" type="NUMBER-FORMAT-TYPE" use="required">
        <xs:enumeration value="standard"></xs:enumeration>
        <xs:enumeration value="scientific"></xs:enumeration>
        <xs:enumeration value="currency"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="decimalPlaces" type="xs:nonNegativeInteger" use="required"></xs:attribute>
    <xs:attribute name="decimalSymbol" type="DECIMAL-SYMBOL" use="required">
        <xs:enumeration value="period"></xs:enumeration>
        <xs:enumeration value="comma"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="groupingSymbol" type="NUMBER-GROUPING-SYMBOL" use="required">
```

```
        <xs:enumeration value="none"></xs:enumeration>
        <xs:enumeration value="period"></xs:enumeration>
        <xs:enumeration value="comma"></xs:enumeration>
        <xs:enumeration value="space"></xs:enumeration>
    </xs:attribute>
</xs:element>
```

Übergeordnet Elemente

FieldFormats Element

FieldGroups Element

Defines the field groups. Field groups are used to associate related fields. For example, a survey question that asks a respondent to select which locations they have visited from a set of options will be represented as a set of flag fields. A field group may be used to identify which fields are associated with that survey question.

Tabelle A-35
Attribute für FieldGroups

Attribut	Verwenden	Beschreibung	Gültige Werte
count	optional		*nonNegativeInteger*

XML-Darstellung

```
<xs:element name="FieldGroups" type="FIELD-GROUPS" minOccurs="0">
    <xs:sequence>
        <xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0"
        maxOccurs="unbounded">
            <xs:sequence>
                <xs:element name="FieldName"></xs:element>
            </xs:sequence>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

DataModel Element

Untergeordnet Elemente

FieldGroup Element

FieldGroup Element

Defines a field group. A field group consists of a list of field names and information about the field group such as the group name and optional label, type of group and for multi-dichotomy groups, the counted value i.e. the value which represents “true”.

Tabelle A-36
Attribute für FieldGroup

Attribut	Verwenden	Beschreibung	Gültige Werte
count	optional		*nonNegativeInteger*
countedValue	optional		*string*
displayLabel	optional		*string*
groupType	**erforderlich**		**fieldGroup** **multiCategorySet** **multiDichotomySet**
name	**erforderlich**		

XML-Darstellung

```
<xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0"
maxOccurs="unbounded">
    <xs:sequence>
        <xs:element name="FieldName"></xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="FIELD-GROUP-NAME" use="required"></xs:attribute>
    <xs:attribute name="displayLabel" type="xs:string"></xs:attribute>
    <xs:attribute name="groupType" type="FIELD-GROUP-TYPE" use="required">
        <xs:enumeration value="fieldGroup"></xs:enumeration>
        <xs:enumeration value="multiCategorySet"></xs:enumeration>
        <xs:enumeration value="multiDichotomySet"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="countedValue" type="xs:string"></xs:attribute>
    <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

FieldGroups Element

Untergeordnet Elemente

FieldName Element

FieldName Element

Tabelle A-37
Attribute für FieldName

Attribut	Verwenden	Beschreibung	Gültige Werte
name	**erforderlich**		

XML-Darstellung

```
<xs:element name="FieldName">
    <xs:attribute name="name" type="FIELD-NAME" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

FieldGroup Element

Fields Element

Tabelle A-38
Attribute für Fields

Attribut	Verwenden	Beschreibung	Gültige Werte
count	optional		*nonNegativeInteger*

XML-Darstellung

```
<xs:element name="Fields" type="FIELDS">
   <xs:sequence>
      <xs:element name="Field" type="FIELD" minOccurs="0" maxOccurs="unbounded">
         <xs:group ref="FIELD-CONTENT">
            <xs:sequence>
               <xs:element ref="DisplayLabel"></xs:element>
               <xs:choice minOccurs="0">
                  <xs:element ref="Range"></xs:element>
                  <xs:element ref="Values"></xs:element>
               </xs:choice>
               <xs:element ref="MissingValues"></xs:element>
            </xs:sequence>
         </xs:group>
      </xs:element>
   </xs:sequence>
   <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

DataModel Element

Untergeordnet Elemente

Field Element

Field Element

Tabelle A-39
Attribute für Field

Attribut	Verwenden	Beschreibung	Gültige Werte
direction	optional		**in** **out** **both** **none** **partition**
displayLabel	optional		*string*
name	**erforderlich**		*string*

Attribut	Verwenden	Beschreibung	Gültige Werte
storage	optional		**unknown** **integer** **real** **string** **date** **time** **timestamp**
type	optional		**auto** **range** **discrete** **set** **orderedSet** **flag** **typeless**

XML-Darstellung

```
<xs:element name="Field" type="FIELD" minOccurs="0" maxOccurs="unbounded">
    <xs:group ref="FIELD-CONTENT">
        <xs:sequence>
            <xs:element ref="DisplayLabel"></xs:element>
            <xs:choice minOccurs="0">
                <xs:element ref="Range"></xs:element>
                <xs:element ref="Values"></xs:element>
            </xs:choice>
            <xs:element ref="MissingValues"></xs:element>
        </xs:sequence>
    </xs:group>
    <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="type" type="FIELD-TYPE" default="auto">
        <xs:enumeration value="auto"></xs:enumeration>
        <xs:enumeration value="range"></xs:enumeration>
        <xs:enumeration value="discrete"></xs:enumeration>
        <xs:enumeration value="set"></xs:enumeration>
        <xs:enumeration value="orderedSet"></xs:enumeration>
        <xs:enumeration value="flag"></xs:enumeration>
        <xs:enumeration value="typeless"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="storage" type="FIELD-STORAGE" default="unknown">
        <xs:enumeration value="unknown"></xs:enumeration>
        <xs:enumeration value="integer"></xs:enumeration>
        <xs:enumeration value="real"></xs:enumeration>
        <xs:enumeration value="string"></xs:enumeration>
        <xs:enumeration value="date"></xs:enumeration>
        <xs:enumeration value="time"></xs:enumeration>
        <xs:enumeration value="timestamp"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="direction" type="FIELD-DIRECTION" default="in">
        <xs:enumeration value="in"></xs:enumeration>
        <xs:enumeration value="out"></xs:enumeration>
        <xs:enumeration value="both"></xs:enumeration>
        <xs:enumeration value="none"></xs:enumeration>
        <xs:enumeration value="partition"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="displayLabel" type="xs:string"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Fields Element

Untergeordnet Elemente

DisplayLabel Element, MissingValues Element, Range Element, Values Element

DBConnectionChooserControl Element

Tabelle A-40
Attribute für DBConnectionChooserControl

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="DBConnectionChooserControl">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
   <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
   <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
   <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

DBTableChooserControl Element

Tabelle A-41
Attribute für DBTableChooserControl

Attribut	Verwenden	Beschreibung	Gültige Werte
connectionProperty	**erforderlich**		*string*
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="DBTableChooserControl">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
   <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
   <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
   <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
```

```
    <xs:attribute name="connectionProperty" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

DefaultValue Element

XML-Darstellung

```
<xs:element name="DefaultValue">
    <xs:choice>
        <xs:element name="ServerTempFile"></xs:element>
        <xs:element name="ServerTempDir"></xs:element>
        <xs:element name="Identifier"></xs:element>
    </xs:choice>
</xs:element>
```

Übergeordnet Elemente

Property Element, PropertyType Element

Untergeordnet Elemente

Identifier Element, ServerTempDir Element, ServerTempFile Element

ServerTempFile Element

Tabelle A-42
Attribute für ServerTempFile

Attribut	Verwenden	Beschreibung	Gültige Werte
basename	**erforderlich**		

XML-Darstellung

```
<xs:element name="ServerTempFile">
```

```
    <xs:attribute name="basename" type="EVALUATED-STRING" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

DefaultValue Element

ServerTempDir Element

Tabelle A-43
Attribute für ServerTempDir

Attribut	Verwenden	Beschreibung	Gültige Werte
basename	**erforderlich**		

XML-Darstellung

```
<xs:element name="ServerTempDir">
    <xs:attribute name="basename" type="EVALUATED-STRING" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

DefaultValue Element

Identifier Element

Tabelle A-44
Attribute für Identifier

Attribut	Verwenden	Beschreibung	Gültige Werte
basename	**erforderlich**		

XML-Darstellung

```
<xs:element name="Identifier">
    <xs:attribute name="basename" type="EVALUATED-STRING" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

DefaultValue Element

DelimitedDataFormat Element

Tabelle A-45
Attribute für DelimitedDataFormat

Attribut	Verwenden	Beschreibung	Gültige Werte
delimeter	optional		**tab** **comma** **semicolon** **colon** **verticalBar** **other**
eol	optional		**cr** **crlf** **lf** **other**
includeFieldNames	optional		*boolean*
otherDelimiter	optional		*string*
otherEol	optional		*string*
quoteStrings	optional		*boolean*
stringQuote	optional		*string*

XML-Darstellung

```
<xs:element name="DelimitedDataFormat">
   <xs:attribute name="delimeter" use="optional" default="tab">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="tab"></xs:enumeration>
            <xs:enumeration value="comma"></xs:enumeration>
            <xs:enumeration value="semicolon"></xs:enumeration>
            <xs:enumeration value="colon"></xs:enumeration>
            <xs:enumeration value="verticalBar"></xs:enumeration>
            <xs:enumeration value="other"></xs:enumeration>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="otherDelimiter" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="eol" use="optional" default="cr">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="cr"></xs:enumeration>
            <xs:enumeration value="crlf"></xs:enumeration>
            <xs:enumeration value="lf"></xs:enumeration>
            <xs:enumeration value="other"></xs:enumeration>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="otherEol" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="includeFieldNames" type="xs:boolean" use="optional" default="true"></xs:attribute>
   <xs:attribute name="quoteStrings" type="xs:boolean" use="optional" default="false"></xs:attribute>
   <xs:attribute name="stringQuote" type="xs:string" use="optional" default=""""></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

DataFile Element, DataFormat Element

DisplayLabel Element

A display label for a field or value in a specified language. The displayLabel attribute can be used where there is no label for a particular language.

Tabelle A-46
Attribute für DisplayLabel

Attribut	Verwenden	Beschreibung	Gültige Werte
lang	optional		*NMTOKEN*
value	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
    <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="lang" type="xs:NMTOKEN" default="en"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Field Element

DocumentBuilder Element

XML-Darstellung

```
<xs:element name="DocumentBuilder">
    <xs:sequence>
        <xs:element name="DocumentGeneration"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

Node Element

Untergeordnet Elemente

DocumentGeneration Element

DocumentGeneration Element

Tabelle A-47
Attribute für DocumentGeneration

Attribut	Verwenden	Beschreibung	Gültige Werte
controlsId	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="DocumentGeneration">
    <xs:attribute name="controlsId" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

DocumentBuilder Element

Verwandt Elemente

ModelingFields Element, ModelGeneration Element, ModelEvaluation Element

DocumentOutput Element

Tabelle A-48
Attribute für DocumentOutput

Attribut	Verwenden	Beschreibung	Gültige Werte
deprecatedScriptNames	optional		*string*
id	**erforderlich**		*string*
scriptName	optional		*string*

XML-Darstellung

```
<xs:element name="DocumentOutput">
    <xs:sequence maxOccurs="unbounded">
        <xs:choice maxOccurs="unbounded">
            <xs:element ref="Properties"></xs:element>
            <xs:element name="Containers" minOccurs="0">
                <xs:sequence maxOccurs="unbounded">
                    <xs:element ref="Container"></xs:element>
                </xs:sequence>
            </xs:element>
            <xs:element ref="UserInterface"></xs:element>
            <xs:element ref="Constructors" minOccurs="0"></xs:element>
            <xs:element ref="ModelProvider" minOccurs="0"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="scriptName" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Extension Element

Untergeordnet Elemente

Constructors Element, Containers Element, ModelProvider Element, Properties Element, UserInterface Element

Verwandt Elemente

InteractiveDocumentBuilder Element, InteractiveModelBuilder Element, ModelOutput Element, Node Element

Containers Element

XML-Darstellung

```
<xs:element name="Containers" minOccurs="0">
    <xs:sequence maxOccurs="unbounded">
        <xs:element ref="Container"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

DocumentOutput Element

Untergeordnet Elemente

Container Element

DocumentType Element

Defines a new document type

Tabelle A-49
Attribute für DocumentType

Attribut	Verwenden	Beschreibung	Gültige Werte
format	**erforderlich**		**utf8** **binary**
id	**erforderlich**		*string*
type	optional		**unknown** **rowSet** **report** **graph**

XML-Darstellung

```
<xs:element name="DocumentType">
    <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="format" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="utf8"></xs:enumeration>
                <xs:enumeration value="binary"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="type" type="DOCUMENT-TYPE" use="optional">
        <xs:enumeration value="unknown"></xs:enumeration>
        <xs:enumeration value="rowSet"></xs:enumeration>
```

```
        <xs:enumeration value="report"></xs:enumeration>
        <xs:enumeration value="graph"></xs:enumeration>
    </xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ContainerTypes Element

Verwandt Elemente

ModelType Element

Enabled Element

XML-Darstellung

```
<xs:element name="Enabled">
    <xs:sequence>
        <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
            <xs:choice>
                <xs:element ref="Condition"></xs:element>
                <xs:element ref="And"></xs:element>
                <xs:element ref="Or"></xs:element>
                <xs:element ref="Not"></xs:element>
            </xs:choice>
        </xs:group>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

ActionButton Element, CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, ComboBoxControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, ExtensionObjectPanel Element, ModelViewerPanel Element, MultiFieldChooserControl Element, MultiItemChooserControl Element, PasswordBoxControl Element, PropertiesPanel Element, PropertiesSubPanel Element, PropertyControl Element, RadioButtonGroupControl Element, SelectorPanel Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SingleItemChooserControl Element, SpinnerControl Element, StaticText Element, SystemControls Element, TabbedPanel Element, TableControl Element, TextAreaControl Element, TextBoxControl Element, TextBrowserPanel Element, ItemChooserControl Geben Sie

Untergeordnet Elemente

And Element, Condition Element, Not Element, Or Element

Enumeration Element

XML-Darstellung

```
<xs:element name="Enumeration">
   <xs:sequence>
      <xs:element name="Enum" maxOccurs="unbounded"></xs:element>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

PropertyType Element

Untergeordnet Elemente

Enum Element

Enum Element

Tabelle A-50
Attribute für Enum

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
imagePath	optional		*string*
imagePathKey	optional		*string*
label	**erforderlich**		*string*
labelKey	optional		*string*
value	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="Enum" maxOccurs="unbounded">
   <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="imagePath" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="imagePathKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Enumeration Element

ErrorDetail Element

Suuplementary information about an error or other condition.

XML-Darstellung

```
<xs:element name="ErrorDetail" type="ERROR-DETAIL">
   <xs:sequence>
      <xs:element name="Diagnostic" type="DIAGNOSTIC" minOccurs="0" maxOccurs="unbounded">
         <xs:sequence>
            <xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0"></xs:element>
            <xs:element  name="Parameter"  type="xs:string"  minOccurs="0"
            maxOccurs="unbounded"></xs:element>
         </xs:sequence>
      </xs:element>
   </xs:sequence>
</xs:element>
```

Untergeordnet Elemente

Diagnostic Element

Diagnostic Element

Tabelle A-51
Attribute für Diagnostic

Attribut	Verwenden	Beschreibung	Gültige Werte
code	**erforderlich**		*integer*
severity	optional		**unknown** **information** **warning** **error** **fatal**
source	optional		*string*
subCode	optional		*integer*

XML-Darstellung

```
<xs:element name="Diagnostic" type="DIAGNOSTIC" minOccurs="0" maxOccurs="unbounded">
   <xs:sequence>
      <xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0"></xs:element>
      <xs:element  name="Parameter"  type="xs:string"  minOccurs="0"
      maxOccurs="unbounded"></xs:element>
   </xs:sequence>
   <xs:attribute name="code" type="xs:integer" use="required"></xs:attribute>
   <xs:attribute name="subCode" type="xs:integer" default="0"></xs:attribute>
   <xs:attribute name="severity" type="DIAGNOSTIC-SEVERITY" default="error">
      <xs:enumeration value="unknown"></xs:enumeration>
      <xs:enumeration value="information"></xs:enumeration>
      <xs:enumeration value="warning"></xs:enumeration>
      <xs:enumeration value="error"></xs:enumeration>
      <xs:enumeration value="fatal"></xs:enumeration>
   </xs:attribute>
   <xs:attribute name="source" type="xs:string"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ErrorDetail Element

Untergeordnet Elemente

Message Element, Parameter Element

Message Element

Tabelle A-52
Attribute für Message

Attribut	Verwenden	Beschreibung	Gültige Werte
lang	optional		*NMTOKEN*

XML-Darstellung

```
<xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
    <xs:attribute name="lang" type="xs:NMTOKEN"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Diagnostic Element

Parameter Element

XML-Darstellung

```
<xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"></xs:element>
```

Übergeordnet Elemente

Diagnostic Element

Executable Element

XML-Darstellung

```
<xs:element name="Executable">
    <xs:sequence>
        <xs:element ref="Run" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

Execution Element

Untergeordnet Elemente

Run Element

Execution Element

XML-Darstellung

```
<xs:element name="Execution">
   <xs:sequence>
      <xs:element ref="Properties" minOccurs="0"></xs:element>
      <xs:element ref="InputFiles"></xs:element>
      <xs:element ref="OutputFiles"></xs:element>
      <xs:choice>
         <xs:element ref="Executable"></xs:element>
         <xs:element ref="Module"></xs:element>
      </xs:choice>
      <xs:element ref="Constructors" minOccurs="0"></xs:element>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

Node Element

Untergeordnet Elemente

Constructors Element, Executable Element, InputFiles Element, Module Element, OutputFiles Element, Properties Element

Extension Element

Defines the top-level extension container.

