COGNOS®

Cognos Impromptu (R)

PowerPrompts-Handbuch

Produktinformationen

Dieses Dokument bezieht sich auf Impromptu (R) Version 7.1 und möglicherweise auch auf zukünftige Versionen. Jüngere Versionen dieses Dokuments finden Sie auf der Website des Cognos Support (http://support.cognos.com).

Copyright

Copyright (C) 2003 Cognos Incorporated

Wir haben uns bemüht, sicherzustellen, dass die Informationen in diesem Dokument so genau und vollständig wie möglich sind; trotzdem ist nicht auszuschließen, dass vereinzelt Druckfehler oder inhaltliche Ungenauigkeiten auftreten können. Cognos übernimmt keine Verantwortung für Verluste, die durch die Verwendung der in diesem Dokument enthaltenen Informationen entstehen.

Dieses Dokument zeigt das Veröffentlichungsdatum. Bei den in diesem Dokument enthaltenen Informationen sind Änderungen vorbehalten. Alle Veränderungen oder Verbesserungen der Software oder des Dokuments werden in zukünftigen Ausgaben dokumentiert.

Diese Software/dieses Dokument enthält urheberrechtlich geschützte Informationen von Cognos Incorporated. Alle Recht vorbehalten. Die Rückentwicklung dieser Software ist nicht gestattet. Diese Software oder dieses Dokument oder Teile davon dürfen ohne die vorherige ausdrückliche, schriftliche Zustimmung von Cognos Incorporated nicht kopiert, reproduziert, in einem Datenabrufsystem gespeichert, in einer beliebigen Form und mit beliebigen Hilfsmitteln übertragen oder in andere Sprachen übersetzt werden.

Cognos, das Cognos Logo, Axiant, COGNOSuite, Cognos Upfront, Cognos DecisionStream, Impromptu, NoticeCast, PowerCube, PowerHouse, PowerPlay, Scenario und 4Thought sind Marken oder eingetragene Marken von Cognos Incorporated in den Vereinigten Staaten und/oder anderen Ländern. Alle anderen genannten Namen sind Marken oder eingetragene Marken der entsprechenden Firmen.

Informationen über Cognos-Produkte und den Zugriff auf sie finden Sie bei www.Cognos.com

Inhaltsverzeichnis

Kapitel 1: Erstellen einer PowerPrompts-Anwendung *7*
- Grundlegende Informationen und Anforderungen *8*
- Unterschiede zwischen PowerPrompts 6.0 und Series 7 *8*
- Aktualisieren von PowerPrompts 6.0-Anwendungen auf 7,1 *8*
- Integrieren von PowerPrompts Security und Access Manager *9*
- Einstellungen für Initialisierungsdateien *10*
- Hinzufügen von virtuellen Verzeichnissen für Abbildungen *11*
- Starten von PowerPrompts *11*
- Einrichten des HTML-Editors und der HTML-Vorlage *11*
- Hinzufügen von Codes zu Seiten mithilfe einer JavaScript-Datei *12*

Erstellen einer neuen Anwendung *12*
- Formatieren der Start-, Schluss- und Fehlerseiten *13*
- Hinzufügen einer Seite *14*
- Importieren einer Seite *14*
- Bearbeiten einer Seite *15*
- Löschen einer Seite *15*

Verknüpfen von Seiten: Überblick *15*
- Hinzufügen einer Verknüpfung durch Ziehen *17*
- Hinzufügen einer Verknüpfung mithilfe des Dialogfelds *17*
- Hinzufügen einer Verknüpfungsbedingung *18*
- Löschen einer Verknüpfung *19*

Dynamos: Überblick *19*
- Erstellen eines Dynamos mithilfe des Assistenten *20*
- Manuelles Erstellen eines Dynamos *21*

Skript: Überblick *23*
- Hinzufügen eines Skripts *23*

Testen und Bereitstellen der Anwendung: Überblick *24*
- Auswählen eines Berichts zum Testen der Anwendung *24*
- Überprüfen der Anwendung *24*
- Testen der Anwendung *25*
- Beheben von Skriptfehlern *25*
- Anzeigen des generierten Skripts *26*
- Benachrichtigen des Impromptu Web Reports-Administrators *27*

Kapitel 2: Verfahrensweisen *29*
- Filtern eines Berichts nach mehreren Werten *29*
- Speichern der von Benutzern gewählten Optionen zwischen Sitzungen *30*
- Erstellen einer Auswertung der vom Benutzer gewählten Optionen *30*
- Hinzufügen von Impromptu-Benutzerklassenfiltern zu Auswahllisten *31*
- Erstellen einer Auswahlliste mit Direkteingabe *32*
- Erstellen einer kaskadierenden Auswahlliste auf einer einzelnen Seite *32*
- Verwenden von ADO aus PowerPrompts *34*
- Abrufen der Benutzerkennung und des Kennworts für die Datenbank von Access Manager *34*
- Erstellen verschiedener Designs für unterschiedliche Upfront-Themen *35*
- Erstellen einer einzelnen PowerPrompts-Anwendung für mehrere Sprachen *36*

Kapitel 3: JavaScript in PowerPrompts *37*
- Quelltabelle serverseitiger JavaScript-Objekte *37*
- #include-Anweisung und andere Präprozessor-Anweisungen *40*

App-Objekt *41*
BackURL-Eigenschaft *42*
CurrentPage-Eigenschaft *43*
Errors-Eigenschaft *43*
FinalURL-Eigenschaft *44*
IsTestMode-Eigenschaft *44*
Path-Eigenschaft *45*
ReportScript-Eigenschaft *45*
Variables-Eigenschaft *45*
RunDynamo-Methode *46*
Connection-Objekt *47*
Execute-Methode *47*
Field-Objekt *47*
HTMLEncodedValue-Eigenschaft *48*
Index-Eigenschaft *48*
Name-Eigenschaft *49*
Type-Eigenschaft *50*
Value-Eigenschaft *51*
Operator()-Methode (Feld) *52*
Query-Objekt *52*
SQL-Eigenschaft *54*
AddColumn-Methode *54*
AndFilterBy-Methode *55*
AndSummaryFilterBy-Methode *55*
AssociateColumn-Methode *56*
GroupBy-Methode *56*
OrFilterBy-Methode *57*
OrSummaryFilterBy-Methode *57*
RemoveColumn-Methode *58*
SetDistinct-Methode *58*
SetPromptValue-Methode *59*
SortBy-Methode *59*
Recordset-Objekt *60*
CurrentRecordIndex-Eigenschaft *60*
EOF-Eigenschaft *61*
Fields-Eigenschaft *62*
MaxRecords-Eigenschaft *62*
Close-Methode *63*
MoveNext-Methode *64*
Open-Methode *64*
Request-Objekt *66*
Cookies-Eigenschaft *66*
ServerVariables-Eigenschaft *66*
Variables-Eigenschaft *67*
Response-Objekt *68*
ContentType-Eigenschaft *68*
AppendCookie-Methode *69*
AppendHeader-Methode *69*
Clear-Methode *69*
ClearContent-Methode *70*
ClearHeaders-Methode *70*
Redirect-Methode *70*
Write-Methode *70*
WriteFile-Methode *71*
Writeln-Methode *71*

Server-Objekt *73*
ScriptTimeout-Eigenschaft *73*
CreateObject-Methode *74*
FormatNumber-Methode *74*
HTMLEncode-Methode *75*
URLDecode-Methode *75*
URLEncode-Methode *76*
StringList-Objekt *76*
Count-Eigenschaft *77*
Contains-Methode *77*
Item-Methode *78*
Join-Methode *79*
Operator() Methode (StringList) *79*
toString-Methode *80*
Upfront-Objekt *81*
Language-Eigenschaft *81*
Locale-Eigenschaft *82*
Theme-Eigenschaft *82*
ExecuteCommand-Methode *82*
GetPageFragment-Methode *83*
User-Objekt *83*
Description-Eigenschaft *83*
Email-Eigenschaft *84*
Telephone-Eigenschaft *84*
UserClass-Eigenschaft *85*
UserClasses-Eigenschaft *85*
UserName-Eigenschaft *86*

JavaScript-Berichtsänderungsmethoden *86*
GetColumnTitle-Methode *86*
GetQuery-Methode *87*
GetReport-Methode *87*
GetReportObject-Methode *88*
AddDataItem-Berichtsmethode *88*
ApplyTemplate-Berichtsmethode *88*
RemoveReportObject-Berichtsmethode *89*
SetListInsertCursor-Berichtsmethode *89*
SetPrimaryFrame-Berichtsmethode *90*
AddConditionalFormat-Berichtsobjektmethode *90*
ApplyStyle-Berichtsobjektmethode *91*
HorizontalAlign-Berichtsobjektmethode *91*
HorizontalAlignToColumn-Berichtsobjektmethode *92*
SetText-Berichtsobjektmethode *92*
SetTextJustification-Berichtsobjektmethode *93*
VerticalAlign-Berichtsobjektmethode *93*
ApplyStyle-Spaltentitelmethode *93*
SetText-Spaltentitelmethode *94*
SetTextJustification-Spaltentitelmethode *94*
AddColumn-Abfragemethode *94*
AddNamedCondition-Abfragemethode *95*
AndFilterBy-Abfragemethode *95*
AndSummaryFilterBy-Abfragemethode *95*
AssociateColumn-Abfragemethode *96*
GroupBy-Abfragemethode *96*
OrFilterBy-Abfragemethode *96*
OrSummaryFilterBy-Abfragemethode *97*
RemoveColumn-Abfragemethode *97*
SetMaxRows-Abfragemethode *98*
SetPromptValue-Abfragemethode *98*
SortBy-Abfragemethode *98*
Impromptu-Ausdrücke in PowerPrompts *99*
JavaScript-Clientmethoden *100*
FormatUserVar-Methode *100*
GetUserVar-Methode *100*
GetUserVarValues-Methode *101*

Index ***103***

Kapitel 1: Erstellen einer PowerPrompts-Anwendung

PowerPrompts ist eine Anwendung, die Berichtskonsumenten durch eine Reihe von HTML-Seiten führt, in denen die für einen Bericht gewünschten Informationen ausgewählt werden. Der Berichtskonsument sieht anschließend den Bericht, der aufgrund dieser gewählten Informationen generiert wurde. Mit einer PowerPrompts-Anwendung können Berichtsbenutzer beispielsweise Folgendes hinzufügen:

- Spalten zu einem Bericht
- Spaltenformatierungen
- eine Berichtsvorlage
- bedingte Formatierungen
- Filter

Verwenden Sie PowerPrompts Developer Studio, um diese webbasierten Anwendungen für Impromptu Web Reports zu erstellen.

PowerPrompts wird mit der Administrator-Version von Impromptu ausgeliefert.

Verwenden Sie beim Erstellen von PowerPrompts-Anwendungen die folgende Checkliste:

- Erstellen Sie eine Reihe von HTML-Seiten, mit denen die Berichtskonsumenten aufgefordert werden, den Inhalt eines Berichts zu definieren.
- Erstellen Sie Anweisungen für die Navigation durch die HTML-Seiten.
- Seiten können zusätzliche Navigations-Links enthalten, so dass Berichtsbenutzer einzelnen Zweigen der PowerPrompts-Anwendung folgen können und nicht alle Seiten der Anwendung durchlaufen müssen.
- Erstellen Sie alle Dynamos, die Daten aus logischen Datenbanken zurückgeben, und fügen Sie diese den entsprechenden HTML-Seiten hinzu.
- Erstellen Sie das Skript, das den mit der PowerPrompts-Anwendung verbundenen Bericht ändert.
- Speichern Sie die Anweisungen als PowerPrompts-Anwendungsdatei (.xxm).
- Wählen Sie einen Impromptu-Bericht zum Testen der PowerPrompts-Anwendung aus.
- Testen Sie die PowerPrompts-Anwendung, und korrigieren Sie alle Fehler.
- Benachrichtigen Sie den Impromptu Web Reports-Administrator, dass die PowerPrompts-Anwendung einsatzbereit ist.

Mit PowerPrompts können Sie die Anzahl der zu verwaltenden Berichte reduzieren, indem Sie statt für jeden Benutzer einen eigenen nur einen einzigen Bericht erstellen, den die Benutzer dann an ihre Bedürfnisse anpassen können. Daher kann ein mit einer PowerPrompts-Anwendung verbundener Bericht sämtliche Daten enthalten, für die Sie ohne PowerPrompts eine Vielzahl von Berichten benötigen würden.

Ihre Berichtskonsumenten können nun den gleichen Bericht für zahlreiche Abfragen verwenden und müssen nicht die dem Bericht zugrunde liegenden Metadaten verstehen, wenn sie ihre Auswahl vornehmen.

Verwandte Themen

- "Einrichten des HTML-Editors und der HTML-Vorlage" (S. 11)
- "Starten von PowerPrompts" (S. 11)
- "Erstellen einer neuen Anwendung" (S. 12)
- "Formatieren der Start-, Schluss- und Fehlerseiten" (S. 13)
- "Hinzufügen einer Seite" (S. 14)
- "Verknüpfen von Seiten: Überblick" (S. 15)
- "Dynamos: Überblick" (S. 19)

Grundlegende Informationen und Anforderungen

PowerPrompts Developer Studio ist ein Entwicklungswerkzeug für Anwendungen. Als PowerPrompts-Anwendungsentwickler sollten Sie über Kenntnisse in den folgenden Bereichen verfügen

- HTML
- JavaScript
- SQL (wenn Sie Dynamos verwenden möchten)
- Logische Datenbanken (Datenbankdefinitionen aus Impromptu)
- Webserver

Unterschiede zwischen PowerPrompts 6.0 und Series 7

Es gibt einige wichtige Unterschiede zwischen PowerPrompts 6.0 und Series 7:

- Die Skriptsprache in Version 7.0 ist ECMAScript (allgemein als JavaScript bezeichnet).
- Wenn Sie bisher FormatUserVar verwendet haben, verwenden Sie stattdessen den folgenden Code:

  ```
  <%"'"+ App.Variables("StateList").Join("','") + "'"%>
  ```
- Alle Indizes sind nun 0-basiert. In Version 6.0 waren sie 1-basiert. Der folgende Index stammt beispielsweise aus PowerPrompts 6.0:

  ```
  <%GetDataCol(1)%>
  ```

  In PowerPrompts Series 7 wurde er folgendermaßen ersetzt:

  ```
  <%=rs.Fields(0)%>
  ```
- Funktionsaufrufe können in Version 7.0 verschachtelt sein.
- Das Dialogfeld *Skript-Manager* ist in Series 7 nicht mehr vorhanden. Skript-Bedingungen wurden im Dialogfeld *Skript-Editor* durch IF-Anweisungen ersetzt, z. B.:

  ```
  if ( GetUserVarAsString("AbsatzZiel") == "J" )
  ```
- Berichtsmethoden erfordern eine vorangestellte GetReport()-Methode, wie z. B. GetReport().AddDataItem.
- JavaScript unterscheidet zwischen Groß- und Kleinschreibung. Verwenden Sie beispielsweise App.RunDynamo und nicht aPP.rUNdYNAMO.

Aktualisieren von PowerPrompts 6.0-Anwendungen auf 7,1

Ihre PowerPrompts 6.0-Anwendungen werden von PowerPrompts so angepasst, dass sie unter PowerPrompts 7,1 ausgeführt werden können. PowerPrompts ändert keine clientseitigen JavaScript-Funktionen zu serverseitigen oder umgekehrt.

PowerPrompts 7.1 erstellt Anwendungen der Version 7.1, kann aber auch solche der Version 6.0 und 7.1 lesen und ausführen.

So aktualisieren Sie eine Anwendung der Version 6.0:

1. Starten Sie *PowerPrompts* über das Startmenü.
2. Klicken Sie im Menü *Datei* auf *Öffnen*.
3. Wählen Sie aus dem 6.0-Anwendungsordner die Anwendungsdatei (.xxm) und klicken Sie auf *Öffnen*.

   Die Anwendung wird in PowerPrompts Developer Studio geöffnet.

4. Klicken Sie im Menü *Datei* auf *Speichern*.
   Wenn Sie eine PowerPrompts 6.0-Anwendung in PowerPrompts 7,1 öffnen, wird eine Sicherungsdatei erstellt. An den Namen der Sicherungsdatei wird die Zahl 60 angehängt. Wenn Sie eine PowerPrompts 6.0 Anwendung mit dem Namen „Beispiel.xxm" öffnen und speichern, heißt die entsprechende Sicherungsdatei „Beispiel60.xxm".

Die Anwendung wird als PowerPrompts 7,1-Anwendung gespeichert.

Verwenden von PowerPrompts 6.0-Anwendungen mit Impromptu Web Reports

Für Impromptu Web Reports 6.0 entwickelte und dort publizierte PowerPrompts-Anwendungen funktionieren unverändert und ohne Neupublizierung auch weiterhin in Impromptu Web Reports 7,1. Impromptu Web Reports 7,1 öffnet die PowerPrompts 6.0-Anwendung und konvertiert sie bei jeder Ausführung in eine PowerPrompts 7,1-Anwendung. Wenn Sie die Zeit für die Konvertierung sparen möchten, öffnen Sie die Anwendungen in PowerPrompts Developer Studio 7,1, speichern sie und publizieren sie erneut in Impromptu Web Reports 7,1.

Integrieren von PowerPrompts Security und Access Manager

Einführung

Beim Ausführen einer Anwendung kann es erforderlich sein, dass PowerPrompts mit einer Datenbank kommuniziert. Zu diesem Zweck muss PowerPrompts die vollständige Verbindungszeichenkette und die Sicherheitseinstellungen bereitstellen. PowerPrompts verfügt über zwei Optionen zur Bereitstellung dieser Informationen: eingebettete Sicherheitsinformationen und die Access Manager-Integration.

Verbindungszeichenkette

Die Verbindungszeichenkette ist die von UDA für das Öffnen einer Datenbankverbindung verwendete Information. Die Zeichenkette enthält Informationen, wie die Datenbankkennung, den Datenbanktyp (beispielsweise Oracle, ODBC, MS SQL) und ob Benutzerkennung und Kennwort erforderlich sind. UDA kann dann festlegen, welche Datenbank geöffnet werden soll und welche Informationen von UDA an die Datenbank übergeben werden sollen.

Die Verbindungszeichenkette einer Datenbank wird nicht direkt in die PowerPrompts-Anwendung oder eine der HTML-Dateien der Anwendung eingebettet. Stattdessen wird die Datenbank durch einen logischen Namen identifiziert, und die Verbindungszeichenkette kann entweder in der Datei Cognos.ini oder in Access Manager gespeichert werden.

Die Datei Cognos.ini wird von Cognos-Produkten vor Series 7 verwendet. Die Datei speichert Datenbank-Verbindungszeichenketten. Eine Datenbank wird mit ihrem logischen Namen und einer von diesem durch ein Gleichheitszeichen (=) getrennten Verbindungszeichenkette aufgelistet. Benutzerkennung und Kennwort werden nicht in der Datei Cognos.ini gespeichert. Sie müssen von der Anwendung auf andere Weise bereitgestellt werden (in einem Skript oder durch Eingabe).

Verbindungszeichenketten werden in Access Manager als Eigenschaft einer Datenquelle gespeichert. Administratoren können durch logische Namen identifizierte, benutzerunabhängige Datenquellenobjekte erstellen (identisch mit Datenquellen in Cognos.ini). Der Administrator muss eine Verbindungszeichenkette für jede einzelne Datenquelle angeben. Für jede Datenquelle kann der Administrator eine oder mehrere Datenquellenanmeldungen erstellen. Diese Anmeldungen werden dann einem oder mehreren Access Manager-Benutzern zugewiesen.

Beim Herstellen einer Verbindung mit einer Datenbank versucht PowerPrompts, die Verbindungszeichenkette von beiden Orten abzurufen. Zunächst wird die Datenquelle von Access Manager abgefragt. Der Benutzer muss in der Lage sein, sich bei Access Manager zu authentifizieren, und die Datenquelle muss in seiner Verweisliste aufgeführt sein. Wenn diese Bedingungen nicht erfüllt sind, sucht PowerPrompts in Cognos.ini nach der logischen Datenbank. Wenn die Datenbank aufgeführt ist, verwendet PowerPrompts diese Verbindungszeichenkette. Wenn die Datenquelle an keinem der Orte gefunden werden kann, wird im Browser eine Fehlermeldung ausgegeben.

Benutzerkennung und Kennwort

Benutzerkennung und Kennwort werden anders behandelt als Verbindungszeichenketten, da sie den Zugriff auf die Datenbank steuern. Sie können nicht unverschlüsselt an einem allgemein zugänglichen Ort (wie der Datei Cognos.ini) gespeichert werden.

Die Benutzerkennung und das Kennwort für eine Datenbank können direkt in die Anwendung eingebettet werden, entweder in die Anwendungsdatei (.xxm), wenn der Zugriff auf die Datenquelle mittels eines Dynamos erfolgt, oder in eine HTML-Seite, wenn auf die Datenquelle mittels eines Datensatzobjekts zugegriffen wird.

Für Dynamos werden Benutzerkennungen und Kennwörter auf der Seite *Dynamo Datenquelle* des Dynamo-Assistenten festgelegt. Sie können die Benutzerkennung und das Kennwort für einen vorhandenen Dynamo im Dialogfeld *Dynamo Datenquelle* ändern. Aktivieren Sie das Kontrollkästchen *Verwenden Sie folgende Authentifizierungs-Information*, und geben Sie die Benutzerkennung und das Kennwort ein.

Um eine Benutzerkennung und ein Kennwort für ein Datensatzobjekt festzulegen, müssen Autoren der Recordset.Open-Methode einen 3. und 4. Parameter hinzufügen. Diese Parameter stehen für die Benutzerkennung bzw. das Kennwort.

Wenn für einen Dynamo oder ein Datensatzobjekt eine Benutzerkennung und ein Kennwort festgelegt wurden, verwendet PowerPrompts ausschließlich diese Werte. PowerPrompts versucht nicht, sich bei Access Manager zu authentifizieren. Wenn die Benutzerkennung und das Kennwort nicht in dem Dynamo oder dem Datensatzobjekt eingebettet ist, versucht PowerPrompts, diese Informationen von Access Manager abzurufen (der Benutzer muss in der Lage sein, sich zu authentifizieren, und er muss Zugriff auf die Anmeldeinformationen für diese Datenquelle haben).

Benutzerkennung und Kennwort sind nicht immer obligatorisch (beispielsweise in der Beispieldatenbank FREIZEIT). PowerPrompts versucht, eine Verbindung zur Datenbank herzustellen, selbst wenn keine Sicherheitsinformationen verfügbar sind.

Wenn eine Benutzerkennung und ein Kennwort erforderlich, diese jedoch fehlerhaft oder nicht verfügbar sind, wird im Browser eine Fehlermeldung ausgegeben.

Metadaten

Anmeldeinformationen und eine Verbindungszeichenkette sind erforderlich, um Metadaten von einer geschützten Datenbank abzurufen. PowerPrompts verhält sich in gleicher Weise wie beim Abrufen regulärer Daten aus einer Datenbank.

Einstellungen für Initialisierungsdateien

PowerPrompts 7.0 besitzt eine ini-Datei file (*xxxpowerprompts.ini)*, die diese Einträge enthält.

ServerIdleLifeTime

Legt die maximale Anzahl von Sekunden fest, während der sich ein neuer DataAccess-Prozess im Leerlauf befinden kann. Jeder DataAccess-Prozess im Leerlauf, der diesen Wert überschreitet, wird zwecks Einsparung von Systemressourcen geschlossen. Der Standardwert ist 900 Sekunden (15 Minuten).

Port

Legt die Port-Nummer fest, die der PowerPrompts DataAccess-Server verwendet. Der Standardwert ist 2425.

DataServersLimit

Legt die maximale Anzahl neuer DataAccess-Prozesse fest, die zeitgleich mit der Verarbeitung von Datenanforderungen ablaufen können. Wenn der Wert 0 ist, gibt es keine Einschränkung, d. h., es werden so viele Prozesse erstellt, wie erforderlich sind. Wenn der Wert eine positive Ganzzahl ist, übersteigt die Anzahl der DataAccess-Prozesse niemals diesen Wert plus 1. Wenn Sie den Wert beispielsweise auf 5 setzen, laufen nicht mehr als 6 Prozesse gleichzeitig ab. Der Standardwert beträgt 0.

Hinzufügen von virtuellen Verzeichnissen für Abbildungen

Wenn Sie auf den HTML-Seiten Ihrer PowerPrompts-Anwendung Grafiken anzeigen möchten, können Sie ein virtuelles Verzeichnis für Abbildungen angeben.

1. Erstellen Sie das folgende virtuelle Verzeichnis auf Ihrem Web-Server.

Alias	Verzeichnis	Zugriff
/Grafiken	Cognos\cer*n*\webcontent\Bilder	Lesezugriff

2. Verschieben Sie Ihre Abbildungen in den Ordner \webcontent\Grafiken.
3. Fügen Sie Ihren HTML-Seiten Abbildungen hinzu, indem Sie das Quellattribut des Abbildungs-Tags verwenden, beispielsweise
   ```
   <IMG SRC="/Grafiken/Datei1.gif">
   ```
4. Überprüfen Sie, ob die PowerPrompts-Anwendung die Abbildungen anzeigt.

Starten von PowerPrompts

Beschreibung

PowerPrompts wird mit der Administrator-Version von Impromptu ausgeliefert.

Weitere Informationen über das Installieren von Impromptu erhalten Sie im Installationshandbuch.

Schritte

- Starten Sie *PowerPrompts* über das Startmenü.

PowerPrompts Developer Studio wird mit einem leeren Arbeitsbereich geöffnet. Eine PowerPrompts-Anwendung wird erstellt, indem Sie dem PowerPrompts-Arbeitsbereich Seiten hinzufügen und Verknüpfungen zwischen diesen Seiten festlegen.

Verwandte Themen

- "Erstellen einer PowerPrompts-Anwendung" (S. 7)
- "Einrichten des HTML-Editors und der HTML-Vorlage" (S. 11)
- "Erstellen einer neuen Anwendung" (S. 12)

Einrichten des HTML-Editors und der HTML-Vorlage

Beschreibung

Richten Sie den Standard-HTML-Editor ein, den Sie verwenden möchten. Wenn Sie sich gut mit HTML auskennen, können Sie beispielsweise Editor als Standard-HTML-Editor festlegen.