Tabelle A-53
Attribute für Extension

Attribut	Verwenden	Beschreibung	Gültige Werte
debug	optional		*boolean*
version	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="Extension">
   <xs:sequence>
      <xs:element ref="ExtensionDetails"></xs:element>
      <xs:element ref="Resources"></xs:element>
      <xs:element ref="License" minOccurs="0"></xs:element>
      <xs:element ref="CommonObjects"></xs:element>
      <xs:element ref="UserInterface" minOccurs="0"></xs:element>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:choice>
            <xs:element ref="Node"></xs:element>
            <xs:element ref="ModelOutput"></xs:element>
            <xs:element ref="DocumentOutput"></xs:element>
            <xs:element ref="InteractiveModelBuilder"></xs:element>
            <xs:element ref="InteractiveDocumentBuilder"></xs:element>
         </xs:choice>
      </xs:sequence>
```

```
    </xs:sequence>
    <xs:attribute name="version" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="debug" type="xs:boolean" use="optional" default="false"></xs:attribute>
</xs:element>
```

Untergeordnet Elemente

CommonObjects Element, DocumentOutput Element, ExtensionDetails Element, InteractiveDocumentBuilder Element, InteractiveModelBuilder Element, License Element, ModelOutput Element, Node Element, Resources Element, UserInterface Element

ExtensionDetails Element

Defines information about the extension such as the extension id, the extension provider and version information.

Tabelle A-54
Attribute für ExtensionDetails

Attribut	Verwenden	Beschreibung	Gültige Werte
copyright	optional		*string*
description	optional		*string*
id	**erforderlich**		*string*
label	**erforderlich**		*string*
provider	optional		*string*
providerTag	**erforderlich**		*string*
version	optional		*string*

XML-Darstellung

```
<xs:element name="ExtensionDetails">
    <xs:attribute name="providerTag" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="version" type="xs:string"></xs:attribute>
    <xs:attribute name="provider" type="xs:string" use="optional" default="(unknown)"></xs:attribute>
    <xs:attribute name="copyright" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Extension Element

ExtensionObjectPanel Element

Tabelle A-55
Attribute für ExtensionObjectPanel

Attribut	Verwenden	Beschreibung	Gültige Werte
id	optional		*string*
panelClass	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="ExtensionObjectPanel">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="panelClass" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="id" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element, Tab Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

ActionButton Element, ComboBoxControl Element, ModelViewerPanel Element, SelectorPanel Element, StaticText Element, SystemControls Element, TabbedPanel Element, TextBrowserPanel Element

Field Element

Tabelle A-56
Attribute für Field

Attribut	Verwenden	Beschreibung	Gültige Werte
direction	optional		**in** **out** **both** **none** **partition**
label	optional		*string*
name	**erforderlich**		
storage	optional		**unknown** **integer** **real** **string** **date** **time** **timestamp**
type	optional		**auto** **range** **discrete** **set** **orderedSet** **flag** **typeless**

XML-Darstellung

```
<xs:element name="Field" type="FIELD-DECLARATION">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Range" minOccurs="0"></xs:element>
            <xs:element ref="Values" minOccurs="0"></xs:element>
            <xs:element ref="NumericInfo" minOccurs="0"></xs:element>
            <xs:element name="MissingValues" minOccurs="0">
                <xs:sequence>
                    <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"></xs:element>
                    <xs:element ref="Range" minOccurs="0"></xs:element>
                </xs:sequence>
            </xs:element>
            <xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="name" type="FIELD-NAME" use="required"></xs:attribute>
    <xs:attribute name="storage" type="FIELD-STORAGE">
        <xs:enumeration value="unknown"></xs:enumeration>
        <xs:enumeration value="integer"></xs:enumeration>
        <xs:enumeration value="real"></xs:enumeration>
        <xs:enumeration value="string"></xs:enumeration>
        <xs:enumeration value="date"></xs:enumeration>
        <xs:enumeration value="time"></xs:enumeration>
        <xs:enumeration value="timestamp"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="type" type="FIELD-TYPE">
        <xs:enumeration value="auto"></xs:enumeration>
        <xs:enumeration value="range"></xs:enumeration>
        <xs:enumeration value="discrete"></xs:enumeration>
        <xs:enumeration value="set"></xs:enumeration>
        <xs:enumeration value="orderedSet"></xs:enumeration>
        <xs:enumeration value="flag"></xs:enumeration>
        <xs:enumeration value="typeless"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="direction" type="FIELD-DIRECTION">
        <xs:enumeration value="in"></xs:enumeration>
        <xs:enumeration value="out"></xs:enumeration>
        <xs:enumeration value="both"></xs:enumeration>
        <xs:enumeration value="none"></xs:enumeration>
        <xs:enumeration value="partition"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="label" type="xs:string"></xs:attribute>
</xs:element>
```

Untergeordnet Elemente

MissingValues Element, ModelField Element, NumericInfo Element, Range Element, Values Element

MissingValues Element

Tabelle A-57
Attribute für MissingValues

Attribut	Verwenden	Beschreibung	Gültige Werte
treatNullAsMissing	optional		*boolean*
treatWhitespaceAsMissing	optional		*boolean*

XML-Darstellung

```
<xs:element name="MissingValues" minOccurs="0">
    <xs:sequence>
        <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"></xs:element>
        <xs:element ref="Range" minOccurs="0"></xs:element>
    </xs:sequence>
    <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" use="optional"
    default="true"></xs:attribute>
    <xs:attribute name="treatNullAsMissing" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Field Element

Untergeordnet Elemente

Range Element, Values Element

ModelField Element

Tabelle A-58
Attribute für ModelField

Attribut	Verwenden	Beschreibung	Gültige Werte
group	optional		
role	**erforderlich**		**unknown** **predictedValue** **predictedDisplayValue** **probability** **residual** **standardError** **entityId** **entityAffinity** **upperConfidenceLimit** **lowerConfidenceLimit** **propensity** **value** **supplementary**
tag	optional		*string*
targetField	optional		*string*
value	optional		*string*

XML-Darstellung

```
<xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
    <xs:attribute name="role" type="MODEL-FIELD-ROLE" use="required">
        <xs:enumeration value="unknown"></xs:enumeration>
        <xs:enumeration value="predictedValue"></xs:enumeration>
        <xs:enumeration value="predictedDisplayValue"></xs:enumeration>
        <xs:enumeration value="probability"></xs:enumeration>
        <xs:enumeration value="residual"></xs:enumeration>
        <xs:enumeration value="standardError"></xs:enumeration>
        <xs:enumeration value="entityId"></xs:enumeration>
        <xs:enumeration value="entityAffinity"></xs:enumeration>
        <xs:enumeration value="upperConfidenceLimit"></xs:enumeration>
        <xs:enumeration value="lowerConfidenceLimit"></xs:enumeration>
        <xs:enumeration value="propensity"></xs:enumeration>
        <xs:enumeration value="value"></xs:enumeration>
        <xs:enumeration value="supplementary"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="targetField" type="xs:string"></xs:attribute>
    <xs:attribute name="value" type="xs:string"></xs:attribute>
    <xs:attribute name="group" type="MODEL-FIELD-GROUP"></xs:attribute>
    <xs:attribute name="tag" type="xs:string"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Field Element

FieldFormats Element

Defines the default field formats. Field formats are used when displaying values in output such as the general format (standard number, scientific or currency formats), number of decimal places to display, decimal separator etc. Currently field formats are only used for numeric fields although this may change in future versions.

Tabelle A-59
Attribute für FieldFormats

Attribut	Verwenden	Beschreibung	Gültige Werte
count	optional		*nonNegativeInteger*
defaultNumberFormat	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="FieldFormats" type="FIELD-FORMATS">
    <xs:sequence>
        <xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0"
        maxOccurs="unbounded"></xs:element>
    </xs:sequence>
    <xs:attribute name="defaultNumberFormat" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>
```

Untergeordnet Elemente

NumberFormat Element

NumberFormat Element

Defines format information for a numeric field.

Tabelle A-60
Attribute für NumberFormat

Attribut	Verwenden	Beschreibung	Gültige Werte
decimalPlaces	**erforderlich**		*nonNegativeInteger*
decimalSymbol	**erforderlich**		**period** **comma**
formatType	**erforderlich**		**standard** **scientific** **currency**
groupingSymbol	**erforderlich**		**none** **period** **comma** **space**
name	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0"
maxOccurs="unbounded">
   <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="formatType" type="NUMBER-FORMAT-TYPE" use="required">
      <xs:enumeration value="standard"></xs:enumeration>
      <xs:enumeration value="scientific"></xs:enumeration>
      <xs:enumeration value="currency"></xs:enumeration>
   </xs:attribute>
   <xs:attribute name="decimalPlaces" type="xs:nonNegativeInteger" use="required"></xs:attribute>
   <xs:attribute name="decimalSymbol" type="DECIMAL-SYMBOL" use="required">
      <xs:enumeration value="period"></xs:enumeration>
      <xs:enumeration value="comma"></xs:enumeration>
   </xs:attribute>
   <xs:attribute name="groupingSymbol" type="NUMBER-GROUPING-SYMBOL" use="required">
      <xs:enumeration value="none"></xs:enumeration>
      <xs:enumeration value="period"></xs:enumeration>
      <xs:enumeration value="comma"></xs:enumeration>
      <xs:enumeration value="space"></xs:enumeration>
   </xs:attribute>
</xs:element>
```

Übergeordnet Elemente

FieldFormats Element

FieldGroup Element

Defines a field group. A field group consists of a list of field names and information about the field group such as the group name and optional label, type of group and for multi-dichotomy groups, the counted value i.e. the value which represents “true”.

Tabelle A-61
Attribute für FieldGroup

Attribut	Verwenden	Beschreibung	Gültige Werte
count	optional		*nonNegativeInteger*
countedValue	optional		*string*
displayLabel	optional		*string*
groupType	**erforderlich**		**fieldGroup** **multiCategorySet** **multiDichotomySet**
name	**erforderlich**		

XML-Darstellung

```
<xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION">
    <xs:sequence>
        <xs:element name="FieldName"></xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="FIELD-GROUP-NAME" use="required"></xs:attribute>
    <xs:attribute name="displayLabel" type="xs:string"></xs:attribute>
    <xs:attribute name="groupType" type="FIELD-GROUP-TYPE" use="required">
        <xs:enumeration value="fieldGroup"></xs:enumeration>
        <xs:enumeration value="multiCategorySet"></xs:enumeration>
        <xs:enumeration value="multiDichotomySet"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="countedValue" type="xs:string"></xs:attribute>
    <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>
```

Untergeordnet Elemente

FieldName Element

FieldName Element

Tabelle A-62
Attribute für FieldName

Attribut	Verwenden	Beschreibung	Gültige Werte
name	**erforderlich**		

XML-Darstellung

```
<xs:element name="FieldName">
    <xs:attribute name="name" type="FIELD-NAME" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

FieldGroup Element

FieldGroups Element

Defines the field groups. Field groups are used to associate related fields. For example, a survey question that asks a respondent to select which locations they have visited from a set of options will be represented as a set of flag fields. A field group may be used to identify which fields are associated with that survey question.

Tabelle A-63
Attribute für FieldGroups

Attribut	Verwenden	Beschreibung	Gültige Werte
count	optional		*nonNegativeInteger*

XML-Darstellung

```
<xs:element name="FieldGroups" type="FIELD-GROUPS">
   <xs:sequence>
      <xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0"
      maxOccurs="unbounded">
         <xs:sequence>
            <xs:element name="FieldName"></xs:element>
         </xs:sequence>
      </xs:element>
   </xs:sequence>
   <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>
```

Untergeordnet Elemente

FieldGroup Element

FieldGroup Element

Defines a field group. A field group consists of a list of field names and information about the field group such as the group name and optional label, type of group and for multi-dichotomy groups, the counted value i.e. the value which represents “true”.

Tabelle A-64
Attribute für FieldGroup

Attribut	Verwenden	Beschreibung	Gültige Werte
count	optional		*nonNegativeInteger*
countedValue	optional		*string*
displayLabel	optional		*string*
groupType	**erforderlich**		**fieldGroup** **multiCategorySet** **multiDichotomySet**
name	**erforderlich**		

XML-Darstellung

```
<xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0"
maxOccurs="unbounded">
    <xs:sequence>
        <xs:element name="FieldName"></xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="FIELD-GROUP-NAME" use="required"></xs:attribute>
    <xs:attribute name="displayLabel" type="xs:string"></xs:attribute>
    <xs:attribute name="groupType" type="FIELD-GROUP-TYPE" use="required">
        <xs:enumeration value="fieldGroup"></xs:enumeration>
        <xs:enumeration value="multiCategorySet"></xs:enumeration>
        <xs:enumeration value="multiDichotomySet"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="countedValue" type="xs:string"></xs:attribute>
    <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

FieldGroups Element

Untergeordnet Elemente

FieldName Element

FieldName Element

Tabelle A-65
Attribute für FieldName

Attribut	Verwenden	Beschreibung	Gültige Werte
name	**erforderlich**		

XML-Darstellung

```
<xs:element name="FieldName">
    <xs:attribute name="name" type="FIELD-NAME" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

FieldGroup Element

FileFormatType Element

Tabelle A-66
Attribute für FileFormatType

Attribut	Verwenden	Beschreibung	Gültige Werte
name	optional		

XML-Darstellung

```
<xs:element name="FileFormatType">
```

```
    <xs:sequence>
        <xs:group ref="FILE-FORMAT">
            <xs:choice>
                <xs:element ref="UTF8Format"></xs:element>
                <xs:element ref="BinaryFormat"></xs:element>
                <xs:element ref="DataFormat"></xs:element>
            </xs:choice>
        </xs:group>
    </xs:sequence>
    <xs:attribute name="name" type="EVALUATED-STRING" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

FileFormatTypes Element

Untergeordnet Elemente

BinaryFormat Element, DataFormat Element, UTF8Format Element

FileFormatTypes Element

XML-Darstellung

```
<xs:element name="FileFormatTypes">
    <xs:sequence>
        <xs:element ref="FileFormatType" minOccurs="0" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

CommonObjects Element

Untergeordnet Elemente

FileFormatType Element

ForEach Element

Tabelle A-67
Attribute für ForEach

Attribut	Verwenden	Beschreibung	Gültige Werte
container	optional		*string*
from	optional		*string*
inFields	optional		*string*
inFieldValues	optional		*string*
inProperty	optional		*string*
step	optional		*string*
to	optional		*string*
var	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="ForEach">
    <xs:sequence maxOccurs="unbounded">
        <xs:group ref="DATA-MODEL-EXPRESSION">
            <xs:choice>
                <xs:element ref="ForEach"></xs:element>
                <xs:element ref="AddField"></xs:element>
                <xs:element ref="ChangeField"></xs:element>
                <xs:element ref="RemoveField"></xs:element>
            </xs:choice>
        </xs:group>
    </xs:sequence>
    <xs:attribute name="var" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="inProperty" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="inFields" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="inFieldValues" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="from" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="to" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="step" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="container" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ModelFields Element

Untergeordnet Elemente

AddField Element, ChangeField Element, ForEach Element, RemoveField Element

Icon Element

Tabelle A-68
Attribute für Icon

Attribut	Verwenden	Beschreibung	Gültige Werte
imagePath	**erforderlich**		*string*
resourceID	optional		*string*
type	**erforderlich**		**standardNode** **smallNode** **standardWindow**

XML-Darstellung

```
<xs:element name="Icon">
    <xs:attribute name="type" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="standardNode"></xs:enumeration>
                <xs:enumeration value="smallNode"></xs:enumeration>
                <xs:enumeration value="standardWindow"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
```

```
    <xs:attribute name="imagePath" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="resourceID" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Icons Element, Palette Element

Icons Element

XML-Darstellung

```
<xs:element name="Icons">
    <xs:sequence>
        <xs:element ref="Icon" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

UserInterface Element

Untergeordnet Elemente

Icon Element

InputFiles Element

XML-Darstellung

```
<xs:element name="InputFiles">
    <xs:group ref="RUNTIME-FILES">
        <xs:sequence>
            <xs:element ref="DataFile"></xs:element>
            <xs:element ref="ContainerFile" minOccurs="0" maxOccurs="unbounded"></xs:element>
        </xs:sequence>
    </xs:group>
</xs:element>
```

Übergeordnet Elemente

Execution Element, Module Element

Untergeordnet Elemente

ContainerFile Element, DataFile Element

InteractiveDocumentBuilder Element

Tabelle A-69
Attribute für InteractiveDocumentBuilder

Attribut	Verwenden	Beschreibung	Gültige Werte
deprecatedScriptNames	optional		*string*
id	**erforderlich**		*string*
scriptName	optional		*string*

XML-Darstellung

```
<xs:element name="InteractiveDocumentBuilder">
   <xs:sequence maxOccurs="unbounded">
      <xs:choice maxOccurs="unbounded">
         <xs:element ref="Properties"></xs:element>
         <xs:element name="Containers" minOccurs="0">
            <xs:sequence maxOccurs="unbounded">
               <xs:element ref="Container"></xs:element>
            </xs:sequence>
         </xs:element>
         <xs:element ref="UserInterface"></xs:element>
         <xs:element ref="Constructors" minOccurs="0"></xs:element>
         <xs:element ref="ModelProvider" minOccurs="0"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="scriptName" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Extension Element

Untergeordnet Elemente

Constructors Element, Containers Element, ModelProvider Element, Properties Element, UserInterface Element

Verwandt Elemente

DocumentOutput Element, InteractiveModelBuilder Element, ModelOutput Element, Node Element

Containers Element

XML-Darstellung

```
<xs:element name="Containers" minOccurs="0">
   <xs:sequence maxOccurs="unbounded">
      <xs:element ref="Container"></xs:element>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

InteractiveDocumentBuilder Element

Untergeordnet Elemente

Container Element

InteractiveModelBuilder Element

Tabelle A-70
Attribute für InteractiveModelBuilder

Attribut	Verwenden	Beschreibung	Gültige Werte
deprecatedScriptNames	optional		*string*
id	**erforderlich**		*string*
scriptName	optional		*string*

XML-Darstellung

```
<xs:element name="InteractiveModelBuilder">
   <xs:sequence maxOccurs="unbounded">
      <xs:choice maxOccurs="unbounded">
         <xs:element ref="Properties"></xs:element>
         <xs:element name="Containers" minOccurs="0">
            <xs:sequence maxOccurs="unbounded">
               <xs:element ref="Container"></xs:element>
            </xs:sequence>
         </xs:element>
         <xs:element ref="UserInterface"></xs:element>
         <xs:element ref="Constructors" minOccurs="0"></xs:element>
         <xs:element ref="ModelProvider" minOccurs="0"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="scriptName" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Extension Element

Untergeordnet Elemente

Constructors Element, Containers Element, ModelProvider Element, Properties Element, UserInterface Element

Verwandt Elemente

DocumentOutput Element, InteractiveDocumentBuilder Element, ModelOutput Element, Node Element

Containers Element

XML-Darstellung

```
<xs:element name="Containers" minOccurs="0">
    <xs:sequence maxOccurs="unbounded">
        <xs:element ref="Container"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

InteractiveModelBuilder Element

Untergeordnet Elemente

Container Element

Layout Element

Tabelle A-71
Attribute für Layout

Attribut	Verwenden	Beschreibung	Gültige Werte
anchor	optional		**north** **northeast** **east** **southeast** **south** **southwest** **west** **northwest** **center**
columnWeight	optional		*double*
fill	optional		**horizontal** **vertical** **both** **none**
gridColumn	optional		*nonNegativeInteger*
gridHeight	optional		*nonNegativeInteger*
gridRow	optional		*nonNegativeInteger*
gridWidth	optional		*nonNegativeInteger*
leftIndent	optional		*nonNegativeInteger*
rowIncrement	optional		*nonNegativeInteger*
rowWeight	optional		*double*

XML-Darstellung

```
<xs:element name="Layout">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="Cell"></xs:element>
    </xs:sequence>
    <xs:attribute name="gridRow" type="xs:nonNegativeInteger" use="optional"></xs:attribute>
```

```
    <xs:attribute name="gridColumn" type="xs:nonNegativeInteger" use="optional"></xs:attribute>
    <xs:attribute name="rowIncrement" type="xs:nonNegativeInteger" use="optional"></xs:attribute>
    <xs:attribute name="gridWidth" type="xs:nonNegativeInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="gridHeight" type="xs:nonNegativeInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="rowWeight" type="xs:double" use="optional"></xs:attribute>
    <xs:attribute name="columnWeight" type="xs:double" use="optional"></xs:attribute>
    <xs:attribute name="fill" type="UI-COMPONENT-FILL" use="optional" default="none">
        <xs:enumeration value="horizontal"></xs:enumeration>
        <xs:enumeration value="vertical"></xs:enumeration>
        <xs:enumeration value="both"></xs:enumeration>
        <xs:enumeration value="none"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="anchor" type="UI-COMPONENT-ANCHOR" use="optional" default="west">
        <xs:enumeration value="north"></xs:enumeration>
        <xs:enumeration value="northeast"></xs:enumeration>
        <xs:enumeration value="east"></xs:enumeration>
        <xs:enumeration value="southeast"></xs:enumeration>
        <xs:enumeration value="south"></xs:enumeration>
        <xs:enumeration value="southwest"></xs:enumeration>
        <xs:enumeration value="west"></xs:enumeration>
        <xs:enumeration value="northwest"></xs:enumeration>
        <xs:enumeration value="center"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="leftIndent" type="xs:nonNegativeInteger" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ActionButton Element, CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, ComboBoxControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, ExtensionObjectPanel Element, ModelViewerPanel Element, MultiFieldChooserControl Element, MultiItemChooserControl Element, PasswordBoxControl Element, PropertiesPanel Element, PropertiesSubPanel Element, PropertyControl Element, RadioButtonGroupControl Element, SelectorPanel Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SingleItemChooserControl Element, SpinnerControl Element, StaticText Element, SystemControls Element, TabbedPanel Element, TableControl Element, TextAreaControl Element, TextBoxControl Element, TextBrowserPanel Element, ItemChooserControl Geben Sie

Untergeordnet Elemente

Cell Element

Cell Element

Tabelle A-72
Attribute für Cell

Attribut	Verwenden	Beschreibung	Gültige Werte
column	**erforderlich**		*nonNegativeInteger*
row	**erforderlich**		*nonNegativeInteger*
width	**erforderlich**		*nonNegativeInteger*

XML-Darstellung

```
<xs:element name="Cell">
    <xs:attribute name="row" type="xs:nonNegativeInteger" use="required"></xs:attribute>
    <xs:attribute name="column" type="xs:nonNegativeInteger" use="required"></xs:attribute>
    <xs:attribute name="width" type="xs:nonNegativeInteger" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Layout Element

License Element

Reserved for system use.

XML-Darstellung

```
<xs:element name="License">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="OptionCode"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

Extension Element

Untergeordnet Elemente

OptionCode Element

ListValue Element

A sequence of values. All values must have the same content type but this is not checked.

XML-Darstellung

```
<xs:element name="ListValue" type="LIST-VALUE">
    <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
        <xs:choice>
            <xs:element ref="MapValue"></xs:element>
            <xs:element ref="StructuredValue"></xs:element>
            <xs:element ref="ListValue"></xs:element>
            <xs:element ref="Value"></xs:element>
            <xs:element ref="DatabaseConnectionValue"></xs:element>
        </xs:choice>
    </xs:group>
</xs:element>
```

Übergeordnet Elemente

Attribute Element, ListValue Element, Parameter Element, Attribute Element, ListValue Element

Untergeordnet Elemente

DatabaseConnectionValue Element, ListValue Element, MapValue Element, StructuredValue Element, Value Element

MapValue Element

A set of map entries, each consisting if a key and a value.

XML-Darstellung

```
<xs:element name="MapValue" type="MAP-VALUE">
   <xs:sequence>
      <xs:element name="MapEntry" type="MAP-ENTRY" maxOccurs="unbounded">
         <xs:sequence>
            <xs:element name="KeyValue" type="KEY-VALUE"></xs:element>
            <xs:element name="StructuredValue" type="STRUCTURED-VALUE">
               <xs:sequence>
                  <xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
                     <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
                        <xs:choice>
                           <xs:element ref="MapValue"></xs:element>
                           <xs:element ref="StructuredValue"></xs:element>
                           <xs:element ref="ListValue"></xs:element>
                           <xs:element ref="Value"></xs:element>
                           <xs:element ref="DatabaseConnectionValue"></xs:element>
                        </xs:choice>
                     </xs:group>
                     <xs:sequence>
                        <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
                           <xs:group ref="PARAMETER-CONTENT" minOccurs="0"
                           maxOccurs="unbounded">
                              <xs:choice>
                                 <xs:element ref="MapValue"></xs:element>
                                 <xs:element ref="StructuredValue"></xs:element>
                                 <xs:element ref="ListValue"></xs:element>
                                 <xs:element ref="Value"></xs:element>
                                 <xs:element ref="DatabaseConnectionValue"></xs:element>
                              </xs:choice>
                           </xs:group>
                        </xs:element>
                     </xs:sequence>
                  </xs:element>
               </xs:sequence>
            </xs:element>
         </xs:sequence>
      </xs:element>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

ListValue Element, Attribute Element, ListValue Element, Parameter Element, Attribute Element, ListValue Element

Untergeordnet Elemente

MapEntry Element

MapEntry Element

An entry in a keyed property map. Each entry consists of a key and an associated value.

XML-Darstellung

```
<xs:element name="MapEntry" type="MAP-ENTRY" maxOccurs="unbounded">
   <xs:sequence>
      <xs:element name="KeyValue" type="KEY-VALUE"></xs:element>
      <xs:element name="StructuredValue" type="STRUCTURED-VALUE">
         <xs:sequence>
            <xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
               <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
                  <xs:choice>
                     <xs:element ref="MapValue"></xs:element>
                     <xs:element ref="StructuredValue"></xs:element>
                     <xs:element ref="ListValue"></xs:element>
                     <xs:element ref="Value"></xs:element>
                     <xs:element ref="DatabaseConnectionValue"></xs:element>
                  </xs:choice>
               </xs:group>
               <xs:sequence>
                  <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
                     <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
                        <xs:choice>
                           <xs:element ref="MapValue"></xs:element>
                           <xs:element ref="StructuredValue"></xs:element>
                           <xs:element ref="ListValue"></xs:element>
                           <xs:element ref="Value"></xs:element>
                           <xs:element ref="DatabaseConnectionValue"></xs:element>
                        </xs:choice>
                     </xs:group>
                  </xs:element>
               </xs:sequence>
            </xs:element>
         </xs:sequence>
      </xs:element>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

MapValue Element

Untergeordnet Elemente

KeyValue Element, StructuredValue Element

KeyValue Element

The key value in a map entry.

Tabelle A-73
Attribute für KeyValue

Attribut	Verwenden	Beschreibung	Gültige Werte
value	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="KeyValue" type="KEY-VALUE">
   <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

MapEntry Element

StructuredValue Element

A sequence of named values (“attributes”).

XML-Darstellung

```
<xs:element name="StructuredValue" type="STRUCTURED-VALUE">
   <xs:sequence>
      <xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
         <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
            <xs:choice>
               <xs:element ref="MapValue"></xs:element>
               <xs:element ref="StructuredValue"></xs:element>
               <xs:element ref="ListValue"></xs:element>
               <xs:element ref="Value"></xs:element>
               <xs:element ref="DatabaseConnectionValue"></xs:element>
            </xs:choice>
         </xs:group>
         <xs:sequence>
            <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
               <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
                  <xs:choice>
                     <xs:element ref="MapValue"></xs:element>
                     <xs:element ref="StructuredValue"></xs:element>
                     <xs:element ref="ListValue"></xs:element>
                     <xs:element ref="Value"></xs:element>
                     <xs:element ref="DatabaseConnectionValue"></xs:element>
                  </xs:choice>
               </xs:group>
            </xs:element>
         </xs:sequence>
```

```
        </xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

MapEntry Element

Untergeordnet Elemente

Attribute Element

Attribute Element

Tabelle A-74
Attribute für Attribute

Attribut	Verwenden	Beschreibung	Gültige Werte
name	**erforderlich**		*string*
value	optional		*string*

XML-Darstellung

```
<xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
    <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
        <xs:choice>
            <xs:element ref="MapValue"></xs:element>
            <xs:element ref="StructuredValue"></xs:element>
            <xs:element ref="ListValue"></xs:element>
            <xs:element ref="Value"></xs:element>
            <xs:element ref="DatabaseConnectionValue"></xs:element>
        </xs:choice>
    </xs:group>
    <xs:sequence>
        <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
            <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
                <xs:choice>
                    <xs:element ref="MapValue"></xs:element>
                    <xs:element ref="StructuredValue"></xs:element>
                    <xs:element ref="ListValue"></xs:element>
                    <xs:element ref="Value"></xs:element>
                    <xs:element ref="DatabaseConnectionValue"></xs:element>
                </xs:choice>
            </xs:group>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="value" type="xs:string"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

StructuredValue Element

Untergeordnet Elemente

DatabaseConnectionValue Element, ListValue Element, ListValue Element, MapValue Element, StructuredValue Element, Value Element

ListValue Element

A sequence of values. All values must have the same content type but this is not checked.

XML-Darstellung

```
<xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
   <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
         <xs:element ref="MapValue"></xs:element>
         <xs:element ref="StructuredValue"></xs:element>
         <xs:element ref="ListValue"></xs:element>
         <xs:element ref="Value"></xs:element>
         <xs:element ref="DatabaseConnectionValue"></xs:element>
      </xs:choice>
   </xs:group>
</xs:element>
```

Übergeordnet Elemente

Attribute Element

Untergeordnet Elemente

DatabaseConnectionValue Element, ListValue Element, MapValue Element, StructuredValue Element, Value Element

Menu Element

Tabelle A-75
Attribute für Menu

Attribut	Verwenden	Beschreibung	Gültige Werte
id	**erforderlich**		*string*
label	**erforderlich**		*string*
labelKey	optional		*string*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
offset	optional		*nonNegativeInteger*
separatorAfter	optional		*boolean*
separatorBefore	optional		*boolean*
showIcon	optional		*boolean*

Attribut	Verwenden	Beschreibung	Gültige Werte
showLabel	optional		*boolean*
systemMenu	**erforderlich**		**file** **edit** **insert** **view** **tools** **window** **help** **generate** **file.project** **file.outputs** **file.models** **edit.stream** **edit.node** **edit.outputs** **edit.models** **edit.project** **tools.repository** **tools.options** **tools.streamProperties**

XML-Darstellung

```
<xs:element name="Menu">
    <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="systemMenu" type="STANDARD-MENU" use="required">
        <xs:enumeration value="file"></xs:enumeration>
        <xs:enumeration value="edit"></xs:enumeration>
        <xs:enumeration value="insert"></xs:enumeration>
        <xs:enumeration value="view"></xs:enumeration>
        <xs:enumeration value="tools"></xs:enumeration>
        <xs:enumeration value="window"></xs:enumeration>
        <xs:enumeration value="help"></xs:enumeration>
        <xs:enumeration value="generate"></xs:enumeration>
        <xs:enumeration value="file.project"></xs:enumeration>
        <xs:enumeration value="file.outputs"></xs:enumeration>
        <xs:enumeration value="file.models"></xs:enumeration>
        <xs:enumeration value="edit.stream"></xs:enumeration>
        <xs:enumeration value="edit.node"></xs:enumeration>
        <xs:enumeration value="edit.outputs"></xs:enumeration>
        <xs:enumeration value="edit.models"></xs:enumeration>
        <xs:enumeration value="edit.project"></xs:enumeration>
        <xs:enumeration value="tools.repository"></xs:enumeration>
        <xs:enumeration value="tools.options"></xs:enumeration>
        <xs:enumeration value="tools.streamProperties"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="separatorBefore" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="separatorAfter" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="offset" type="xs:nonNegativeInteger" use="optional" default="0"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Controls Element

MenuItem Element

Tabelle A-76
Attribute für MenuItem

Attribut	Verwenden	Beschreibung	Gültige Werte
action	**erforderlich**		*string*
customMenu	optional		*string*
offset	optional		*nonNegativeInteger*
separatorAfter	optional		*boolean*
separatorBefore	optional		*boolean*
showIcon	optional		*boolean*
showLabel	optional		*boolean*
systemMenu	optional		**file** **edit** **insert** **view** **tools** **window** **help** **generate** **file.project** **file.outputs** **file.models** **edit.stream** **edit.node** **edit.outputs** **edit.models** **edit.project** **tools.repository** **tools.options** **tools.streamProperties**

XML-Darstellung

```
<xs:element name="MenuItem">
   <xs:attribute name="action" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="systemMenu" type="STANDARD-MENU" use="optional">
      <xs:enumeration value="file"></xs:enumeration>
      <xs:enumeration value="edit"></xs:enumeration>
      <xs:enumeration value="insert"></xs:enumeration>
      <xs:enumeration value="view"></xs:enumeration>
      <xs:enumeration value="tools"></xs:enumeration>
      <xs:enumeration value="window"></xs:enumeration>
      <xs:enumeration value="help"></xs:enumeration>
      <xs:enumeration value="generate"></xs:enumeration>
      <xs:enumeration value="file.project"></xs:enumeration>
      <xs:enumeration value="file.outputs"></xs:enumeration>
      <xs:enumeration value="file.models"></xs:enumeration>
      <xs:enumeration value="edit.stream"></xs:enumeration>
      <xs:enumeration value="edit.node"></xs:enumeration>
      <xs:enumeration value="edit.outputs"></xs:enumeration>
```

```
        <xs:enumeration value="edit.models"></xs:enumeration>
        <xs:enumeration value="edit.project"></xs:enumeration>
        <xs:enumeration value="tools.repository"></xs:enumeration>
        <xs:enumeration value="tools.options"></xs:enumeration>
        <xs:enumeration value="tools.streamProperties"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="customMenu" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="separatorBefore" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="separatorAfter" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="offset" type="xs:nonNegativeInteger" use="optional" default="0"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Controls Element

MissingValues Element

Tabelle A-77
Attribute für MissingValues

Attribut	Verwenden	Beschreibung	Gültige Werte
treatNullAsMissing	optional		*boolean*
treatWhitespaceAsMissing	optional		*boolean*

XML-Darstellung

```
<xs:element name="MissingValues" type="MISSING-VALUES" minOccurs="0">
    <xs:sequence>
        <xs:element name="Range" type="RANGE"></xs:element>
        <xs:element name="Values" type="FIELD-VALUES">
            <xs:sequence>
                <xs:element name="Value" type="FIELD-VALUE" minOccurs="0" maxOccurs="unbounded">
                    <xs:sequence>
                        <xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0"
                        maxOccurs="unbounded"></xs:element>
                    </xs:sequence>
                </xs:element>
            </xs:sequence>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="treatNullAsMissing" type="xs:boolean" default="true"></xs:attribute>
    <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" default="false"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Field Element

Untergeordnet Elemente

Range Element, Values Element

Range Element

Tabelle A-78
Attribute für Range

Attribut	Verwenden	Beschreibung	Gültige Werte
maxValue	**erforderlich**		*string*
minValue	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="Range" type="RANGE">
   <xs:attribute name="minValue" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="maxValue" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

MissingValues Element

Values Element

Tabelle A-79
Attribute für Values

Attribut	Verwenden	Beschreibung	Gültige Werte
count	optional		*nonNegativeInteger*

XML-Darstellung

```
<xs:element name="Values" type="FIELD-VALUES">
   <xs:sequence>
      <xs:element name="Value" type="FIELD-VALUE" minOccurs="0" maxOccurs="unbounded">
         <xs:sequence>
            <xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0"
            maxOccurs="unbounded"></xs:element>
         </xs:sequence>
      </xs:element>
   </xs:sequence>
   <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

MissingValues Element

Untergeordnet Elemente

Value Element

Value Element

Tabelle A-80
Attribute für Value

Attribut	Verwenden	Beschreibung	Gültige Werte
code	**erforderlich**		*integer*
displayLabel	optional		*string*
flagValue	optional		*boolean*
value	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="Value" type="FIELD-VALUE" minOccurs="0" maxOccurs="unbounded">
    <xs:sequence>
        <xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0"
        maxOccurs="unbounded"></xs:element>
    </xs:sequence>
    <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="code" type="xs:integer" use="required"></xs:attribute>
    <xs:attribute name="flagValue" type="xs:boolean"></xs:attribute>
    <xs:attribute name="displayLabel" type="xs:string"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Values Element

Untergeordnet Elemente

DisplayLabel Element

DisplayLabel Element

A display label for a field or value in a specified language. The displayLabel attribute can be used where there is no label for a particular language.

Tabelle A-81
Attribute für DisplayLabel

Attribut	Verwenden	Beschreibung	Gültige Werte
lang	optional		*NMTOKEN*
value	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
    <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="lang" type="xs:NMTOKEN" default="en"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Value Element

ModelBuilder Element

Tabelle A-82
Attribute für ModelBuilder

Attribut	Verwenden	Beschreibung	Gültige Werte
allowNoInputs	optional		*boolean*
allowNoOutputs	optional		*boolean*
miningFunctions	**erforderlich**		*beliebig*
nullifyBlanks	optional		*boolean*

XML-Darstellung

```
<xs:element name="ModelBuilder">
   <xs:sequence>
      <xs:element name="Algorithm"></xs:element>
      <xs:element name="ModelingFields" minOccurs="0">
         <xs:sequence>
            <xs:element name="InputFields" minOccurs="0"></xs:element>
            <xs:element name="OutputFields" minOccurs="0"></xs:element>
         </xs:sequence>
      </xs:element>
      <xs:element name="ModelGeneration"></xs:element>
      <xs:element name="ModelFields">
         <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:group ref="DATA-MODEL-EXPRESSION">
               <xs:choice>
                  <xs:element ref="ForEach"></xs:element>
                  <xs:element ref="AddField"></xs:element>
                  <xs:element ref="ChangeField"></xs:element>
                  <xs:element ref="RemoveField"></xs:element>
               </xs:choice>
            </xs:group>
         </xs:sequence>
      </xs:element>
      <xs:element name="ModelEvaluation" minOccurs="0">
         <xs:sequence>
            <xs:element name="RawPropensity" minOccurs="0"></xs:element>
            <xs:element name="AdjustedPropensity" minOccurs="0"></xs:element>
            <xs:element name="VariableImportance" minOccurs="0"></xs:element>
         </xs:sequence>
      </xs:element>
      <xs:element name="AutoModeling" minOccurs="0">
         <xs:sequence>
            <xs:element name="SimpleSettings">
               <xs:sequence>
                  <xs:element ref="PropertyGroup" maxOccurs="unbounded"></xs:element>
               </xs:sequence>
            </xs:element>
            <xs:element name="ExpertSettings" minOccurs="0">
               <xs:sequence>
                  <xs:element ref="Condition" minOccurs="0"></xs:element>
                  <xs:element ref="PropertyGroup" maxOccurs="unbounded"></xs:element>
               </xs:sequence>
            </xs:element>
            <xs:element name="PropertyMap" minOccurs="0">
               <xs:sequence>
```

```
                    <xs:element name="PropertyMapping" maxOccurs="unbounded"></xs:element>
                </xs:sequence>
            </xs:element>
            <xs:element ref="Constraint" minOccurs="0" maxOccurs="unbounded"></xs:element>
        </xs:sequence>
    </xs:element>
</xs:sequence>
<xs:attribute name="miningFunctions" use="required"></xs:attribute>
<xs:attribute name="allowNoInputs" type="xs:boolean" use="optional" default="false"></xs:attribute>
<xs:attribute name="allowNoOutputs" type="xs:boolean" use="optional" default="false"></xs:attribute>
<xs:attribute name="nullifyBlanks" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Node Element

Untergeordnet Elemente

Algorithm Element, AutoModeling Element, ModelEvaluation Element, ModelFields Element, ModelGeneration Element, ModelingFields Element

Algorithm Element

Tabelle A-83
Attribute für Algorithm

Attribut	Verwenden	Beschreibung	Gültige Werte
label	**erforderlich**		*string*
labelKey	optional		*string*
value	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="Algorithm">
    <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ModelBuilder Element

ModelingFields Element

Tabelle A-84
Attribute für ModelingFields

Attribut	Verwenden	Beschreibung	Gültige Werte
controlsId	**erforderlich**		*string*
ignoreBOTH	optional		*boolean*

XML-Darstellung

```
<xs:element name="ModelingFields" minOccurs="0">
    <xs:attribute name="controlsId" type="xs:string" use="required"></xs:attribute>
    <xs:sequence>
        <xs:element name="InputFields" minOccurs="0"></xs:element>
        <xs:element name="OutputFields" minOccurs="0"></xs:element>
    </xs:sequence>
    <xs:attribute name="ignoreBOTH" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ModelBuilder Element

Untergeordnet Elemente

InputFields Element, OutputFields Element

Verwandt Elemente

DocumentGeneration Element, ModelGeneration Element, ModelEvaluation Element

InputFields Element

Tabelle A-85
Attribute für InputFields

Attribut	Verwenden	Beschreibung	Gültige Werte
label	**erforderlich**		*string*
labelKey	optional		*string*
multiple	**erforderlich**		*boolean*
onlyDatetime	optional		*boolean*
onlyDiscrete	optional		*boolean*
onlyNumeric	optional		*boolean*
onlyRanges	optional		*boolean*
onlySymbolic	optional		*boolean*
property	**erforderlich**		*string*
storage	optional		*string*
types	optional		*string*

XML-Darstellung

```
<xs:element name="InputFields" minOccurs="0">
    <xs:attribute name="storage" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="types" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="multiple" type="xs:boolean" use="required"></xs:attribute>
```

```
    <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ModelingFields Element

Verwandt Elemente

OutputFields Element

OutputFields Element

Tabelle A-86
Attribute für OutputFields

Attribut	Verwenden	Beschreibung	Gültige Werte
label	**erforderlich**		*string*
labelKey	optional		*string*
multiple	**erforderlich**		*boolean*
onlyDatetime	optional		*boolean*
onlyDiscrete	optional		*boolean*
onlyNumeric	optional		*boolean*
onlyRanges	optional		*boolean*
onlySymbolic	optional		*boolean*
property	**erforderlich**		*string*
storage	optional		*string*
types	optional		*string*

XML-Darstellung