Verwenden Sie für die Formatierung von PowerPrompts-Seiten reguläre HTML-Tags. Eine Seite kann alle Formatierungen enthalten, die Sie unter Verwendung von HTML erstellen können. Wenn Sie eine Vorlage festlegen, sind alle der Anwendung hinzugefügten Seiten Kopien der Vorlagendatei (mit Ausnahme der Fehler- und der Schlussseite). Sie können dann entweder zusätzliche Steuerelemente hinzufügen oder nicht gewünschte Elemente entfernen. Auf diese Weise erhalten Ihre Seiten ein einheitliches Aussehen, und der erforderliche Formatierungsaufwand wird reduziert.

Hinweise

- Die HTML-Vorlage wirkt sich nicht auf Seiten aus, die Sie gegebenenfalls in eine Anwendung importieren.
- Sie können aber auch jeder Seite mithilfe der #include-Anweisung den Inhalt einer JavaScript-Datei (.js) hinzufügen. Ein Beispiel finden Sie im Handbuch *Entdecken Sie PowerPrompts*.

Schritte

1. Klicken Sie im Menü *Extras* auf *Optionen*.
2. Klicken Sie auf die Registerkarte *Dateipfade*.
3. Geben Sie im Feld *HTML-Editor* den HTML-Texteditor an, den Sie für die Bearbeitung der Seiten Ihrer Anwendung verwenden möchten.
   Dieser Editor wird immer verwendet, wenn Sie eine HTML-Seite aus PowerPrompts Developer Studio heraus bearbeiten.
4. Legen Sie im Feld *HTML-Vorlagendatei* die HTML-Vorlagendatei für diese Anwendung fest, und klicken Sie auf *OK*.
   Nachdem Sie eine HTML-Vorlagendatei festgelegt haben, sind alle der Anwendung hinzugefügten Seiten Kopien der Vorlagendatei (mit Ausnahme der Fehler- und der Schlussseite).

Verwandte Themen

- "Erstellen einer PowerPrompts-Anwendung" (S. 7)
- "Erstellen einer neuen Anwendung" (S. 12)
- "Importieren einer Seite" (S. 14)
- "Hinzufügen von Codes zu Seiten mithilfe einer JavaScript-Datei" (S. 12)

Hinzufügen von Codes zu Seiten mithilfe einer JavaScript-Datei

Anstatt eine HTML-Vorlage festzulegen, können Sie die #include-Anweisung verwenden, um den Seiten Ihrer Anwendung JavaScript- und HTML-Codes aus einer JavaScript-Datei (.js) hinzuzufügen. Sie können beispielsweise allen Ihren Seiten eine gemeinsame Kopfzeile oder ein Firmenlogo als Grafik hinzufügen. Die betreffenden Elemente entnehmen Sie der JavaScript-Datei.

Das Lernprogramm *Entdecken Sie PowerPrompts* verwendet für ein einheitliches Erscheinungsbild dieser Anwendung die Datei Kopfzeile.js.

Schritte

1. Klicken Sie im Menü *Extras* auf *HTML-Editor*.
   Der HTML-Editor wird angezeigt.
2. Geben Sie den gemeinsamen Code ein, den Sie auf allen Seiten Ihrer Anwendung einfügen möchten.
   Der folgende Code fügt beispielsweise die Datei Kopfzeile.js ein:
   ```
   <%#include "Kopfzeile.js"%>
   ```
3. Speichern Sie die Datei. Stellen Sie sicher, dass die Dateierweiterung .js ist.
4. Schließen Sie den HTML-Editor.
5. Verschieben Sie die .js-Datei in Ihren PowerPrompts-Anwendungsordner.

Weitere Informationen über die Verwendung der #include-Anweisung finden Sie unter "#include-Anweisung und andere Präprozessor-Anweisungen" (S. 40).

Verwandte Themen

- "Einrichten des HTML-Editors und der HTML-Vorlage" (S. 11)

Erstellen einer neuen Anwendung

Beschreibung

Strukturieren Sie Ihre HTML-Seiten im PowerPrompts Developer Studio-Arbeitsbereich.

Jede neue PowerPrompts-Anwendung wird erstellt mit einer:

- *Startseite*, der ersten Seite, die Berichtskonsumenten angezeigt wird
- *Schlussseite*, die das auf der Auswahl des Berichtskonsumenten basierende Skript initialisiert. Alle Navigationspfade durch die PowerPrompts-Anwendung müssen auf der *Schlussseite* enden, die keine weiterführende Verknüpfung enthalten darf.
- *Fehlerseite*, die beim Auftreten eines Fehlers in der PowerPrompts-Anwendung angezeigt wird. Auf der *Fehlerseite* können keine Verknüpfungen angelegt werden.

Der Pfad zwischen der *Startseite* und der *Schlussseite* kann sich aus beliebig vielen Seiten zusammensetzen. Weitere Informationen über das Erstellen von Navigationspfaden finden Sie unter "Verknüpfen von Seiten: Überblick" (S. 15).

Schritte

1. Klicken Sie im Menü *Datei* auf *Neu*.
   Das Dialogfeld *Neue Anwendung* wird angezeigt.
2. Geben Sie in das Feld *Anwendungsname* den gewünschten Namen ein.
   Das Feld *Pfad* legt automatisch im Standardanwendungsordner einen Unterordner fest. Der Unterordner wird mit demselben Namen wie Ihre PowerPrompts-Anwendung erstellt. Der Unterordner wird beim Speichern der Anwendung eingerichtet.
   Die in der Anwendung verwendete Anwendungsdatei (.xxm) und die verwendeten HTML-Seiten (.htm) werden in diesem Ordner gespeichert. Die HTML-Seiten dürfen nicht aus dem Anwendungsordner entfernt werden.
3. Übernehmen Sie den Pfad im Feld *Pfad* oder geben Sie einen anderen an, und klicken Sie auf *OK*.
4. Klicken Sie auf *Ausführen*, um zu testen, ob PowerPrompts und der Web-Server ordnungsgemäß funktionieren.
5. Wenn in Ihrem Browser die Startseite angezeigt wird, schließen Sie den Browser, und kehren Sie zu PowerPrompts Developer Studio zurück.

Hinweise

- Sie können Seiten für eine Anwendung überall im Arbeitsbereich einfügen. Ebenso können Sie sie jederzeit verschieben, ohne dass die Verknüpfungen verloren gehen. Sie werden außerdem automatisch als HTML-Seiten im Anwendungsordner gespeichert.
- Das Dialogfeld *Neue Anwendung* enthält das Feld *Auszuführender Bericht*. Bevor Sie Ihre PowerPrompts-Anwendung nicht fertiggestellt haben, müssen Sie keinen Bericht für die Ausführung auswählen. Weitere Informationen erhalten Sie unter "Auswählen eines Berichts zum Testen der Anwendung" (S. 24).

Verwandte Themen

- "Hinzufügen einer Seite" (S. 14)
- "Erstellen einer PowerPrompts-Anwendung" (S. 7)
- "Dynamos: Überblick" (S. 19)
- "Verknüpfen von Seiten: Überblick" (S. 15)
- "Einrichten des HTML-Editors und der HTML-Vorlage" (S. 11)
- "Skript: Überblick" (S. 23)

Formatieren der Start-, Schluss- und Fehlerseiten

Wenn Sie eine HTML-Vorlage einrichten, wird die *Startseite* unter Verwendung dieser Vorlage erstellt. Wenn Sie keine HTML-Vorlage einrichten, enthält die *Startseite* lediglich eine Weiterleitungsschaltfläche, mit deren Hilfe der Benutzer zur nächsten Seite wechseln kann. Die *Fehlerseite* und die *Schlussseite* werden nicht mithilfe einer HTML-Vorlage erstellt. Die *Fehlerseite* führt eine PowerPrompts-Methode aus, die dem Berichtskonsumenten eine Fehlermeldung anzeigt. Die *Schlussseite* wird mithilfe eines HTML-Fomulars erstellt, das das Skript mit dem Bericht ausführt. Ein Beispiel, wie Sie diesen Seiten zusätzliche Formatierungen hinzufügen können, finden Sie im Handbuch *Entdecken Sie PowerPrompts*.

Sie können eine neue *Startseite* einrichten. Die *Schluss-* und die *Fehlerseite* können Sie lediglich umbenennen, nicht jedoch bearbeiten.

Alternativ haben Sie die Möglichkeit, jeder Seite mithilfe der #include-Anweisung den Inhalt einer JavaScript-Datei (.js) hinzuzufügen. Ein Beispiel finden Sie im Handbuch *Entdecken Sie PowerPrompts*.

Verwandte Themen

- "Erstellen einer PowerPrompts-Anwendung" (S. 7)
- "Einrichten des HTML-Editors und der HTML-Vorlage" (S. 11)

Hinzufügen einer Seite

Beschreibung

Jede Seite, die Sie der Anwendung hinzufügen, verkörpert eine einzelne HTML-Seite, die Berichtskonsumenten eine Eingabeaufforderung anzeigt. Sie können auf einer HTML-Seite beliebig viele Eingabeaufforderungen einfügen.

Sie können einer PowerPrompts-Anwendung beliebig viele HTML-Seiten hinzufügen.

Schritte

1. Klicken Sie auf *Seite einfügen*.
2. Klicken Sie auf den Arbeitsbereich, in den Sie die neue Seite einfügen möchten.
   Die Seite wird mit dem gewählten Namen eingefügt.
3. Geben Sie für die neue Seite einen eindeutigen Namen ein, und drücken Sie die Eingabetaste.

Verwandte Themen

- "Erstellen einer neuen Anwendung" (S. 12)
- "Löschen einer Seite" (S. 15)
- "Bearbeiten einer Seite" (S. 15)
- "Importieren einer Seite" (S. 14)
- "Verknüpfen von Seiten: Überblick" (S. 15)

Importieren einer Seite

Beschreibung

Anstatt eine leere Seite oder eine Kopie der HTML-Vorlage einzufügen, können Sie eine vorhandene HTML-Seite importieren, die bereits über die gewünschten Steuerelemente oder die gewünschte Formatierung verfügt.

Tipp: Sie können auch Seiten aus dem Windows Explorer durch Kopieren und Einfügen oder Ziehen in den Arbeitsbereich importieren.

Schritte

1. Klicken Sie im Menü *Datei* auf *HTML-Dateien importieren*.
2. Wählen Sie die gewünschte HTML-Datei, und klicken Sie auf *Öffnen*.
   Die Seite wird im PowerPrompts-Arbeitsbereich angezeigt, und eine Kopie der HTML-Datei wird im Anwendungsordner abgelegt.

Verwandte Themen

- "Hinzufügen einer Seite" (S. 14)
- "Erstellen einer neuen Anwendung" (S. 12)
- "Löschen einer Seite" (S. 15)
- "Bearbeiten einer Seite" (S. 15)
- "Verknüpfen von Seiten: Überblick" (S. 15)

Bearbeiten einer Seite

Beschreibung

Die Steuerelemente, Tabellen, Farben und Hintergründe der HTML-Seiten können Sie mithilfe eines HTML-Editors hinzufügen.

Ein Beispiel, wie Sie einer Seite HTML-Steuerelemente hinzufügen, finden Sie im Handbuch *Entdecken Sie PowerPrompts*.

Schritte

1. Klicken Sie im Arbeitsbereich auf die zu bearbeitende Seite.
2. Klicken Sie im Menü *Extras* auf *HTML-Editor*.
   Die Seite wird im Standard-HTML-Editor angezeigt.
   Hinweis: Sie können zum Bearbeiten auch mit der rechten Maustaste auf eine Seite klicken und anschließend auf *HTML bearbeiten* klicken.

Weitere Informationen über HTML finden Sie in der Dokumentation Ihres HTML-Editors.

Verwandte Themen

- "Hinzufügen einer Seite" (S. 14)
- "Erstellen einer neuen Anwendung" (S. 12)
- "Löschen einer Seite" (S. 15)
- "Importieren einer Seite" (S. 14)
- "Verknüpfen von Seiten: Überblick" (S. 15)
- "Einrichten des HTML-Editors und der HTML-Vorlage" (S. 11)

Löschen einer Seite

Beschreibung

Sie können nicht mehr benötigte Seiten Ihrer PowerPompts-Anwendung löschen. Wenn Sie eine Seite löschen, werden auch alle Verknüpfungen zu und von dieser Seite gelöscht.

Schritte

1. Klicken Sie im Arbeitsbereich auf eine Seite.

2. Klicken Sie auf *Löschen*.
   PowerPrompts löscht die Seite und alle Verknüpfungen von und zu der Seite.

Tipp

- Wenn Sie mehrere Seiten gleichzeitig löschen möchten, markieren Sie mehrere Seiten, und klicken Sie anschließend auf *Löschen*.

Verwandte Themen

- "Hinzufügen einer Seite" (S. 14)
- "Erstellen einer neuen Anwendung" (S. 12)
- "Bearbeiten einer Seite" (S. 15)
- "Importieren einer Seite" (S. 14)
- "Verknüpfen von Seiten: Überblick" (S. 15)

Verknüpfen von Seiten: Überblick

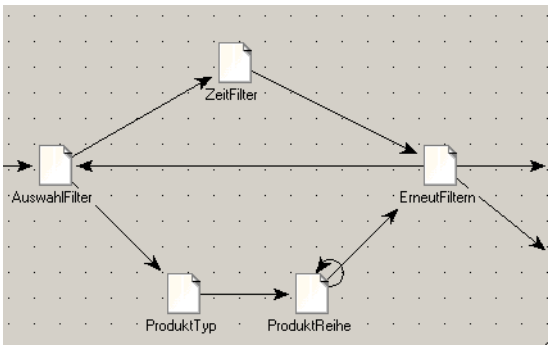
Die von Ihnen hinzugefügten Verknüpfungen stehen für die Navigationspfade, denen Berichtskonsumenten folgen können, wenn sie einen mit dieser PowerPrompts-Anwendung verbundenen Bericht anzeigen.

Sie können so viele Verknüpfungen verwenden, wie sich Seiten in der Anwendung befinden, vorausgesetzt, dass von jeder dieser Seiten die *Schlussseite* erreicht werden kann. Sie können Navigationspfade erstellen, die:

- die *Start-* und die *Schlussseite* direkt, d. h. ohne Zweige und Schleifen, miteinander verbinden
- die gleichen Seiten mehrfach anzeigen (über eine Schleife verfügen)
- von einzelnen Seiten abzweigen
- mit sich selbst verknüpfte Seiten besitzen

Ein Beispiel eines Navigationspfades mit Schleife ist der Pfad von *AuswahlFilter* über *ZeitFilter* zu ErneutFiltern und zurück zu *AuswahlFilter*.

Ein Beispiel für einen verzweigten Navigationspfad ist *AuswahlFilter* zu *ZeitFilter* oder *ProduktTyp*.



Dialogfeld „Seiteneigenschaften"

Verwenden Sie das Dialogfeld *Seiteneigenschaften* zum Bearbeiten und Löschen von Verknüpfungen. Wenn Sie eine Verknüpfung hinzufügen, verbinden Sie zwei Seiten im Arbeitsbereich. Wenn Sie für die Seite, auf der die Verknüpfung beginnt, das Dialogfeld *Seiteneigenschaften* öffnen, können Sie die Eigenschaften dieser Verknüpfung anzeigen lassen. Jede von dieser Seite ausgehende Verknüpfung wird auf der Registerkarte *Verknüpfungen* aufgeführt. Jede Verknüpfung besitzt einen Eintrag in den Spalten *Bedingung* und *Seite*.

Während der Ausführung wird die Liste der Verknüpfungen für jede Seite von oben nach unten ausgewertet. Die erste Verknüpfung, deren Bedingung „wahr" ist, führt zur nächsten zu öffnenden Seite. Verknüpfungen unterhalb dieser Verknüpfung werden nicht mehr ausgewertet. Wenn in der Spalte *Bedingung* der Eintrag <STANDARD> angezeigt wird, ist diese Verknüpfung die Standardverknüpfung. Die Bedingung der Standardverknüpfung ist immer „wahr". Stellen Sie sicher, dass die Standardverknüpfung die letzte Verknüpfung auf der Registerkarte *Verknüpfungen* ist. Wenn Sie andere Verknüpfungen unterhalb der Standardverknüpfung platzieren, werden diese niemals überprüft. Weitere Informationen über Verknüpfungsbedingungen finden Sie unter "Hinzufügen einer Verknüpfungsbedingung" (S. 18).

Wenn Sie eine Verknüpfung hinzufügen, wird in der Spalte *Bedingung* der Eintrag <STANDARD> angezeigt, da noch keine Verknüpfungsbedingung hinzugefügt wurde. Wenn Sie eine vorhandene Verknüpfungsbedingung löschen, wird für diese Verknüpfung in der Spalte *Bedingung* erneut <STANDARD> eingetragen.

Tipp

- Verwenden Sie den Nach unten-Pfeil, um die Standardverknüpfung (<STANDARD>) zu verschieben.

Verwandte Themen

- "Hinzufügen einer Verknüpfung durch Ziehen" (S. 17)
- "Hinzufügen einer Verknüpfung mithilfe des Dialogfelds" (S. 17)
- "Hinzufügen einer Verknüpfungsbedingung" (S. 18)
- "Erstellen einer neuen Anwendung" (S. 12)
- "Erstellen einer PowerPrompts-Anwendung" (S. 7)
- "Löschen einer Verknüpfung" (S. 19)
- "Skript: Überblick" (S. 23)

Hinzufügen einer Verknüpfung durch Ziehen

Beschreibung

Fügen Sie eine Verknüpfung hinzu, indem Sie sie durch Ziehen zwischen zwei Seiten im Arbeitsbereich erstellen. Diese Methode beschleunigt die Erstellung von PowerPrompts-Anwendungen.

Sie können auch auf der Registerkarte *Verknüpfungen* des Dialogfelds *Seiteneigenschaften* eine Verknüpfung hinzufügen.

Schritte

1. Klicken Sie auf *Verknüpfung erstellen*.
2. Ziehen Sie eine Linie zwischen den zwei Seiten.

Verwandte Themen

- "Hinzufügen einer Verknüpfung mithilfe des Dialogfelds" (S. 17)
- "Hinzufügen einer Verknüpfungsbedingung" (S. 18)
- "Löschen einer Verknüpfung" (S. 19)
- "Verknüpfen von Seiten: Überblick" (S. 15)

Hinzufügen einer Verknüpfung mithilfe des Dialogfelds

Beschreibung

Wenn eine Seite mehrere Verknüpfungen besitzt, müssen Sie die Verknüpfungen im Dialogfeld *Seiteneigenschaften* erstellen, da Sie Verknüpfungsbedingungen hinzufügen müssen, damit Berichtskonsumenten die gewünschte Seite wählen können. Geben Sie nach dem Hinzufügen der Verknüpfungsbedingung die Seite an, zu der gewechselt werden soll, wenn die Bedingung zutrifft.

Sie können eine Verknüpfung auch durch Ziehen zwischen zwei Seiten hinzufügen.

Schritte

1. Klicken Sie auf *Seite auswählen*.
2. Klicken Sie mit der rechten Maustaste auf die Seite, auf der die Verknüpfung beginnen soll, und klicken Sie anschließend auf *Eigenschaften*.
3. Klicken Sie auf die Registerkarte *Verknüpfungen*.
4. Klicken Sie auf die Schaltfläche *Neu*.
   Eine neue Verknüpfungszeile wird angezeigt.
5. Wenn Sie für diese Verknüpfung eine Bedingung benötigen, geben Sie diese an.
6. Wählen Sie in der Spalte *Seite* die Seite, zu der Sie eine Verknüpfung erstellen möchten, und klicken Sie auf *OK*.

Verwandte Themen

- "Hinzufügen einer Verknüpfung durch Ziehen" (S. 17)
- "Hinzufügen einer Verknüpfungsbedingung" (S. 18)
- "Löschen einer Verknüpfung" (S. 19)
- "Verknüpfen von Seiten: Überblick" (S. 15)

Hinzufügen einer Verknüpfungsbedingung

Beschreibung

Wenn eine Seite mehrere Verknüpfungen besitzt, müssen Sie Verknüpfungsbedingungen hinzufügen, damit Berichtskonsumenten die gewünschte Seite auswählen können. Nachdem Sie im Dialogfeld *Seiteneigenschaften* die Verknüpfungsbedingungen festgelegt haben, können Sie der Seite den HTML-Code hinzufügen, damit Berichtskonsumenten die Bedingungen angeben können, nach denen die gewünschte Seite ausgewählt wird. Während der Ausführung wird die Liste der Verknüpfungen für jede Seite von oben nach unten ausgewertet. Die erste Verknüpfung, deren Bedingung „wahr" ist, führt zur nächsten zu öffnenden Seite. Bedingungen basieren auf den während der Vervollständigung der Anwendung festgelegten Form-Variablen. So wird beispielsweise die Bedingung FilterTyp = "NachZeit" aus dem Lernprogramm *Entdecken Sie PowerPrompts* als „wahr" bewertet, wenn Sie auf der Seite *AuswahlFilter* auf die Optionsschaltfläche *Nach Zeiträumen filtern* klicken. Verknüpfungen unterhalb dieser Verknüpfung werden nicht mehr ausgewertet. Achten Sie daher darauf, dass sich Ihre Verknüpfungen in der richtigen Reihenfolge befinden.

Schritte

1. Klicken Sie auf *Seite auswählen*.
2. Klicken Sie mit der rechten Maustaste auf die Seite, für die Sie eine Verknüpfungsbedingung hinzufügen möchten, und klicken Sie anschließend auf *Eigenschaften*.
3. Klicken Sie auf die Registerkarte *Verknüpfungen*.
4. Klicken Sie auf die Spalte *Bedingung* und anschließend auf die Schaltfläche *Bedingung bearbeiten*.

   Das Dialogfeld *Verknüpfungsbedingung* wird angezeigt.
5. Klicken Sie auf die Schaltfläche *Neu*.

   PowerPrompts sucht alle Namenwerte (beispielsweise NAME="FilterTyp") auf der ausgewählten Seite und zeigt sie im Feld *Name* an.
6. Geben Sie im Feld *Name* das in Ihrem Ausdruck auszuwertende Element ein. Im Ausdruck Preis < 10 ist der Name beispielsweise Preis.
7. Klicken Sie im Feld *Operator* auf den relationalen Operator, den Sie in Ihrem Ausdruck verwenden möchten. Im Ausdruck „Preis < 10" ist der Operator beispielsweise 'kleiner als' (<).
8. Geben Sie im Feld *Wert* den in Ihrem Ausdruck zu verwendenden Wert ein. Im Ausdruck „Preis < 10" ist der Wert beispielsweise 10.
9. Klicken Sie zweimal auf *OK*.

Tipp

- Wenn Sie in Ihrer Verknüpfungsbedingung mehrere Ausdrücke verwenden möchten, fügen Sie mehrere Ausdrücke hinzu, und verbinden Sie sie mithilfe der logischen Operatoren AND oder OR.

Verwandte Themen

- "Hinzufügen einer Verknüpfung durch Ziehen" (S. 17)
- "Hinzufügen einer Verknüpfung mithilfe des Dialogfelds" (S. 17)
- "Löschen einer Verknüpfung" (S. 19)
- "Verknüpfen von Seiten: Überblick" (S. 15)

Löschen einer Verknüpfung

Beschreibung

Beim Erstellen Ihrer PowerPrompts-Anwendung stoßen Sie möglicherweise auf nicht mehr benötigte Verknüpfungen. Löschen Sie diese Verknüpfungen mithilfe der Registerkarte *Verknüpfungen* im Dialogfeld *Seiteneigenschaften*. Durch das Löschen einer nicht mehr benötigten Seite werden auch alle Verknüpfungen von und zu dieser Seite entfernt.

Schritte

1. Klicken Sie auf *Seite auswählen*.
2. Klicken Sie mit der rechten Maustaste auf die Seite, auf der die zu löschende Verknüpfung beginnt, und klicken Sie anschließend auf *Eigenschaften*.
3. Klicken Sie auf die Registerkarte *Verknüpfungen*.
4. Klicken Sie auf die zu löschende Verknüpfung.
   Die Zielseite dieser Verknüpfung wird in der Spalte *Seite* aufgeführt.
5. Klicken Sie auf *Löschen* und anschließend auf *OK*.
   Die Verknüpfung wird aus dem Arbeitsbereich entfernt.

Verwandte Themen

- "Hinzufügen einer Verknüpfung durch Ziehen" (S. 17)
- "Hinzufügen einer Verknüpfung mithilfe des Dialogfelds" (S. 17)
- "Hinzufügen einer Verknüpfungsbedingung" (S. 18)
- "Löschen einer Seite" (S. 15)
- "Verknüpfen von Seiten: Überblick" (S. 15)

Dynamos: Überblick

Ein Dynamo wird erstellt, um das Ergebnis einer SQL-Abfrage in einer logischen Datenbank entsprechend der Definition im Database Definition Manager zu formatieren. So können Sie beispielsweise einen Dynamo erstellen, der Länder aus einer Datenquelle abruft. Dabei kann ein und derselbe Dynamo unterschiedliche Länder enthalten, je nach dem, was in der Datenquelle enthalten ist oder wie diese aktualisiert wird. Die Liste wird angezeigt, wenn der Berichtsbenutzer die HTML-Seite, die die App.RunDynamo-Methode enthält, anzeigt.

Für einen Dynamo legen sie Folgendes fest:

- Datenquelle: die abzufragende logische Datenquelle.
- SQL: der die Abfrage definierende SQL-Code.
- Definition: der den Ergebnissatz formatierende JavaScript-Code.

Bevor Sie eine Datenbank abfragen können, müssen Sie bereits über eine in der Datei Cognos.ini oder in Access Manager definierte logische Datenbank verfügen. Wenn Sie eine ODBC-Verbindung verwenden möchten, erstellen Sie eine logische Datenbank für die ODBC-Verbindung, oder verwenden Sie direkt ADO mithilfe der Server.CreateObject JavaScript-Methode. Alte PowerPrompts 6.0 Anwendungen, die ODBC-Verbindungen verwenden, funktionieren auch in PowerPrompts Series 7.