```
<xs:element name="OutputFields" minOccurs="0">
    <xs:attribute name="storage" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="types" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="multiple" type="xs:boolean" use="required"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ModelingFields Element

Verwandt Elemente

InputFields Element

ModelGeneration Element

Tabelle A-87
Attribute für ModelGeneration

Attribut	Verwenden	Beschreibung	Gültige Werte
controlsId	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="ModelGeneration">
   <xs:attribute name="controlsId" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ModelBuilder Element

Verwandt Elemente

DocumentGeneration Element, ModelingFields Element, ModelEvaluation Element

ModelFields Element

XML-Darstellung

```
<xs:element name="ModelFields">
   <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:group ref="DATA-MODEL-EXPRESSION">
         <xs:choice>
            <xs:element ref="ForEach"></xs:element>
            <xs:element ref="AddField"></xs:element>
            <xs:element ref="ChangeField"></xs:element>
            <xs:element ref="RemoveField"></xs:element>
         </xs:choice>
      </xs:group>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

ModelBuilder Element

Untergeordnet Elemente

AddField Element, ChangeField Element, ForEach Element, RemoveField Element

ModelEvaluation Element

Tabelle A-88
Attribute für ModelEvaluation

Attribut	Verwenden	Beschreibung	Gültige Werte
container	**erforderlich**		*beliebig*
controlsId	**erforderlich**		*string*
outputContainer	optional		*beliebig*

XML-Darstellung

```
<xs:element name="ModelEvaluation" minOccurs="0">
   <xs:attribute name="controlsId" type="xs:string" use="required"></xs:attribute>
   <xs:sequence>
      <xs:element name="RawPropensity" minOccurs="0"></xs:element>
      <xs:element name="AdjustedPropensity" minOccurs="0"></xs:element>
      <xs:element name="VariableImportance" minOccurs="0"></xs:element>
   </xs:sequence>
   <xs:attribute name="container" use="required"></xs:attribute>
   <xs:attribute name="outputContainer" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ModelBuilder Element

Untergeordnet Elemente

AdjustedPropensity Element, RawPropensity Element, VariableImportance Element

Verwandt Elemente

DocumentGeneration Element, ModelingFields Element, ModelGeneration Element

RawPropensity Element

Tabelle A-89
Attribute für RawPropensity

Attribut	Verwenden	Beschreibung	Gültige Werte
availabilityProperty	optional		*string*
defaultValue	optional		*boolean*
property	optional		*string*

XML-Darstellung

```
<xs:element name="RawPropensity" minOccurs="0">
   <xs:attribute name="property" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="availabilityProperty" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="defaultValue" type="xs:boolean" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ModelEvaluation Element

AdjustedPropensity Element

Tabelle A-90
Attribute für AdjustedPropensity

Attribut	Verwenden	Beschreibung	Gültige Werte
availabilityProperty	optional		*string*
defaultValue	optional		*boolean*
property	optional		*string*

XML-Darstellung

```
<xs:element name="AdjustedPropensity" minOccurs="0">
    <xs:attribute name="property" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="availabilityProperty" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="defaultValue" type="xs:boolean" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ModelEvaluation Element

VariableImportance Element

Tabelle A-91
Attribute für VariableImportance

Attribut	Verwenden	Beschreibung	Gültige Werte
availabilityProperty	optional		*string*
defaultValue	optional		*boolean*
property	optional		*string*

XML-Darstellung

```
<xs:element name="VariableImportance" minOccurs="0">
    <xs:attribute name="property" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="availabilityProperty" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="defaultValue" type="xs:boolean" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ModelEvaluation Element

AutoModeling Element

Tabelle A-92
Attribute für AutoModeling

Attribut	Verwenden	Beschreibung	Gültige Werte
enabledByDefault	optional		*beliebig*

XML-Darstellung

```
<xs:element name="AutoModeling" minOccurs="0">
    <xs:sequence>
        <xs:element name="SimpleSettings">
            <xs:sequence>
                <xs:element ref="PropertyGroup" maxOccurs="unbounded"></xs:element>
            </xs:sequence>
        </xs:element>
        <xs:element name="ExpertSettings" minOccurs="0">
            <xs:sequence>
                <xs:element ref="Condition" minOccurs="0"></xs:element>
                <xs:element ref="PropertyGroup" maxOccurs="unbounded"></xs:element>
            </xs:sequence>
        </xs:element>
        <xs:element name="PropertyMap" minOccurs="0">
            <xs:sequence>
                <xs:element name="PropertyMapping" maxOccurs="unbounded"></xs:element>
            </xs:sequence>
        </xs:element>
        <xs:element ref="Constraint" minOccurs="0" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
    <xs:attribute name="enabledByDefault" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ModelBuilder Element

Untergeordnet Elemente

Constraint Element, ExpertSettings Element, PropertyMap Element, SimpleSettings Element

SimpleSettings Element

XML-Darstellung

```
<xs:element name="SimpleSettings">
    <xs:sequence>
        <xs:element ref="PropertyGroup" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

AutoModeling Element

Untergeordnet Elemente

PropertyGroup Element

ExpertSettings Element

XML-Darstellung

```
<xs:element name="ExpertSettings" minOccurs="0">
   <xs:sequence>
      <xs:element ref="Condition" minOccurs="0"></xs:element>
      <xs:element ref="PropertyGroup" maxOccurs="unbounded"></xs:element>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

AutoModeling Element

Untergeordnet Elemente

Condition Element, PropertyGroup Element

PropertyMap Element

XML-Darstellung

```
<xs:element name="PropertyMap" minOccurs="0">
   <xs:sequence>
      <xs:element name="PropertyMapping" maxOccurs="unbounded"></xs:element>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

AutoModeling Element

Untergeordnet Elemente

PropertyMapping Element

PropertyMapping Element

Tabelle A-93
Attribute für PropertyMapping

Attribut	Verwenden	Beschreibung	Gültige Werte
property	**erforderlich**		*string*
systemProperty	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="PropertyMapping" maxOccurs="unbounded">
   <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="systemProperty" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertyMap Element

ModelOutput Element

Tabelle A-94
Attribute für ModelOutput

Attribut	Verwenden	Beschreibung	Gültige Werte
deprecatedScriptNames	optional		*string*
helpLink	optional		*string*
id	**erforderlich**		*string*
label	optional		*string*
labelKey	optional		*string*
scriptName	optional		*string*

XML-Darstellung

```
<xs:element name="ModelOutput">
   <xs:sequence maxOccurs="unbounded">
      <xs:choice maxOccurs="unbounded">
         <xs:element ref="Properties"></xs:element>
         <xs:element name="Containers" minOccurs="0">
            <xs:sequence maxOccurs="unbounded">
               <xs:element ref="Container"></xs:element>
            </xs:sequence>
         </xs:element>
         <xs:element ref="UserInterface"></xs:element>
         <xs:element ref="Constructors" minOccurs="0"></xs:element>
         <xs:element ref="ModelProvider" minOccurs="0"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="scriptName" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="helpLink" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Extension Element

Untergeordnet Elemente

Constructors Element, Containers Element, ModelProvider Element, Properties Element, UserInterface Element

Verwandt Elemente

DocumentOutput Element, InteractiveDocumentBuilder Element, InteractiveModelBuilder Element, Node Element

Containers Element

XML-Darstellung

```
<xs:element name="Containers" minOccurs="0">
    <xs:sequence maxOccurs="unbounded">
        <xs:element ref="Container"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

ModelOutput Element

Untergeordnet Elemente

Container Element

ModelProvider Element

Tabelle A-95
Attribute für ModelProvider

Attribut	Verwenden	Beschreibung	Gültige Werte
container	**erforderlich**		*string*
isPMML	optional		*boolean*

XML-Darstellung

```
<xs:element name="ModelProvider">
    <xs:attribute name="container" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="isPMML" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

DocumentOutput Element, InteractiveDocumentBuilder Element, InteractiveModelBuilder Element, ModelOutput Element, Node Element

ModelType Element

Defines a new model type

Tabelle A-96
Attribute für ModelType

Attribut	Verwenden	Beschreibung	Gültige Werte
format	**erforderlich**		**utf8** **binary**
id	**erforderlich**		*string*
inputDirection	optional		*string*
outputDirection	optional		*string*

XML-Darstellung

```
<xs:element name="ModelType">
    <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="format" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="utf8"></xs:enumeration>
                <xs:enumeration value="binary"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="inputDirection" type="xs:string" use="optional" default="[in]"></xs:attribute>
    <xs:attribute name="outputDirection" type="xs:string" use="optional" default="[out]"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ContainerTypes Element

Verwandt Elemente

DocumentType Element

ModelViewerPanel Element

Tabelle A-97
Attribute für ModelViewerPanel

Attribut	Verwenden	Beschreibung	Gültige Werte
container	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="ModelViewerPanel">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="container" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Tab Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

ActionButton Element, ComboBoxControl Element, ExtensionObjectPanel Element, SelectorPanel Element, StaticText Element, SystemControls Element, TabbedPanel Element, TextBrowserPanel Element

Module Element

Tabelle A-98
Attribute für Module

Attribut	Verwenden	Beschreibung	Gültige Werte
libraryId	optional		*string*
name	optional		*string*
systemModule	optional		**SmartScore**

XML-Darstellung

```
<xs:element name="Module">
   <xs:sequence>
      <xs:element ref="InputFiles"></xs:element>
      <xs:element ref="OutputFiles"></xs:element>
      <xs:element ref="StatusCodes" minOccurs="0" maxOccurs="1"></xs:element>
   </xs:sequence>
   <xs:attribute name="systemModule" use="optional" default="SmartScore">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="SmartScore"></xs:enumeration>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="libraryId" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="name" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Execution Element

Untergeordnet Elemente

InputFiles Element, OutputFiles Element, StatusCodes Element

MultiFieldChooserControl Element

Tabelle A-99
Attribute für MultiFieldChooserControl

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*

Attribut	Verwenden	Beschreibung	Gültige Werte
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
onlyDatetime	optional		*boolean*
onlyDiscrete	optional		*boolean*
onlyNumeric	optional		*boolean*
onlyRanges	optional		*boolean*
onlySymbolic	optional		*boolean*
property	**erforderlich**		*string*
showLabel	optional		*boolean*
storage	optional		*string*
types	optional		*string*

XML-Darstellung

```
<xs:element name="MultiFieldChooserControl">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="storage" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="types" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

MultiItemChooserControl Element

Tabelle A-100
Attribute für MultiItemChooserControl

Attribut	Verwenden	Beschreibung	Gültige Werte
catalog	**erforderlich**		*string*
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="MultiItemChooserControl">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
   <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
   <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
   <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="catalog" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

SingleItemChooserControl Element

Node Element

Tabelle A-101
Attribute für Node

Attribut	Verwenden	Beschreibung	Gültige Werte
deprecatedScriptNames	optional		*string*
description	optional		*string*
descriptionKey	optional		*string*
helpLink	optional		*string*
id	**erforderlich**		*string*
label	**erforderlich**		*string*
labelKey	optional		*string*
palette	optional		**import** **fieldOp** **recordOp** **modeling** **dbModeling** **graph** **output** **export** **modeling.classification** **modeling.association** **modeling.segmentation** **modeling.auto**
relativePosition	optional		*string*
relativeTo	optional		*string*
scriptName	optional		*string*
type	**erforderlich**		**dataReader** **dataWriter** **dataTransformer** **modelApplier** **modelBuilder** **documentBuilder**

XML-Darstellung

```
<xs:element name="Node">
   <xs:sequence maxOccurs="unbounded">
      <xs:choice maxOccurs="unbounded">
         <xs:element ref="Properties"></xs:element>
         <xs:element name="Containers" minOccurs="0">
```

```
                <xs:sequence maxOccurs="unbounded">
                    <xs:element ref="Container"></xs:element>
                </xs:sequence>
            </xs:element>
            <xs:element ref="UserInterface"></xs:element>
            <xs:element ref="Constructors" minOccurs="0"></xs:element>
            <xs:element ref="ModelProvider" minOccurs="0"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="scriptName" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"></xs:attribute>
    <xs:sequence>
        <xs:element ref="ModelBuilder" minOccurs="0"></xs:element>
        <xs:element ref="DocumentBuilder" minOccurs="0"></xs:element>
        <xs:element ref="Execution"></xs:element>
        <xs:element ref="OutputDataModel" minOccurs="0"></xs:element>
    </xs:sequence>
    <xs:attribute name="type" type="NODE-TYPE" use="required">
        <xs:enumeration value="dataReader"></xs:enumeration>
        <xs:enumeration value="dataWriter"></xs:enumeration>
        <xs:enumeration value="dataTransformer"></xs:enumeration>
        <xs:enumeration value="modelApplier"></xs:enumeration>
        <xs:enumeration value="modelBuilder"></xs:enumeration>
        <xs:enumeration value="documentBuilder"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="palette" type="SYSTEM-PALETTE" use="optional">
        <xs:enumeration value="import"></xs:enumeration>
        <xs:enumeration value="fieldOp"></xs:enumeration>
        <xs:enumeration value="recordOp"></xs:enumeration>
        <xs:enumeration value="modeling"></xs:enumeration>
        <xs:enumeration value="dbModeling"></xs:enumeration>
        <xs:enumeration value="graph"></xs:enumeration>
        <xs:enumeration value="output"></xs:enumeration>
        <xs:enumeration value="export"></xs:enumeration>
        <xs:enumeration value="modeling.classification"></xs:enumeration>
        <xs:enumeration value="modeling.association"></xs:enumeration>
        <xs:enumeration value="modeling.segmentation"></xs:enumeration>
        <xs:enumeration value="modeling.auto"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="helpLink" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="relativeTo" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="relativePosition" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Extension Element

Untergeordnet Elemente

Constructors Element, Containers Element, DocumentBuilder Element, Execution Element, ModelBuilder Element, ModelProvider Element, OutputDataModel Element, Properties Element, UserInterface Element

Verwandt Elemente

DocumentOutput Element, InteractiveDocumentBuilder Element, InteractiveModelBuilder Element, ModelOutput Element

Containers Element

XML-Darstellung

```
<xs:element name="Containers" minOccurs="0">
   <xs:sequence maxOccurs="unbounded">
      <xs:element ref="Container"></xs:element>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

Node Element

Untergeordnet Elemente

Container Element

Not Element

XML-Darstellung

```
<xs:element name="Not">
   <xs:sequence>
      <xs:group ref="CONDITION-EXPRESSION">
         <xs:choice>
            <xs:element ref="Condition"></xs:element>
            <xs:element ref="And"></xs:element>
            <xs:element ref="Or"></xs:element>
            <xs:element ref="Not"></xs:element>
         </xs:choice>
      </xs:group>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

And Element, Command Element, Constraint Element, CreateDocument Element, CreateDocumentOutput Element, CreateInteractiveDocumentBuilder Element, CreateInteractiveModelBuilder Element, CreateModel Element, CreateModelApplier Element,

CreateModelOutput Element, Enabled Element, Option Element, Or Element, Run Element, Visible Element

Untergeordnet Elemente

And Element, Condition Element, Not Element, Or Element

NumberFormat Element

Defines format information for a numeric field.

Tabelle A-102
Attribute für NumberFormat

Attribut	Verwenden	Beschreibung	Gültige Werte
decimalPlaces	**erforderlich**		*nonNegativeInteger*
decimalSymbol	**erforderlich**		**period** **comma**
formatType	**erforderlich**		**standard** **scientific** **currency**
groupingSymbol	**erforderlich**		**none** **period** **comma** **space**
name	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION">
   <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="formatType" type="NUMBER-FORMAT-TYPE" use="required">
      <xs:enumeration value="standard"></xs:enumeration>
      <xs:enumeration value="scientific"></xs:enumeration>
      <xs:enumeration value="currency"></xs:enumeration>
   </xs:attribute>
   <xs:attribute name="decimalPlaces" type="xs:nonNegativeInteger" use="required"></xs:attribute>
   <xs:attribute name="decimalSymbol" type="DECIMAL-SYMBOL" use="required">
      <xs:enumeration value="period"></xs:enumeration>
      <xs:enumeration value="comma"></xs:enumeration>
   </xs:attribute>
   <xs:attribute name="groupingSymbol" type="NUMBER-GROUPING-SYMBOL" use="required">
      <xs:enumeration value="none"></xs:enumeration>
      <xs:enumeration value="period"></xs:enumeration>
      <xs:enumeration value="comma"></xs:enumeration>
      <xs:enumeration value="space"></xs:enumeration>
   </xs:attribute>
</xs:element>
```

NumericInfo Element

Tabelle A-103
Attribute für NumericInfo

Attribut	Verwenden	Beschreibung	Gültige Werte
mean	optional		*double*
standardDeviation	optional		*double*

XML-Darstellung

```
<xs:element name="NumericInfo">
   <xs:attribute name="mean" type="xs:double"></xs:attribute>
   <xs:attribute name="standardDeviation" type="xs:double"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

AddField Element, ChangeField Element, Field Element

Option Element

Tabelle A-104
Attribute für Option

Attribut	Verwenden	Beschreibung	Gültige Werte
ifProperty	optional		*string*
unlessProperty	optional		*string*
value	**erforderlich**		

XML-Darstellung

```
<xs:element name="Option">
   <xs:sequence>
      <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
         <xs:choice>
            <xs:element ref="Condition"></xs:element>
            <xs:element ref="And"></xs:element>
            <xs:element ref="Or"></xs:element>
            <xs:element ref="Not"></xs:element>
         </xs:choice>
      </xs:group>
   </xs:sequence>
   <xs:attribute name="value" type="EVALUATED-STRING" use="required"></xs:attribute>
   <xs:attribute name="ifProperty" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="unlessProperty" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Run Element

Untergeordnet Elemente

And Element, Condition Element, Not Element, Or Element

OptionCode Element

Tabelle A-105
Attribute für OptionCode

Attribut	Verwenden	Beschreibung	Gültige Werte
code	optional		*long*
description	optional		*string*
type	optional		**mandatory** **optional**

XML-Darstellung

```
<xs:element name="OptionCode">
   <xs:attribute name="code" type="xs:long"></xs:attribute>
   <xs:attribute name="type" type="LicenseType">
      <xs:enumeration value="mandatory"></xs:enumeration>
      <xs:enumeration value="optional"></xs:enumeration>
   </xs:attribute>
   <xs:attribute name="description" type="xs:string"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

License Element

Or Element

XML-Darstellung

```
<xs:element name="Or">
   <xs:sequence minOccurs="2" maxOccurs="unbounded">
      <xs:group ref="CONDITION-EXPRESSION">
         <xs:choice>
            <xs:element ref="Condition"></xs:element>
            <xs:element ref="And"></xs:element>
            <xs:element ref="Or"></xs:element>
            <xs:element ref="Not"></xs:element>
         </xs:choice>
      </xs:group>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

And Element, Command Element, Constraint Element, CreateDocument Element, CreateDocumentOutput Element, CreateInteractiveDocumentBuilder Element, CreateInteractiveModelBuilder Element, CreateModel Element, CreateModelApplier Element, CreateModelOutput Element, Enabled Element, Not Element, Option Element, Run Element, Visible Element

Untergeordnet Elemente

And Element, Condition Element, Not Element, Or Element

OutputDataModel Element

Tabelle A-106
Attribute für OutputDataModel

Attribut	Verwenden	Beschreibung	Gültige Werte
libraryId	optional		*string*
method	optional		**xml** **dataModelProvider** **sharedLibrary**
mode	optional		**fixed** **modify** **extend** **replace**
providerClass	optional		*string*

XML-Darstellung

```
<xs:element name="OutputDataModel">
    <xs:attribute name="mode" use="optional" default="fixed">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="fixed"></xs:enumeration>
                <xs:enumeration value="modify"></xs:enumeration>
                <xs:enumeration value="extend"></xs:enumeration>
                <xs:enumeration value="replace"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="method" use="optional" default="xml">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="xml"></xs:enumeration>
                <xs:enumeration value="dataModelProvider"></xs:enumeration>
                <xs:enumeration value="sharedLibrary"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="providerClass" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="libraryId" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Node Element

OutputFiles Element

XML-Darstellung

```
<xs:element name="OutputFiles">
   <xs:group ref="RUNTIME-FILES">
      <xs:sequence>
         <xs:element ref="DataFile"></xs:element>
         <xs:element ref="ContainerFile" minOccurs="0" maxOccurs="unbounded"></xs:element>
      </xs:sequence>
   </xs:group>
</xs:element>
```

Übergeordnet Elemente

Execution Element, Module Element

Untergeordnet Elemente

ContainerFile Element, DataFile Element

Palette Element

Tabelle A-107
Attribute für Palette

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
id	**erforderlich**		*string*
label	**erforderlich**		*string*
labelKey	optional		*string*
position	optional		**atStart** **atEnd** **before** **after**
systemPalette	optional		**import** **fieldOp** **recordOp** **modeling** **dbModeling** **graph** **output** **export** **modeling.classification** **modeling.association** **modeling.segmentation** **modeling.auto**

XML-Darstellung

```
<xs:element name="Palette">
   <xs:sequence>
      <xs:element ref="Icon"></xs:element>
```

```
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="position" use="optional">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="atStart"></xs:enumeration>
                <xs:enumeration value="atEnd"></xs:enumeration>
                <xs:enumeration value="before"></xs:enumeration>
                <xs:enumeration value="after"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="systemPalette" type="SYSTEM-PALETTE" use="optional">
        <xs:enumeration value="import"></xs:enumeration>
        <xs:enumeration value="fieldOp"></xs:enumeration>
        <xs:enumeration value="recordOp"></xs:enumeration>
        <xs:enumeration value="modeling"></xs:enumeration>
        <xs:enumeration value="dbModeling"></xs:enumeration>
        <xs:enumeration value="graph"></xs:enumeration>
        <xs:enumeration value="output"></xs:enumeration>
        <xs:enumeration value="export"></xs:enumeration>
        <xs:enumeration value="modeling.classification"></xs:enumeration>
        <xs:enumeration value="modeling.association"></xs:enumeration>
        <xs:enumeration value="modeling.segmentation"></xs:enumeration>
        <xs:enumeration value="modeling.auto"></xs:enumeration>
    </xs:attribute>
</xs:element>
```

Untergeordnet Elemente

Icon Element

Parameters Element

Configuration parameters from the extension node.