Weitere Informationen über die Erstellung logischer Datenbanken finden Sie in der Hilfe für Datenbankdefinition-Manager.

Dialogfeld „Dynamo-Manager"

Im Dialogfeld *Dynamo-Manager* können Sie Dynamos hinzufügen, bearbeiten und löschen. Jeder von Ihnen erstellte Dynamo besitzt je einen Eintrag in den Spalten *Name* und *Datenquelle* und gegebenenfalls auch in der Spalte *Beschreibung*. Sie können Dynamos aus diesem Dialogfeld kopieren und sie auf HTML-Seiten einfügen.

Hinweis: Wenn Sie versuchen, eine Datenquelle in Windows 2000 oder Windows 2000 Professional hinzuzufügen, können Sie eventuell eine Fehlermeldung erhalten. Wenn dies auftritt, suchen Sie in Ihrem lokalen Laufwerk die Datei MSSTDFMT.DLL. Stellen Sie sicher, dass sich diese Datei im Pfad \system32 Ihres Computers befindet. Wenn Sie diese Datei nicht finden können, wenden Sie sich an Ihren System-Administrator.

Tipp: Anstatt einen Dynamo zu erstellen, können Sie auf der HTML-Seite einen JavaScript-Code schreiben, um Werte einer SQL-Abfrage zurückzugeben. Ein Beispiel zur Verwendung von JavaScript anstelle eines Dynamos finden Sie im Handbuch *Entdecken Sie PowerPrompts*.

Verwandte Themen

- "Erstellen eines Dynamos mithilfe des Assistenten" (S. 20)
- "Manuelles Erstellen eines Dynamos" (S. 21)
- "Erstellen einer neuen Anwendung" (S. 12)
- "Erstellen einer PowerPrompts-Anwendung" (S. 7)

Erstellen eines Dynamos mithilfe des Assistenten

Beschreibung

Verwenden Sie den *Dynamo-Assistenten*, wenn Sie schrittweise durch den Erstellungsprozess eines Dynamos, der eine Datenquelle verwendet, geführt werden möchten. Die manuelle Erstellung eines Dynamos ist in der Regel komplizierter.

Schritte

1. Klicken Sie auf *Dynamo-Manager*.
2. Klicken Sie auf *Dynamo-Assistent*, und folgen Sie den Anweisungen bis zur Seite *Datenquelle auswählen*.
3. Wenn Sie eine Benutzerkennung und ein Kennwort vergeben möchten, aktivieren Sie das Kontrollkästchen *Verwenden Sie folgende Authentifizierungs-Information*, und geben Sie die gewünschte Benutzerkennung und das gewünschte Kennwort ein.
4. Wählen Sie im Feld *Logische Datenbank* die gewünschte logische Datenbank, und klicken Sie auf *Weiter*.
5. Wenn ausschließlich eindeutige Werte abgerufen werden sollen, aktivieren Sie das Kontrollkästchen *Nur eindeutige Werte verwenden*.
6. Wählen Sie die Tabelle, die die Daten für den Dynamo bereitstellen soll, und klicken Sie auf *Weiter*.

7. Wenn Sie eine Liste, ein Optionsfeld oder ein Kontrollkästchen erstellen, geben Sie den Namen, den Wert und die Bezeichnung des HTML-Steuerelements an, und klicken Sie auf *Weiter*.

Steuerelement	Verwendung
Name des HTML-Steuerelements	Der in Bedingungen verwendete Name. Im folgenden HTML-Code trägt das HTML-Steuerelement den Namen FilterTyp. <INPUT type="radio" NAME="FilterTyp" VALUE="NachZeit">>
Wert des HTML-Steuerelements	Der in Bedingungen verwendete Wert. Im folgenden HTML-Code trägt das HTML-Steuerelement den Namen FilterTyp. <INPUT type="radio" NAME="FilterTyp" VALUE="NachZeit">
Bezeichnung des Steuerelements	Die tatsächlich von diesem Dynamo angezeigten Daten. Diese können dem Wert (VALUE) entsprechen, müssen es aber nicht.

   Wenn Sie eine Tabelle erstellen, müssen Sie in der Spalte *HTML Title* die in der Tabelle anzuzeigenden Spalten auswählen. Verwenden Sie die Schaltflächen *Nach oben* und *Nach unten*, um die Anzeigereihenfolge festzulegen, und klicken Sie auf *Weiter*.
8. Überprüfen Sie im Feld *Auswertung*, ob die Informationen richtig sind, und klicken Sie auf *Beenden*.
9. Klicken Sie mit der rechten Maustaste auf den Dynamo und anschließend auf *Kopieren*.
10. Fügen Sie den Dynamo in den Abschnitt <form> der HTML-Seite ein, und speichern Sie die Seite.
11. Schließen Sie das Dialogfeld *Dynamo-Manager*.

Verwandte Themen

- "Manuelles Erstellen eines Dynamos" (S. 21)
- "Dynamos: Überblick" (S. 19)

Manuelles Erstellen eines Dynamos

Beschreibung

Dynamos können auch manuell erstellt werden. Im Gegensatz zur assistentengestützten Dynamo-Erstellung können Sie hierbei anspruchsvollere SQL-Codes schreiben.

Tipp: Sie können auch Dynamos erstellen, die keine logische Datenbank verwenden. Wenn Sie beispielsweise in Ihrer Anwendung zahlreiche Seiten besitzen, können Sie einen Dynamo erstellen, der auf jeder Seite einen Titel einfügt. Wenn Sie den Titel später ändern möchten, müssen Sie nur die Dynamo-Definition ändern, um alle Seiten mit diesem Titel an die Änderung anzupassen. Erstellen Sie zu diesem Zweck einen Dynamo, der keine Datenquelle verwendet, und fügen Sie im Dialogfeld *Dynamo-Definition* den Titel und die HTML-Formatierung ein. Fügen Sie auf jeder Seite, auf der der Titel angezeigt werden soll, den Tag `<%=App.RunDynamo("Dynamo Name")%>` ein.

Vorgehensweise zum Festlegen der Dynamo-Datenquelle

1. Klicken Sie auf *Dynamo-Manager*.
2. Klicken Sie auf *Neu*.
   Eine neue Dynamozeile wird angezeigt.
3. Geben Sie in der Spalte *Name* den gewünschten Namen ein.
4. Klicken Sie auf *Bearbeiten* und anschließend auf *Datenquelle*.
   Das Dialogfeld *Dynamo Datenquelle* wird angezeigt.
5. Klicken Sie auf *Keine Datenverbindung*, wenn der Dynamo keine Daten abrufen soll.
6. Wenn Daten dynamisch abgerufen werden sollen, klicken Sie auf *Verwenden Sie folgende logische Datenbank*, und wählen Sie aus der Dropdownliste den Namen der logischen Datenbank.
7. Wenn Sie eine Benutzerkennung und ein Kennwort vergeben möchten, aktivieren Sie das Kontrollkästchen *Verwenden Sie folgende Authentifizierungs-Information*, und geben Sie die gewünschte Benutzerkennung und das gewünschte Kennwort ein.
8. Wenn die Anzahl der abgerufenen Datensätze beschränkt werden soll, aktivieren Sie das Kontrollkästchen *Dynamo beschränken auf*, geben Sie im Feld *Datensätze* die gewünschte Zahl ein, und klicken Sie anschließend auf *OK*.
   Das Dialogfeld *Dynamo-Manager* wird erneut angezeigt.

Vorgehensweise zum Festlegen von Dynamo-SQL und Definition

1. Klicken Sie auf *Bearbeiten* und anschließend auf *SQL*.
   Das Dialogfeld *Dynamo-SQL* wird angezeigt.
2. Geben Sie den SQL-Code ein, um die Daten von der Datenquelle abzurufen. Folgendermaßen weisen Sie Ihre SQL der SQL-Eigenschaft eines Dynamo-Objekts zu:
   ```
   Dynamo.SQL = ...
   ```
   Das folgende Beispiel gibt Produktreihencode und ProduktReihe zurück und sortiert die Werte nach ProduktReihe.
   ```
   Dynamo.SQL = 'Select Produktreihencode, ProduktReihe from ProduktReihe order by
   ProduktReihe';
   ```
   Hinweis: Sie können auch Ordner und Spalten aus dem Metadatenbereich in den SQL-Bereich ziehen. Dies erspart Ihnen die manuelle Eingabe.
3. Klicken Sie auf *OK*.
   Das Dialogfeld *Dynamo-Manager* wird erneut angezeigt.
4. Klicken Sie auf *Bearbeiten* und anschließend auf *Definition*.
   Das Dialogfeld *Dynamo-Definition* wird angezeigt.
5. Geben Sie den gewünschten JavaScript-Code ein.
   Das folgende Beispiel verwendet Produktreihencode als HTML-Ausgabewerte in einer Dropdownliste, zeigt jedoch ProduktReihe an.
   ```
   var sOutput = '<p><select name="ProduktReihe">';

   while (!rs.EOF)
   {
          sOutput = sOutput + "<option value='" + rs.Fields("Produktreihencode") + "'>"
   + rs.Fields("ProduktReihe") + "</option>";
          rs.MoveNext();
   }

   sOutput += '</select></p>';
   return sOutput;
   ```
   Die rs.EOF-Anweisung überprüft das Ende des Datensatzes. Durch die gesamte Anweisung „while (!rs.EOF)" werden die Anweisungen in den geschweiften Klammern so lange ausgeführt, bis das Ende des Datensatzes erreicht ist.
6. Klicken Sie auf *OK*, um das Dialogfeld *Dynamo-Definition* zu schließen.
7. Klicken Sie mit der rechten Maustaste auf den Dynamo und anschließend auf *Kopieren*.
8. Schließen Sie das Dialogfeld *Dynamo-Manager*.

9. Fügen Sie den Dynamo in den Abschnitt <form> der HTML-Seite ein, und speichern Sie die Seite.

Verwandte Themen

- "Erstellen eines Dynamos mithilfe des Assistenten" (S. 20)
- "Dynamos: Überblick" (S. 19)

Skript: Überblick

Durch ein Skript wird der mit der PowerPrompts-Anwendung verbundene Bericht auf der Grundlage der vom Berichtskonsumenten vorgenommenen Auswahl geändert. Die Skriptsprache für PowerPrompts Series ist JavaScript. In PowerPrompts 6.0 war die Skriptsprache eine Kombination aus PowerPrompts-Methoden und Impromptu Report-Methoden.

Dialogfeld „Skript-Editor"

Verwenden Sie das Dialogfeld *Skript-Editor*, um in Ihrer PowerPrompts-Anwendung Skripts zu erstellen, zu bearbeiten und zu löschen.

Die Skripts können nur durch die Ausführung der Anwendung überprüft werden. Möglicherweise vorhandene Skriptfehler werden ausschließlich im Web-Browser angezeigt.

Verwandte Themen

- "Hinzufügen eines Skripts" (S. 23)
- "Erstellen einer neuen Anwendung" (S. 12)
- "Erstellen einer PowerPrompts-Anwendung" (S. 7)

Hinzufügen eines Skripts

Beschreibung

Über das Dialogfeld *Skript-Editor* können Sie ein Skript hinzufügen. Ein Skript besteht aus JavaScript-Code, der Änderungen an dem mit der PowerPrompts-Anwendung verbundenen Bericht vornimmt. Wenn beispielsweise ein Berichtskonsument ein bestimmtes Land auswählt, kann ein Berichtsfilter erstellt werden, der nur Datensätze für dieses Land anzeigt. Ein Skriptbeispiel finden Sie im Lernprogramm *Entdecken Sie PowerPrompts*.

Schritte

1. Klicken Sie auf *Skript-Editor*.
2. Erstellen Sie das Skript mit den gewünschten Methoden und Konstanten, und klicken Sie anschließend auf *OK*.

   Das folgende Skript filtert beispielsweise den Bericht auf der Grundlage des vom Berichtskonsumenten eingegebenen Wertes für Absatzländercode.

   ```
   if (App.Variables("LänderCd"))
   {
          GetQuery().AndFilterBy("[\\Land\\Absatzländercode] = " +
   App.Variables("LänderCd"));
   }
   ```

   Tipp: Zum Einfügen von Impromptu Report-Methoden und -Konstanten klicken Sie mit der rechten Maustaste in das Feld *Skript-Editor* und wählen dann das gewünschte Element aus. Wenn Sie Tabellen- und Spaltenverweise einfügen möchten, öffnen Sie Impromptu, kopieren Sie die betreffenden Elemente aus dem Dialogfeld *Ordner*, und fügen Sie sie in das Dialogfeld *Skript-Editor* ein.
3. Schließen Sie das Dialogfeld *Skript-Editor*.

Wenn der Berichtskonsument eine Auswahl vornimmt, die diesen Wert setzt, wird das Skript vor dem Öffnen des Berichts ausgeführt.

Verwandte Themen

- "Skript: Überblick" (S. 23)

Testen und Bereitstellen der Anwendung: Überblick

Zum Testen Ihrer PowerPrompts-Anwendung müssen Sie zunächst einen Impromptu-Bericht auswählen. Anschließend können Sie die Anwendung überprüfen und testen und sämtliche Skriptfehler beheben. Funktioniert die Anwendung reibungslos, steht einer Verteilung nichts mehr im Weg.

Verwandte Themen:

- "Beheben von Skriptfehlern" (S. 25)
- "Benachrichtigen des Impromptu Web Reports-Administrators" (S. 27)
- "Auswählen eines Berichts zum Testen der Anwendung" (S. 24)
- "Testen der Anwendung" (S. 25)
- "Überprüfen der Anwendung" (S. 24)
- "Anzeigen des generierten Skripts" (S. 26)

Auswählen eines Berichts zum Testen der Anwendung

Beschreibung

Eine PowerPrompts-Anwendung kann mit unterschiedlichen Berichten arbeiten, vorausgesetzt, die erstellten Skripte sind auf den Bericht anwendbar. Wenn Sie die Skripte testen möchten, müssen Sie einen Bericht angeben.

Schritte

1. Klicken Sie im Menü *Datei* auf *Eigenschaften*.
   Die Registerkarte *Allgemein* des Dialogfelds *Anwendungseigenschaften* wird angezeigt.
2. Klicken Sie auf *Durchsuchen*, wählen Sie den Bericht, und klicken Sie anschließend auf *Öffnen*.
   Der ausgewählte Bericht wird im Feld *Auszuführender Bericht* angezeigt.
3. Klicken Sie auf *OK*.

Verwandte Themen

- "Beheben von Skriptfehlern" (S. 25)
- "Benachrichtigen des Impromptu Web Reports-Administrators" (S. 27)
- "Testen der Anwendung" (S. 25)
- "Überprüfen der Anwendung" (S. 24)

Überprüfen der Anwendung

Beschreibung

Jedes Mal, wenn Sie eine Anwendung speichern, überprüft PowerPrompts Developer Studio die Anwendung und zeigt gegebenenfalls Fehler- und Warnmeldungen an. Sie können den Pfad und die Verknüpfungen in Ihrer Anwendung jedoch auch mit dem Befehl *Anwendung überprüfen* testen, ohne die Anwendung speichern zu müssen.

Hinweis: Wenn Sie ein Hilfethema zu einem Anwendungsfehler anzeigen möchten, klicken Sie mit der rechten Maustaste auf den Fehler und anschließend auf *Hilfe anzeigen*.

Schritte

1. Öffnen Sie die zu überprüfende Anwendung.
2. Wählen Sie im Menü *Datei* den Befehl *Überprüfen*.
   Das Dialogfeld *Anwendung überprüfen* wird geöffnet und zeigt eine Liste sämtlicher Navigationsfehler, fehlender HTML-Seiten und überzähliger Verknüpfungen an.
   Hinweis: Sie können das Dialogfeld *Anwendung überprüfen* während der Fehlerbehebung geöffnet lassen.

3. Beheben Sie alle Fehler, und überprüfen Sie die Anwendung dann erneut.

Nachdem sämtliche Anwendungsfehler behoben wurden, wird dem Benutzer in einer Meldung mitgeteilt, dass die Anwendung nun einsatzfähig ist.

Verwandte Themen

- "Beheben von Skriptfehlern" (S. 25)
- "Benachrichtigen des Impromptu Web Reports-Administrators" (S. 27)
- "Auswählen eines Berichts zum Testen der Anwendung" (S. 24)
- "Testen der Anwendung" (S. 25)
- "Anzeigen des generierten Skripts" (S. 26)

Testen der Anwendung

Beschreibung

Nach der Fertigstellung Ihrer Anwendung und der Überprüfung auf Anwendungsfehler können Sie die PowerPrompts-Anwendung testen. Die Anwendung und der resultierende Bericht werden im Browser wie bei einem Berichtskonsumenten angezeigt.

Damit Sie die Anwendung testen können, muss auf Ihrem Computer die folgende Software installiert sein:

- ein Web-Server
- ein Web-Browser
- ein Anzeigeprogramm für PDF-Dateien, wie Adobe Acrobat Reader

Außerdem müssen Sie Ihre Katalog- und Verbindungsinformationen angeben. In Abhängigkeit von Ihren Sicherheitseinstellungen müssen Sie möglicherweise Ihre Benutzerklasse, Ihr Benutzerklassenkennwort, Ihre Datenbankkennung oder Ihr Datenbankkennwort eingeben.

In welchem Fall schlägt eine Verbindung von PowerPrompts mit einem Katalog fehl?

Wenn die Anmeldeinformationen des aktuellen Katalogs fehlerhaft oder unvollständig sind oder bisher nicht festgelegt wurden.

Schritte

1. Starten Sie Ihren Web-Server.
2. Klicken Sie auf *Ausführen*.
   Im Browser wird die *Startseite* der Anwendung angezeigt.
3. Folgen Sie den von Ihnen erstellten Anweisungen und legen Sie die Informationen fest, die in dem Bericht angezeigt werden sollen.
4. Klicken Sie auf der letzten Seite auf *Bericht ausführen*.
   Wenn keine Skriptfehler aufgetreten sind und PowerPrompts eine Verbindung zum Katalog herstellen kann, wird der Bericht als PDF-Datei im Browser angezeigt.

Verwandte Themen

- "Beheben von Skriptfehlern" (S. 25)
- "Benachrichtigen des Impromptu Web Reports-Administrators" (S. 27)
- "Auswählen eines Berichts zum Testen der Anwendung" (S. 24)
- "Überprüfen der Anwendung" (S. 24)
- "Anzeigen des generierten Skripts" (S. 26)

Beheben von Skriptfehlern

Beschreibung

Beim Testen Ihrer Anwendung könnten sich Skriptfehler herausstellen. Diese werden im Browser angezeigt, nachdem die Anwendung mit dem Bericht ausgeführt wurde.

Tipp: Überprüfen Sie, ob die Anführungszeichen in den Skripten ordnungsgemäß gesetzt wurden. Überprüfen Sie auch die richtige Schreibweise von Impromptu-Katalogordnern und -Spaltennamen. Weitere Informationen erhalten Sie unter "Impromptu-Ausdrücke in PowerPrompts" (S. 99).

Schritte

1. Führen Sie die PowerPrompts-Anwendung aus.
2. Lesen Sie im Browser alle Fehlerbeschreibungen.
3. Beheben Sie die Fehler, und führen Sie die Anwendung erneut aus.

Verwandte Themen

- "Auswählen eines Berichts zum Testen der Anwendung" (S. 24)
- "Benachrichtigen des Impromptu Web Reports-Administrators" (S. 27)
- "Testen der Anwendung" (S. 25)
- "Überprüfen der Anwendung" (S. 24)
- "Anzeigen des generierten Skripts" (S. 26)

Anzeigen des generierten Skripts

Beschreibung

Sie können auch ein auf den Bericht anzuwendendes Skript generieren, das auf der Auswahl basiert, die Sie während der Ausführung der Anwendung vorgenommen haben. Dies ist eine Alternativmethode, mit der sie die PowerPrompts-Anwendung testen und etwaige Fehler im Skript beheben können.

Schritte

1. Klicken Sie im Menü *Datei* auf *Eigenschaften*.
2. Löschen Sie das Feld *Auszuführender Bericht*, und klicken Sie auf *OK*.
   Dieser Bericht ist nicht mehr mit der PowerPrompts-Anwendung verbunden.
3. Klicken Sie im Menü *Datei* auf *Speichern*.
   Das Dialogfeld *Anwendung überprüfen* wird angezeigt und informiert Sie, dass mit der Anwendung kein Bericht verbunden ist.
4. Klicken Sie auf *Speichern*.
5. Klicken Sie auf *Ausführen*.
6. Wählen Sie die Informationen für den Bericht aus.
7. Klicken Sie auf der *Schlussseite* auf *Bericht ausführen*.

Im Browser wird ein Skript mit den ausgewählten Werten angezeigt. Wenn Sie für die Skripte Kommentare eingefügt haben, sind diese ebenfalls in der Liste enthalten.

Tipps

- Wenn das Dialogfeld *Anwendung überprüfen* aus dem Menü *Extras* nicht angezeigt werden soll, klicken Sie auf *Optionen*, und deaktivieren Sie das Kontrollkästchen *Beim Überprüfen der Anwendung Warnungen ignorieren*.
- Wenn Sie vor dem Testen Ihrer PowerPrompts-Anwendung nicht zum Speichern der Änderungen aufgefordert werden möchten, klicken Sie im Menü *Extras* auf *Optionen* und anschließend auf *Änderungen speichern*.

Verwandte Themen

- "Beheben von Skriptfehlern" (S. 25)
- "Benachrichtigen des Impromptu Web Reports-Administrators" (S. 27)
- "Auswählen eines Berichts zum Testen der Anwendung" (S. 24)
- "Testen der Anwendung" (S. 25)
- "Überprüfen der Anwendung" (S. 24)

Benachrichtigen des Impromptu Web Reports-Administrators

Benachrichtigen Sie den Impromptu Web Reports-Administrator, wenn Sie Ihre PowerPrompts-Anwendung fertiggestellt und getestet haben. Wenn Berichtskonsumenten einen mit einer PowerPrompts-Anwendung verbundenen Bericht öffnen, wird die Anwendung in ihrem Browser ausgeführt, und Sie werden zur Eingabe der erforderlichen Informationen aufgefordert.

Weitere Informationen über erforderliche Hinweise für den Berichts-Administrator nach der Fertigstellung der PowerPrompts-Anwendung finden Sie im *Administrations-Handbuch* für Impromptu Web Reports.

Verwandte Themen

- "Beheben von Skriptfehlern" (S. 25)
- "Auswählen eines Berichts zum Testen der Anwendung" (S. 24)
- "Testen der Anwendung" (S. 25)
- "Überprüfen der Anwendung" (S. 24)
- "Anzeigen des generierten Skripts" (S. 26)

Kapitel 2: Verfahrensweisen

Filtern eines Berichts nach mehreren Werten

Beschreibung

Sie können Berichte nach mehreren Werten filtern. Beispielsweise können Sie sich die Verkaufszahlen von Deutschland und Frankreich gemeinsam anzeigen lassen.

Schritte

1. Fügen Sie Ihrer Seite oder Ihrem Dynamo das HTML-Steuerelement <select> mit dem multiple-Attribut hinzu, beispielsweise

   ```
   <select name="Länder" multiple>...</select>
   ```

2. Erstellen Sie das Skript, um den Filter mithilfe der GetQuery().AndFilterBy()-Methode hinzuzufügen. Für Daten ohne Zeichenketten ist das Skript einfach, beispielsweise

   ```
   GetQuery().AndFilterBy("[LänderCd] in (" + App.Variables("LänderCd") + ")");
   ```

   Hinweis: Verwenden Sie in Skripts „in" anstelle von „=", da „in" mehrere Werte verarbeitet.

   Wenn es sich bei den Daten um eine Zeichenkette handelt, schließen Sie jeden einzelnen Wert in einfache Anführungszeichen ein. Verwenden Sie dazu die Join-Eigenschaft des StringList-Objekts, beispielsweise

   ```
   GetQuery().AndFilterBy("[Stadt] in ('" +  App.Variables("Stadt").Join("','") +
   "')");
   ```

Beispiel

Dieses Beispiel fügt auf einer Seite eine Dropdownliste für Länder hinzu und filtert den Bericht anschließend nach den gewählten Ländern.

HTML-Seite

```
<select name="LänderCd" multiple>
<%
var rs = new Recordset();
rs.Open("Select Absatzländercode,Land from Land", "AUFUmsatz");
while(!rs.EOF)
{
    Response.Write("<option value='" + rs.Fields("Absatzländercode") + "'>" +
rs.Fields("Land") + "</option>");
    Response.Write("<br>");
    rs.MoveNext();
}
rs.Close();
%>
</select>
```

Skript

```
if (App.Variables("Länder"))
{
    GetQuery().AndFilterBy("[\\Land\\Absatzländercode] in (" + App.Variables("LänderCd")
+ ")");
}
```

Verwandte Themen

- "Hinzufügen von Impromptu-Benutzerklassenfiltern zu Auswahllisten" (S. 31)
- "Erstellen einer kaskadierenden Auswahlliste auf einer einzelnen Seite" (S. 32)
- "Erstellen einer Auswertung der vom Benutzer gewählten Optionen" (S. 30)
- "Erstellen einer Auswahlliste mit Direkteingabe" (S. 32)

Speichern der von Benutzern gewählten Optionen zwischen Sitzungen

Beschreibung

Sie können von Benutzern ausgewählte Optionen speichern und sie bei erneuter Ausführung der Anwendung wiederverwenden.

Speichern Sie die von den Benutzern ausgewählten Optionen, indem Sie sie mithilfe des PowerPrompts Connection-Objekts oder ADO (unter Windows) in eine Datenbank schreiben.

Schritte

1. Schreiben Sie die App.Variables-Werte in die Datenbank am Kopf jeder HTML-Seite Ihrer Anwendung. Eine einfache Datenbank wäre eine Tabelle mit drei Spalten.
2. Führen Sie jedes Mal, wenn eine Seite erzeugt wird, eine select-Abfrage durch, um zu bestimmen, ob in der Datenbank Werte für Eingaben auf dieser Seite vorhanden sind. Wenn dies der Fall ist, initialisieren Sie die Steuerelemente für die richtigen Werte.