Tabelle A-108
Attribute für Parameters

Attribut	Verwenden	Beschreibung	Gültige Werte
count	optional		*nonNegativeInteger*

XML-Darstellung

```
<xs:element name="Parameters" type="PARAMETERS">
    <xs:sequence>
        <xs:element name="Parameter" type="PARAMETER" minOccurs="0" maxOccurs="unbounded">
            <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
                <xs:choice>
                    <xs:element ref="MapValue"></xs:element>
                    <xs:element ref="StructuredValue"></xs:element>
                    <xs:element ref="ListValue"></xs:element>
```

```
                <xs:element ref="Value"></xs:element>
                <xs:element ref="DatabaseConnectionValue"></xs:element>
            </xs:choice>
        </xs:group>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>
```

Untergeordnet Elemente

Parameter Element

Parameter Element

A parameter has a name and a value. A simple value can be expressed with the value attribute; a compound value uses the content model described by ParameterContent. This combination of attribute and content is repeated for nested values.

Tabelle A-109
Attribute für Parameter

Attribut	Verwenden	Beschreibung	Gültige Werte
name	**erforderlich**		*string*
value	optional		*string*

XML-Darstellung

```
<xs:element name="Parameter" type="PARAMETER" minOccurs="0" maxOccurs="unbounded">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
    <xs:choice>
      <xs:element ref="MapValue"></xs:element>
      <xs:element ref="StructuredValue"></xs:element>
      <xs:element ref="ListValue"></xs:element>
      <xs:element ref="Value"></xs:element>
      <xs:element ref="DatabaseConnectionValue"></xs:element>
    </xs:choice>
  </xs:group>
  <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="value" type="xs:string"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Parameters Element

Untergeordnet Elemente

DatabaseConnectionValue Element, ListValue Element, MapValue Element, StructuredValue Element, Value Element

PasswordBoxControl Element

Tabelle A-110
Attribute für PasswordBoxControl

Attribut	Verwenden	Beschreibung	Gültige Werte
columns	optional		*positiveInteger*
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="PasswordBoxControl">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PropertyControl

Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

Properties Element

XML-Darstellung

```
<xs:element name="Properties">
   <xs:sequence>
      <xs:element ref="Property" minOccurs="0" maxOccurs="unbounded"></xs:element>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

DocumentOutput Element, Execution Element, InteractiveDocumentBuilder Element, InteractiveModelBuilder Element, ModelOutput Element, Node Element

Untergeordnet Elemente

Property Element

PropertiesPanel Element

Tabelle A-111
Attribute für PropertiesPanel

Attribut	Verwenden	Beschreibung	Gültige Werte
id	optional		*string*
label	optional		*string*
labelKey	optional		*string*

XML-Darstellung

```
<xs:element name="PropertiesPanel">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:sequence maxOccurs="unbounded">
      <xs:choice>
         <xs:element ref="CheckBoxControl"></xs:element>
         <xs:element ref="TextBoxControl"></xs:element>
         <xs:element ref="PasswordBoxControl"></xs:element>
         <xs:element ref="TextAreaControl"></xs:element>
         <xs:element ref="RadioButtonGroupControl"></xs:element>
         <xs:element ref="CheckBoxGroupControl"></xs:element>
         <xs:element ref="ComboBoxControl"></xs:element>
```

```
            <xs:element ref="SpinnerControl"></xs:element>
            <xs:element ref="ServerFileChooserControl"></xs:element>
            <xs:element ref="ServerDirectoryChooserControl"></xs:element>
            <xs:element ref="ClientFileChooserControl"></xs:element>
            <xs:element ref="ClientDirectoryChooserControl"></xs:element>
            <xs:element ref="TableControl"></xs:element>
            <xs:element ref="SingleFieldChooserControl"></xs:element>
            <xs:element ref="MultiFieldChooserControl"></xs:element>
            <xs:element ref="SingleFieldValueChooserControl"></xs:element>
            <xs:element ref="SingleItemChooserControl"></xs:element>
            <xs:element ref="MultiItemChooserControl"></xs:element>
            <xs:element ref="DBConnectionChooserControl"></xs:element>
            <xs:element ref="DBTableChooserControl"></xs:element>
            <xs:element ref="PropertyControl"></xs:element>
            <xs:element ref="StaticText"></xs:element>
            <xs:element ref="SystemControls"></xs:element>
            <xs:element ref="ActionButton"></xs:element>
            <xs:element ref="PropertiesPanel"></xs:element>
            <xs:element ref="PropertiesSubPanel"></xs:element>
            <xs:element ref="SelectorPanel"></xs:element>
            <xs:element ref="ExtensionObjectPanel"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesSubPanel Element, Tab Element

Untergeordnet Elemente

ActionButton Element, CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, ComboBoxControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, Enabled Element, ExtensionObjectPanel Element, Layout Element, MultiFieldChooserControl Element, MultiItemChooserControl Element, PasswordBoxControl Element, PropertiesPanel Element, PropertiesSubPanel Element, PropertyControl Element, RadioButtonGroupControl Element, SelectorPanel Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SingleItemChooserControl Element, SpinnerControl Element, StaticText Element, SystemControls Element, TableControl Element, TextAreaControl Element, TextBoxControl Element, Visible Element

Verwandt Elemente

PropertiesSubPanel Element

PropertiesSubPanel Element

Tabelle A-112
Attribute für PropertiesSubPanel

Attribut	Verwenden	Beschreibung	Gültige Werte
buttonLabel	optional		*string*
buttonLabelKey	optional		*string*
dialogTitle	optional		*string*
dialogTitleKey	optional		*string*
helpLink	optional		*string*
mnemonic	optional		*string*
mnemonicKey	optional		*string*

XML-Darstellung

```
<xs:element name="PropertiesSubPanel">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:sequence maxOccurs="unbounded">
      <xs:choice>
         <xs:element ref="CheckBoxControl"></xs:element>
         <xs:element ref="TextBoxControl"></xs:element>
         <xs:element ref="PasswordBoxControl"></xs:element>
         <xs:element ref="TextAreaControl"></xs:element>
         <xs:element ref="RadioButtonGroupControl"></xs:element>
         <xs:element ref="CheckBoxGroupControl"></xs:element>
         <xs:element ref="ComboBoxControl"></xs:element>
         <xs:element ref="SpinnerControl"></xs:element>
         <xs:element ref="ServerFileChooserControl"></xs:element>
         <xs:element ref="ServerDirectoryChooserControl"></xs:element>
         <xs:element ref="ClientFileChooserControl"></xs:element>
         <xs:element ref="ClientDirectoryChooserControl"></xs:element>
         <xs:element ref="TableControl"></xs:element>
         <xs:element ref="SingleFieldChooserControl"></xs:element>
         <xs:element ref="MultiFieldChooserControl"></xs:element>
         <xs:element ref="SingleFieldValueChooserControl"></xs:element>
         <xs:element ref="SingleItemChooserControl"></xs:element>
         <xs:element ref="MultiItemChooserControl"></xs:element>
         <xs:element ref="DBConnectionChooserControl"></xs:element>
         <xs:element ref="DBTableChooserControl"></xs:element>
         <xs:element ref="PropertyControl"></xs:element>
         <xs:element ref="StaticText"></xs:element>
         <xs:element ref="SystemControls"></xs:element>
         <xs:element ref="ActionButton"></xs:element>
         <xs:element ref="PropertiesPanel"></xs:element>
         <xs:element ref="PropertiesSubPanel"></xs:element>
         <xs:element ref="SelectorPanel"></xs:element>
         <xs:element ref="ExtensionObjectPanel"></xs:element>
      </xs:choice>
   </xs:sequence>
```

```
    <xs:attribute name="buttonLabel" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="buttonLabelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="dialogTitle" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="dialogTitleKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="helpLink" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element

Untergeordnet Elemente

ActionButton Element, CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, ComboBoxControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, Enabled Element, ExtensionObjectPanel Element, Layout Element, MultiFieldChooserControl Element, MultiItemChooserControl Element, PasswordBoxControl Element, PropertiesPanel Element, PropertiesSubPanel Element, PropertyControl Element, RadioButtonGroupControl Element, SelectorPanel Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SingleItemChooserControl Element, SpinnerControl Element, StaticText Element, SystemControls Element, TableControl Element, TextAreaControl Element, TextBoxControl Element, Visible Element

Verwandt Elemente

PropertiesPanel Element

Property Element

Tabelle A-113
Attribute für Property

Attribut	Verwenden	Beschreibung	Gültige Werte
defaultValue	optional		
defaultValueKey	optional		*string*
deprecatedScriptNames	optional		*string*
description	optional		*string*
descriptionKey	optional		*string*
isList	optional		*boolean*
label	optional		*string*
labelKey	optional		*string*
max	optional		*string*
min	optional		*string*
name	**erforderlich**		*string*
scriptName	optional		*string*

Attribut	Verwenden	Beschreibung	Gültige Werte
type	optional		*string*
valueType	optional		**string** **encryptedString** **fieldName** **integer** **double** **boolean** **date** **enum** **structure** **databaseConnection**

XML-Darstellung

```
<xs:element name="Property">
    <xs:choice>
        <xs:element ref="DefaultValue" minOccurs="0"></xs:element>
    </xs:choice>
    <xs:attribute name="valueType" type="PROPERTY-VALUE-TYPE">
        <xs:enumeration value="string"></xs:enumeration>
        <xs:enumeration value="encryptedString"></xs:enumeration>
        <xs:enumeration value="fieldName"></xs:enumeration>
        <xs:enumeration value="integer"></xs:enumeration>
        <xs:enumeration value="double"></xs:enumeration>
        <xs:enumeration value="boolean"></xs:enumeration>
        <xs:enumeration value="date"></xs:enumeration>
        <xs:enumeration value="enum"></xs:enumeration>
        <xs:enumeration value="structure"></xs:enumeration>
        <xs:enumeration value="databaseConnection"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="isList" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="min" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="max" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="scriptName" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="type" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="defaultValue" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="defaultValueKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Properties Element, PropertySets Element

Untergeordnet Elemente

DefaultValue Element

Verwandt Elemente

PropertyType Element

PropertyControl Element

Tabelle A-114
Attribute für PropertyControl

Attribut	Verwenden	Beschreibung	Gültige Werte
controlClass	**erforderlich**		*string*
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="PropertyControl">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
   <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
   <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
   <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="controlClass" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

PropertyGroup Element

Tabelle A-115
Attribute für PropertyGroup

Attribut	Verwenden	Beschreibung	Gültige Werte
label	optional		*string*
labelKey	optional		*string*
properties	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="PropertyGroup">
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="properties" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

SimpleSettings Element, ExpertSettings Element

PropertySets Element

XML-Darstellung

```
<xs:element name="PropertySets">
    <xs:sequence>
        <xs:element ref="Property" minOccurs="0" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

CommonObjects Element

Untergeordnet Elemente

Property Element

PropertyType Element

Tabelle A-116
Attribute für PropertyType

Attribut	Verwenden	Beschreibung	Gültige Werte
id	**erforderlich**		*string*
isKeyed	optional		*boolean*
isList	optional		*boolean*
max	optional		*string*
min	optional		*string*
valueType	optional		**string** **encryptedString** **fieldName** **integer** **double** **boolean** **date** **enum** **structure** **databaseConnection**

XML-Darstellung

```
<xs:element name="PropertyType">
    <xs:choice>
        <xs:element ref="DefaultValue" minOccurs="0"></xs:element>
    </xs:choice>
    <xs:attribute name="valueType" type="PROPERTY-VALUE-TYPE">
        <xs:enumeration value="string"></xs:enumeration>
        <xs:enumeration value="encryptedString"></xs:enumeration>
        <xs:enumeration value="fieldName"></xs:enumeration>
        <xs:enumeration value="integer"></xs:enumeration>
        <xs:enumeration value="double"></xs:enumeration>
        <xs:enumeration value="boolean"></xs:enumeration>
        <xs:enumeration value="date"></xs:enumeration>
        <xs:enumeration value="enum"></xs:enumeration>
        <xs:enumeration value="structure"></xs:enumeration>
        <xs:enumeration value="databaseConnection"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="isList" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="min" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="max" type="xs:string" use="optional"></xs:attribute>
    <xs:choice>
        <xs:element ref="Enumeration" minOccurs="0"></xs:element>
        <xs:element ref="Structure" minOccurs="0"></xs:element>
    </xs:choice>
    <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="isKeyed" type="xs:boolean" use="optional" default="false"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertyTypes Element

Untergeordnet Elemente

DefaultValue Element, Enumeration Element, Structure Element

Verwandt Elemente

Property Element

PropertyTypes Element

XML-Darstellung

```
<xs:element name="PropertyTypes">
    <xs:sequence>
        <xs:element ref="PropertyType" minOccurs="0" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

CommonObjects Element

Untergeordnet Elemente

PropertyType Element

RadioButtonGroupControl Element

Tabelle A-117
Attribute für RadioButtonGroupControl

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
falseLabel	optional		*string*
falseLabelKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
layoutByRow	optional		*boolean*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
rows	optional		*positiveInteger*
showLabel	optional		*boolean*
trueFirst	optional		*boolean*
trueLabel	optional		*string*
trueLabelKey	optional		*string*
useSubPanel	optional		*boolean*

XML-Darstellung

```
<xs:element name="RadioButtonGroupControl">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="layoutByRow" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="useSubPanel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="falseLabel" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="falseLabelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="trueLabel" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="trueLabelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="trueFirst" type="xs:boolean" use="optional" default="false"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

Range Element

Tabelle A-118
Attribute für Range

Attribut	Verwenden	Beschreibung	Gültige Werte
max	optional		*string*
min	optional		*string*

XML-Darstellung

```
<xs:element name="Range">
    <xs:attribute name="min" type="xs:string"></xs:attribute>
    <xs:attribute name="max" type="xs:string"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

AddField Element, MissingValues Element, ChangeField Element, MissingValues Element, Field Element, MissingValues Element

Range Element

Tabelle A-119
Attribute für Range

Attribut	Verwenden	Beschreibung	Gültige Werte
maxValue	**erforderlich**		*string*
minValue	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="Range" type="RANGE">
    <xs:attribute name="minValue" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="maxValue" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Field Element

RemoveField Element

Tabelle A-120
Attribute für RemoveField

Attribut	Verwenden	Beschreibung	Gültige Werte
fieldRef	**erforderlich**		

XML-Darstellung

```
<xs:element name="RemoveField">
    <xs:attribute name="fieldRef" type="EVALUATED-STRING" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ForEach Element, ModelFields Element

Resources Element

Defines common resources such as client-side libraries and resource bundles, and server-side libraries.

XML-Darstellung

```
<xs:element name="Resources">
   <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
         <xs:element name="Bundle" minOccurs="0"></xs:element>
         <xs:element name="JarFile" minOccurs="0"></xs:element>
         <xs:element name="SharedLibrary" minOccurs="0"></xs:element>
         <xs:element name="HelpInfo" minOccurs="0"></xs:element>
      </xs:choice>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

Extension Element

Untergeordnet Elemente

Bundle Element, HelpInfo Element, JarFile Element, SharedLibrary Element

Bundle Element

Tabelle A-121
Attribute für Bundle

Attribut	Verwenden	Beschreibung	Gültige Werte
id	**erforderlich**		*string*
path	**erforderlich**		
type	**erforderlich**		**list** **properties**

XML-Darstellung

```
<xs:element name="Bundle" minOccurs="0">
   <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="type" use="required">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="list"></xs:enumeration>
            <xs:enumeration value="properties"></xs:enumeration>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="path" type="EVALUATED-STRING" use="required"></xs:attribute>
```

```
</xs:element>
```

Übergeordnet Elemente

Resources Element

JarFile Element

Tabelle A-122
Attribute für JarFile

Attribut	Verwenden	Beschreibung	Gültige Werte
id	**erforderlich**		*string*
path	**erforderlich**		

XML-Darstellung

```
<xs:element name="JarFile" minOccurs="0">
    <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="path" type="EVALUATED-STRING" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Resources Element

SharedLibrary Element

Tabelle A-123
Attribute für SharedLibrary

Attribut	Verwenden	Beschreibung	Gültige Werte
id	**erforderlich**		*string*
path	**erforderlich**		

XML-Darstellung

```
<xs:element name="SharedLibrary" minOccurs="0">
    <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="path" type="EVALUATED-STRING" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Resources Element

HelpInfo Element

Tabelle A-124
Attribute für HelpInfo

Attribut	Verwenden	Beschreibung	Gültige Werte
default	optional		*string*
helpset	optional		

Attribut	Verwenden	Beschreibung	Gültige Werte
id	optional		*string*
missing	optional		*string*
path	optional		
type	**erforderlich**		**native** **javahelp** **html**

XML-Darstellung

```
<xs:element name="HelpInfo" minOccurs="0">
    <xs:attribute name="id" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="type" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="native"></xs:enumeration>
                <xs:enumeration value="javahelp"></xs:enumeration>
                <xs:enumeration value="html"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="path" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="helpset" type="EVALUATED-STRING" use="optional"></xs:attribute>
    <xs:attribute name="default" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="missing" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Resources Element

Run Element

XML-Darstellung