Beispiel

Dieses Beispiel erstellt eine einfache Datenbank, die drei Werte speichert: 1) Benutzername, 2) Variablenname und 3) Wert.

```
<%
var conn = new Connection();
for (var sName in App.Variables)
{
    conn.Execute("INSERT INTO BenutzerLand (BenutzerName, VariableName, Wert) VALUES (
'"+ Benutzer.BenutzerName + "','" + sName + "','" + App.Variables(sName) + "')");
}
%>
```

Verwandte Themen

- "Erstellen verschiedener Designs für unterschiedliche Upfront-Themen" (S. 35)
- "Erstellen einer Auswertung der vom Benutzer gewählten Optionen" (S. 30)

Erstellen einer Auswertung der vom Benutzer gewählten Optionen

Beschreibung

Sie können eine Auswertung der Optionen erstellen, die vom Benutzer bereits ausgewählt wurden.

Schritte

- Fügen Sie auf der Schlussseite der PowerPrompts-Anwendung JavaScript-Code hinzu, der die vom Benutzer festgelegten Variablen anzeigt. Der folgende Code zeigt in einer HTML-Tabelle das vom Benutzer für die Ansicht ausgewählte Land an.

```
<br><b>Tabelle der ausgewählten Variablen</b>
<table>
<table border=2 bgcolor=#80FFFF>
<tr width="15%">
<th>Variablenname</th>
<th>Wert</th>
</tr>
<%
if (App.Variables("LänderCd"))
{
    var rs = new Recordset;
    rs.Open("SELECT Land FROM Land WHERE Absatzländercode=" +
App.Variables("LänderCd"),"AUFUmsatz");
    if (!rs.EOF)
    {
        Response.Write("<tr><td>Land</td><td>" + rs.Fields("Land") + "</td></tr>");
    }
}
...
```

Hinweis: In Kapitel 2 des Lernprogramms *Entdecken Sie PowerPrompts* finden Sie ein vollständiges JavaScript-Beispiel, das die vom Benutzer ausgewählten Optionen anzeigt. Der Code für dieses Beispiel befindet sich auch in der Datei *pwp_html.txt*, die im Ordner *Installationsverzeichnis*\Beispiele\PowerPrompts gespeichert ist.

Verwandte Themen

- "Erstellen verschiedener Designs für unterschiedliche Upfront-Themen" (S. 35)
- "Speichern der von Benutzern gewählten Optionen zwischen Sitzungen" (S. 30)

Hinzufügen von Impromptu-Benutzerklassenfiltern zu Auswahllisten

Beschreibung

In Ihren Katalogen befinden sich möglicherweise vorhandene Benutzerklassenfilter. Diese Informationen können Sie in Ihren PowerPrompts-Anwendungen verwenden. Wenn ein Katalog beispielsweise eine Benutzerklasse für jedes Land besäße, könnten Sie die Daten in jedem Bericht auf der Grundlage der Benutzerklasse filtern, der der aktuelle Benutzer angehört. Wenn Sie daher eine PowerPrompts-Anwendung für einen Bericht aus diesem Katalog erstellen, können Sie eine Dropdownliste einfügen, die nur die Städte für die entsprechende Benutzerklasse enthält. Dies können Sie mit dem Query-Objekt implementieren.

Schritte

- Verwenden Sie das Query-Objekt, um SQL mit dem entsprechenden, auf der aktuellen Benutzerklasse basierenden Filter an das Recordset-Objekt zu senden.

Beispiel

Dieses Beispiel erstellt auf der Grundlage des Benutzerklassenfilters eine Dropdownliste mit Städten.

```
<%
var myQuery = new Query();
myQuery.AddColumn("[\\Land\\Stadt]","Stadt");
var sql = myQuery.SQL;
var rs = new Recordset();
rs.Open(sql,"VERTRIEBAUF");
Response.Write("<select name=Stadt>");
while (!rs.EOF)
{
    Response.Write("<option>" + rs.Fields(0) + "</option>");
    rs.MoveNext();
}
Response.Write("</select>");
%>
```

Verwandte Themen

- "Erstellen einer kaskadierenden Auswahlliste auf einer einzelnen Seite" (S. 32)
- "Erstellen einer Auswahlliste mit Direkteingabe" (S. 32)
- "Filtern eines Berichts nach mehreren Werten" (S. 29)

Erstellen einer Auswahlliste mit Direkteingabe

Beschreibung

Sie können eine Auswahlliste mit direkter Eingabe erstellen, indem Sie auf der HTML-Seite ein Texteingabe-Steuerelement hinzufügen. Nachdem PowerPrompts den Wert des Eingabesteuerelements abgerufen hat, kann der Wert entweder im Abschlussskript oder einem Impromptu-Ausdruck verwendet werden.

Schritte

1. Sie können eine Auswahlliste mit direkter Eingabe erstellen, indem Sie auf der HTML-Seite ein Texteingabe-Steuerelement hinzufügen.
2. Nachdem PowerPrompts den Wert des Eingabesteuerelements abgerufen hat, kann der Wert entweder im Abschlussskript oder einem Impromptu-Ausdruck verwendet werden.

Beispiel

Dieses Beispiel filtert den Abschlussbericht nach dem vom Benutzer eingegebenen Bestellcode.

HTML-Seite

```
<html>
<body>
<h1>Bestellcode eingeben</h1>
Geben Sie einen Bestellcode ein.<p>
<form method="post">
Bestellcode<br>
<input type="text" name="Bestellung">
<input type="submit" name="Eingeben">
</form>
</body>
</html>
```

Skript

```
GetQuery().AndFilterBy("[Bestellcode] = " + App.Variables("Bestellung"));
```

Verwandte Themen

- "Hinzufügen von Impromptu-Benutzerklassenfiltern zu Auswahllisten" (S. 31)
- "Erstellen einer kaskadierenden Auswahlliste auf einer einzelnen Seite" (S. 32)
- "Filtern eines Berichts nach mehreren Werten" (S. 29)

Erstellen einer kaskadierenden Auswahlliste auf einer einzelnen Seite

Beschreibung

Sie können zunächst in einer Auswahlliste eine Option auswählen und auf diese Weise den Inhalt der zweiten Auswahlliste auf ein und derselben Seite einschränken. So wäre es beispielsweise möglich, den Inhalt einer Städteauswahlliste von dem ausgewählten Land abhängig zu machen.

Schritte

1. Schreiben Sie den clientseitigen JavaScript-Code, um die beiden Auswahllisten zu erstellen.
2. Fügen Sie von der Seite mit den Auswahllisten zwei Verknüpfungen hinzu.
3. Legen Sie fest, dass die erste Verknüpfung die Bedingung btn=Next (Weiter) besitzt und auf die nächste Seite der Anwendung führt.

4. Legen Sie fest, dass die Standardverknüpfung auf die eigene Seite führt. Dies ist eine Verknüpfung mit Selbstverweis.

   Wenn Sie in der Anwendung eine Länderauswahl vornehmen, wird die Seite übermittelt und die entsprechende Städteliste erzeugt. Wenn Sie auf die Schaltfläche *Weiter* klicken, wird die nächste Seite angezeigt.

Beispiel

Dieses Beispiel filtert eine Städteliste entsprechend Ihrer Auswahl in einer Länderliste.

HTML-Seite

```
<html>
<body>
<h1>Kaskadierende Auswahlliste</h1>
<form method="post">

Land<br>
<select name="Land" OnChange="document.forms[0].submit()">
<%
// Länderliste abrufen
var rs = new Recordset();
rs.Open("SELECT Land FROM Land ORDER BY Land ","AUFUmsatz");

// Jedes Land in der Liste ausschreiben
var sSelectedText;
var sPreviousCountry = App.Variable("Land");
while (!rs.EOF)
{
    // Dieser Code trifft eine Vorauswahl für das zuvor ausgewählte Land
    if (sPreviousCountry == rs.Fields(0))
    {
        sSelectedText = " selected ";
    }
    else
    {
        sSelectedText = "";
    }

    Response.Writeln("<option " + sSelectedText + " >" + rs.Fields(0));
    rs.MoveNext();
}
%>
</select>
<p>
Stadt<br>
<select name="Stadt">
<%
// Städteliste basierend auf Land abrufen
var rs2 = new Recordset();
rs2.Open("SELECT Stadt FROM Land WHERE Land = '" + App.Variables("Land") + "' ORDER BY
Stadt","AUFUmsatz");
while (!rs2.EOF)
{
    Response.Writeln("<option>" + rs2.Fields(0));
    rs2.MoveNext();
}
%>
</select>
<p>
<input type=submit name=btn value=Weiter>
</form>
</body>
</html>
```

Verwandte Themen

- "Hinzufügen von Impromptu-Benutzerklassenfiltern zu Auswahllisten" (S. 31)
- "Erstellen einer Auswahlliste mit Direkteingabe" (S. 32)
- "Filtern eines Berichts nach mehreren Werten" (S. 29)

Verwenden von ADO aus PowerPrompts

Beschreibung

Sie können in PowerPrompts-Anwendungen Microsoft ActiveX Data Objects (ADO) verwenden, um auf ODBC-Daten zuzugreifen. Verwenden Sie die Server.CreateObject-Methode, um ein COM-Objekt zu instanziieren, und anschließend dessen Eigenschaften und Methoden.

Hinweis: JavaScript unterstützt nicht das Konzept der Standardeigenschaften und -methoden, daher müssen Sie jede ADO-Methode explizit aufrufen.

Schritte

1. Schreiben Sie den Code, um ein ADO-Objekt zu erstellen.
2. Fügen Sie diesen Code einer Ihrer HTML-Seiten hinzu, speichern Sie Ihre Änderungen, und führen Sie die Anwendung aus.

Beispiel

Dieses Beispiel erstellt ein ADO-Verbindungsobjekt, stellt eine Verbindung zu einer ODBC-Datenquelle her, füllt einen Datensatz und schreibt dann das erste Feld in Ihre HTML-Seite.

```
<%
var oCmd = Server.CreateObject("ADODB.Command");
oCmd.ActiveConnection = "DB_AUF";
oCmd.CommandText = "SELECT * FROM Land";
var oRS = oCmd.Execute();
while (!oRS.EOF)
{
    Response.Writeln(oRS.Fields("Land").Value + "<br>");
    oRS.MoveNext();
}
%>
```

Abrufen der Benutzerkennung und des Kennworts für die Datenbank von Access Manager

Beschreibung

Sie müssen die Informationen für die Datenbankkennung und das -kennwort nicht unmittelbar in Ihre PowerPrompts-Anwendung einfügen, sondern können sie von Access Manager abrufen. Zu diesem Zweck müssen Sie zunächst Access Manager ordnungsgemäß einrichten. Stellen Sie sicher, dass jeder Benutzer, der die PowerPrompts-Anwendung verwenden soll, Zugriff auf die erforderliche Datenbank besitzt und über die erforderlichen Anmeldeinformationen für jede Datenbank verfügt. Informationen dazu finden Sie in der Access Manager-Dokumentation.

Geben Sie anschließend beim Aufruf der Recordset.Open-Methode weder Benutzerkennung noch Kennwort ein.

Schritte

1. Richten Sie Access Manager ein.
2. Rufen Sie die Recordset.Open-Methode ohne Eingabe von Benutzerkennung oder Kennwort auf.
3. Deaktivieren Sie zum Abrufen von Benutzerkennung und Kennwort aus Access Manager das Kontrollkästchen *Verwenden Sie diese Authentifizierungs-Informationen*.

Beispiel

Dieses Beispiel überprüft in Access Manager, ob für den aktuellen Benutzer Datenbank-Anmeldeinformationen vorhanden sind. Wenn dies der Fall ist, werden diese Daten verwendet.

```
rs.Open("SELECT * FROM LAND","Datenbank");
```

Erstellen verschiedener Designs für unterschiedliche Upfront-Themen

Beschreibung

Upfront unterstützt unterschiedliche Themen, von denen jedes ein eigenes Design besitzen kann. Berichtskonsumenten können das gewünschte Design auswählen. Standardmäßig sehen die Themen in PowerPrompts-Anwendungen alle gleich aus. Sie können jedoch bei der Erstellung Ihrer Anwendungen festlegen, dass die Seiten in Abhängigkeit von den gewählten Themen jeweils unterschiedlich aussehen. Öffnen Sie zu diesem Zweck die Upfront.Theme-Eigenschaft, und ändern Sie die auf dieser Eigenschaft basierende Seite.

Schritte

- Erstellen Sie den Code, der Ihre HTML-Seiten auf der Grundlage des vom Benutzer gewählten Upfront-Themas ändert.

Beispiel

Dieses Beispiel erstellt zwei Themen mit den Namen „Corporate" und „Flashy".

HTML-Seite

```
<html>
<head>
<%
    switch (Upfront.Theme)
    {
        case "Corporate":
%>
    <link rel="stylesheet" type="text/css" href="/corp.css">
<%
        break;

        case "Flashy":
%>
    <link rel="stylesheet" type="text/css" href="/flashy.css">
<%
        break;
    }
%>
</head>
<body>
<%
    switch (Upfront.Theme)
    {
        case "Corporate":
%>
    <h1>Corporate-Thema</h1>
    Dies ist das Corporate-Thema mit dem Corporate-Design.
<%
        break;

        case "Flashy":
%>
    <h1>Flashy-Thema</h1>
    Dies ist das Flashy-Thema mit dem Flashy-Design.
<%
        break;
    }
%>

</body>
</html>
```

Erstellen einer einzelnen PowerPrompts-Anwendung für mehrere Sprachen

Sie können bei der Erstellung Ihrer PowerPrompts-Anwendung festlegen, dass der Text für jeden Benutzer in der entsprechenden Sprache angezeigt wird.

1. Erstellen Sie auf Ihrer HTML-Seite für jedes Wort oder jede Wortgruppe eine JavaScript-Variable, anstatt den Text auf der Seite einzufügen.
2. Verwenden Sie die Response.Write-Methode, um den Text der Variable an der gewünschten Position auf der Seite anzeigen zu lassen.
3. Fragen Sie im Kopfteil der Seite bei Upfront die aktuell verwendete Sprache ab, und setzen Sie jede Variable auf den entsprechenden Text.

Beispiel

Dieses Beispiel gibt Text auf der Grundlage der vom Benutzer gewählten Sprache zurück.

HTML-Seite

```
<html>
<head>
<%
// Sprache abfragen und speichern
var slang = Upfront.Language;
var sTitle;
var sCountry;
var SelectText;
if (sLang == "en")
{
    sTitle = "Welcome Page";
    sCountry = "Country";
    sSelectText = "Select a Country";
}
else if (slang == "de")
{
    sTitle = "Startseite";
    sCountry = "Land";
    sSelectText = "Land wählen";
}
else
{
    // Standardmäßig nur Englisch verwenden
    sTitle = "Welcome Page";
    sCountry = "Country";
    sSelectText = "Select a Country";
}
%>
</head>
<body>
<h1><%=sTitle%></h1>
<%=sSelectText%><p>
<%=sCountry%>
<select name=Country>
</select>
</body>
</html>
```

Kapitel 3: JavaScript in PowerPrompts

Sie können auf Ihren HTML-Seiten clientseitiges und serverseitiges JavaScript verwenden.

Clientseitiges JavaScript wird vom Browser ausgeführt. Informationen über die Syntaxregeln finden Sie in der Dokumentation für clientseitiges JavaScript. Serverseitiges JavaScript wird von PowerPrompts ausgeführt und dazu verwendet, sowohl die Inhalte einer Seite zu erzeugen als auch auf die PowerPrompts- und Impromptu Web Reports-Objekte zuzugreifen.

Im Folgenden finden Sie einige Regeln, die Ihnen bei der Erstellung von serverseitigem JavaScript-Code behilflich sein können.

Serverseitiges JavaScript

- Serverseitiges JavaScript ist vollständig ECMA-262-kompatibel.
- Sämtlicher serverseitiger JavaScript-Code muss in spitzen Klammern (<%...%>) eingeschlossen sein, damit er vom PowerPrompts-Server erkannt wird.
- Kommentare in serverseitigem JavaScript müssen durch die C++ (//)- oder C (/*...*/)-Formatkommentare gekennzeichnet sein.

  Beispiel:

```
<%
/* Führen Sie einen Dynamo aus, der eine Zahl zurückgibt, und weisen Sie diese einer
Zeichenkette zu */
var sReturn = App.RunDynamo("MeinDynamo2");
if (sReturn > 0)
{
    Response.Write("<b>Retouren:" + sReturn +"</b>");
}
%>
```

- JavaScript unterscheidet zwischen Groß- und Kleinschreibung. Verwenden Sie also nicht RESPONSE.write, sondern Response.Write.
- Ein Gleichheitszeichen (=) in spitzen Klammern (<%...%>) ist ein Kurzzeichen für Response.Write.

  Beispiel:

```
<%=Request.Variables%>
```

  ist identisch mit

```
<%Response.Write(Request.Variables)%>
```

Quelltabelle serverseitiger JavaScript-Objekte

Mit den folgenden serverseitigen Objekten können Sie Seiteninhalte erstellen und mit den PowerPrompts- und Impromptu Web Reports-Komponenten kommunizieren. Weitere Informationen über serverseitiges JavaScript finden Sie unter "JavaScript in PowerPrompts" (S. 37).

Objekt	Eigenschaften	Methoden
"App-Objekt" (S. 41)	"BackURL-Eigenschaft" (S. 42) "CurrentPage-Eigenschaft" (S. 43) "Errors-Eigenschaft" (S. 43)	"RunDynamo-Methode" (S. 46)

Objekt	Eigenschaften	Methoden
	"FinalURL-Eigenschaft" (S. 44)	
	"IsTestMode-Eigenschaft" (S. 44)	
	"Path-Eigenschaft" (S. 45)	
	"ReportScript-Eigenschaft" (S. 45)	
	"Variables-Eigenschaft" (S. 45)	
"Connection-Objekt" (S. 47)		"Execute-Methode" (S. 47)
"Field-Objekt" (S. 47)	"HTMLEncodedValue-Eigenschaft" (S. 48)	"Operator()-Methode (Feld)" (S. 52)
	"Index-Eigenschaft" (S. 48)	
	"Name-Eigenschaft" (S. 49)	
	"Type-Eigenschaft" (S. 50)	
	"Value-Eigenschaft" (S. 51)	
"Query-Objekt" (S. 52)	"SQL-Eigenschaft" (S. 54)	"AddColumn-Methode" (S. 54)
		"AndFilterBy-Methode" (S. 55)
		"AndSummaryFilterBy-Methode" (S. 55)
		"AssociateColumn-Methode" (S. 56)
		"GroupBy-Methode" (S. 56)
		"OrFilterBy-Methode" (S. 57)
		"OrSummaryFilterBy-Methode" (S. 57)
		"RemoveColumn-Methode" (S. 58)
		"SetDistinct-Methode" (S. 58)
		"SetPromptValue-Methode" (S. 59)
		"SortBy-Methode" (S. 59)
"Recordset-Objekt" (S. 60)	"CurrentRecordIndex-Eigenschaft" (S. 60)	"Close-Methode" (S. 63)

Objekt	Eigenschaften	Methoden
	"EOF-Eigenschaft" (S. 61)	"MoveNext-Methode" (S. 64)
	"Fields-Eigenschaft" (S. 62)	"Open-Methode" (S. 64)
	"MaxRecords-Eigenschaft" (S. 62)	
"Request-Objekt" (S. 66)	"Cookies-Eigenschaft" (S. 66)	
	"ServerVariables-Eigenschaft" (S. 66)	
	"Variables-Eigenschaft" (S. 45)	
"Response-Objekt" (S. 68)	"ContentType-Eigenschaft" (S. 68)	"AppendCookie-Methode" (S. 69)
		"AppendHeader-Methode" (S. 69)
		"Clear-Methode" (S. 69)
		"ClearContent-Methode" (S. 70)
		"ClearHeaders-Methode" (S. 70)
		"Redirect-Methode" (S. 70)
		"Write-Methode" (S. 70)
		"WriteFile-Methode" (S. 71)
		"Writeln-Methode" (S. 71)
"Server-Objekt" (S. 73)	"ScriptTimeout-Eigenschaft" (S. 73)	"CreateObject-Methode" (S. 74)
		"FormatNumber-Methode" (S. 74)
		"HTMLEncode-Methode" (S. 75)
		"URLDecode-Methode" (S. 75)
		"URLEncode-Methode" (S. 76)
"StringList-Objekt" (S. 76)	"Count-Eigenschaft" (S. 77)	"Contains-Methode" (S. 77)
		"Item-Methode" (S. 78)
		"Join-Methode" (S. 79)
		"Operator() Methode (StringList)" (S. 79)
		"toString-Methode" (S. 80)

Objekt	Eigenschaften	Methoden
"Upfront-Objekt" (S. 81)	"Language-Eigenschaft" (S. 81)	"ExecuteCommand-Methode" (S. 82)
	"Locale-Eigenschaft" (S. 82)	"GetPageFragment-Methode" (S. 83)
	"Theme-Eigenschaft" (S. 82)	
"User-Objekt" (S. 83)	"Description-Eigenschaft" (S. 83)	
	"Email-Eigenschaft" (S. 84)	
	"Telephone-Eigenschaft" (S. 84)	
	"UserClass-Eigenschaft" (S. 85)	
	"UserClasses-Eigenschaft" (S. 85)	
	"UserName-Eigenschaft" (S. 86)	

#include-Anweisung und andere Präprozessor-Anweisungen

#include-Anweisung

Mit der #include-Anweisung können Sie einen anderen JavaScript-Code auf der Seite einfügen. Sie können diese Anweisung dazu verwenden, den Code auf mehreren Seiten Ihrer Anwendung oder sogar in unterschiedlichen Anwendungen gemeinsam zu nutzen. Der angegebene Pfad ist immer relativ zum Verzeichnis der Anwendung. Verwenden Sie in Beispiel1 den relativen Pfad, wenn Sie einen gemeinsamen Pfad für gemeinsam genutzte JavaScript-Dateien auf einem Server angeben möchten.

Beispiel1

Dieses Beispiel erläutert die #include-Anweisung, wenn sich SaveState.js im selben Ordner wie die Anwendung befindet.

```
<%
#include "SaveState.js"
%>
```

Beispiel2

Dieses Beispiel zeigt die Verwendung der #include-Anweisung, wenn sich SaveState.js in einem Ordner CommonScripts befindet, der sich in dem Ordner unterhalb der Anwendung selbst befindet.

```
<%
#include "../CommonScripts/SaveState.js"
%>
```

Präprozessor-Anweisungen

Neben der #include-Anweisung können Sie weitere Präprozessor-Anweisungen verwenden:

- #define
- #ifdef
- #else
- #endif

Die #define-Anweisung wird zum Festlegen globaler Variablen verwendet, beispielsweise:

```
#define COMPANY_NAME Cognos
```

Die anderen drei werden dazu verwendet, bedingte define-Anweisungen zu erstellen.

Beispiel

Das folgende Beispiel zeigt eine bedingte define-Anweisung, die auf UNIX überprüft.

```
#ifdef _UNIX_
    Response.Write("Keine UNIX-Unterstützung");
#else
    var ocom = Server.CreateObject("ADODB.Connection");
#endif
```

Vordefinierte Konstanten

Es gibt zwei vordefinierte Konstanten, die Sie für den Test von Anwendungen verwenden können:

- TEST_MODE
- _UNIX_

TEST_MODE wird definiert, wenn Sie die PowerPrompts-Anwendung unter Verwendung eines Impromptu-Clients verwenden. _UNIX_ wird definiert, wenn ein UNIX-Server versucht, eine PowerPrompts-Anwendung auszuführen.

App-Objekt

Gibt den Status des aktuellen Benutzers der PowerPrompts-Anwendung zurück.

Dieses Objekt ist nicht erstellbar.

Eigenschaft	Beschreibung
BackURL-Eigenschaft	Gibt den URL an Upfront zurück. Ermöglicht die Rückkehr zu Upfront aus Ihrer PowerPrompts-Anwendung. Verwenden Sie diese Eigenschaft als Teil einer Aktion auf einem Formular.
CurrentPage-Eigenschaft	Gibt den Namen der aktuellen Seite zurück.
Errors-Eigenschaft	Gibt eine Liste von Fehlermeldungen für die aktuelle Server-Anforderung zurück.
FinalURL-Eigenschaft	Gibt den URL an Upfront zurück, nachdem eine PowerPrompts-Anwendung beendet wurde. Verwenden Sie diese Methode auf der Schlussseite als Teil einer Aktion auf einem Formular.
IsTestMode-Eigenschaft	Gibt zurück, ob die PowerPrompts-Anwendung im Test-Modus ausgeführt wird. Gibt „true" zurück, wenn die Anwendung von PowerPrompts Developer Studio aus ausgeführt wird, andernfalls „false". Verwenden Sie diese Eigenschaft, um Fehler in Ihrer Anwendung zu beheben.
Path-Eigenschaft	Gibt den Ordner der PowerPrompts-Anwendungsdatei (.xxm) zurück. Dies ist eine zweckmäßige Eigenschaft für die Response.WriteFile-Methode.
ReportScript-Eigenschaft	Gibt das Skript zurück, das an Impromptu gesendet wird. Diese Eigenschaft ist nur im Testmodus und auf der Schlussseite verfügbar. In allen anderen Fällen wird eine leere Zeichenkette zurückgegeben. Verwenden Sie diese Eigenschaft, um Fehler in Ihrer Anwendung zu beheben.
Variables-Eigenschaft	Gibt den Status der Anwendung zurück. Es handelt sich um eine Auflistung von Variablen.

Methode	Beschreibung
RunDynamo-Methode	Führt zur Erzeugung einer Zeichenkette den angegebenen Dynamo aus. Verwenden Sie diese Methode auf Ihren HTML-Seiten.

BackURL-Eigenschaft

Gibt den URL an Upfront zurück. Ermöglicht die Rückkehr zu Upfront aus Ihrer PowerPrompts-Anwendung. Diese Eigenschaft funktioniert nur von unter Impromptu Web Reports eingesetzten Anwendungen. Diese Eigenschaft funktioniert nicht im Testmodus, da kein Upfront vorhanden ist, zu dem der Benutzer zurückkehren kann.