```
<xs:element name="Run">
    <xs:sequence>
        <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
            <xs:choice>
                <xs:element ref="Condition"></xs:element>
                <xs:element ref="And"></xs:element>
                <xs:element ref="Or"></xs:element>
                <xs:element ref="Not"></xs:element>
            </xs:choice>
        </xs:group>
        <xs:element ref="Command" minOccurs="0" maxOccurs="unbounded"></xs:element>
        <xs:element ref="Option" minOccurs="0" maxOccurs="unbounded"></xs:element>
        <xs:element ref="StatusCodes" minOccurs="0"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

Executable Element

Untergeordnet Elemente

And Element, Command Element, Condition Element, Not Element, Option Element, Or Element, StatusCodes Element

SelectorPanel Element

Tabelle A-125
Attribute für SelectorPanel

Attribut	Verwenden	Beschreibung	Gültige Werte
control	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="SelectorPanel">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element name="Selector"></xs:element>
   </xs:sequence>
   <xs:attribute name="control" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Selector Element, Visible Element

Verwandt Elemente

ActionButton Element, ComboBoxControl Element, ExtensionObjectPanel Element, ModelViewerPanel Element, StaticText Element, SystemControls Element, TabbedPanel Element, TextBrowserPanel Element

Selector Element

Tabelle A-126
Attribute für Selector

Attribut	Verwenden	Beschreibung	Gültige Werte
controlValue	**erforderlich**		*string*
panelId	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="Selector">
    <xs:attribute name="panelId" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="controlValue" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

SelectorPanel Element

ServerDirectoryChooserControl Element

Tabelle A-127
Attribute für ServerDirectoryChooserControl

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
mode	**erforderlich**		**open** **save** **import** **export**
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="ServerDirectoryChooserControl">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
        <xs:enumeration value="open"></xs:enumeration>
```

```
        <xs:enumeration value="save"></xs:enumeration>
        <xs:enumeration value="import"></xs:enumeration>
        <xs:enumeration value="export"></xs:enumeration>
    </xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

ServerFileChooserControl Element

Tabelle A-128
Attribute für ServerFileChooserControl

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
mode	**erforderlich**		**open** **save** **import** **export**
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="ServerFileChooserControl">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
```

```
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
        <xs:enumeration value="open"></xs:enumeration>
        <xs:enumeration value="save"></xs:enumeration>
        <xs:enumeration value="import"></xs:enumeration>
        <xs:enumeration value="export"></xs:enumeration>
    </xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

SetContainer Element

Tabelle A-129
Attribute für SetContainer

Attribut	Verwenden	Beschreibung	Gültige Werte
source	**erforderlich**		*string*
target	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="SetContainer">
    <xs:attribute name="source" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="target" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

CreateDocumentOutput Element, CreateInteractiveDocumentBuilder Element, CreateInteractiveModelBuilder Element, CreateModelApplier Element, CreateModelOutput Element

Verwandt Elemente

SetProperty Element

SetProperty Element

Tabelle A-130
Attribute für SetProperty

Attribut	Verwenden	Beschreibung	Gültige Werte
source	**erforderlich**		*string*
target	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="SetProperty">
    <xs:attribute name="source" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="target" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

CreateDocumentOutput Element, CreateInteractiveDocumentBuilder Element, CreateInteractiveModelBuilder Element, CreateModelApplier Element, CreateModelOutput Element

Verwandt Elemente

SetContainer Element

SingleFieldChooserControl Element

Tabelle A-131
Attribute für SingleFieldChooserControl

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
onlyDatetime	optional		*boolean*
onlyDiscrete	optional		*boolean*

Attribut	Verwenden	Beschreibung	Gültige Werte
onlyNumeric	optional		*boolean*
onlyRanges	optional		*boolean*
onlySymbolic	optional		*boolean*
property	**erforderlich**		*string*
showLabel	optional		*boolean*
storage	optional		*string*
types	optional		*string*

XML-Darstellung

```
<xs:element name="SingleFieldChooserControl">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="storage" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="types" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"></xs:attribute>
    <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element,

SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

SingleFieldValueChooserControl Element

Tabelle A-132
Attribute für SingleFieldValueChooserControl

Attribut	Verwenden	Beschreibung	Gültige Werte
description	optional		*string*
descriptionKey	optional		*string*
fieldControl	optional		*string*
fieldDirection	optional		**in** **out** **both** **none** **partition**
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="SingleFieldValueChooserControl">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
   <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
   <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
   <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="fieldControl" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="fieldDirection" type="FIELD-DIRECTION" use="optional">
      <xs:enumeration value="in"></xs:enumeration>
      <xs:enumeration value="out"></xs:enumeration>
      <xs:enumeration value="both"></xs:enumeration>
      <xs:enumeration value="none"></xs:enumeration>
      <xs:enumeration value="partition"></xs:enumeration>
```

```
    </xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

SingleItemChooserControl Element

Tabelle A-133
Attribute für SingleItemChooserControl

Attribut	Verwenden	Beschreibung	Gültige Werte
catalog	**erforderlich**		*string*
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="SingleItemChooserControl">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
```

```
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="catalog" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

MultiItemChooserControl Element

SpinnerControl Element

Tabelle A-134
Attribute für SpinnerControl

Attribut	Verwenden	Beschreibung	Gültige Werte
columns	optional		*positiveInteger*
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
maxDecimalDigits	optional		*positiveInteger*
minDecimalDigits	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
showLabel	optional		*boolean*
stepSize	optional		*decimal*

XML-Darstellung

```
<xs:element name="SpinnerControl">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
```

```
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="5"></xs:attribute>
    <xs:attribute name="stepSize" type="xs:decimal" use="optional" default="1.0"></xs:attribute>
    <xs:attribute name="minDecimalDigits" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="maxDecimalDigits" type="xs:positiveInteger" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, TableControl Element, TextAreaControl Element, TextBoxControl Element

SPSSDataFormat Element

XML-Darstellung

```
<xs:element name="SPSSDataFormat"></xs:element>
```

Übergeordnet Elemente

DataFormat Element

StaticText Element

Tabelle A-135
Attribute für StaticText

Attribut	Verwenden	Beschreibung	Gültige Werte
text	optional		*string*
textKey	optional		*string*

XML-Darstellung

```
<xs:element name="StaticText">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="text" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="textKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

ActionButton Element, ComboBoxControl Element, ExtensionObjectPanel Element, ModelViewerPanel Element, SelectorPanel Element, SystemControls Element, TabbedPanel Element, TextBrowserPanel Element

StatusCode Element

Tabelle A-136
Attribute für StatusCode

Attribut	Verwenden	Beschreibung	Gültige Werte
code	**erforderlich**		*integer*
message	optional		*string*
messageKey	optional		*string*
status	optional		**success** **warning** **error**

XML-Darstellung

```
<xs:element name="StatusCode">
   <xs:attribute name="code" type="xs:integer" use="required"></xs:attribute>
```

```
    <xs:attribute name="status" use="optional" default="success">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="success"></xs:enumeration>
                <xs:enumeration value="warning"></xs:enumeration>
                <xs:enumeration value="error"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="message" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="messageKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

StatusCodes Element

StatusCodes Element

Tabelle A-137
Attribute für StatusCodes

Attribut	Verwenden	Beschreibung	Gültige Werte
defaultMessage	optional		*string*
defaultMessageKey	optional		*string*

XML-Darstellung

```
<xs:element name="StatusCodes">
    <xs:sequence>
        <xs:element ref="StatusCode" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
    <xs:attribute name="defaultMessage" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="defaultMessageKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Module Element, Run Element

Untergeordnet Elemente

StatusCode Element

StatusDetail Element

Supplementary information about a progress or other conditions.

Tabelle A-138
Attribute für StatusDetail

Attribut	Verwenden	Beschreibung	Gültige Werte
destination	optional		**client** **tracefile** **console**

XML-Darstellung

```
<xs:element name="StatusDetail" type="STATUS-DETAIL">
    <xs:sequence>
        <xs:element name="Diagnostic" type="DIAGNOSTIC" minOccurs="0" maxOccurs="unbounded">
            <xs:sequence>
                <xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0"></xs:element>
                <xs:element  name="Parameter"  type="xs:string"  minOccurs="0"
                maxOccurs="unbounded"></xs:element>
            </xs:sequence>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="destination" type="STATUS-DESTINATION" default="client">
        <xs:enumeration value="client"></xs:enumeration>
        <xs:enumeration value="tracefile"></xs:enumeration>
        <xs:enumeration value="console"></xs:enumeration>
    </xs:attribute>
</xs:element>
```

Untergeordnet Elemente

Diagnostic Element

Diagnostic Element

Tabelle A-139
Attribute für Diagnostic

Attribut	Verwenden	Beschreibung	Gültige Werte
code	**erforderlich**		*integer*
severity	optional		**unknown** **information** **warning** **error** **fatal**
source	optional		*string*
subCode	optional		*integer*

XML-Darstellung

```
<xs:element name="Diagnostic" type="DIAGNOSTIC" minOccurs="0" maxOccurs="unbounded">
    <xs:sequence>
        <xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0"></xs:element>
        <xs:element  name="Parameter"  type="xs:string"  minOccurs="0"
        maxOccurs="unbounded"></xs:element>
    </xs:sequence>
    <xs:attribute name="code" type="xs:integer" use="required"></xs:attribute>
    <xs:attribute name="subCode" type="xs:integer" default="0"></xs:attribute>
    <xs:attribute name="severity" type="DIAGNOSTIC-SEVERITY" default="error">
        <xs:enumeration value="unknown"></xs:enumeration>
        <xs:enumeration value="information"></xs:enumeration>
        <xs:enumeration value="warning"></xs:enumeration>
        <xs:enumeration value="error"></xs:enumeration>
        <xs:enumeration value="fatal"></xs:enumeration>
    </xs:attribute>
    <xs:attribute name="source" type="xs:string"></xs:attribute>
```

```
</xs:element>
```

Übergeordnet Elemente

StatusDetail Element

Untergeordnet Elemente

Message Element, Parameter Element

Message Element

Tabelle A-140
Attribute für Message

Attribut	Verwenden	Beschreibung	Gültige Werte
lang	optional		*NMTOKEN*

XML-Darstellung

```
<xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
    <xs:attribute name="lang" type="xs:NMTOKEN"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Diagnostic Element

Parameter Element

XML-Darstellung

```
<xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"></xs:element>
```

Übergeordnet Elemente

Diagnostic Element

Structure Element

XML-Darstellung

```
<xs:element name="Structure">
    <xs:sequence>
        <xs:element ref="Attribute" minOccurs="0" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

PropertyType Element

Untergeordnet Elemente

Attribute Element

StructuredValue Element

A sequence of named values (“attributes”).

XML-Darstellung

```
<xs:element name="StructuredValue" type="STRUCTURED-VALUE">
   <xs:sequence>
      <xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
         <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
            <xs:choice>
               <xs:element ref="MapValue"></xs:element>
               <xs:element ref="StructuredValue"></xs:element>
               <xs:element ref="ListValue"></xs:element>
               <xs:element ref="Value"></xs:element>
               <xs:element ref="DatabaseConnectionValue"></xs:element>
            </xs:choice>
         </xs:group>
         <xs:sequence>
            <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
               <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
                  <xs:choice>
                     <xs:element ref="MapValue"></xs:element>
                     <xs:element ref="StructuredValue"></xs:element>
                     <xs:element ref="ListValue"></xs:element>
                     <xs:element ref="Value"></xs:element>
                     <xs:element ref="DatabaseConnectionValue"></xs:element>
                  </xs:choice>
               </xs:group>
            </xs:element>
         </xs:sequence>
      </xs:element>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

ListValue Element, Attribute Element, ListValue Element, Parameter Element, Attribute Element, ListValue Element

Untergeordnet Elemente

Attribute Element

Attribute Element

Tabelle A-141
Attribute für Attribute

Attribut	Verwenden	Beschreibung	Gültige Werte
name	**erforderlich**		*string*
value	optional		*string*

XML-Darstellung

```
<xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
   <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
      <xs:choice>
         <xs:element ref="MapValue"></xs:element>
         <xs:element ref="StructuredValue"></xs:element>
         <xs:element ref="ListValue"></xs:element>
         <xs:element ref="Value"></xs:element>
         <xs:element ref="DatabaseConnectionValue"></xs:element>
      </xs:choice>
   </xs:group>
   <xs:sequence>
      <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
         <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
            <xs:choice>
               <xs:element ref="MapValue"></xs:element>
               <xs:element ref="StructuredValue"></xs:element>
               <xs:element ref="ListValue"></xs:element>
               <xs:element ref="Value"></xs:element>
               <xs:element ref="DatabaseConnectionValue"></xs:element>
            </xs:choice>
         </xs:group>
      </xs:element>
   </xs:sequence>
   <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="value" type="xs:string"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

StructuredValue Element

Untergeordnet Elemente

DatabaseConnectionValue Element, ListValue Element, ListValue Element, MapValue Element, StructuredValue Element, Value Element

ListValue Element

A sequence of values. All values must have the same content type but this is not checked.

XML-Darstellung

```
<xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
   <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
```

```
        <xs:choice>
            <xs:element ref="MapValue"></xs:element>
            <xs:element ref="StructuredValue"></xs:element>
            <xs:element ref="ListValue"></xs:element>
            <xs:element ref="Value"></xs:element>
            <xs:element ref="DatabaseConnectionValue"></xs:element>
        </xs:choice>
    </xs:group>
</xs:element>
```

Übergeordnet Elemente

Attribute Element

Untergeordnet Elemente

DatabaseConnectionValue Element, ListValue Element, MapValue Element, StructuredValue Element, Value Element

SystemControls Element

Tabelle A-142
Attribute für SystemControls

Attribut	Verwenden	Beschreibung	Gültige Werte
controlsId	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="SystemControls">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="controlsId" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

ActionButton Element, ComboBoxControl Element, ExtensionObjectPanel Element, ModelViewerPanel Element, SelectorPanel Element, StaticText Element, TabbedPanel Element, TextBrowserPanel Element

Tab Element

Tabelle A-143
Attribute für Tab

Attribut	Verwenden	Beschreibung	Gültige Werte
helpLink	optional		*string*
id	optional		*string*
label	**erforderlich**		*string*
labelKey	optional		*string*
mnemonic	optional		*string*
mnemonicKey	optional		*string*

XML-Darstellung

```
<xs:element name="Tab">
   <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
         <xs:element ref="PropertiesPanel"></xs:element>
         <xs:element ref="ExtensionObjectPanel"></xs:element>
         <xs:element ref="TextBrowserPanel"></xs:element>
         <xs:element ref="ModelViewerPanel"></xs:element>
         <xs:element ref="TabbedPanel"></xs:element>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="id" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
   <xs:attribute name="helpLink" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Tabs Element

Untergeordnet Elemente

ExtensionObjectPanel Element, ModelViewerPanel Element, PropertiesPanel Element, TabbedPanel Element, TextBrowserPanel Element

TabbedPanel Element

Tabelle A-144
Attribute für TabbedPanel

Attribut	Verwenden	Beschreibung	Gültige Werte
style	optional		**standard** **sidebar**

XML-Darstellung

```
<xs:element name="TabbedPanel">
```

```
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:sequence maxOccurs="unbounded">
        <xs:element ref="Tabs"></xs:element>
    </xs:sequence>
    <xs:attribute name="style" use="optional">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="standard"></xs:enumeration>
                <xs:enumeration value="sidebar"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Tab Element

Untergeordnet Elemente

Enabled Element, Layout Element, Tabs Element, Visible Element

Verwandt Elemente

ActionButton Element, ComboBoxControl Element, ExtensionObjectPanel Element, ModelViewerPanel Element, SelectorPanel Element, StaticText Element, SystemControls Element, TextBrowserPanel Element

TableControl Element

Tabelle A-145
Attribute für TableControl

Attribut	Verwenden	Beschreibung	Gültige Werte
columns	optional		*positiveInteger*
columnWidths	optional		*string*
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
rows	optional		*positiveInteger*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="TableControl">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="8"></xs:attribute>
    <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"></xs:attribute>
    <xs:attribute name="columnWidths" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TextAreaControl Element, TextBoxControl Element

Tabs Element

Tabelle A-146
Attribute für Tabs

Attribut	Verwenden	Beschreibung	Gültige Werte
defaultTab	optional		*nonNegativeInteger*

XML-Darstellung

```
<xs:element name="Tabs">
```

```
    <xs:sequence>
        <xs:element ref="Tab" minOccurs="0" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
    <xs:attribute name="defaultTab" type="xs:nonNegativeInteger" use="optional" default="0"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

TabbedPanel Element, UserInterface Element

Untergeordnet Elemente

Tab Element

TextAreaControl Element

Tabelle A-147
Attribute für TextAreaControl

Attribut	Verwenden	Beschreibung	Gültige Werte
columns	optional		*positiveInteger*
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
rows	optional		*positiveInteger*
showLabel	optional		*boolean*
wrapLines	optional		*boolean*

XML-Darstellung

```
<xs:element name="TextAreaControl">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
```

```
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="8"></xs:attribute>
    <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"></xs:attribute>
    <xs:attribute name="wrapLines" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextBoxControl Element

TextBoxControl Element

Tabelle A-148
Attribute für TextBoxControl

Attribut	Verwenden	Beschreibung	Gültige Werte
columns	optional		*positiveInteger*
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="TextBoxControl">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
```

```
    </xs:sequence>
    <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
    <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

PropertiesPanel Element, PropertiesSubPanel Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, MultiFieldChooserControl Element, PasswordBoxControl Element, PropertyControl Element, RadioButtonGroupControl Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SpinnerControl Element, TableControl Element, TextAreaControl Element

TextBrowserPanel Element

Tabelle A-149
Attribute für TextBrowserPanel

Attribut	Verwenden	Beschreibung	Gültige Werte
columns	optional		*string*
container	**erforderlich**		*string*
rows	optional		*string*
textFormat	**erforderlich**		**plainText** **html** **rtf**
wrapLines	optional		*boolean*

XML-Darstellung

```
<xs:element name="TextBrowserPanel">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
            <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
```

```
            <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="container" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="textFormat" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="plainText"></xs:enumeration>
                <xs:enumeration value="html"></xs:enumeration>
                <xs:enumeration value="rtf"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="rows" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="columns" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="wrapLines" type="xs:boolean" use="optional" default="false"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Tab Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Verwandt Elemente

ActionButton Element, ComboBoxControl Element, ExtensionObjectPanel Element, ModelViewerPanel Element, SelectorPanel Element, StaticText Element, SystemControls Element, TabbedPanel Element

ToolbarItem Element

Tabelle A-150
Attribute für ToolbarItem

Attribut	Verwenden	Beschreibung	Gültige Werte
action	**erforderlich**		*string*
offset	optional		*nonNegativeInteger*
separatorAfter	optional		*boolean*
separatorBefore	optional		*boolean*
showIcon	optional		*boolean*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:element name="ToolbarItem">
    <xs:attribute name="action" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="true"></xs:attribute>
    <xs:attribute name="separatorBefore" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="separatorAfter" type="xs:boolean" use="optional" default="false"></xs:attribute>
    <xs:attribute name="offset" type="xs:nonNegativeInteger" use="optional" default="0"></xs:attribute>
```

```
</xs:element>
```

Übergeordnet Elemente

Controls Element

UserInterface Element

Tabelle A-151
Attribute für UserInterface

Attribut	Verwenden	Beschreibung	Gültige Werte
actionHandler	optional		*beliebig*
frameClass	optional		*beliebig*

XML-Darstellung

```
<xs:element name="UserInterface">
   <xs:sequence>
      <xs:element ref="Icons" minOccurs="0"></xs:element>
      <xs:element ref="Controls" minOccurs="0"></xs:element>
      <xs:element ref="Tabs" minOccurs="0"></xs:element>
   </xs:sequence>
   <xs:attribute name="frameClass" use="optional"></xs:attribute>
   <xs:attribute name="actionHandler" use="optional"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

DocumentOutput Element, Extension Element, InteractiveDocumentBuilder Element, InteractiveModelBuilder Element, ModelOutput Element, Node Element

Untergeordnet Elemente

Controls Element, Icons Element, Tabs Element

UTF8Format Element

XML-Darstellung

```
<xs:element name="UTF8Format"></xs:element>
```

Übergeordnet Elemente

FileFormatType Element

Value Element

A simple value.