Verwenden Sie diese Eigenschaft als Teil einer Aktion auf einem Formular.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

Durch das folgende Beispiel kehrt der Benutzer zu Upfront zurück, wenn er auf die Schaltfläche *Abbrechen* klickt.

```
<form method="post" action="<%=App.BackURL%>">
<input type="submit" value="Abbrechen">
</form>
```

Gilt für folgende Objekte:

App-Objekt

CurrentPage-Eigenschaft

Gibt den Namen der aktuellen Seite zurück.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

Dieses Beispiel zeigt einen Seitennamen in einer Überschrift 1 (h1).

```
<!--Setzen Sie den Seitennamen als Überschrift ein-->
<h1><%=App.CurrentPage%></h1>
```

Beispielausgabe

```
LandAuswählen
```

Gilt für folgende Objekte:

App-Objekt

Errors-Eigenschaft

Gibt eine Liste von Fehlermeldungen für die aktuelle Server-Anforderung zurück.

Diese Eigenschaft ist schreibgeschützt und funktioniert nur auf der Fehlerseite.

Typ

StringList

Beispiel

```
Dieses Beispiel zeigt jeden Fehler getrennt durch eine horizontale Linie an.
<%
// Funktioniert nur auf der Fehlerseite.
for ( var i = 0; i < App.Errors.Count; i++ )
{
    if ( i > 0 )
    {
        Response.Writeln( "<hr>" );
    }
Response.Write(<p><pre>Server.HTMLEncode( App.Errors( i ) )</pre></p>)
}
%>
```

Beispielausgabe

```
Fehler auf Seite [LandAuswählen] Zeile 10

<%=App.RunDynamo("Länder")%>
---------------------------------------------------------------
Methodenaufruf [App.RunDynamo] schlug fehl

Error near no filename:10 [RunDynamo()].
      from no filename:10 [:Global Initialization:()]
---------------------------------------------------------------
Fehler in Dynamo [Länder] auf Zeile 5 bei der Ausführung der Definition.
---------------------------------------------------------------
TypeError 1406: Variable is not a function type.
Error near no filename:5 [Global Code].
```

Gilt für folgende Objekte:

App-Objekt

FinalURL-Eigenschaft

Gibt den URL an Upfront zurück, nachdem eine PowerPrompts-Anwendung beendet wurde. Verwenden Sie diese Methode auf der Schlussseite als Teil einer Aktion auf einem Formular.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

Dieses Beispiel erstellt eine Schaltfläche, durch die ein Bericht ausgeführt wird.

```
<form action="<%=App.FinalURL%>">
<input type = "submit" value = "Fertig stellen">
</form>
```

Gilt für folgende Objekte:

App-Objekt

IsTestMode-Eigenschaft

Gibt zurück, ob die PowerPrompts-Anwendung im Test-Modus ausgeführt wird. Gibt „true" zurück, wenn die Anwendung von PowerPrompts Developer Studio aus ausgeführt wird, andernfalls „false". Verwenden Sie diese Eigenschaft, um Fehler in Ihrer Anwendung zu beheben.

Diese Eigenschaft ist schreibgeschützt.

Typ

Boolesch

Beispiel

Dieses Beispiel erzeugt ein leeres Textfeld, wenn die Anwendung getestet wird.

```
<!--Das Berichtsskript nur anzeigen, wenn die Anwendung gerade getestet wird-->
<%
if (App.IsTestMode);
{
%>
<textarea>
<%=Server.HTMLEncode(App.ReportScript)%>
</textarea>
<%
}
%>
```

Gilt für folgende Objekte:

App-Objekt

Path-Eigenschaft

Gibt den Ordner der PowerPrompts-Anwendungsdatei (.xxm) zurück. Dieser Pfad beinhaltet das Trennzeichen für aufeinanderfolgende Pfade. Das Pfadtrennzeichen ist entweder ein einfacher oder ein umgekehrter Schrägstrich. Dies ist abhängig von dem Betriebssystem, unter dem PowerPrompts läuft. Diese Eigenschaft ist zweckmäßig für die Response.WriteFile-Methode.

Diese Eigenschaft ist schreibgeschützt.

Typ

String

Beispiel

Dieses Beispiel zeigt den Pfad zu der Anwendung.

```
<%Response.Write(App.Path);%>
```

Beispielausgabe

```
C:\PowerPrompts\Verkaufsrep\
```

Gilt für folgende Objekte:

App-Objekt

ReportScript-Eigenschaft

Gibt das Skript zurück, das an Impromptu gesendet wird. Diese Eigenschaft ist nur im Testmodus und auf der Schlussseite verfügbar. In allen anderen Fällen wird eine leere Zeichenkette zurückgegeben. Verwenden Sie diese Eigenschaft, um Fehler in Ihrer Anwendung zu beheben.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

Dieses Beispiel erzeugt ein Textfeld, das das Skript zeigt.

```
<!--Das Berichtsskript in einem Textfeld anzeigen-->
<textarea>
<%=Server.HTMLEncode( App.ReportScript )%>
</textarea>
```

Gilt für folgende Objekte:

App-Objekt

Variables-Eigenschaft

Gibt den Status der Anwendung zurück. Es handelt sich um eine Auflistung von Variablen. Im Gegensatz zu Anforderungsvariablen bleiben Anwendungsvariablen zwischen Anforderungen erhalten. Mit dieser Eigenschaft können Sie in der Anwendung festgelegte Variablen abrufen. Verwenden Sie die Request.Variables-Eigenschaft nur, wenn Sie die unverarbeiteten Name/Wert-Paare für Formulare benötigen.

Diese Eigenschaft ist schreibgeschützt.

Typ

StringList

Beispiel

Dieses Beispiel gibt den Ländercode für das Land an, das Sie auf der Seite *LandAuswählen* ausgewählt haben.

```
<%
// Ausgabe eines Textsteuerelements mit dem Text für LänderCd, falls kein Wert
// vorhanden, Text auf Normal einstellen
if (App.Variables("LänderCd"))
{
    Response.Writeln("<input type='text' name='text1' value='" +
App.Variables("LänderCd") + "'");
}
else
{
    Response.Writeln("<input type='text' name='text1' value='Normal'>");
}
%>
```

Beispielausgabe

```
16
```

Gilt für folgende Objekte:

App-Objekt

RunDynamo-Methode

Führt zur Erzeugung einer Zeichenkette den angegebenen Dynamo aus. Verwenden Sie diese Methode auf Ihren HTML-Seiten. Diese Methode kann nicht in den Dialogfeldern *Dynamo-Definition* oder *Dynamo-SQL* verwendet werden.

Parameter

DynamoName: Zeichenkette, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel1

Dieses Beispiel führt MeinDynamo1 aus und schreibt dessen Ausgabe auf die Seite.

```
<!--Einen mit dem Dynamo-Assistenten erstellten Dynamo ausführen-->
<%=App.RunDynamo("MeinDynamo1")%>
```

Beispiel2

Dieses Beispiel führt MeinDynamo2 aus. Wenn der zurückgegebene Wert größer als 0 ist, wird eine Meldung ausgegeben.

```
<%
var iReturn = parseInt(App.RunDynamo("MeinDynamo2"));

if (iReturn > 0 )
{
    Response.Write("Retouren: " + iReturn);
}
%>
```

Beispielausgabe

```
Retouren: 4
```

Gilt für folgende Objekte:

App-Objekt

Connection-Objekt

Ein Objekt, das Ihre Datenbank unter Verwendung von SQL-Anweisungen ändert. Dies ermöglicht die Verbindung zu einer Datenquelle und die Ausführung von SQL ohne die Rückgabe eines Ergebnissatzes.

Dieses Objekt ist erstellbar.

Vorgehensweise	Beschreibung
Execute-Methode	Führt die SQL-Anweisung aus, die die Datenbank ändert.

Execute-Methode

Führt die SQL-Anweisung aus, die die Datenbank ändert.

Wenn diese Methode fehlschlägt, wird die Fehlerseite angezeigt.

Parameter

Der SQL-Parameter ist die auszuführende SQL. Der DBName-Parameter ist der Name der logischen Cognos-Datenbank.

Es gibt zwei optionale Zeichenkettenparameter. Der DBBenutzerkennung-Parameter ist die Datenbankbenutzerkennung, die für das Herstellen einer Verbindung zur Datenbank erforderlich ist. Das DBKennwort ist das Datenbankkennwort für diese Benutzerkennung.

SQL: Zeichenkette, Obligatorisch

DBName: Zeichenkette, Obligatorisch

DBBenutzerkennung: Zeichenkette, Optional

DBKennwort: Zeichenkette, Optional

Beispiel

Dieses Beispiel fügt der Datenbank die Werte Jürgen Schmitt und Programmierer hinzu.

```
<%
var conn = new Connection();
conn.Execute("INSERT INTO User(UserName,UserDescription) VALUES('Jürgen
Schmitt','Programmierer')", "User");
Response.Write("<b>Abfrage erfolgreich</b>");
%>
```

Gilt für folgende Objekte:

Connection-Objekt

Field-Objekt

Ein Recordset-Objekt besitzt eine Fields-Auflistung, die aus Field-Objekten besteht. Jedes Field-Objekt entspricht einer Datenspalte im Datensatz.

Dieses Objekt ist nicht erstellbar.

Eigenschaft	Beschreibung
HTMLEncodedValue-Eigenschaft	Gibt den Wert der Spalte im Ergebnissatz für die aktuelle Zeile zurück, es sei denn, sie ist HTML-codiert.
Index-Eigenschaft	Gibt die Position der Spalte im Ergebnissatz zurück. Der Wert ist nullbasiert.
Name-Eigenschaft	Gibt den Namen der Spalte im Ergebnissatz zurück.
Type-Eigenschaft	Gibt den Datentyp des Feldes zurück.
Value-Eigenschaft	Gibt den Wert der Spalte im Ergebnissatz der aktuellen Zeile zurück.

Vorgehensweise	Beschreibung
Operator()-Methode (Feld)	Gibt den Index-Operator an, mit dem auf die Elemente in der Auflistung zugegriffen wird. Sie können entweder einen numerischen Index oder einen Zeichenkettenschlüssel angeben.

HTMLEncodedValue-Eigenschaft

Gibt den Wert der Spalte im Ergebnissatz für die aktuelle Zeile zurück, es sei denn, sie ist HTML-codiert. Das entsprechende JavaScript ist

```
Server.HTMLEncode(rs.Fields(0).Value)
```

Dies ist die Standardeigenschaft des Field-Objekts. Diese Eigenschaft wird unabhängig vom ursprünglichen Datenbanktyp als Zeichenkette dargestellt. Wenn Sie sie als Zahl verwenden möchten, können Sie mit einer beliebigen JavaScript-Standardmethode die Zeichenkette in eine Zahl konvertieren. Wenn der Wert in der Datenbank NULL ist, wird eine JavaScript-Null zurückgegeben.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel1

```
<%=rs.Fields(0).HTMLEncodedValue%>
```

Beispiel2

```
<%=rs.Fields(0)%>
```

Gilt für folgende Objekte:

Field-Objekt

Index-Eigenschaft

Gibt die Position der Spalte im Ergebnissatz zurück. Dies ist ein nullbasierter Wert, was bedeutet, dass die erste Spalte einen Index von 0 besitzt, die zweite einen Index von 1 usw. Beispielsweise besitzt die SQL "SELECT Land, Absatzländercode FROM Land" zwei Felder. Land besitzt einen Index von 0 und Absatzländercode einen Index von 1.

Diese Eigenschaft ist schreibgeschützt.

Typ

Zahl

Beispiel

```
<%
// Wert der Spalte plus Name und Index ausgeben
var rs = new Recordset();
rs.Open("Select Land,Absatzländercode from Land","AUFUmsatz");
// Eine Tabelle nur bei vollem Ergebnissatz ausgeben
if (!rs.EOF)
{
    Response.Writeln("<table><tr>");
    // Tabellenüberschriften ausschreiben
    for (var i = 0; i < rs.Fields.Count; i++)
    {
        Response.Write("<td>" + rs.Fields(i).Name + " Index: " + rs.Fields(i).Index + "</
td>");
    }
    Response.Writeln("</tr>");
    while (!rs.EOF)
    {
        Response.Write("<tr>");
        for (var j = 0; j < rs.Fields.Count; j++)
        {
            Response.Write("<td>" + rs.Fields(j).Value + "</td>");
            rs.MoveNext();
        }
        Response.Writeln("</tr>");
    }
    Response.Writeln("</table>");
}
else
{
    Response.Writeln("Keine Daten.<p>");
}
%>
```

Beispielausgabe

```
Land Index: 0 Absatzländercode Index: 1
Frankreich 2
USA 4
Österreich 6
Niederlande 8
England 10
Japan 12
Korea 14
Australien 17
Dänemark 19
Mexiko 21
Finnland 23
```

Gilt für folgende Objekte:

Field-Objekt

Name-Eigenschaft

Gibt den Namen der Spalte im Ergebnissatz zurück. Beispielsweise hat die SQL "SELECT Land, LänderCd AS Code FROM Land" zwei Felder. Das erste wird mit Land bezeichnet, das zweite mit Code.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

```
<%
// Wert der Spalte plus Name und Index ausgeben
var rs = new Recordset();
rs.Open("Select Land,Absatzländercode from Land","AUFUmsatz");
// Eine Tabelle nur bei vollem Ergebnissatz ausgeben
if (!rs.EOF)
{
    Response.Writeln("<table><tr>");
    // Tabellenüberschriften ausschreiben
    for (var i = 0; i < rs.Fields.Count; i++)
    {
        Response.Write("<td>" + rs.Fields(i).Name + " Index: " + rs.Fields(i).Index + "</
td>");
    }
    Response.Writeln("</tr>");
    while (!rs.EOF)
    {
        Response.Write("<tr>");
        for (var j = 0; j < rs.Fields.Count; j++)
        {
            Response.Write("<td>" + rs.Fields(j).Value + "</td>");
            rs.MoveNext();
        }
        Response.Writeln("</tr>");
    }
    Response.Writeln("</table>");
}
else
{
    Response.Writeln("Keine Daten.<p>");
}
%>
```

Beispielausgabe

```
Land Index: 0 Absatzländercode Index: 1
Frankreich 2
USA 4
Österreich 6
Niederlande 8
England 10
Japan 12
Korea 14
Australien 17
Dänemark 19
Mexiko 21
Finnland 23
```

Gilt für folgende Objekte:

Field-Objekt

Type-Eigenschaft

Gibt den Datentyp des Feldes zurück. Die folgenden numerischen Konstanten können verwendet werden:

- ftDate
- ftDateTime
- ftInterval
- ftNumeric
- ftString
- ftTime
- ftUnknown

Diese Eigenschaft ist schreibgeschützt.

Typ

Zahl

Beispiel

```
<%
// Wert der Spalte plus Name und Index ausgeben
var rs = new Recordset();
rs.Open("Select * from Land","AUFUmsatz");
// Namen und Datentypen der Spalten ausgeben
for (var i = 0; i < rs.Fields.Count; i++)
{
var sText = rs.Fields(i).Name + ": ";
    switch(rs.Fields(i).Type)
    {
        case ftDate:
            sText += "Datum";
            break;
        case ftDateTime:
            sText += "Datum/Uhrzeit";
            break;
        case ftInterval:
            sText += "Intervall";
            break;
        case ftNumeric:
            sText += "Numerisch";
            break;
        case ftString:
            sText += "Zeichenkette";
            break;
        case ftTime:
            sText += "Uhrzeit";
            break;
        case ftUnknown:
            sText += "Unbekannt";
            break;
    }
Response.Writeln(sText + "<br>");
}
%>
```

Beispielausgabe

```
Absatzländercode: Numerisch
Land: Zeichenkette
ISOCode3Buchstaben: Zeichenkette
ISOCode2Buchstaben: Zeichenkette
ISOCode3Ziffern: Zeichenkette
Währungsbezeichng: Zeichenkette
InEuroSeit: Datum/Uhrzeit
```

Gilt für folgende Objekte:

Field-Objekt

Value-Eigenschaft

Gibt den Wert der Spalte im Ergebnissatz der aktuellen Zeile zurück. Wenn der Wert in der Datenbank NULL ist, wird eine JavaScript-Null zurückgegeben.

Diese Eigenschaft wird unabhängig vom ursprünglichen Datenbanktyp als Zeichenkette dargestellt. Wenn Sie sie als Zahl verwenden möchten, können Sie mit einer beliebigen JavaScript-Standardmethode die Zeichenkette in eine Zahl konvertieren.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

```
<%
// Ausgabe des Feldwertes oder der Zeichenkette "NULL", wenn das Feld keinen Wert hat
var rs = new Recordset();
rs.Open("Select Absatzländercode from Land","AUFUmsatz");
var sValue = rs.Fields( 0 ).Value;
Response.Write( ( sValue == null ) ? "NULL" : sValue );
%>
```

Beispielausgabe

```
1
```

Gilt für folgende Objekte:

Field-Objekt

Operator()-Methode (Feld)

Gibt den Index-Operator an, mit dem auf die Elemente in der Auflistung zugegriffen wird. Sie können entweder einen numerischen Index oder einen Zeichenkettenschlüssel angeben.

Parameter

Index: Numerisch, Obligatorisch

oder

Schlüssel: Zeichenkette, Obligatorisch

Rückgabetyp

Ein Element in einer Auflistung.

Beispiel

```
<%
// Wert der Spalte plus Name und Index ausgeben
var rs = new Recordset();
rs.Open("Select * from Land","AUFUmsatz");
// Beide Möglichkeiten der Verwendung der Operator()-Methode anzeigen
Response.Write(rs.Fields(0) + " ist der Wert der ersten Spalte in der
Datensatzgruppe.");
Response.Write("<br>");
Response.Write(rs.Fields("Land") + " ist der Wert der Spalte Land in der
Datensatzgruppe.");
%>
```

Beispielausgabe

```
1 ist der Wert der ersten Spalte in der Datensatzgruppe.
Frankreich ist der Wert der Spalte Land in der Datensatzgruppe.
```

Gilt für folgende Objekte:

Field-Objekt

Query-Objekt

Verwenden Sie dieses Objekt zum Abrufen eines Impromptu-Berichts. Die SQL enthält alle Benutzerklassenfilter und alle im Katalog definierten Datenbankqualifikatoren. Die SQL kann anschließend in einem Dynamo oder einer Auswahlliste für das Ausfüllen von Benutzersteuerelementen verwendet werden. Alle im Impromptu-Katalog festgelegten verweigerten Tabellen werden ignoriert.

Sie können das Query-Objekt mit dem vollständigen Pfad zu einem Impromptu-Bericht erstellen. Dies bedeutet, dass das Query-Objekt die Hauptabfrage für diesen Bericht zurückgibt. PowerPrompts kann nicht auf Unterberichtsabfragen zugreifen. Wenn Sie mit einer leeren Abfrage beginnen möchten, übergeben Sie keine Objekte an die Erstellungsanweisung für die Abfrage, beispielsweise

```
var myQuery = new Query()
```

Dieses Objekt ist erstellbar.

Eigenschaft	Beschreibung
SQL-Eigenschaft	Gibt basierend auf der Benutzerklasse des aktuellen Benutzers die SQL für das Query-Objekt zurück, einschließlich der Impromptu-Katalogfilter.

Vorgehensweise	Beschreibung
AddColumn-Methode	Fügt der Abfrage eine Impromptu-Katalogspalte oder eine Berechnung hinzu.
AndFilterBy-Methode	Fügt der Abfrage einen Detailfilterausdruck hinzu. Wenn bereits ein Detailfilter vorhanden ist, wird der Filter unter Verwendung eines AND-Operators an diesen angehängt.
AndSummaryFilterBy-Methode	Fügt der Abfrage einen Auswertungsfilterausdruck hinzu. Wenn bereits ein Auswertungsfilter vorhanden ist, wird der Filter unter Verwendung des AND-Operators an diesen angehängt.
AssociateColumn-Methode	Fügt der Abfrage eine Zuordnung zu einer Spalte hinzu.
GroupBy-Methode	Fügt der Abfrage die Gruppierung für eine bestimmte Spalte hinzu.
OrFilterBy-Methode	Fügt der Abfrage einen Detailfilterausdruck hinzu. Wenn bereits ein Detailfilter vorhanden ist, wird der Filter unter Verwendung des OR-Operators an diesen angehängt.
OrSummaryFilterBy-Methode	Fügt der Abfrage einen Auswertungsfilterausdruck hinzu. Wenn bereits ein Auswertungsfilter vorhanden ist, wird der Filter unter Verwendung des OR-Operators an diesen angehängt.
RemoveColumn-Methode	Entfernt eine Spalte aus der Abfrage und dem Bericht.
SetDistinct-Methode	Legt fest, dass die Abfrage nur eindeutige Elemente enthält.

Vorgehensweise	Beschreibung
SetPromptValue-Methode	Legt für eine benannte Eingabeaufforderung in der Abfrage einen Wert fest.
SortBy-Methode	Sortiert die Abfrage zusätzlich zu der vorhandenen Sortierfolge nach der festgelegten Spalte.

SQL-Eigenschaft

Gibt basierend auf der Benutzerklasse des aktuellen Benutzers die SQL für das Query-Objekt zurück, einschließlich der Impromptu-Katalogfilter.

Die zurückgegebene SQL beginnt immer mit einem Ausrufezeichen (!). Dieses Zeichen teilt dem Recordset-Objekt mit, dass die SQL von dem Query-Objekt erzeugt wurde.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

Dieses Beispiel öffnet den Impromptu-Bericht pwp_Lernprogramm_js.imr, der sich im selben Ordner wie die PowerPrompts-Anwendung befindet, und gibt anschließend die SQL dieses Berichts an das Recordset-Objekt zurück. Alle Benutzerklassenfilter werden vor der Rückgabe der SQL angewendet.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_Lernprogramm_js.imr");
var rs = new Recordset();
rs.Open(myQuery.SQL, "AUFUmsatz");
%>
```

Gilt für folgende Objekte:

Query-Objekt

AddColumn-Methode

Fügt der Abfrage eine Impromptu-Katalogspalte oder eine Berechnung hinzu. Weitere Informationen zum Erstellen von Impromptu-Katalogausdrücken finden Sie unter "Impromptu-Ausdrücke in PowerPrompts" (S. 99).

Der Name-Parameter ist der Name der Spalte (oder Berechnung) und muss innerhalb der Abfrage eindeutig sein. Der Ausdruck-Parameter ist der Impromptu-Katalogausdruck, der die Spalte definiert.

Parameter

Name: Zeichenkette, Obligatorisch

Ausdruck: Zeichenkette, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel fügt dem Bericht eine neue Spalte hinzu, bevor die SQL zurückgegeben wird.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_Lernprogramm_js.imr");
myQuery.AddColumn("Jahr","[Jahr([Bestellungen\\Bestelldatum])]");
rs.Open(myQuery.SQL, "AUFUmsatz");
%>
```

Gilt für folgende Objekte:

Query-Objekt

AndFilterBy-Methode

Fügt der Abfrage einen Detailfilterausdruck hinzu. Wenn bereits ein Detailfilter vorhanden ist, wird der Filter unter Verwendung eines AND-Operators an diesen angehängt. Weitere Informationen zum Erstellen von Impromptu-Katalogausdrücken finden Sie unter "Impromptu-Ausdrücke in PowerPrompts" (S. 99).

Der Detailfilter-Parameter ist der hinzuzufügende Detailfilterausdruck.

Parameter

Detailfilter: Zeichenkette, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel filtert den Bericht vor der Rückgabe der SQL nach Jahr.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_Lernprogramm_js.imr");
myQuery.AndFilterBy("Jahr([Bestellungen\\Bestelldatum]) in (1999,2000)");
rs.Open(myQuery.SQL, "AUFUmsatz");
%>
```

Gilt für folgende Objekte:

Query-Objekt

AndSummaryFilterBy-Methode

Fügt der Abfrage einen Auswertungsfilterausdruck hinzu. Wenn bereits ein Auswertungsfilter vorhanden ist, wird der Filter unter Verwendung des AND-Operators an diesen angehängt. Weitere Informationen zum Erstellen von Impromptu-Katalogausdrücken finden Sie unter "Impromptu-Ausdrücke in PowerPrompts" (S. 99).

Der Auswertungsfilter-Parameter ist der hinzuzufügende Auswertungsfilterausdruck.

Parameter

Auswertungsfilter: Zeichenkette, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel filtert den Bericht vor der Rückgabe der SQL nach Jahr.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_Lernprogramm_js.imr");
myQuery.AndSummaryFilterBy("Jahr([Bestellungen\\Bestelldatum]) in (1999,2000)");
rs.Open(myQuery.SQL, "AUFUmsatz");
%>
```

Gilt für folgende Objekte:

Query-Objekt

AssociateColumn-Methode

Fügt der Abfrage eine Zuordnung zu einer Spalte hinzu.

Der NameSpalte-Parameter ist der Name der Spalte, die zugeordnet werden soll. Der Gruppenspalte-Parameter ist der Name der gruppierten Spalte, der die erste Spalte zugeordnet wird.

Parameter

NameSpalte: Zeichenkette, Obligatorisch

Gruppenspalte: Zeichenkette, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel ordnet die Spalte *Gesamtumsatz* der Spalte *Land* zu, bevor die SQL zurückgegeben wird.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_Lernprogramm_js.imr");
myQuery.AssociateColumn("Gesamtumsatz","Land");
rs.Open(myQuery.SQL, "AUFUmsatz");
%>
```

Gilt für folgende Objekte:

Query-Objekt

GroupBy-Methode

Fügt der Abfrage die Gruppierung für eine bestimmte Spalte hinzu.

Der Gruppenspalte-Parameter ist der Name der zu gruppierenden Spalte. Der Sortierfolge-Parameter legt fest, wie die Gruppenspalte sortiert wird: *Aufsteigend* oder *Absteigend*.

Diese Methode fügt eine Gruppe einer beliebigen vorhandenen Gruppe im Bericht hinzu.

Parameter

Gruppenspalte: Zeichenkette, Obligatorisch

Sortierfolge: Zeichenkette (*Aufsteigend* oder *Absteigend*), Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel gruppiert die Spalte *Anbietername*, bevor die SQL zurückgegeben wird.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_Lernprogramm_js.imr");
myQuery.GroupBy("Anbietername","Absteigend");
rs.Open(myQuery.SQL, "AUFUmsatz");
%>
```

Gilt für folgende Objekte:

Query-Objekt

OrFilterBy-Methode

Fügt der Abfrage einen Detailfilterausdruck hinzu. Wenn bereits ein Detailfilter vorhanden ist, wird der Filter unter Verwendung des OR-Operators an diesen angehängt. Weitere Informationen zum Erstellen von Impromptu-Katalogausdrücken finden Sie unter "Impromptu-Ausdrücke in PowerPrompts" (S. 99).