Tabelle A-152
Attribute für Value

Attribut	Verwenden	Beschreibung	Gültige Werte
value	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="Value" type="SIMPLE-VALUE">
    <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

ListValue Element, Attribute Element, ListValue Element, Parameter Element, Attribute Element, ListValue Element

Values Element

XML-Darstellung

```
<xs:element name="Values">
    <xs:sequence>
        <xs:element name="Value" minOccurs="0" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

AddField Element, MissingValues Element, ChangeField Element, MissingValues Element, Field Element, MissingValues Element

Untergeordnet Elemente

Value Element

Value Element

Tabelle A-153
Attribute für Value

Attribut	Verwenden	Beschreibung	Gültige Werte
flagProperty	optional		**trueValue** **falseValue**
value	**erforderlich**		*string*
valueLabel	optional		*string*

XML-Darstellung

```
<xs:element name="Value" minOccurs="0" maxOccurs="unbounded">
    <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="valueLabel" type="xs:string" use="optional"></xs:attribute>
    <xs:attribute name="flagProperty">
```

```
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="trueValue"></xs:enumeration>
                <xs:enumeration value="falseValue"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Values Element

Values Element

Tabelle A-154
Attribute für Values

Attribut	Verwenden	Beschreibung	Gültige Werte
code	**erforderlich**		*integer*
displayLabel	optional		*string*
flagValue	optional		*boolean*
value	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="Values" type="FIELD-VALUE">
    <xs:sequence>
        <xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0"
        maxOccurs="unbounded"></xs:element>
    </xs:sequence>
    <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="code" type="xs:integer" use="required"></xs:attribute>
    <xs:attribute name="flagValue" type="xs:boolean"></xs:attribute>
    <xs:attribute name="displayLabel" type="xs:string"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Field Element

Untergeordnet Elemente

DisplayLabel Element

DisplayLabel Element

A display label for a field or value in a specified language. The displayLabel attribute can be used where there is no label for a particular language.

Tabelle A-155
Attribute für DisplayLabel

Attribut	Verwenden	Beschreibung	Gültige Werte
lang	optional		*NMTOKEN*
value	**erforderlich**		*string*

XML-Darstellung

```
<xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
   <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="lang" type="xs:NMTOKEN" default="en"></xs:attribute>
</xs:element>
```

Übergeordnet Elemente

Values Element

Visible Element

XML-Darstellung

```
<xs:element name="Visible">
   <xs:sequence>
      <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
         <xs:choice>
            <xs:element ref="Condition"></xs:element>
            <xs:element ref="And"></xs:element>
            <xs:element ref="Or"></xs:element>
            <xs:element ref="Not"></xs:element>
         </xs:choice>
      </xs:group>
   </xs:sequence>
</xs:element>
```

Übergeordnet Elemente

ActionButton Element, CheckBoxControl Element, CheckBoxGroupControl Element, ClientDirectoryChooserControl Element, ClientFileChooserControl Element, ComboBoxControl Element, DBConnectionChooserControl Element, DBTableChooserControl Element, ExtensionObjectPanel Element, ModelViewerPanel Element, MultiFieldChooserControl Element, MultiItemChooserControl Element, PasswordBoxControl Element, PropertiesPanel Element, PropertiesSubPanel Element, PropertyControl Element, RadioButtonGroupControl Element, SelectorPanel Element, ServerDirectoryChooserControl Element, ServerFileChooserControl Element, SingleFieldChooserControl Element, SingleFieldValueChooserControl Element, SingleItemChooserControl Element, SpinnerControl Element, StaticText Element, SystemControls Element, TabbedPanel Element, TableControl Element, TextAreaControl Element, TextBoxControl Element, TextBrowserPanel Element, ItemChooserControl Geben Sie

Untergeordnet Elemente

And Element, Condition Element, Not Element, Or Element

Erweitert Typen

Erweiterte Typen erweitern Elemente in einem XML-Dokument durch Hinzufügen von Attributen und untergeordneten Elementen. Um einen erweiterten Typ in einem XML-Dokument zu verwenden, geben Sie den erweiterten Typ mit dem Attribut “xsi:type” für das Element an. Anschließend können Sie die durch den erweiterten Typ definierten Attribute und Elemente verwenden.

ItemChooserControl Geben Sie

Tabelle A-156
Attribute für ItemChooserControl

Attribut	Verwenden	Beschreibung	Gültige Werte
catalog	**erforderlich**		*string*
description	optional		*string*
descriptionKey	optional		*string*
label	optional		*string*
labelAbove	optional		*boolean*
labelKey	optional		*string*
labelWidth	optional		*positiveInteger*
mnemonic	optional		*string*
mnemonicKey	optional		*string*
property	**erforderlich**		*string*
showLabel	optional		*boolean*

XML-Darstellung

```
<xs:complexType name="ItemChooserControl" mixed="false">
   <xs:sequence>
      <xs:choice>
         <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
      </xs:choice>
   </xs:sequence>
</xs:complexType>
```

Erweitert

ComboBoxControl Element

Untergeordnet Elemente

Enabled Element, Layout Element, Visible Element

Anhang

B

Hinweise

Diese Informationen wurden für weltweit angebotene Produkte und Dienstleistungen erarbeitet.

IBM bietet die in diesem Dokument behandelten Produkte, Dienstleistungen oder Merkmale möglicherweise nicht in anderen Ländern an. Informationen zu den derzeit in Ihrem Land erhältlichen Produkten und Dienstleistungen erhalten Sie bei Ihrem zuständigen IBM-Mitarbeiter vor Ort. Mit etwaigen Verweisen auf Produkte, Programme oder Dienste von IBM soll nicht behauptet oder impliziert werden, dass nur das betreffende Produkt oder Programm bzw. der betreffende Dienst von IBM verwendet werden kann. Stattdessen können alle funktional gleichwertigen Produkte, Programme oder Dienste verwendet werden, die keine geistigen Eigentumsrechte von IBM verletzen. Es obliegt jedoch der Verantwortung des Benutzers, die Funktionsweise von Produkten, Programmen oder Diensten von Drittanbietern zu bewerten und zu überprüfen.

IBM verfügt möglicherweise über Patente oder hat Patentanträge gestellt, die sich auf in diesem Dokument beschriebene Inhalte beziehen. Durch die Bereitstellung dieses Dokuments werden Ihnen keinerlei Lizenzen an diesen Patenten gewährt. Lizenzanfragen können schriftlich an folgende Adresse gesendet werden:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.

Bei Lizenzanfragen in Bezug auf DBCS-Daten (Double-Byte Character Set) wenden Sie sich an die für geistiges Eigentum zuständige Abteilung von IBM in Ihrem Land. Schriftliche Anfragen können Sie auch an folgende Adresse senden:

Intellectual Property Licensing, Legal and Intellectual Property Law, IBM Japan Ltd., 1623-14, Shimotsuruma, Yamato-shi, Kanagawa 242-8502 Japan.

Der folgende Abschnitt findet in Großbritannien und anderen Ländern keine Anwendung, in denen solche Bestimmungen nicht mit der örtlichen Gesetzgebung vereinbar sind: INTERNATIONAL BUSINESS MACHINES STELLT DIESE VERÖFFENTLICHUNG IN DER VERFÜGBAREN FORM OHNE GARANTIEN BEREIT, SEIEN ES AUSDRÜCKLICHE ODER STILLSCHWEIGENDE, EINSCHLIESSLICH JEDOCH NICHT NUR DER GARANTIEN BEZÜGLICH DER NICHT-RECHTSVERLETZUNG, DER GÜTE UND DER EIGNUNG FÜR EINEN BESTIMMTEN ZWECK. Manche Rechtsprechungen lassen den Ausschluss ausdrücklicher oder implizierter Garantien bei bestimmten Transaktionen nicht zu, sodass die oben genannte Ausschlussklausel möglicherweise nicht für Sie relevant ist.

Diese Informationen können technische Ungenauigkeiten oder typografische Fehler aufweisen. An den hierin enthaltenen Informationen werden regelmäßig Änderungen vorgenommen. Diese Änderungen werden in neuen Ausgaben der Veröffentlichung aufgenommen. IBM kann jederzeit und ohne vorherige Ankündigung Optimierungen und/oder Änderungen an den Produkten und/oder Programmen vornehmen, die in dieser Veröffentlichung beschrieben werden.

Jegliche Verweise auf Drittanbieter-Websites in dieser Information werden nur der Vollständigkeit halber bereitgestellt und dienen nicht als Befürwortung dieser. Das Material auf diesen Websites ist kein Bestandteil des Materials zu diesem IBM-Produkt und die Verwendung erfolgt auf eigene Gefahr.

IBM kann die von Ihnen angegebenen Informationen verwenden oder weitergeben, wie dies angemessen erscheint, ohne Ihnen gegenüber eine Verpflichtung einzugehen.

Lizenznehmer dieses Programms, die Informationen dazu benötigen, wie (i) der Austausch von Informationen zwischen unabhängig erstellten Programmen und anderen Programmen und (ii) die gegenseitige Verwendung dieser ausgetauschten Informationen ermöglicht wird, wenden sich an:

IBM Software Group, Attention: Licensing, 233 S. Wacker Dr., Chicago, IL 60606, USA.

Derartige Informationen stehen ggf. in Abhängigkeit von den jeweiligen Geschäftsbedingungen sowie in einigen Fällen der Zahlung einer Gebühr zur Verfügung.

Das in diesem Dokument beschriebene lizenzierte Programm und sämtliche dafür verfügbaren lizenzierten Materialien werden von IBM gemäß dem IBM-Kundenvertrag, den Internationalen Nutzungsbedingungen für Programmpakete der IBM oder einer anderen zwischen uns getroffenen Vereinbarung bereitgestellt.

Jegliche hier enthaltene Daten zur Leistung wurden in einer überwachten Umgebung ermittelt. Aus diesem Grund können in anderen Betriebsumgebungen gewonnene Ergebnisse stark davon abweichen. Einige Messungen wurden unter Umständen auf Systemen im Entwicklungsstadium durchgeführt und es kann nicht garantiert werden, dass diese Messungen auf allgemein verfügbaren Systemen zum gleichen Ergebnis führen. Darüber hinaus wurden einige Messungen unter Umständen durch Extrapolation bestimmt. Die tatsächlichen Ergebnisse können hiervon abweichen. Die Benutzer dieses Dokuments sollten die entsprechenden Daten für die jeweils vorliegende Umgebung prüfen.

Informationen zu Produkten von Drittanbietern wurden von den Anbietern des jeweiligen Produkts, aus deren veröffentlichten Ankündigungen oder anderen, öffentlich verfügbaren Quellen bezogen. IBM hat diese Produkte nicht getestet und kann die Genauigkeit bezüglich Leistung, Kompatibilität oder anderen Behauptungen nicht bestätigen, die sich auf Drittanbieter-Produkte beziehen. Fragen bezüglich der Funktionen von Drittanbieter-Produkten sollten an die Anbieter der jeweiligen Produkte gerichtet werden.

Alle Aussagen bezüglich der zukünftigen Ausrichtung von IBM oder der Absichten des Unternehmens können ohne vorherige Ankündigung geändert oder zurückgenommen werden und stellen lediglich Ziele und Vorgaben dar.

Diese Informationen enthalten Beispiele zu Daten und Berichten, die im täglichen Geschäftsbetrieb Verwendung finden. Um diese so vollständig wie möglich zu illustrieren, umfassen die Beispiele Namen von Personen, Unternehmen, Marken und Produkten. Alle diese Namen sind fiktiv und jegliche Ähnlichkeit mit Namen und Adressen realer Unternehmen ist rein zufällig.

Unter Umständen werden Fotografien und farbige Abbildungen nicht angezeigt, wenn Sie diese Informationen nicht in gedruckter Form verwenden.

Marken

IBM, das IBM-Logo, ibm.com und SPSS sind Marken der IBM Corporation und in vielen Ländern weltweit registriert. Eine aktuelle Liste der IBM-Marken finden Sie im Internet unter *http://www.ibm.com/legal/copytrade.shtml*.

Intel, das Intel-Logo, Intel Inside, das Intel Inside-Logo, Intel Centrino, das Intel Centrino-Logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium und Pentium sind Markten oder eingetragene Marken der Intel Corporation oder der Tochtergesellschaften des Unternehmens in den USA und anderen Ländern.

Linux ist eine eingetragene Marke von Linus Torvalds in den USA, anderen Ländern oder beidem.

Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA, anderen Ländern oder beidem.

UNIX ist eine eingetragene Marke der The Open Group in den USA und anderen Ländern.

Java und alle Java-basierten Marken sowie Logos sind Marken von Sun Microsystems, Inc. in den USA, anderen Ländern oder beidem.

Andere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein.

Index

Abschnitt ‘Benutzeroberfläche’, Spezifikationsdatei, 134
  benutzerdefinierte Paletten, 55
Abschnitt ‘Objektdefinition’, Spezifikationsdatei, 61
Abwärtskompatibilität, Erhaltung, 97
action
  Handler, 135
  Schaltflächen, 155
Action Element, 257
Action-Element, Spezifikationsdatei, 51
ActionButton Element, 258
ActionButton-Element, Spezifikationsdatei, 155
Actions Element, 258
Actions-Element, Spezifikationsdatei, 51
AddField Element, 259
AddField-Element, Spezifikationsdatei, 82, 88
AdjustedPropensity Element, 341
Algorithm Element, 336
Algorithm-Element, Spezifikationsdatei, 103
Algorithmus, Spezifikation für Modellerstellungsknoten, 103
And Element, 263
And-Element, Spezifikationsdatei, 90
Angepasste Neigungen, 82
Anmerkungen (Registerkarte), Knotendialogfeld, 28
Anwenden von Modellen, 124
Anwendungsprogrammierschnittstelle (API)
  C-basiert, 5
  clientseitig, 5, 221
  Dokumentation, 220
  Java-basiert, 5
  PSAPI, 5, 222
  Serverseitig, 5, 222
Architektur
  Serverseitige API, 222
  System, 1
Attribute Element, 263, 328, 390
Attribute, Controller, 162
Attribute-Element (Catalogs), Spezifikationsdatei, 52
Attribute-Element, Spezifikationsdatei, 79
aufgezählte Eigenschaften, 76–77
Ausblenden von Paletten und Unterpaletten, 60
Ausführung, externe (des Erweiterungsprozesses), 252
Ausführungsanforderungsdokument, XML-Ausgabe, 239
Ausgabe
  Dateien, 5, 70
  Dokumente (XML), 235
Ausgabefenster, 131
  benutzerdefiniert, 204
  erstellen, 30
Ausgaben, Registerkarte im Managerfenster, 125
Ausgabeobjekte
  Dokument, 14
  Modell, 12
Auswahlelement für mehrere Felder, 171
Automatisierte Modellierung, 116
AutoModeling Element, 341
AutoModeling-Element, Spezifikationsdatei, 116

Bedingungen, in der Spezifikationsdatei, 91
  einfache, 94
  für die Steuerung der Anzeigeeigenschaften verwenden, 199
  zur Steuerung der Sichtbarkeit von Bildschirmkomponenten verwenden, 204
  zusammengesetzt, 95
Beispielknoten, CLEF, 33
benutzerdefinierte Ausgabefenster, 204
Benutzeroberfläche
  Definieren, 131
  erstellen, 22
Benutzeroberflächenkomponenten
  Aktionsschaltflächen, 155
  statischer Text, 156
  Systemsteuerelemente, 157
Berichte, 50
Beschriftungen, über Komponenten positionieren, 189
Bibliotheken, freigegeben (serverseitig), 45, 72, 231
Bilder, für Symbole erstellen, 21
BinaryFormat Element, 264
Bundle Element, 372
Bundle-Element, Spezifikationsdatei, 44

C++
  Helper, 244
  language, 222
C-basierte API, 5
Cache, Daten, 239
Caching-Status, Knoten, 19
Callback-Funktionen, API, 222, 225
Catalog Element, 264
Catalog-Element, Spezifikationsdatei, 52
Catalogs Element, 265
Cell Element, 323
Cell-Element, Spezifikationsdatei, 192
ChangeField Element, 265
ChangeField-Element, Spezifikationsdatei, 84
CheckBoxControl Element, 269
CheckBoxControl-Element, Spezifikationsdatei, 163
CheckBoxGroupControl Element, 270
CheckBoxGroupControl-Element, Spezifikationsdatei, 164
Client
  Dateiauswahlelement, 166
  Verzeichnisauswahlelement, 165
ClientDirectoryChooserControl Element, 271
ClientDirectoryChooserControl-Element, Spezifikationsdatei, 165
ClientFileChooserControl Element, 272
ClientFileChooserControl-Element, Spezifikationsdatei, 166
Clientseitige API, 5, 221
  Klassen, 221

verwenden, 221
Clientseitige Komponenten, 1
ColumnControl-Element, Spezifikationsdatei, 185
ComboBoxControl Element, 273
ComboBoxControl-Element, Spezifikationsdatei, 167
Command Element, 275
CommonObjects Element, 275
CommonObjects-Element, Spezifikationsdatei, 46
Condition Element, 276
Condition-Element, Spezifikationsdatei, 91
Constraint Element, 278
Constraint-Element, Spezifikationsdatei, 122
Constructors Element, 279
Constructors-Element, Spezifikationsdatei, 126
Container, 49, 68
  Dateien, 72
  Inhalt untersuchen, 253
  Typen, 49
Container Element, 279
Container-Element, Spezifikationsdatei, 68
ContainerFile Element, 280
ContainerFile-Element für Ausgabedateien, Spezifikationsdatei, 72
ContainerFile-Element für Eingabedateien, Spezifikationsdatei, 71
Containers Element, 302, 320, 322, 345, 352
ContainerTypes Element, 280
ContainerTypes-Element, Spezifikationsdatei, 49
Controller, 161
  Attribute von, 162
  Auswahl eines einzelnen Elements, 182
  Auswahl mehrerer Elemente, 174
  Auswahlelement für mehrere Felder, 171
  Client-Dateiauswahlelement, 166
  Client-Verzeichnisauswahlelement, 165
  Datenbanktabellenauswahlelement, 170
  Datenbankverbindungsauswahlelement, 168
  Drehfeld, 183
  Eigenschaftssteuerelement, 175
  Element für die Auswahl eines einzelnen Felds, 180
  Element für die Serverdateiauswahl, 179
  Element zur Server-Verzeichnisauswahl, 178
  Kombinationsfeld, 167
  Kontrollkästchen, 163
  Kontrollkästchengruppe, 164
  Optionsschaltflächengruppe, 176
  Passwortfeld, 175
  Tabelle, 184
  Textbereich, 185
  Textfeld, 186
Controls Element, 280
Controls-Element, Spezifikationsdatei, 138
CreateDocument Element, 281
CreateDocument-Element, Spezifikationsdatei, 127
CreateDocumentOutput Element, 282
CreateDocumentOutput-Element, Spezifikationsdatei, 128
CreateInteractiveDocumentBuilder Element, 282
CreateInteractiveModelBuilder Element, 283
CreateInteractiveModelBuilder-Element, Spezifikationsdatei, 113
CreateModel Element, 284
CreateModel-Element, Spezifikationsdatei, 127
CreateModelApplier Element, 285
CreateModelApplier-Element, Spezifikationsdatei, 129
CreateModelOutput Element, 286
CreateModelOutput-Element, Spezifikationsdatei, 127