Der Detailfilter-Parameter ist der hinzuzufügende Detailfilterausdruck.

Parameter

Detailfilter: Zeichenkette, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel fügt einen Detailfilter hinzu, bevor die SQL zurückgegeben wird.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_Lernprogramm_js.imr");
myQuery.OrFilterBy("[\\Produkte\\Produkt] beginnt mit 'AUF'");
rs.Open(myQuery.SQL, "AUFUmsatz");
%>
```

Gilt für folgende Objekte:

Query-Objekt

OrSummaryFilterBy-Methode

Fügt der Abfrage einen Auswertungsfilterausdruck hinzu. Wenn bereits ein Auswertungsfilter vorhanden ist, wird der Filter unter Verwendung des OR-Operators an diesen angehängt. Weitere Informationen zum Erstellen von Impromptu-Katalogausdrücken finden Sie unter "Impromptu-Ausdrücke in PowerPrompts" (S. 99).

Der Auswertungsfilter-Parameter ist der hinzuzufügende Auswertungsfilterausdruck.

Parameter

Auswertungsfilter: Zeichenkette, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel fügt einen Auswertungsfilter hinzu, bevor die SQL zurückgegeben wird.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_Lernprogramm_js.imr");
myQuery.OrSummaryFilterBy("[\\Produkte\\Produkt] beginnt mit 'AUF'");
rs.Open(myQuery.SQL, "AUFUmsatz");
%>
```

Gilt für folgende Objekte:

Query-Objekt

RemoveColumn-Methode

Entfernt eine Spalte aus der Abfrage und dem Bericht.

Der Spaltenname-Parameter bezeichnet die Spalte, die aus der Abfrage entfernt werden soll.

Parameter

Spaltenname: Zeichenkette, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel entfernt die Spalte *Anbietername* aus der Abfrage, bevor die SQL zurückgegeben wird.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_Lernprogramm_js.imr");
myQuery.RemoveColumn("Anbietername");
rs.Open(myQuery.SQL, "AUFUmsatz");
%>
```

Gilt für folgende Objekte:

Query-Objekt

SetDistinct-Methode

Legt fest, dass die Abfrage nur eindeutige Elemente enthält.

Parameter

Umschalter: Boolesch, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel entfernt alle doppelten Werte, bevor die SQL zurückgegeben wird.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_Lernprogramm_js.imr");
myQuery.SetDistinct("true");
rs.Open(myQuery.SQL, "AUFUmsatz");
%>
```

Gilt für folgende Objekte:

Query-Objekt

SetPromptValue-Methode

Legt für eine benannte Eingabeaufforderung in der Hauptabfrage einen Wert fest. Diese Methode funktioniert nur, wenn die Abfrage innerhalb eines vorhandenen Impromptu-Berichts erstellt wird. Alle Eingabeaufforderungen müssen Werte besitzen, bevor Sie die SQL für die Abfrage abrufen können. Um sicher zu stellen, dass der Impromptu-Bericht mehrere Eingabewerte akzeptiert, verwenden Sie in der Filteranweisung des Impromptu-Berichts anstelle eines Ist-Gleich-Operators (=) einen „In"-Operator.

Tipp: Verwenden Sie zum Trennen mehrerer Werte ein Komma. Wenn ein Komma als Teil des Wertes erforderlich ist, stellen Sie dem Komma ein Escape-Zeichen voran. Standardmäßig ist das Caret-Zeichen (^) das Escape-Zeichen. Dieses können Sie jedoch ändern, indem Sie den Eintrag für *Separator Escape Character* im Abschnitt [Startup Options] der Datei Impromptu.ini aktualisieren.

Parameter

Der Aufforderungsname-Parameter ist der Name der Eingabeaufforderung innerhalb der Abfrage. Verwenden Sie den Eingabeaufforderungsnamen aus dem Feld *Verfügbare Eingabeaufforderungen* des Dialogfeldes *Eingabeaufforderungs-Manager*.

Der Aufforderungswert-Parameter ist der Wert, den die Eingabeaufforderung besitzen soll.

Aufforderungsname: Zeichenkette, Obligatorisch

Aufforderungswert: Zeichenkette, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel setzt den Wert *Kanada* für die Eingabeaufforderung *Land*, bevor die SQL zurückgegeben wird.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_Lernprogramm_js.imr");
myQuery.SetPromptValue("Land", "Kanada");
rs.Open(myQuery.SQL, "AUFUmsatz");
%>
```

Gilt für folgende Objekte:

Query-Objekt

SortBy-Methode

Sortiert die Abfrage zusätzlich zu der vorhandenen Sortierfolge nach der festgelegten Spalte.

Parameter

Der Sortierspalte-Parameter ist der Name der zu sortierenden Spalte. Der Sortierfolge-Parameter legt fest, wie die Sortierspalte sortiert wird: *Aufsteigend* oder *Absteigend*.

Sortierspalte: Zeichenkette, Obligatorisch

Sortierfolge: Zeichenkette (*Aufsteigend* oder *Absteigend*), Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel sortiert die Abfrage nach der Spalte *Produkttyp*, bevor die SQL zurückgegeben wird.

```
<%
var myQuery = new Query("pwp_Lernprogramm_js.imr");
myQuery.SortBy("Produkttyp","Aufsteigend");
rs.Open(myQuery.SQL, "AUFUmsatz");
%>
```

Gilt für folgende Objekte:

Query-Objekt

Recordset-Objekt

Entnimmt Daten aus logischen Cognos-Datenbanken. Das Recordset-Objekt stellt Methoden für die Verbindung zu Datenbanken, für die Ausführung von SQL und das Zurückholen von Datenzeilen bereit.

Dieses Objekt ist erstellbar.

Eigenschaft	Beschreibung
CurrentRecordIndex-Eigenschaft	Gibt den aktuellen Datensatzindex zurück, der nullbasiert ist. Dieser Wert erhöht sich nach jedem Aufruf der MoveNext-Methode. Diese Eigenschaft ist nur verfügbar, wenn die EOF-Eigenschaft auf false gesetzt ist.
EOF-Eigenschaft	Überprüft auf das Ende der Datensatzgruppe.
Fields-Eigenschaft	Eine Auflistung von Feldobjekten. Jedes Feld steht für eine Spalte im Ergebnissatz der Abfrage.
MaxRecords-Eigenschaft	Legt die maximale Anzahl von zurückzugebenden Datensätzen fest oder gibt sie zurück.

Vorgehensweise	Beschreibung
Close-Methode	Schließt eine offene Datensatzgruppe.
MoveNext-Methode	Wechselt in die nächste Zeile in der Datensatzgruppe.
Open-Methode	Füllt den Datensatz mit den durch die Ausführung der SQL zurückgegebenen Zeilen.

CurrentRecordIndex-Eigenschaft

Gibt den aktuellen Datensatzindex zurück, der nullbasiert ist. Dieser Wert erhöht sich nach jedem Aufruf der MoveNext-Methode. Diese Eigenschaft ist nur verfügbar, wenn die EOF-Eigenschaft auf false gesetzt ist.

Diese Eigenschaft ist schreibgeschützt.

Typ

Integer (Ganzzahl)

Beispiel

Dieses Beispiel führt hinter der Zeilennummer in einer separaten Zeile den Namen eines jeden Landes auf.

```
<%
var rs = new Recordset();
rs.Open("Select Land from Land","AUFUmsatz");
while (!rs.EOF)
{
    Response.Writeln("Zeile: " + rs.CurrentRecordIndex + " " + rs.Fields("Land"));
    Response.Write("<br>");
    rs.MoveNext();
}
%>
```

Beispielausgabe

```
Zeile: 0 Frankreich
Zeile: 1 Deutschland
Zeile: 2 USA
Zeile: 3 Kanada
Zeile: 4 Österreich
Zeile: 5 Italien
Zeile: 6 Niederlande
Zeile: 7 Schweiz
Zeile: 8 England
Zeile: 9 Schweden
Zeile: 10 Japan
Zeile: 11 Taiwan
Zeile: 12 Korea
Zeile: 13 China
Zeile: 14 Australien
Zeile: 15 Belgien
Zeile: 16 Dänemark
Zeile: 17 Spanien
Zeile: 18 Mexiko
Zeile: 19 Brasilien
Zeile: 20 Finnland
Zeile: 21 Euroland
```

Gilt für folgende Objekte:

Recordset-Objekt

EOF-Eigenschaft

Überprüft auf das Ende der Datensatzgruppe. EOF steht für End Of File (Ende der Datei). Wenn die resultierende Abfrage keine Zeilen enthält, ist EOF wahr (true).

Diese Eigenschaft ist schreibgeschützt.

Typ

Boolesch

Beispiel

Dieses Beispiel führt den Namen eines jeden Landes in einer separaten Zeile auf.

```
<%
var rs = new Recordset();
rs.Open("Select Absatzländercode,Land from Land", "AUFUmsatz");
while(!rs.EOF)
{
    Response.Write("<option value='" + rs.Fields("Absatzländercode") + "'>" +
rs.Fields("Land") + "</option>");
    Response.Write("<br>");
    rs.MoveNext();
}
rs.Close();
%>
```

Beispielausgabe

```
Frankreich
Deutschland
USA
Kanada
Österreich
Italien
Niederlande
Schweiz
England
Schweden
Japan
Taiwan
Korea
China
Australien
Belgien
Dänemark
Spanien
Mexiko
Brasilien
Finnland
Euroland
```

Gilt für folgende Objekte:

Recordset-Objekt

Fields-Eigenschaft

Eine Auflistung von Field-Objekten. Jedes Feld steht für eine Spalte im Ergebnissatz der Abfrage.

Diese Eigenschaft ist schreibgeschützt.

Typ

Auflistung

Beispiel

```
<%
var rs = new Recordset();
rs.Open("Select * from Land","AUFUmsatz");
Response.Write("Anzahl der Felder im Ergebnissatz: " + rs.Fields.Count);
%>
```

Beispielausgabe

```
Anzahl der Felder im Ergebnissatz: 7
```

Gilt für folgende Objekte:

Recordset-Objekt

MaxRecords-Eigenschaft

Legt die maximale Anzahl von zurückzugebenden Datensätzen fest oder gibt sie zurück.

Damit diese Eigenschaft ordnungsgemäß funktioniert, muss sie vor dem Aufruf der Recordset.Open-Methode festgelegt werden.

Für diese Eigenschaft besteht Lese- und Schreibzugriff.

Typ

Integer (Ganzzahl)

Standardwert

0 -- Gibt keine Datensätze zurück

-1 -- Unbegrenzte Anzahl von Datensätzen

Beispiel

```
<%
var rs = new Recordset();
rs.MaxRecords = 10;
rs.Open("Select Absatzländercode,Land from Land", "AUFUmsatz");
while(!rs.EOF)
{
    Response.Write("<option value='" + rs.Fields("Absatzländercode") + "'>" +
rs.Fields("Land") + "</option>");
    Response.Write("<br>");
    rs.MoveNext();
}
%>
```

Beispielausgabe

```
Frankreich
Deutschland
USA
Kanada
Österreich
Italien
Niederlande
Schweiz
England
Schweden
```

Gilt für folgende Objekte:

Recordset-Objekt

Close-Methode

Schließt die Datensatzgruppe. Wenn eine Datensatzgruppe nicht ausdrücklich geschlossen wird, wird sie von PowerPrompts am Ende der Seite geschlossen. Mit dieser Methode können Sie vorhandene Recordset-Objekte erneut verwenden.

Parameter

Ohne

Rückgabetyp

Ohne

Beispiel

Dieses Beispiel gibt jeden Ländernamen und anschließend jede Produktreihe aus.

```
<%
var rs = new Recordset();
rs.Open("Select Absatzländercode,Land from Land", "AUFUmsatz");
// Datensatzgruppe muss geschlossen werden, bevor Recordset-Objekt wiederverwendet
werden kann
while(!rs.EOF)
{
    Response.Write("<option value='" + rs.Fields("Absatzländercode") + "'>" +
rs.Fields("Land") + "</option>");
    Response.Write(", ");
    rs.MoveNext();
}
Response.Write("<br><br>");
rs.Close();
rs.Open("Select Produktreihe from Produktreihe","AUFUmsatz");
while(!rs.EOF)
{
    Response.Write("<option>" + rs.Fields("Produktreihe") + "</option>");
    Response.Write(", ");
    rs.MoveNext();
}
%>
```

Beispielausgabe

```
Frankreich, Deutschland, USA, Kanada, Österreich, Italien, Niederland, Schweiz, England,
Schweden, Japan, Taiwan, Korea, China, Australien, Belgien, Dänemark, Spanien, Mexiko,
Brasilien, Finnland, Euroland,
Campingausrüstung, Bergsteigerausrüstung, Accessoires, Outdoor-Bekleidung,
Golfausrüstung,
```

Gilt für folgende Objekte:

Recordset-Objekt

MoveNext-Methode

Wechselt in die nächste Zeile in der Datensatzgruppe.

Parameter

Ohne

Rückgabetyp

Ohne

Beispiel

Dieses Beispiel gibt jeden Ländernamen und die Währung in einer eigenen Zeile aus.

```
<%
var rs = new Recordset();
rs.Open("Select Währungsbezeichng,Land from Land order by Land", "AUFUmsatz");
while(!rs.EOF)
{
    Response.Writeln(rs.Fields("Land") + " " + rs.Fields("Währungsbezeichng"));
    Response.Write("<br>");
    rs.MoveNext();
}
rs.Close();
%>
```

Beispielausgabe

```
Australien Dollar
Österreich Schillinge
Belgien Franc
Brasilien Real
Kanada Dollar
China Renminbi
Dänemark Kronen
England Pfund
Euroland Euro
Finnland Finnmark
Frankreich Franc
Deutschland D-Mark
Italien Lire
Japan Yen
Korea Won
Mexiko Pesos
Niederlande Gulden
Spanien Peseten
Schweden Kronen
Schweiz Franken
Taiwan Neuer Dollar
USA Dollar
```

Gilt für folgende Objekte:

Recordset-Objekt

Open-Methode

Füllt den Datensatz mit den durch die Ausführung der SQL zurückgegebenen Zeilen.

Parameter

Der SQL-Parameter ist die auszuführende SQL. Der DBName-Parameter ist der Name der logischen Cognos-Datenbank.

Es gibt zwei optionale Zeichenkettenparameter. Der DBBenutzerkennung-Parameter ist die Datenbankbenutzerkennung, die für das Herstellen einer Verbindung zur Datenbank erforderlich ist. Das DBKennwort ist das Datenbankkennwort für diese Benutzerkennung.

SQL: Zeichenkette, Obligatorisch

DBName: Zeichenkette, Obligatorisch

DBBenutzerkennung: Zeichenkette, Optional

DBKennwort: Zeichenkette, Optional

Rückgabetyp

Ohne

Beispiel1

Dieses Beispiel zeigt, wie für eine logische Datenbank, für die keine Datenbankanmeldung erforderlich ist, eine Datensatzgruppe geöffnet wird.

```
<%
var rs = new Recordset();
rs.Open("Select Produktreihencode, Produktreihe from Produktreihe order by
Produktreihe", "AUFUmsatz");
while(!rs.EOF)
{
    Response.Write("<option value='" + rs.Fields("Produktreihencode") + "'>" +
rs.Fields("Produkreihe") + "</option>");
    Response.Write("<br>");
    rs.MoveNext();
}
rs.Close();
%>
```

Ausgabe von Beispiel1

```
Campingausrüstung
Golfausrüstung
Bergsteigerausrüstung
Schutzartikel
Persönliches Zubehör
```

Beispiel2

Dieses Beispiel erläutert die Verwendung von SQL aus einem Impromptu-Bericht.

Der Beispielcode gleicht dem in Beispiel1, mit dem Unterschied, dass Zeile 2 (rs.Open...) durch die folgenden Zeilen ersetzt werden muss:

```
var myQuery = new Query("pwp_Lernprogramm_js.imr");
rs.Open(myQuery.SQL, "AUFUmsatz");
```

Beispiel3

Dieses Beispiel erläutert die Verwendung von SQL über eine Zeichenkette.

Der Beispielcode gleicht dem in Beispiel1, mit dem Unterschied, dass Zeile 2 (rs.Open...) durch die folgenden Zeilen ersetzt werden muss:

```
var sSQL = new String;
sSQL = "Select Land from Land";
rs.Open(sSQL.SQL, "AUFUmsatz");
```

Gilt für folgende Objekte:

Recordset-Objekt

Request-Objekt

Stellt Informationen über die aktuelle Anforderung des Clients bereit.

Eigenschaft	Beschreibung
Cookies-Eigenschaft	Gibt alle eingehenden Cookies als Zeichenkettenliste zurück.
ServerVariables-Eigenschaft	Gibt alle Servervariablen als Zeichenkettenliste zurück.
Variables-Eigenschaft	Gibt die Variablen für die aktuelle Anforderung zurück. Diese unterscheiden sich von Anwendungsvariablen, die zwischen Anforderungen als Zeichenkettenliste erhalten bleiben.

Cookies-Eigenschaft

Gibt alle eingehenden Cookies als Zeichenkettenliste zurück. Ein Cookie ist ein vom Browser gesendetes Name/Wert-Paar. Dabei werden die Cookies mit genaueren Pfadattributen denen mit weniger genauen Attributen vorangestellt.

Diese Eigenschaft ist schreibgeschützt.

Typ

StringList

Beispiel1

Dieses Beispiel zeigt eine Liste von Cookies und ihre zugehörigen Werte.

```
<%
// Alle Cookies auflisten
for (var myCookie in Request.Cookies)
{
    Response.Write(myCookie + " = " + Request.Cookies(myCookie) + "<br/>");
}
%>
```

Beispiel2

Dieses Beispiel zeigt den Wert des Cookies mit dem Namen CookieName.

```
<% =Request.Cookies("CookieName") %>
```

Gilt für folgende Objekte:

Request-Objekt

ServerVariables-Eigenschaft

Gibt alle Servervariablen als Zeichenkettenliste zurück. Sie können Servervariablen nicht nach Zahl indizieren, da diese Variablen keine Reihenfolge besitzen.

Die folgenden Servervariablen können zurückgegeben werden:

AUTH_TYPE	AUTH_USER
CONTENT_LENGTH	CONTENT_TYPE
GATEWAY_INTERFACE	HTTP_ACCEPT
HTTP_ACCEPT_LANGUAGE	HTTP_COOKIE
HTTP_USER_AGENT	HTTP_ACCEPT_CHARSET

HTTP_HOST	HTTP_RANGE
HTTP_REFERER	HTTPS
LOGON_USER	PATH_INFO
PATH_TRANSLATED	QUERY_STRING
REMOTE_ADDR	REMOTE_HOST
REMOTE_IDENT	REMOTE_USER
REQUEST_METHOD	SCRIPT_NAME
SERVER_NAME	SERVER_PORT
SERVER_PROTOCOL	SERVER_SOFTWARE

Diese Eigenschaft ist schreibgeschützt.

Typ

StringList

Beispiel

Dieses Beispiel zeigt den Wert der Servervariable mit dem Namen SERVER_NAME.

```
<%=Request.ServerVariables("SERVER_NAME")%>
```

Gilt für folgende Objekte:

Request-Objekt

Variables-Eigenschaft

Gibt die Variablen für die aktuelle Anforderung zurück. Diese unterscheiden sich von Anwendungsvariablen, die zwischen Anforderungen als Zeichenkettenliste erhalten bleiben. Verwenden Sie diese Eigenschaft nur, wenn Sie die unverarbeiteten Formvariablenpaare benötigen. Verwenden Sie die App.Variables-Eigenschaft, um in der Anwendung festgelegte Variablen abzurufen.

Diese Eigenschaft ist schreibgeschützt.

Typ

StringList

Beispiel

Dieses Beispiel ordnet den Wert der Variable LänderCd der JavaScript-Variable myValue zu. Die Variable LänderCd muss auf einer vorangegangenen Seite festgelegt werden.

```
<%
var myValue = Request.Variables("LänderCd");
%>
```

Gilt für folgende Objekte:

Request-Objekt

Response-Objekt

Sendet Inhalte zurück an den Browser.

Eigenschaft	Beschreibung
ContentType-Eigenschaft	Legt den HTTP-Inhaltstyp der Antwort fest.

Vorgehensweise	Beschreibung
AppendCookie-Methode	Setzt ein Cookie.
AppendHeader-Methode	Fügt der HTTP-Kopfzeile der Antwort ein Feld hinzu.
Clear-Methode	Löscht sowohl die Inhalts- als auch die HTTP-Kopfzeilenfelder.
ClearContent-Methode	Löscht den gesamten Inhalt der Antwort.
ClearHeaders-Methode	Löscht alle HTTP-Kopfzeilenfelder der Antwort.
Redirect-Methode	Leitet den Browser an eine andere URL-Adresse um.
Write-Methode	Fügt in die vom Browser empfangene Antwort eine Zeichenkette ein.
WriteFile-Methode	Fügt in die vom Browser empfangene Antwort den Inhalt einer Datei ein.
Writeln-Methode	Fügt der vom Browser empfangenen Antwort eine Zeichenkette ein (gefolgt von einer neuen Zeile).

ContentType-Eigenschaft

Legt den HTTP-Inhaltstyp der Antwort fest. Informationen über den MIME-Typ finden Sie unter dem HTTP-Standard.

Wenn es sich um eine HTML-Seite handelt, muss diese Eigenschaft nicht festgelegt werden, da HTML die Standardeinstellung ist. Legen Sie diese Eigenschaft nur einmal pro Seite fest.

Für diese Eigenschaft besteht Lese- und Schreibzugriff.

Typ

String (Zeichenkette)

Standardwert

Text/HTML

Beispiel1

Dieses Beispiel erläutert, wie unformatierter Text an den Browser gesendet wird.

```
<%
Response.ContentType = "text/plain";
Response.Write("Dies ist eine unformatierte Textnachricht.");
%>
```

Ausgabe von Beispiel1

```
Dies ist eine unformatierte Textnachricht.
```

Beispiel2

Dieses Beispiel erläutert, wie eine Excel-Arbeitsmappe an den Browser gesendet wird.

```
<%
Response.ContentType = "application/vnd.ms-excel";
Response.AppendHeader("Content-Disposition", "inline; filename=file1.xls");
%>
<table>
<tr>
<td>Produktnummer</td>
<td>Produktname</td>
<td>Produktkosten</td>
</tr>
<tr>
<td align=left>105</td>
<td>Hailstorm Hölzerset - Titan</td>
<td align=left>555,90 EUR</td>
</tr>
</table>
```

Gilt für folgende Objekte:

Response-Objekt

AppendCookie-Methode

Setzt ein Cookie. Weitere Informationen über Cookies finden Sie in der Cookie-Spezifikation von Netscape.

Parameter

CookieName: Zeichenkette, Obligatorisch

Beispiel

```
<% Response.AppendCookie( "MyCookie=Laventus3; path=/" ), %>
```

Gilt für folgende Objekte:

Response-Objekt

AppendHeader-Methode

Fügt der HTML-Kopfzeile der Antwort ein Feld hinzu. Diese Methode ist eine Zusatzfunktion, die für die normale Programmbenutzung nicht erforderlich ist.

Parameter

KopfzeileFeld: Zeichenkette, Obligatorisch

Beispiel

```
<%Response.AppendHeader("Content-Type: text/plain");%>
```

Gilt für folgende Objekte:

Response-Objekt

Clear-Methode

Löscht sowohl die Inhalts- als auch die HTTP-Kopfzeilenfelder. Diese Methode besitzt den gleichen Effekt wie ein gleichzeitiger Aufruf der Methoden Response.ClearContent und Response.ClearHeader. Diese Methode ist eine Zusatzfunktion, die für die normale Programmbenutzung nicht erforderlich ist.

Beispiel

```
<% Response.Clear(); %>
```

Gilt für folgende Objekte:

Response-Objekt

ClearContent-Methode

Löscht den gesamten Inhalt der Antwort. Diese Methode ist eine Zusatzfunktion, die für die normale Programmbenutzung nicht erforderlich ist.

Beispiel

```
<%Response.ClearContent();%>
```

Gilt für folgende Objekte:

Response-Objekt

ClearHeaders-Methode

Löscht alle HTTP-Kopfzeilenfelder der Antwort. Diese Methode ist eine Zusatzfunktion, die für die normale Programmbenutzung nicht erforderlich ist.

Beispiel

```
<%Response.ClearHeaders()%>
```

Gilt für folgende Objekte:

Response-Objekt

Redirect-Methode

Leitet den Browser an eine andere URL-Adresse um.

Parameter

URL: Zeichenkette, Obligatorisch

Beispiel

Dieses Beispiel leitet den Browser auf die Cognos-Website um.

```
<%Response.Redirect( "www.cognos.com" );%>
```

Gilt für folgende Objekte:

Response-Objekt

Write-Methode

Fügt in die vom Browser empfangene Antwort eine Zeichenkette ein. Diese Methode konvertiert automatisch numerische Daten in eine Zeichenkette.

Parameter

Zeichenkette: Zeichenkette, Obligatorisch

Rückgabetyp

Ohne

Beispiel1

Dieses Beispiel sendet Text an den Browser.

```
<%
// Zeichenkette für Browser schreiben
Response.Write("<b>Dies ist eine fett formatierte Zeichenkette.</b>");
%>
```

Beispiel2

Dieses Beispiel sendet ein Dropdown-Listenfeld mit Produkten an den Browser.

```
<select name="ProduktTyp">
<%
var rs = new Recordset;
rs.Open("Select ProdukttypCode, ProduktTyp from ProduktTyp where ProdukttypCode = " +
App.Variables("ProduktReihe"), "AUFUmsatz");
while(!rs.EOF)
{
    Response.Write("<option value='" + rs.Fields("ProdukttypCode") + "'>" +
rs.Fields("ProduktTyp") + "</option>");
    rs.MoveNext();
}
rs.Close();
%>
</select>
```

Beispielausgabe

```
Eisen
Hölzer
Putter
Golfzubehör
```

Gilt für folgende Objekte:

Response-Objekt

WriteFile-Methode

Fügt in die vom Browser empfangene Antwort den Inhalt einer Datei ein.

Dieser Text wird nicht von der serverseitigen JavaScript-Engine verarbeitet.

Parameter

Dateiname: Zeichenkette, Obligatorisch

Beispiel

Dieses Beispiel übergibt den Inhalt der Datei CompanyPageFooter.txt an den Browser.

```
<% Response.WriteFile( App.Path + "CompanyPageFooter.txt" ); %>
```

Gilt für folgende Objekte:

Response-Objekt

WriteIn-Methode

Fügt der vom Browser empfangenen Antwort eine Zeichenkette ein (gefolgt von einer neuen Zeile).

Hinweis: Wenn der Inhaltstyp HTML ist und im Browser eine neue Zeile angezeigt werden soll, verwenden Sie den Tag
.

Parameter

Zeichenkette: Zeichenkette, Obligatorisch

Rückgabetyp

Ohne

Beispiel1

Dieses Beispiel gibt in separaten Zeilen Namen und Währung der einzelnen Länder aus.

```
<%
var rs = new Recordset();
rs.Open("Select Währungsbezeichng,Land from Land order by Land", "AUFUmsatz");
while(!rs.EOF)
{
    Response.Writeln(rs.Fields("Land") + " " + rs.Fields("Währungsbezeichng"));
    Response.Write("<br>");
    rs.MoveNext();
}
rs.Close();
%>
```

Ausgabe von Beispiel1

```
Australien Dollar
Österreich Schillinge
Belgien Franc
Brasilien Real
Kanada Dollar
China Renminbi
Dänemark Kronen
England Pfund
Euroland Euro
Finnland Finnmark
Frankreich Franc
Deutschland D-Mark
Italien Lire
Japan Yen
Korea Won
Mexiko Pesos
Niederlande Gulden
Spanien Peseten
Schweden Kronen
Schweiz Franken
Taiwan Neuer Dollar
USA Dollar
```

Beispiel2

Dieses Beispiel übergibt den Body-Tag an eine neue Zeile im Browser.

```
Response.Writeln('<BODY BGCOLOR="#FFFFFF"><table width="100%" bgcolor="#cddfff"
cellpadding="0" cellspacing="0"><tr><td height="1" bgcolor="#003366" width="100%"></
td></tr><tr><td><H1>Vertriebsbeauftragten-Einzelheiten</H1></td></tr></table>')
```

Gilt für folgende Objekte:

Response-Objekt

Server-Objekt

Ermöglicht den Zugriff auf Methoden und Eigenschaften auf dem Server. Die meisten dieser Methoden und Eigenschaften besitzen eine Dienstprogrammfunktion.

Eigenschaft	Beschreibung
ScriptTimeout-Eigenschaft	Setzt die Anzahl von Sekunden (oder gibt sie zurück), die der Server wartet, bevor er die Fehlermeldung ausgibt, dass es zu einem Skript-Timeout gekommen ist („Script Timeout reached").

Vorgehensweise	Beschreibung
CreateObject-Methode	Erstellt eine Instanz eines COM-Objekts (Component Object Model), dem eine Programmkennung zugewiesen wird. Dies funktioniert nur unter Windows.
FormatNumber-Methode	Formatiert eine Zahl.
HTMLEncode-Methode	Wendet auf die angegebene Zeichenkette HTML-Codierung an.
URLDecode-Methode	Wendet URL-Decodierungsregeln an. Diese Methode ist das Gegenteil der URLEncode-Methode.
URLEncode-Methode	Wendet URL-Codierungsregeln an. Diese Methode ist das Gegenteil der URLDecode-Methode.

ScriptTimeout-Eigenschaft

Setzt die Anzahl von Sekunden (oder gibt sie zurück), die der Server wartet, bevor er die Fehlermeldung ausgibt, dass es zu einem Skript-Timeout gekommen ist („Script Timeout reached"). Verwenden Sie diese Eigenschaft, um eine Endlosschleife auf dem Server zu verhindern.

Für diese Eigenschaft besteht Lese- und Schreibzugriff.

Typ

Integer (Ganzzahl)

Standardwert

90

Beispiel

```
<%
// Gibt nach 20 Sekunden die Fehlerseite zurück
Server.ScriptTimeout = 20;
var i = 0;
while (true)
{
    i++
}
%>
```

Beispielausgabe

```
Fehler
Server.ScriptTimeout überschritten (20 Sekunden) während Zeile Nummer Zeilennummer von
Seite Seitenname ausgeführt wurde.
i++
Überprüfen Sie das Server-Ereignisprotokoll für mehr Einzelheiten.
```

Gilt für folgende Objekte:

Server-Objekt

CreateObject-Methode

Erstellt eine Instanz eines COM-Objekts (Component Object Model), dem eine Programmkennung zugewiesen wird. Diese Methode funktioniert nur unter Windows, da UNIX keine COM-Unterstützung bietet.

Parameter

Programmkennung: Zeichenkette

Rückgabetyp

Objekt

Beispiel

Dieses Beispiel zeigt die Datenentnahme unter Verwendung von ADO. Weitere Informationen über ADO finden Sie in der entsprechenden Microsoft-Dokumentation.

```
<%
var oConn = Server.CreateObject("ADODB.Connection");
// Dies ist jedes gültige ODBC-DSN
oConn.Open("GOScer3");
// Dies ist jeder SQL-Wert für obiges DSN
var oRs = oConn.Execute("Select Land,Absatzländercode from Land order by Land");
Response.Writeln("<select name='LänderCd'>");
while (!oRs.EOF)
{
    Response.Writeln("<option value='" + oRs.Fields("Absatzländercode").Value + "'>" +
oRs.Fields("Land").Value + "</option>");
    oRs.MoveNext();
}
Response.Writeln("</select>");
%>
```

Gilt für folgende Objekte:

Server-Objekt

FormatNumber-Methode

Formatiert eine Zahl. Weitere Informationen über numerische Formate in Impromptu finden Sie in der Impromptu Benutzer-Onlinehilfe unter „Numerische Formatsymbole".

Parameter

Zahl: Zeichenkette, Obligatorisch

Format: Zeichenkette, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

```
<%
var myNumber = "67.0000012";
Response.Write(Server.FormatNumber(myNumber,"#,#.00"));
%>
```

Beispielausgabe

```
67.00
```

Gilt für folgende Objekte:

Server-Objekt

HTMLEncode-Methode

Wendet auf die angegebene Zeichenkette HTML-Codierung an. Beispielsweise ändert diese Methode alle Kleiner-als-Symbole (<) zu <, die Größer-als-Symbole (>) zu > und das kaufmännische Und-Zeichen (&) zu &. Weitere Informationen über die HTML-Codierung finden Sie in Ihrer HTML-Dokumentation.

Parameter

Wert: Zeichenkette

Rückgabetyp

String (Zeichenkette)

Beispiel

```
<%
var myString;
myString = "<>&'\"";
Response.Write(Server.HTMLEncode(myString));
%>
```

Beispielausgabe

```
<>&'"
```

Browser-Sequenz

```
&amp;lt;&amp;gt;&amp;amp;&amp;apos;&amp;quot;
```

Gilt für folgende Objekte:

Server-Objekt

URLDecode-Methode

Wendet URL-Decodierungsregeln an. Beispielsweise ändert diese Methode Hexadezimalwerte in ihre entsprechenden Schriftzeichen. Diese Methode ist das Gegenteil der URLEncode-Methode.

Parameter

Wert: Zeichenkette

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel codiert eine Zeichenkette und decodiert sie anschließend.

```
<%
var myString;
myString= " &^*@";
var myString2;
myString2 = ""
Response.Write(Server.URLEncode(myString));
Response.Write("<br>")
myString2 = Server.URLEncode(myString);
Response.Write(Server.URLDecode(myString2));
%>
```

Beispielausgabe

```
%20%26%5E%2A%40
&^*@
```

Gilt für folgende Objekte:

Server-Objekt

URLEncode-Methode

Wendet URL-Codierungsregeln an. Beispielsweise ändert diese Methode alle nicht-alphanumerischen Zeichen in ihre entsprechenden Hexadezimalwerte. Diese Methode ist das Gegenteil der URLDecode-Methode.

Parameter

Wert: Zeichenkette

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel codiert eine Zeichenkette und decodiert sie anschließend.

```
<%
var myString;
myString= " &^*@";
var myString2;
myString2 = ""
Response.Write(Server.URLEncode(myString));
Response.Write("<br>")
myString2 = Server.URLEncode(myString);
Response.Write(Server.URLDecode(myString2));
%>
```

Beispielausgabe

```
%20%26%5E%2A%40
&^*@
```

Gilt für folgende Objekte:

Server-Objekt

StringList-Objekt

Eine Auflistung von Zeichenkettenwerten.

Eigenschaft	Beschreibung
Count-Eigenschaft	Gibt die Anzahl der Zeichenkettenwerte in der StringList zurück.

Vorgehensweise	Beschreibung
Contains-Methode	Überprüft, ob die StringList einen Wert enthält.
Item-Methode	Ruft einen indizierten Zeichenkettenwert ab.
Join-Methode	Gibt alle verketteten Werte zurück. Es kann ein optionales Trennzeichen angegeben werden. Standardmäßig ist dies ein Komma (,).

Vorgehensweise	Beschreibung
Operator() Methode (StringList)	Gibt den Index-Operator an, mit dem auf die Elemente in der Auflistung zugegriffen wird. Sie können einen numerischen Index angeben.
toString-Methode	Gibt alle verketteten Werte zurück. Diese Methode ist identisch mit der Join-Eigenschaft, mit dem Unterschied, dass sie keine Parameter verwendet.

Count-Eigenschaft

Gibt die Anzahl der Zeichenkettenwerte in der StringList zurück.

Diese Eigenschaft ist schreibgeschützt.

Typ

Integer (Ganzzahl)

Beispiel

Dieses Beispiel fügt einer Seite ein Dropdownlistenfeld mit Ländern hinzu und gibt auf der nächsten Seite die Anzahl der gewählten Länder aus.

Seite 1

```
<select name="Länder" multiple>
<%
var rs = new Recordset;
rs.Open("Select Land from Land order by Land", "AUFUmsatz");
while(!rs.EOF)
{
    Response.Write("<option>" + rs.Fields("Land") + "</option>");
    rs.MoveNext();
}
rs.Close();
%>
</select>
```

Seite 2

```
<%
var myStringList = App.Variables("Länder");
Response.Write("Es gibt " + myStringList.Count + " Zeichenkettenelemente.");
%>
```

Beispielausgabe

```
Es gibt 22 Zeichenkettenelemente.
```

Gilt für folgende Objekte:

StringList-Objekt

Contains-Methode

Überprüft, ob die StringList einen Wert enthält.

Parameter

GesuchterWert: Zeichenkette, Obligatorisch

Rückgabetyp

Boolesch

Beispiel

Dieses Beispiel fügt einer Seite ein Dropdownlistenfeld mit Ländern hinzu und überprüft auf der nächsten Seite, ob Kanada zu den ausgewählten Ländern gehörte.

Seite 1

```
<select name="Länder" multiple>
<%
var rs = new Recordset;
rs.Open("Select Land from Land order by Land", "AUFUmsatz");
while(!rs.EOF)
{
    Response.Write("<option>" + rs.Fields("Land") + "</option>");
    rs.MoveNext();
}
rs.Close();
%>
</select>
```

Seite 2

```
<%
var myStringList = App.Variables("Länder");
// StringList-Objekt muss gefüllt werden
if (myStringList.Contains("Kanada"))
{
    Response.Write("Kanada wurde ausgewählt.");
}
else
{
    Response.Write("Kanada wurde nicht ausgewählt.");
}
%>
```

Beispielausgabe

```
Kanada wurde ausgewählt.
```

Gilt für folgende Objekte:

StringList-Objekt

Item-Methode

Ruft einen indizierten Zeichenkettenwert ab. Der Index ist nullbasiert, das heißt, er beginnt mit null.

Parameter

Index: Ganzzahl, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel fügt einer Seite ein Dropdownlistenfeld mit Ländern hinzu und zeigt auf der nächsten Seite das zweite Element der Zeichenkette an.

Seite 1

```
<select name="Länder" multiple>
<%
var rs = new Recordset;
rs.Open("Select Land from Land order by Land", "AUFUmsatz");
while(!rs.EOF)
{
    Response.Write("<option>" + rs.Fields("Land") + "</option>");
    rs.MoveNext();
}
rs.Close();
%>
</select>
```

Seite 2

```
<%
var myStringList = App.Variables("Länder");
var x = new String();
x = myStringList.Item(1);
// Da der Index bei 0 beginnt, wird das zweite Element angezeigt
Response.Write(x + " ist das zweite Element in der Zeichenkette.");
%>
```

Beispielausgabe

```
Österreich ist das zweite Element in der Zeichenkette.
```

Gilt für folgende Objekte:

StringList-Objekt

Join-Methode

Gibt alle verketteten Werte zurück. Es kann ein optionales Trennzeichen angegeben werden. Standardmäßig ist dies ein Komma (,).

Parameter

Trennzeichen: Zeichenkette, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

Dieses Beispiel fügt einer Seite ein Dropdownlistenfeld mit Ländern hinzu und zeigt auf der nächsten Seite die ausgewählten Länder mit einem Komma zwischen jedem Namen an.

Seite 1

```
<select name="Länder" multiple>
<%
var rs = new Recordset;
rs.Open("Select Land from Land order by Land", "AUFUmsatz");
while(!rs.EOF)
{
    Response.Write("<option>" + rs.Fields("Land") + "</option>");
    rs.MoveNext();
}
rs.Close();
%>
</select>
```

Seite 2

```
<%
var myStringList = App.Variables("Länder").Join(", ");
Response.Write(myStringList);
%>
```

Beispielausgabe

```
Australien, Österreich, Belgien
```

Gilt für folgende Objekte:

StringList-Objekt

Operator() Methode (StringList)

Gibt den Index-Operator an, mit dem auf die Elemente in der Auflistung zugegriffen wird. Der Index ist nullbasiert, das heißt, er beginnt mit null.

Parameter

Sie können einen numerischen Index angeben.

Index: Numerisch

Rückgabetyp

Ein Element in einer Auflistung.

Beispiel

Dieses Beispiel gibt das erste Land in der StringList zurück.

```
<%=App.Variables("Land")(0)%>
```

Gilt für folgende Objekte:

StringList-Objekt

toString-Methode

Gibt alle verketteten Werte zurück. Diese Methode ist identisch mit der Join-Eigenschaft, mit dem Unterschied, dass sie keine Parameter verwendet.

Dies ist der Standardparameter des StringList-Objekts.

Parameter

Ohne

Rückgabetyp

String (Zeichenkette)

Beispiel

```
<%
switch(App.Variables("BestellMonat").toString())
{
    case "1" :
        Response.Write("<tr><td>Bestellmonat</td><td>" + "Januar" + "</td></tr>");
        break;
...
}
%>
```

Gilt für folgende Objekte:

StringList-Objekt

Upfront-Objekt

Stellt die Integration mit dem Upfront-Server bereit, so dass PowerPrompts-Anwendungen Themen und Sprachen verwenden können.

Eigenschaft	Beschreibung
Language-Eigenschaft	Gibt die aktuell in Upfront ausgewählte Sprache zurück.
Locale-Eigenschaft	Gibt die aktuell in Upfront ausgewählte Ländereinstellung zurück.
Theme-Eigenschaft	Gibt den Namen des aktuell in Upfront ausgewählten Themas zurück.

Vorgehensweise	Beschreibung
ExecuteCommand-Methode	Wird für die Übergabe eines XML-Befehls an den Upfront-Server verwendet. Der Rückgabewert ist ein XML-Dokument, das entweder den Erfolg oder das Scheitern sowie alle Ergebnisse des Befehls anzeigt.
GetPageFragment-Methode	Gibt in Upfront ein auf dem aktuellen Thema und der aktuellen Sprache basierendes HTML-Seitenfragment zurück. Der Rückgabewert ist der Seiteninhalt, wie er von der Upfront-Vorlagen-Engine verarbeitet wurde.

Language-Eigenschaft

Gibt die aktuell in Upfront ausgewählte Sprache zurück.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

```
<%
// Dem Benutzer mitteilen, dass diese Anwendung nur auf Englisch verfügbar ist
if (Upfront.Language != "en")
{
    Response.Write("<b>Sie haben eine andere Sprache als Englisch ausgewählt. Diese
Anwendung ist nur auf Englisch verfügbar.</b>");
}
else
{
    Response.Write("<bAbfrage erfolgreich</b>");
}
%>
```

Beispielausgabe

```
Upfront mag Englisch.
```

Gilt für folgende Objekte:

Upfront-Objekt

Locale-Eigenschaft

Gibt die aktuell in Upfront ausgewählte Ländereinstellung zurück.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

```
<%
// Dem Benutzer mitteilen, welche Ländereinstellung derzeit ausgewählt ist
Response.Write("Ländereinstellung: " + Upfront.Locale);
%>
```

Beispielausgabe

```
Ländereinstellung: en-us
```

Gilt für folgende Objekte:

Upfront-Objekt

Theme-Eigenschaft

Gibt den Namen des aktuell in Upfront ausgewählten Themas zurück.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

```
<!-- Dem Benutzer mitteilen, welches Thema aktuell ausgewählt ist-->
Thema: <%=Upfront.Theme%>
```

Beispielausgabe

```
Thema: standard70
```

Gilt für folgende Objekte:

Upfront-Objekt

ExecuteCommand-Methode

Wird für die Übergabe eines XML-Befehls an den Upfront-Server verwendet. Der Rückgabewert ist ein XML-Dokument, das entweder den Erfolg oder das Scheitern sowie alle Ergebnisse des Befehls anzeigt.

Der zugehörige XML-Namespace muss für das Root-Element des Befehls festgelegt werden. Der Namespace ist http://developer.cognos.com/schemas/upfront.

Parameter

XMLBefehl: Zeichenkette, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

```
<%=Server.HTMLEncode(Upfront.ExecuteCommand("<DescribeUser xmlns=\"http://
developer.cognos.com/schemas/upfront/\"/>"))%>
```

Gilt für folgende Objekte:

Upfront-Objekt

GetPageFragment-Methode

Gibt in Upfront ein auf dem aktuellen Thema und der aktuellen Sprache basierendes HTML-Seitenfragment zurück. Der Rückgabewert ist der Seiteninhalt, wie er von der Upfront-Vorlagen-Engine verarbeitet wurde.

Parameter

SeitenfragmentName: Zeichenkette, Obligatorisch

Rückgabetyp

String (Zeichenkette)

Beispiel

```
<%=Upfront.GetPageFragment("header.html")%>
```

Gilt für folgende Objekte:

Upfront-Objekt

User-Objekt

Stellt die Benutzerinformationen für ein gültiges Ticket in Access Manager bereit.

Eigenschaft	Beschreibung
Description-Eigenschaft	Gibt die Beschreibung des aktuellen in Access Manager festgelegten Benutzers zurück.
Email-Eigenschaft	Gibt die E-Mail-Adresse des aktuellen in Access Manager festgelegten Benutzers zurück.
Telephone-Eigenschaft	Gibt die Telefonnummer des aktuellen in Access Manager festgelegten Benutzers zurück.
UserClass-Eigenschaft	Gibt den Benutzerklassennamen für den aktuellen Benutzer zurück. UserClass ist die Benutzerklasse, unter der der Bericht ausgeführt wird.
UserClasses-Eigenschaft	Führt die verfügbaren Benutzerklassen für diesen Benutzer auf. UserClasses ist die Liste aller Benutzerklassen, von denen der Benutzer momentan in Access Manager Mitglied ist.
UserName-Eigenschaft	Gibt den Namen des aktuellen in Access Manager festgelegten Benutzers zurück.

Description-Eigenschaft

Gibt die Beschreibung des aktuellen in Access Manager festgelegten Benutzers zurück.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

Dieses Beispiel zeigt den Benutzernamen, die Benutzerklasse, E-Mail-Adresse, Telefonnummer und die Beschreibung des Benutzers in einer Tabelle an.

```
<%
// Eigenschaften des Benutzers in einer Tabelle anzeigen
Response.Writeln("<table>");
Response.Writeln("<tr><td>Benutzername:</td><td>" + User.UserName + "</td></tr>");
Response.Writeln("<tr><td>Benutzerklasse:</td><td>" + User.UserClass + "</td></tr>");
Response.Writeln("<tr><td>E-Mail:</td><td>" + User.Email + "</td></tr>");
Response.Writeln("<tr><td>Telefon:</td><td>" + User.Telephone + "</td></tr>");
Response.Writeln("<tr><td>Beschreibung:</td><td>" + User.Description + "</td></tr>");
Response.Writeln("</table>");
%>
```

Gilt für folgende Objekte:

User-Objekt

Email-Eigenschaft

Gibt die E-Mail-Adresse des aktuellen in Access Manager festgelegten Benutzers zurück.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

Dieses Beispiel zeigt den Benutzernamen, die Benutzerklasse, E-Mail-Adresse, Telefonnummer und die Beschreibung des Benutzers in einer Tabelle an.

```
<%
// Eigenschaften des Benutzers in einer Tabelle anzeigen
Response.Writeln("<table>");
Response.Writeln("<tr><td>Benutzername:</td><td>" + User.UserName + "</td></tr>");
Response.Writeln("<tr><td>Benutzerklasse:</td><td>" + User.UserClass + "</td></tr>");
Response.Writeln("<tr><td>E-Mail:</td><td>" + User.Email + "</td></tr>");
Response.Writeln("<tr><td>Telefon:</td><td>" + User.Telephone + "</td></tr>");
Response.Writeln("<tr><td>Beschreibung:</td><td>" + User.Description + "</td></tr>");
Response.Writeln("</table>");
%>
```

Gilt für folgende Objekte:

User-Objekt

Telephone-Eigenschaft

Gibt die Telefonnummer des aktuellen in Access Manager festgelegten Benutzers zurück.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

Dieses Beispiel zeigt den Benutzernamen, die Benutzerklasse, E-Mail-Adresse, Telefonnummer und die Beschreibung des Benutzers in einer Tabelle an.

```
<%
// Eigenschaften des Benutzers in einer Tabelle anzeigen
Response.Writeln("<table>");
Response.Writeln("<tr><td>Benutzername:</td><td>" + User.UserName + "</td></tr>");
Response.Writeln("<tr><td>Benutzerklasse:</td><td>" + User.UserClass + "</td></tr>");
Response.Writeln("<tr><td>E-Mail:</td><td>" + User.Email + "</td></tr>");
Response.Writeln("<tr><td>Telefon:</td><td>" + User.Telephone + "</td></tr>");
Response.Writeln("<tr><td>Beschreibung:</td><td>" + User.Description + "</td></tr>");
Response.Writeln("</table>");
%>
```

Gilt für folgende Objekte:

User-Objekt

UserClass-Eigenschaft

Gibt den Benutzerklassennamen für den aktuellen Benutzer zurück. UserClass ist die Benutzerklasse, unter der der Bericht ausgeführt wird.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

Dieses Beispiel zeigt den Benutzernamen, die Benutzerklasse, E-Mail-Adresse, Telefonnummer und die Beschreibung des Benutzers in einer Tabelle an.

```
<%
// Eigenschaften des Benutzers in einer Tabelle anzeigen
Response.Writeln("<table>");
Response.Writeln("<tr><td>Benutzername:</td><td>" + User.UserName + "</td></tr>");
Response.Writeln("<tr><td>Benutzerklasse:</td><td>" + User.UserClass + "</td></tr>");
Response.Writeln("<tr><td>E-Mail:</td><td>" + User.Email + "</td></tr>");
Response.Writeln("<tr><td>Telefon:</td><td>" + User.Telephone + "</td></tr>");
Response.Writeln("<tr><td>Beschreibung:</td><td>" + User.Description + "</td></tr>");
Response.Writeln("</table>");
%>
```

Gilt für folgende Objekte:

User-Objekt

UserClasses-Eigenschaft

Führt die verfügbaren Benutzerklassen für diesen Benutzer auf. UserClasses ist die Liste aller Benutzerklassen, von denen der Benutzer momentan in Access Manager Mitglied ist.

Diese Eigenschaft ist schreibgeschützt.

Typ

StringList

Beispiel

Dieses Beispiel verknüpft alle Benutzerklassen, die für diesen Benutzer verfügbar sind.

```
<%=UserClasses.Join()%>
```

Gilt für folgende Objekte:

User-Objekt

UserName-Eigenschaft

Gibt den Namen des aktuellen in Access Manager festgelegten Benutzers zurück.

Diese Eigenschaft ist schreibgeschützt.

Typ

String (Zeichenkette)

Beispiel

Dieses Beispiel zeigt den Benutzernamen, die Benutzerklasse, E-Mail-Adresse, Telefonnummer und die Beschreibung des Benutzers in einer Tabelle an.

```
<%
// Eigenschaften des Benutzers in einer Tabelle anzeigen
Response.Writeln("<table>");
Response.Writeln("<tr><td>Benutzername:</td><td>" + User.UserName + "</td></tr>");
Response.Writeln("<tr><td>Benutzerklasse:</td><td>" + User.UserClass + "</td></tr>");
Response.Writeln("<tr><td>E-Mail:</td><td>" + User.Email + "</td></tr>");
Response.Writeln("<tr><td>Telefon:</td><td>" + User.Telephone + "</td></tr>");
Response.Writeln("<tr><td>Beschreibung:</td><td>" + User.Description + "</td></tr>");
Response.Writeln("</table>");
%>
```

Gilt für folgende Objekte:

User-Objekt

JavaScript-Berichtsänderungsmethoden

Verwenden Sie die JavaScript-Berichtsänderungsmethoden zum Ändern von Impromptu-Berichten. Verwenden Sie das Dialogfeld *Skript-Editor*, um Ihrer PowerPrompts-Anwendung diese Methoden hinzuzufügen. Diese Methoden können Sie nicht in HTML-Seiten verwenden. Weitere Informationen zu Skripten finden Sie unter "Skript: Überblick" (S. 23).

Syntaxregeln

- Report-Methoden erfordern eine vorangestellte GetReport()-Methode.
- Report Object-Methoden erfordern eine vorangestellte GetReportObject()-Methode.
- Column-Methoden erfordern eine vorangestellte GetColumnTitle()-Methode.
- Query-Methoden erfordern die vorangestellte GetQuery()-Methode.
- JavaScript unterscheidet zwischen Groß- und Kleinschreibung. Verwenden Sie also nicht ADDdataITEM, sondern AddDataItem.
- Schließen Sie Impromptu-Ausdrücke, die Sie an diese Methoden übergeben, in doppelte Anführungszeichen ein.
- Schließen Sie Zeichenkettenwerte innerhalb von Impromptu-Ausdrücken in einfache Anführungszeichen ein.

  Ein Beispiel, das diese Regeln verwendet, finden Sie unter "Hinzufügen eines Skripts" (S. 23).

Hinweis

Wenn Sie für eine Methode ein Hilfethema im Dialogfeld *Skript-Editor* öffnen möchten, wählen Sie den Methodennamen, und drücken Sie auf F1.

GetColumnTitle-Methode

Gibt den Spaltentitel für die angegebene Spalte zurück.

Syntax

```
GetColumnTitle(SpaltenName)
```

Parameter

SpaltenName: Zeichenkette

Beispiel

Dieses Beispiel ändert den Titel der Spalte Produkt zu Produktname.