DatabaseConnectionValue Element, 287
DataFile Element, 288
DataFormat Element, 288
DataModel Element, 288
Dateistruktur, 6
Daten
  Leserknoten, 11, 34, 63
  Mining-Funktionen, Model Builder, 101
  Schreiberknoten, 15, 63
  Transformationsknoten, 12, 35, 63
  Typen, 230
Datenbank
  Tabellenauswahlelement, 170
  Verbindungsauswahlelement, 168
Datenmodell, 4, 237
  Behandlung, 232
  Provider, 87
Datenmodelldokument, XML-Ausgabe, 237
DBConnectionChooserControl Element, 295
DBConnectionChooserControl-Element, Spezifikationsdatei, 168
DBTableChooserControl Element, 296
DBTableChooserControl-Element, Spezifikationsdatei, 170
DefaultValue Element, 297
DefaultValue-Element, Spezifikationsdatei, 71
Deinstallieren von Erweiterungen, 256
DelimitedDataFormat Element, 299
Diagnostic Element, 305, 387
Diagnostic-Element, Statusdetaildokument, 243
Dialogfeld “Algorithmuseinstellungen”, 118–119, 122
Dialogfelder erstellen, 22
Direktzugriffstasten, 145
  in CLEF, 52, 145
DisplayLabel Element, 300, 334, 401
DocumentBuilder Element, 300
DocumentBuilder-Element, Spezifikationsdatei, 124
DocumentGeneration Element, 300
DocumentGeneration-Element, Spezifikationsdatei, 124
DocumentOutput Element, 301
DocumentOutput-Element, Spezifikationsdatei, 125
DocumentType Element, 302
DocumentType-Element, Spezifikationsdatei, 50
Dokument
  Ausgabe, für Knoten festlegen, 125
  Ausgabeobjekte, 14
  Erstellungsknoten, 14, 36, 63, 100, 124

Typen, 50
Dokumente, 50, 100
Erstellen, 124
Drehfeldsteuerelemente, 183

Eigenschaften
aufgezählt, 76
definieren, 66
Einstellungen untersuchen, 253
Fenster (verschachtelt), 161
keyed, 47, 79
panel, 150
Runtime, 70
Unterfenster, 158
Eigenschaften, Typen
aufgezählt, 77
strukturiert, 78
Eigenschaftendateien (.properties), 212
Eigenschafteneinstellungen untersuchen, 253
Eigenschaftssteuerelemente, 154
Benutzeroberflächenkomponenten, 154
Controller, 161
Eigenschaftsfenster, 158
PropertyControl-Element, 175
Eingabedateien, 5, 70
Eingabehilfen, 211, 218
Enabled Element, 303
Enabled-Element, Spezifikationsdatei, 199
Ensemble-Modellierungsknoten, 116
Enum Element, 304
Enum-Element, Spezifikationsdatei, 78
Enumeration Element, 304
Enumeration-Element, Spezifikationsdatei, 78
ErrorDetail Element, 304
Erstellen
interaktive Modelle, 100, 112
Modelle, 100
Erweiterung
Objektfenster, 149
Erweiterungen, 1
deinstallieren, 256
Erhalten der Abwärtskompatibilität, 97
Installieren, 256
lokalisieren, 211
verteilen, 255
Erweiterungselement, Spezifikationsdatei, 42
evaluierte Zeichenketten, 81
Exclude-Element, Spezifikationsdatei, 88
Executable Element, 306
Execution Element, 307
Execution-Element, Spezifikationsdatei, 70
ExpertSettings Element, 343
ExpertSettings-Element, Spezifikationsdatei, 119
extension
Module, 222
Ordner, 6
Extension Element, 307
extension.xml (Datei), 7, 39
ExtensionDetails Element, 308
ExtensionDetails-Element, Spezifikationsdatei, 42
ExtensionObjectPanel Element, 308
ExtensionObjectPanel-Element, Spezifikationsdatei, 150
Externe Ausführung des Erweiterungsprozesses, 252

Fehlerbehandlung, 245
Fehlerdetaildokument, XML-Ausgabe, 239
Fehlermeldungen, lokalisieren, 243
Fehlersuche
Ändern der Serverkonfigurationsoptionen, 254
Erweiterungen, 252
Registerkarte “Fehlersuche”, Dialogfeld “Knoten”, 42, 253
Feld
groups, 108–109
Metadaten, 87
Sets, 88
Fenster
Angeben, 147
Eigenschaftsfenster, 150, 158
Eigenschaftsunterfenster, 158
Erweiterungsobjekt, 149
Modell-Viewer, 152
Textbrowser, 147
Fensterbereich, Dialogfeld, 27
Field Element, 293, 309
FieldFormats Element, 289, 312
FieldGroup Element, 291, 314–315
FieldGroups Element, 291, 315
FieldName Element, 292, 314, 316
Fields Element, 293
FieldSet-Element, Spezifikationsdatei, 88
FileFormatType Element, 316
FileFormatTypes Element, 317
Filespace, 231
ForEach Element, 317
ForEach-Element, Spezifikationsdatei, 85, 88
Fortschrittsfunktionen, API, 227
Frame-Klasse, 135
Freigegebene Bibliotheken, 45, 72, 231

Generierte Objekte
Diagramm oder Bericht, 124
Modell, 100
Glyphen, 18
Grafikanforderungen, Symbole, 20
Grafiken, 50
Gruppen, Feld, 108–109

Handles, in Callback-Funktionen, 225
Hauptfenster, anpassen, 142
HelpInfo Element, 373
HelpInfo-Element, Spezifikationsdatei, 208
Helpset-Dateien, JavaHelp, 207

Hilfesysteme
  lokalisieren, 216
  Speicherort von, 208
  verknüpfen mit, 206
Hilfethemen, anzuzeigendes Thema festlegen, 209
Hilfeverknüpfungen, für Knoten spezifizieren, 64
Hintergründe, Symbol, 20
Hostfunktionen, APIs, 226
Hostinformationsdokument, XML-Ausgabe, 240
HTML Help
  lokalisieren, 216
  verknüpfen mit, 206

Icon Element, 318
Icon-Element, Spezifikationsdatei, 137
Icons Element, 319
Icons-Element, Spezifikationsdatei, 137
Identifier Element, 298
Include-Element, Spezifikationsdatei, 88
InputFields Element, 337
InputFields-Element, Spezifikationsdatei, 105
InputFiles Element, 319
InputFiles-Element, Spezifikationsdatei, 71
Installieren von Erweiterungen, 256
InteractiveDocumentBuilder Element, 320
InteractiveModelBuilder Element, 321
InteractiveModelBuilder-Element, Spezifikationsdatei, 114
Interaktionsfenster, 112
interaktiv
  Modelle erstellen, 100, 112
ISO-Standard, Sprachcodes, 212
ItemChooserControl Typ, 403
Iteration, in der Spezifikationsdatei, 85
Iteratorfunktionen, API, 226

JarFile Element, 373
JarFile-Element, Spezifikationsdatei, 44
Java, 7
  API, 5
  Klassen, 44, 51, 75, 135, 149, 175, 204
JavaHelp
  lokalisieren, 217
  verknüpfen mit, 207

Kanalfunktionen, API, 227
Kataloge, 52
Katalogelement, Spezifikationsdatei, 52
KeyValue Element, 327
Klassen, 7
  Clientseitige API, 221
Klonen, Modell, 49
Knoten, 4, 10
  Attribute, 63
  Caching-Status, 19
  Datenleser, 11
  Datenschreiber, 15
  Datentransformer, 12
  definieren, 62
  Dokumentersteller, 14
  Ensemble, 116
  Funktionen, API, 226
  Informationsdokument (XML), 231
  Modellanwender, 14
  Modellersteller, 12
  Name, benutzerdefiniert, 29
  Symbole erstellen, 18
  Testen von CLEF-Erweiterungen, 251
  Typen, 4, 230
Knoteninformationsdokument, XML-Ausgabe, 240
Kombinationsfelder, 167
Kommentarzeile, in Spezifikationsdatei, 40
Kompatibel zu Vorversionen, Aufrechterhaltung für eine Erweiterung, 97
Konstruktoren, 100
Konstruktoren verwenden, 126
Kontrollkästchen, 163
Kontrollkästchengruppen, 164
  Ändern der Anzeigereihenfolge in, 190
  Ändern der Zeilenzahl, 189

Ländereinstellung in Windows, 211
language
  Codes, ISO-Standard, 212
  festlegen, 211
Laufzeiteigenschaften, 70
Layout Element, 322
Layout für benutzerdefiniertes Eigenschaftssteuerelement, 189
  einfach, 189
  erweitert, 190
Layout-Element, Spezifikationsdatei, 191
Layouts von Eigenschaftssteuerelementen
  benutzerdefiniert, 189
  Standard, 187
Layouts, Eigenschaftssteuerelement
  benutzerdefiniert, 189
  Standard, 187
License Element, 324
Liste von Werten, die von aufgezählten Eigenschaften verwendet wird, 77
ListValue Element, 324, 329, 390
lokalisieren
  Erweiterungen, 211
  Fehlermeldungen, 243
  Hilfesysteme, 216
löschen
  Paletten und Unterpaletten, 60

MapEntry Element, 326
MapValue Element, 325
Marken, 406
Menü
  Bereich, Dialogfeld, 26

Element, benutzerdefiniert, 15, 140
Menu Element, 329
Menü-Element, Spezifikationsdatei, 138
MenuItem Element, 331
MenuItem-Element, Spezifikationsdatei, 140
Menüs, Standard und benutzerdefiniert, 15, 138
Message Element, 306, 388
Message-Element, Statusdetaildokument, 243
Metadaten, Feld, 87
Mining-Funktionen, Model Builder, 101
MissingValues Element, 261, 267, 311, 332
model
Viewer-Fenster, 152
ModelBuilder Element, 335
ModelBuilder-Element, Spezifikationsdatei, 101
ModelDetail Element, 285
ModelEvaluation Element, 340
ModelField Element, 262, 268, 311
ModelFields Element, 339
ModelFields-Element, Spezifikationsdatei, 108
ModelGeneration Element, 339
ModelGeneration-Element, Spezifikationsdatei, 108
ModelingFields Element, 336
ModelingFields-Element, Spezifikationsdatei, 103
Modell
Anwendungsknoten, 14, 63, 99–100, 129
Ausgabeobjekte, 99
Erstellungsknoten, 12, 36, 63, 99–100
Nugget, 12
Signatur, 108
Typen, 50
Modellausgabe
für Knoten festlegen, 110
Objekte, 12, 100
Modelle, 99
automatisiert, 116
Daten, 4
Erstellen, 100
interaktiv, 112
zuweisen, 124
Modelle, Registerkarte im Managerfenster, 110
ModelOutput Element, 344
ModelOutput-Element, Spezifikationsdatei, 110
ModelProvider Element, 345
ModelProvider-Element, Spezifikationsdatei, 65
ModelType Element, 345
ModelType-Element, Spezifikationsdatei, 50
ModelViewerPanel Element, 346
ModelViewerPanel-Element, Spezifikationsdatei, 153
Module Element, 347
Module, Erweiterung, 222
Module-Element, Spezifikationsdatei, 72
Modulfunktionen, API, 224
Modulinformationsdokument, XML-Ausgabe, 240
MultiFieldChooserControl Element, 347
MultiFieldChooserControl-Element, Spezifikationsdatei, 171
MultiItemChooserControl Element, 349
MultiItemChooserControl-Element, Spezifikationsdatei, 174

Neigungen, Angeben in Datenmodell, 82, 89
Node Element, 350
Node-Element, Spezifikationsdatei, 62
Not Element, 352
Not-Element, Spezifikationsdatei, 90
Nugget, Modell, 12
NumberFormat Element, 290, 313, 353
NumericInfo Element, 354

Objekt-IDs, 62
Operationen, in der Spezifikationsdatei, 81
Option Element, 354
OptionCode Element, 355
Optionsschaltflächengruppen, 176
Ändern der Anzeigereihenfolge in, 190
Ändern der Zeilenzahl, 189
Or Element, 355
Or-Element, Spezifikationsdatei, 90
Ordner, Erweiterung, 6
OutputDataModel Element, 356
OutputDataModel-Element, Spezifikationsdatei, 75
OutputFields Element, 338
OutputFields-Element, Spezifikationsdatei, 106
OutputFiles Element, 357
OutputFiles-Element, Spezifikationsdatei, 72

Palette Element, 357
Palette-Element, Spezifikationsdatei, 55
Paletten
Ausblenden, 60
für Knoten spezifizieren, 17, 55, 63
löschen, 60
Palettes-Element, Spezifikationsdatei, 55
Parameter Element, 306, 359, 388
Parameter-Element, Statusdetaildokument, 243
Parameterdokument, XML-Ausgabe, 241
Parameters Element, 358
Parsing, XML, 246
PasswordBoxControl Element, 360
PasswordBoxControl-Element, Spezifikationsdatei, 175
Passwortfeld, 175
Peer, 222
Funktionen, API, 224
PMML-Format, Modellausgabe, 65, 152
Präzise Steuerelementpositionen, festlegen, 191
Predictive Server-API (PSAPI), 222
Properties Element, 361
Properties-Element, Spezifikationsdatei, 66
Runtime, 70
PropertiesPanel Element, 361

PropertiesPanel-Element, Spezifikationsdatei
  verschachtelt, 161
  verwendet von Registerkarte oder Eigenschaftsunterfenster, 152
PropertiesSubPanel Element, 363
PropertiesSubPanel-Element, Spezifikationsdatei, 159
Property Element, 364
Property-Element, Spezifikationsdatei, 66
  Runtime, 70
PropertyControl Element, 366
PropertyControl-Element, Spezifikationsdatei, 175
PropertyGroup Element, 367
PropertyGroup-Element, Spezifikationsdatei, 118–119
PropertyMap Element, 343
PropertyMapping Element, 343
PropertySet-Element, Spezifikationsdatei, 48
PropertySets Element, 367
PropertySets-Element, Spezifikationsdatei, 48
PropertyType Element, 368
PropertyType-Element, Spezifikationsdatei, 47
PropertyTypes Element, 369
PropertyTypes-Element, Spezifikationsdatei, 47
Provider, Datenmodell, 87
Prozessfluss, serverseitige API, 227
PSAPI, 5

QuickInfo-Text, spezifizieren, 29, 51

RadioButtonGroupControl Element, 369
RadioButtonGroupControl-Element, Spezifikationsdatei, 176
Rahmen, Symbol, 19
Range Element, 333, 371
RawPropensity Element, 340
Rechtliche Hinweise, 404
Registerkarten, in Dialogfeld oder Fenster definieren, 143
Registerkartenbereich, Dialogfeld, 27
Reihenfolge von Steuerelementen, ändern, 190
RemoveField Element, 371
RemoveField-Element, Spezifikationsdatei, 85
Resources Element, 372
Resources-Element, Spezifikationsdatei, 43
Ressourcen, Erweiterung, 222
Ressourcenpakete, 44
Rohe Neigungen, 82
Rollen, in der Modellausgabe, 89
Run Element, 374

Schaltflächenbereich, Dialogfeld, 30
Schlüsseleigenschaften, 47, 79
Scoring von Daten, 30
Selector Element, 375
SelectorPanel Element, 375
Server
  Element für die Dateiauswahl, 179
  Konfigurationsoptionen für Fehlersuche ändern, 254
  Steuerelement zur Verzeichnisauswahl, 178
  temporäre Datei, 71
ServerDirectoryChooserControl Element, 376
ServerDirectoryChooserControl-Element, Spezifikationsdatei, 178
ServerFileChooserControl Element, 377
ServerFileChooserControl-Element, Spezifikationsdatei, 179
Serverseitig
  Bibliotheken, 45, 72, 231
  Komponenten, 2
Serverseitige API, 5, 222
  Architektur, 222
  Funktionen, 229
  verwenden, 246
ServerTempDir Element, 298
ServerTempFile Element, 297
Servicefunktionen, API, 222–223
SetContainer Element, 378
SetProperty Element, 379
SharedLibrary Element, 373
SharedLibrary-Element, Spezifikationsdatei, 45
Sichtbarkeit von Bildschirmkomponenten, Steuerung, 204
Signatur, Modell, 108
SimpleSettings Element, 342
SimpleSettings-Element, Spezifikationsdatei, 118
SingleFieldChooserControl Element, 379
SingleFieldChooserControl-Element, Spezifikationsdatei, 180
SingleFieldValueChooserControl Element, 381
SingleItemChooserControl Element, 382
SingleItemChooserControl-Element, Spezifikationsdatei, 182
Skriptnamen, 62
  für Eigenschaften spezifizieren, 67
  für Knoten spezifizieren, 64, 96
Speichertypen, 230
Spezifikationsdatei, 1, 4, 39
SpinnerControl Element, 383
SpinnerControl-Element, Spezifikationsdatei, 183
SPSSDataFormat Element, 384
SQL-Generierungsdokument, XML-Ausgabe, 241
SQL-Pushback, 232
StaticText Element, 385
StaticText-Element, Spezifikationsdatei, 156
statischer Text, 156
Statusbereich, Dialogfeld, 27
StatusCode Element, 385
StatusCode-Element, Spezifikationsdatei, 73, 239
StatusCodes Element, 386
StatusCodes-Element, Spezifikationsdatei, 73
StatusDetail Element, 386
Statusdetaildokument, XML-Ausgabe, 243
Steuerelement für die Auswahl eines einzelnen Elements, 182
Steuerelement für die Auswahl eines einzelnen Felds, 180
Steuerelement für die Auswahl mehrerer Elemente, 174

Steuerelemente des Eigenschaftsfensters
  Eigenschaftsfenster (verschachtelt), 161
  Eigenschaftsunterfenster, 158
Steuerelemente, Bildschirmeigenschaft, 154
  Benutzeroberflächenkomponenten, 154
  Controller, 161
  Eigenschaftsfenster, 158
Steuerelemente, Knotendialogfeld, 22
Structure Element, 388
Structure-Element, Spezifikationsdatei, 78
StructuredValue Element, 327, 389
Strukturdeklarationen, 76
Strukturierte Eigenschaften, 78
Symbol
  Bereich, Dialogfeld, 26
  Typen, 137
Symbole
  Bilder erstellen, 21
  erstellen, 18
  generiertes Modell, 19
  Grafikanforderungen, 20
  Knoten, 19
Symbolleiste
  Bereich, Dialogfeld, 26
  Element, benutzerdefiniert, 16, 141
System
  Menüs, 138
  Steuerelemente, 157
SystemControls Element, 391
SystemControls-Element, Spezifikationsdatei, 157

Tab Element, 392
Tab-Element, Spezifikationsdatei, 144
TabbedPanel Element, 392
Tabellensteuerelemente, 184
TableControl Element, 393
TableControl-Element, Spezifikationsdatei, 184
Tabs Element, 394
Tabs-Element, Spezifikationsdatei, 143
Temporäre Dateien, 231
  Server, 71
Testen
  CLEF-Erweiterungen, 251
  lokalisierten Knoten und Hilfe, 217
Text
  Bereichssteuerelemente, 185
  Browserfenster, 147
  Feldsteuerelemente, 186
TextAreaControl Element, 395
TextAreaControl-Element, Spezifikationsdatei, 185
TextBoxControl Element, 396
TextBoxControl-Element, Spezifikationsdatei, 186
TextBrowserPanel Element, 397
TextBrowserPanel-Element, Spezifikationsdatei, 148
Titelleiste, Dialogfeld, 25
ToolbarItem Element, 398
ToolbarItem-Element, Spezifikationsdatei, 141

Unterpaletten
  Ausblenden, 60
  für Knoten spezifizieren, 17, 55, 63
  löschen, 60
UserInterface Element, 399
UserInterface-Element, Spezifikationsdatei, 69
  benutzerdefinierte Paletten, 55
UTF8Format Element, 399

Value Element, 334, 399–400
Values Element, 333, 400–401
VariableImportance Element, 341
verschlüsselte Zeichenketten, 76
Verteilen von Erweiterungen, 255
Visible Element, 402
Visible-Element, Spezifikationsdatei, 204

Werteliste, 52
Werttypen, Eigenschaft, 76

XML
  Ausgabedokumente, 235
  Deklaration, Spezifikationsdatei, 42
  Parsing-API, 246

Zeichenketten
  evaluierte, 81
  verschlüsselt, 76
Zeilen, Ändern der Anzahl für Kontrollkästchen- und Optionsschaltflächengruppen, 189
Zugriffstasten, 145
Zusammengesetzte Bedingungen, 95