```
GetColumnTitle("Produkt").SetText("Produktname");
```

Verwandte Themen

- "ApplyStyle-Spaltentitelmethode" (S. 93)
- "GetQuery-Methode" (S. 87)
- "GetReport-Methode" (S. 87)
- "GetReportObject-Methode" (S. 88)
- "SetText-Spaltentitelmethode" (S. 94)
- "SetTextJustification-Spaltentitelmethode" (S. 94)

GetQuery-Methode

Gibt die Hauptabfrage des Berichts zurück.

Syntax

```
GetQuery()
```

Beispiel

```
GetQuery().AddColumn("Gesamtmenge","Summe([Menge])");
```

Verwandte Themen

- "AddColumn-Abfragemethode" (S. 94)
- "AndFilterBy-Abfragemethode" (S. 95)
- "AssociateColumn-Abfragemethode" (S. 96)
- "GetColumnTitle-Methode" (S. 86)
- "GetReport-Methode" (S. 87)
- "GetReportObject-Methode" (S. 88)
- "GroupBy-Abfragemethode" (S. 96)
- "SetMaxRows-Abfragemethode" (S. 98)

GetReport-Methode

Gibt den Bericht mit dem angegebenen Namen zurück.

Syntax

```
GetReport()
```

Beispiel

Dieses Beispiel fügt dem Abschlussbericht die Katalogspalte *Menge* hinzu.

```
GetReport().AddDataItem("Menge","[\\Bestellungen\\Einzelheiten\\Menge]");
```

Verwandte Themen

- "ApplyTemplate-Berichtsmethode" (S. 88)
- "GetColumnTitle-Methode" (S. 86)
- "GetQuery-Methode" (S. 87)
- "GetReportObject-Methode" (S. 88)
- "RemoveReportObject-Berichtsmethode" (S. 89)
- "SetListInsertCursor-Berichtsmethode" (S. 89)
- "SetPrimaryFrame-Berichtsmethode" (S. 90)

GetReportObject-Methode

Gibt das Berichtsobjekt mit dem angegebenen Namen zurück.

Syntax

```
GetReportObject(repObjName)
```

Parameter

repObjName: Zeichenkette

Beispiel

Dieses Beispiel weist der Spalte Spal1 das Format mit dem Namen UglyRed zu.

```
GetReportObject("Spal1").ApplyStyle("UglyRed");
```

Verwandte Themen

- "ApplyStyle-Berichtsobjektmethode" (S. 91)
- "GetColumnTitle-Methode" (S. 86)
- "GetQuery-Methode" (S. 87)
- "GetReport-Methode" (S. 87)
- "SetText-Berichtsobjektmethode" (S. 92)
- "VerticalAlign-Berichtsobjektmethode" (S. 93)

AddDataItem-Berichtsmethode

Fügt der Abfrage den angegebenen Ausdruck als Spalte und ein entsprechendes Berichtsobjekt hinzu. Das Objekt wird an der Standardposition des Primärbausteins eingefügt.

Syntax

```
AddDataItem(repObjName,Daten)
```

Parameter

repObjName: Zeichenkette

Daten: Zeichenkette

Beispiel

Dieses Beispiel fügt dem Abschlussbericht die Katalogspalte *Menge* hinzu.

```
GetReport().AddDataItem("Menge","[\\Bestellungen\\Einzelheiten\\Menge]");
```

Verwandte Themen

- "ApplyTemplate-Berichtsmethode" (S. 88)
- "GetReport-Methode" (S. 87)
- "GetReportObject-Methode" (S. 88)
- "RemoveReportObject-Berichtsmethode" (S. 89)
- "SetListInsertCursor-Berichtsmethode" (S. 89)
- "SetPrimaryFrame-Berichtsmethode" (S. 90)

ApplyTemplate-Berichtsmethode

Weist dem Bericht die angegebene Vorlage zu.

Syntax

```
ApplyTemplate(VorlageName)
```

Parameter

VorlageName: Zeichenkette

Beispiel1

Dieses Beispiel weist dem Bericht eine Vorlage mit dem Namen Laventus.imt zu. Laventus.imt befindet sich im Ordner C:\PowerPrompts Apps.

```
GetReport().ApplyTemplate("C:\\PowerPrompts Apps\\Laventus.imt");
```

Beispiel2

Dieses Beispiel weist dem Bericht eine Vorlage mit dem Namen Laventus.imt zu. Die Datei Laventus.imt befindet sich im Ordner Vorlagen, einem Unterordner des PowerPrompts-Anwendungsordners.

```
GetReport().ApplyTemplate(".\\Vorlagen\\Laventus.imt");
```

Hinweis: Der Vorlagenname muss immer angegeben werden, Sie können jedoch einen absoluten oder einen relativen Pfad angeben. Relative Pfade verhalten sich relativ zum Berichtspfad. Den Pfad zu der Anwendung erhalten Sie mithilfe der App.Path-Eigenschaft.

Verwandte Themen

- "AddDataItem-Berichtsmethode" (S. 88)
- "GetReport-Methode" (S. 87)
- "RemoveReportObject-Berichtsmethode" (S. 89)
- "SetListInsertCursor-Berichtsmethode" (S. 89)
- "SetPrimaryFrame-Berichtsmethode" (S. 90)

RemoveReportObject-Berichtsmethode

Entfernt das angegebene Berichtsobjekt aus dem Bericht.

Diese Methode ist auf verborgene Spalten in einem Impromptu-Bericht nicht anwendbar.

Syntax

```
RemoveReportObject(repObjName)
```

Parameter

repObjName: Zeichenkette

Beispiel

Dieses Beispiel entfernt das Berichtsobjekt mit dem Namen *Land* aus dem Abschlussbericht.

```
GetReport().RemoveReportObject("Land");
```

Verwandte Themen

- "AddDataItem-Berichtsmethode" (S. 88)
- "ApplyTemplate-Berichtsmethode" (S. 88)
- "GetReport-Methode" (S. 87)
- "SetListInsertCursor-Berichtsmethode" (S. 89)
- "SetPrimaryFrame-Berichtsmethode" (S. 90)

SetListInsertCursor-Berichtsmethode

Legt fest, wo Listenbausteinen Spalten hinzugefügt werden. Standardmäßig werden neue Spalten rechts von vorhandenen Spalten eingefügt.

Wert	Einfügeposition der neuen Spalten
-1	rechts neben den vorhandenen Spalten
0	links neben den vorhandenen Spalten
1	rechts neben der ersten vorhandenen Spalte

Wert	Einfügeposition der neuen Spalten
2	rechts neben der zweiten vorhandenen Spalte
N (dabei ist N eine positive Ganzzahl)	rechts neben der n-ten vorhandenen Spalte

Syntax

```
SetListInsertCursor(SpalteIndexEinfügen)
```

Parameter

SpalteIndexEinfügen: Ganzzahl

Beispiel

```
GetReport().SetListInsertCursor(0);
```

Hinweis: Im Dialogfeld *Skript-Editor* muss diese Methode oberhalb der AddDataItem-Methode erscheinen, da das Skript den Einfügepunkt vor der Hinzufügung der neuen Spalte festlegen muss.

Verwandte Themen

- "AddDataItem-Berichtsmethode" (S. 88)
- "ApplyTemplate-Berichtsmethode" (S. 88)
- "GetReport-Methode" (S. 87)
- "RemoveReportObject-Berichtsmethode" (S. 89)
- "SetPrimaryFrame-Berichtsmethode" (S. 90)

SetPrimaryFrame-Berichtsmethode

Legt den Primärbaustein für den Bericht fest. Der Primärbaustein ist die Stelle, an der Impromptu neue Berichtsobjekte hinzufügt.

Syntax

```
SetPrimaryFrame(BausteinName)
```

Parameter

BausteinName: Zeichenkette

Beispiel

Dieses Beispiel legt *Produkttyp-Fußzeile* als Primärbaustein für den Bericht fest.

```
GetReport().SetPrimaryFrame("Produkttyp-Fußzeile");
```

Verwandte Themen

- "AddDataItem-Berichtsmethode" (S. 88)
- "ApplyTemplate-Berichtsmethode" (S. 88)
- "GetReport-Methode" (S. 87)
- "RemoveReportObject-Berichtsmethode" (S. 89)
- "SetListInsertCursor-Berichtsmethode" (S. 89)

AddConditionalFormat-Berichtsobjektmethode

Fügt dem Berichtsobjekt ein bedingtes Formatobjekt hinzu. Ein bedingtes Format setzt sich aus einer benannten Bedingung und einem benannten Format zusammen.

Syntax

```
AddConditionalFormat(BenannteBedingung,FormatName)
```

Parameter

BenannteBedingung: Zeichenkette

FormatName: Zeichenkette

Beispiel

Dieses Beispiel fügt dem Menge-Objekt ein bedingtes Format hinzu.

```
GetReportObject("Menge").AddConditionalFormat("Small Amount","Bad");
```

Verwandte Themen

- "ApplyStyle-Berichtsobjektmethode" (S. 91)
- "GetReportObject-Methode" (S. 88)
- "HorizontalAlign-Berichtsobjektmethode" (S. 91)
- "HorizontalAlignToColumn-Berichtsobjektmethode" (S. 92)
- "SetText-Berichtsobjektmethode" (S. 92)
- "SetTextJustification-Berichtsobjektmethode" (S. 93)
- "VerticalAlign-Berichtsobjektmethode" (S. 93)

ApplyStyle-Berichtsobjektmethode

Wendet das benannte Format auf das Berichtsobjekt an. Das Format muss in der Datei Impromptu.ini auf dem Server vorhanden sein, auf dem die PowerPrompts-Anwendung ausgeführt wird.

Syntax

```
ApplyStyle(FormatName)
```

Parameter

FormatName: Zeichenkette

Beispiel

```
GetReportObject("Produkt").ApplyStyle("Seeing Red 1");
```

Verwandte Themen

- "AddConditionalFormat-Berichtsobjektmethode" (S. 90)
- "GetReportObject-Methode" (S. 88)
- "HorizontalAlign-Berichtsobjektmethode" (S. 91)
- "HorizontalAlignToColumn-Berichtsobjektmethode" (S. 92)
- "SetText-Berichtsobjektmethode" (S. 92)
- "SetTextJustification-Berichtsobjektmethode" (S. 93)
- "VerticalAlign-Berichtsobjektmethode" (S. 93)

HorizontalAlign-Berichtsobjektmethode

Richtet das Berichtsobjekt horizontal relativ zu seinem übergeordneten Objekt aus.

Syntax

```
HorizontalAlign(AusrichtungTyp)
```

Parameter

AusrichtungTyp: Zeichenkette (links, rechts oder zentriert)

Beispiel

```
GetReportObject("Titel").HorizontalAlign("Zentriert");
```

Verwandte Themen

- "AddConditionalFormat-Berichtsobjektmethode" (S. 90)
- "ApplyStyle-Berichtsobjektmethode" (S. 91)
- "GetReportObject-Methode" (S. 88)
- "HorizontalAlignToColumn-Berichtsobjektmethode" (S. 92)
- "SetText-Berichtsobjektmethode" (S. 92)
- "SetTextJustification-Berichtsobjektmethode" (S. 93)
- "VerticalAlign-Berichtsobjektmethode" (S. 93)

HorizontalAlignToColumn-Berichtsobjektmethode

Richtet das Berichtsobjekt horizontal relativ zu der angegebenen Spalte aus. Verwenden Sie diese Methode beispielsweise für Auswertungen in Fußzeilen.

Syntax

```
HorizontalAlignToColumn(AusrichtungsspalteName,AusrichtungTyp)
```

Parameter

AusrichtungsspalteName: Zeichenkette

AusrichtungTyp: Zeichenkette (links, rechts oder zentriert)

Beispiel

```
GetReportObject("Gesamtmenge").HorizontalAlignToColumn("Menge","Rechts");
```

Verwandte Themen

- "AddConditionalFormat-Berichtsobjektmethode" (S. 90)
- "ApplyStyle-Berichtsobjektmethode" (S. 91)
- "GetReportObject-Methode" (S. 88)
- "HorizontalAlign-Berichtsobjektmethode" (S. 91)
- "SetText-Berichtsobjektmethode" (S. 92)
- "SetTextJustification-Berichtsobjektmethode" (S. 93)
- "VerticalAlign-Berichtsobjektmethode" (S. 93)

SetText-Berichtsobjektmethode

Legt den Text für einen Textbaustein fest.

Syntax

```
SetText(Text)
```

Parameter

Text: Zeichenkette

Beispiel

```
GetReportObject("Titel").SetText("Finanzbericht");
```

Verwandte Themen

- "AddConditionalFormat-Berichtsobjektmethode" (S. 90)
- "ApplyStyle-Berichtsobjektmethode" (S. 91)
- "GetReportObject-Methode" (S. 88)
- "HorizontalAlign-Berichtsobjektmethode" (S. 91)
- "HorizontalAlignToColumn-Berichtsobjektmethode" (S. 92)
- "SetTextJustification-Berichtsobjektmethode" (S. 93)
- "VerticalAlign-Berichtsobjektmethode" (S. 93)

SetTextJustification-Berichtsobjektmethode

Richtet den Text innerhalb des Textbausteins aus.

Syntax

```
SetTextJustification(AusrichtungTyp)
```

Parameter

AusrichtungTyp: Zeichenkette (links, rechts oder zentriert)

Beispiel

```
GetReportObject("Produkt").SetTextJustification("Rechts");
```

Verwandte Themen

- "AddConditionalFormat-Berichtsobjektmethode" (S. 90)
- "ApplyStyle-Berichtsobjektmethode" (S. 91)
- "GetReportObject-Methode" (S. 88)
- "HorizontalAlign-Berichtsobjektmethode" (S. 91)
- "HorizontalAlignToColumn-Berichtsobjektmethode" (S. 92)
- "SetText-Berichtsobjektmethode" (S. 92)
- "VerticalAlign-Berichtsobjektmethode" (S. 93)

VerticalAlign-Berichtsobjektmethode

Richtet das Berichtsobjekt vertikal relativ zu seinem übergeordneten Baustein aus.

Syntax

```
VerticalAlign(AusrichtungTyp)
```

Parameter

AusrichtungTyp: Zeichenkette (oben, unten oder zentriert)

Beispiel

```
GetReportObject("Gesamtmenge").VerticalAlign("Zentriert");
```

Verwandte Themen

- "AddConditionalFormat-Berichtsobjektmethode" (S. 90)
- "ApplyStyle-Berichtsobjektmethode" (S. 91)
- "GetReportObject-Methode" (S. 88)
- "HorizontalAlign-Berichtsobjektmethode" (S. 91)
- "HorizontalAlignToColumn-Berichtsobjektmethode" (S. 92)
- "SetText-Berichtsobjektmethode" (S. 92)
- "SetTextJustification-Berichtsobjektmethode" (S. 93)

ApplyStyle-Spaltentitelmethode

Wendet das benannte Format auf den Spaltentitel an.

Syntax

```
ApplyStyle(FormatName)
```

Parameter

FormatName: Zeichenkette

Beispiel

```
GetColumnTitle("Produkt").ApplyStyle("Seeing Red 1");
```

Verwandte Themen

- "GetColumnTitle-Methode" (S. 86)
- "SetText-Spaltentitelmethode" (S. 94)
- "SetTextJustification-Spaltentitelmethode" (S. 94)

SetText-Spaltentitelmethode

Legt den Text für einen Spaltentitel fest.

Syntax

```
SetText(Text)
```

Parameter

Text: Zeichenkette

Beispiel

```
GetColumnTitle("Produkttyp").SetText("Typ des Produkts");
```

Verwandte Themen

- "ApplyStyle-Spaltentitelmethode" (S. 93)
- "GetColumnTitle-Methode" (S. 86)
- "SetTextJustification-Spaltentitelmethode" (S. 94)

SetTextJustification-Spaltentitelmethode

Richtet den Text innerhalb des Textbausteins aus.

Syntax

```
SetTextJustification(AusrichtungTyp)
```

Parameter

AusrichtungTyp: Zeichenkette (links, rechts oder zentriert)

Beispiel

```
GetColumnTitle("Produkt").SetTextJustification("Rechts");
```

Verwandte Themen

- "ApplyStyle-Spaltentitelmethode" (S. 93)
- "GetColumnTitle-Methode" (S. 86)
- "SetText-Spaltentitelmethode" (S. 94)

AddColumn-Abfragemethode

Fügt der Abfrage eine neue Spalte mit dem angegebenen Namen und Ausdruck hinzu.

Syntax

```
AddColumn(SpaltenName,Daten)
```

Parameter

Spaltenname: Zeichenkette

Daten: Zeichenkette

Beispiel

```
GetQuery().AddColumn("Gesamtmenge","Summe([Menge])");
```

Verwandte Themen

- "AssociateColumn-Abfragemethode" (S. 96)
- "GetQuery-Methode" (S. 87)
- "Impromptu-Ausdrücke in PowerPrompts" (S. 99)
- "RemoveColumn-Abfragemethode" (S. 97)

AddNamedCondition-Abfragemethode

Fügt der Abfrage eine benannte Bedingung hinzu. Verwenden Sie diese Methode, um bedingte Formatierungen zu definieren.

Syntax

```
AddNamedCondition(Bedingungsname,Bedingung)
```

Parameter

Bedingungsname: Zeichenkette

Bedingung: Zeichenkette

Beispiel

```
GetQuery().AddNamedCondition("Geringe Umsätze","[\\Produkte\\Umsatzentwicklung\\Umsatz
95] < 5000");
```

Verwandte Themen

- "GetQuery-Methode" (S. 87)

AndFilterBy-Abfragemethode

Fügt dem aktuellen Filter die angegebene Bedingung hinzu. Wenn der aktuelle Filter nicht leer ist, wird die Bedingung unter Verwendung von AND hinzugefügt.

Hinweis: Diese Methode kann nicht mit Impromptu-Berichten verwendet werden, bei denen SQL für die Abfrage geschrieben wurde.

Syntax

```
AndFilterBy(Bedingung)
```

Parameter

Bedingung: Zeichenkette

Beispiel

```
GetQuery().AndFilterBy("Jahr([Bestelldatum]) in (95,96)");
```

Verwandte Themen

- "AndSummaryFilterBy-Abfragemethode" (S. 95)
- "GetQuery-Methode" (S. 87)
- "OrFilterBy-Abfragemethode" (S. 96)

AndSummaryFilterBy-Abfragemethode

Fügt dem aktuellen Auswertungsfilter die angegebene Bedingung hinzu. Wenn der aktuelle Filter nicht leer ist, wird die Bedingung unter Verwendung von AND hinzugefügt.

Hinweis: Diese Methode kann nicht mit Impromptu-Berichten verwendet werden, bei denen SQL für die Abfrage geschrieben wurde.

Syntax

```
AndSummaryFilterBy(Bedingung)
```

Parameter

Bedingung: Zeichenkette

Beispiel

```
GetQuery().AndSummaryFilterBy("[Gesamtumsatz] > 10000");
```

Verwandte Themen

- "AndFilterBy-Abfragemethode" (S. 95)
- "GetQuery-Methode" (S. 87)
- "OrSummaryFilterBy-Abfragemethode" (S. 97)

AssociateColumn-Abfragemethode

Verbindet die angegebene Spalte mit der angegebenen Gruppenspalte.

Syntax

```
AssociateColumn(Spaltenname,GruppenspalteName)
```

Parameter

Spaltenname: Zeichenkette

GruppenspalteName: Zeichenkette

Beispiel

```
GetQuery().AssociateColumn("Gesamtumsatz","Land");
```

Verwandte Themen

- "AddColumn-Abfragemethode" (S. 94)
- "GetQuery-Methode" (S. 87)
- "RemoveColumn-Abfragemethode" (S. 97)

GroupBy-Abfragemethode

Fügt die benannte Spalte am Ende der gruppierten Spalten hinzu. Die Spalte wird je nach Festlegung des Sortiertyps auf- oder absteigend sortiert.

Syntax

```
GroupBy(Spaltenname,SortierungTyp)
```

Parameter

Spaltenname: Zeichenkette

SortierungTyp: Zeichenkette (aufsteigend oder absteigend)

Beispiel

```
GetQuery().GroupBy("Kunde","Absteigend");
```

Verwandte Themen

- "AddColumn-Abfragemethode" (S. 94)
- "GetQuery-Methode" (S. 87)
- "SortBy-Abfragemethode" (S. 98)

OrFilterBy-Abfragemethode

Fügt dem aktuellen Filter die angegebene Bedingung hinzu. Wenn der aktuelle Filter nicht leer ist, wird die Bedingung unter Verwendung von OR hinzugefügt.

Hinweis: Diese Methode kann nicht mit Impromptu-Berichten verwendet werden, bei denen SQL für die Abfrage geschrieben wurde.

Syntax

```
OrFilterBy(Bedingung)
```

Parameter

Bedingung: Zeichenkette

Beispiel

```
GetQuery().OrFilterBy("[\\Produkte\\Produkt] beginnt mit('AUF')");
```

Verwandte Themen

- "AndFilterBy-Abfragemethode" (S. 95)
- "GetQuery-Methode" (S. 87)
- "OrSummaryFilterBy-Abfragemethode" (S. 97)

OrSummaryFilterBy-Abfragemethode

Fügt dem aktuellen Auswertungsfilter die angegebene Bedingung hinzu. Wenn der aktuelle Filter nicht leer ist, wird die Bedingung unter Verwendung von OR hinzugefügt.

Hinweis: Diese Methode kann nicht mit Impromptu-Berichten verwendet werden, bei denen SQL für die Abfrage geschrieben wurde.

Syntax

```
OrSummaryFilterBy(Bedingung)
```

Parameter

Bedingung: Zeichenkette

Beispiel

```
GetQuery().OrSummaryFilterBy("[Land] = 'Kanada'");
```

Verwandte Themen

- "AndSummaryFilterBy-Abfragemethode" (S. 95)
- "GetQuery-Methode" (S. 87)
- "OrFilterBy-Abfragemethode" (S. 96)

RemoveColumn-Abfragemethode

Entfernt die angegebene Spalte aus der Abfrage sowie alle Berichtsobjekte, die auf die Spalte verweisen.

Syntax

```
RemoveColumn(Spaltenname)
```

Parameter

Spaltenname: Zeichenkette

Beispiel

```
GetQuery().RemoveColumn("Produkttyp");
```

Verwandte Themen

- "AddColumn-Abfragemethode" (S. 94)
- "GetQuery-Methode" (S. 87)
- "AssociateColumn-Abfragemethode" (S. 96)

SetMaxRows-Abfragemethode

Legt die maximale Anzahl von Zeilen fest, die die Abfrage im Ergebnissatz zurückgibt.

Syntax

```
SetMaxRows(maxZeilen)
```

Parameter

maxZeilen: Ganzzahl

Beispiel

```
GetQuery().SetMaxRows(3000);
```

Verwandte Themen

- "GetQuery-Methode" (S. 87)

SetPromptValue-Abfragemethode

Legt den Wert für die Eingabeaufforderung in dem mit der PowerPrompts-Anwendung verbundenen Impromptu-Bericht fest. Um sicher zu stellen, dass der Impromptu-Bericht mehrere Eingabewerte akzeptiert, verwenden Sie in der Filteranweisung des Berichts anstelle eines Ist-Gleich-Operators (=) einen In-Operator.

Tipp: Verwenden Sie zum Trennen der Werte ein Komma. Wenn ein Komma als Teil des Wertes erforderlich ist, stellen Sie dem Komma ein Escape-Zeichen voran. Standardmäßig ist das Caret-Zeichen (^) das Escape-Zeichen. Dieses können Sie jedoch ändern, indem Sie den Eintrag für *Separator Escape Character* im Abschnitt [Startup Options] der Datei Impromptu.ini aktualisieren.

Syntax

```
SetPromptValue(Aufforderungsname,Wert1,...,WertN)
```

Parameter

Verwenden Sie für den Aufforderungsname-Parameter den Namen der Eingabeaufforderung aus dem Feld *Verfügbare Eingabeaufforderungen* im Dialogfeld *Eingabeaufforderungs-Manager*.

Aufforderungsname: Zeichenkette

Wert1-WertN: Variant

Beispiele

Im folgenden Beispiel ist „Filterdatum" der Name und „28.02.1999" der Wert der Eingabeaufforderung.

```
GetQuery().SetPromptValue("Filterdatum","28.02.1999")
GetQuery().SetPromptValue("Ländercode","App.Variables("meineVar")");
```

Verwandte Themen

- "GetQuery-Methode" (S. 87)

SortBy-Abfragemethode

Fügt die benannte Spalte am Ende der sortierten Spalten hinzu. Die Spalte wird je nach Festlegung des Sortiertyps auf- oder absteigend sortiert.

Syntax

```
SortBy(Spaltenname,SortierungTyp)
```

Parameter

Spaltenname: Zeichenkette

SortierungTyp: Zeichenkette (aufsteigend oder absteigend)

Beispiel

```
GetQuery().SortBy("Land","Aufsteigend")
```

Verwandte Themen

- "AddColumn-Abfragemethode" (S. 94)
- "GetQuery-Methode" (S. 87)
- "GroupBy-Abfragemethode" (S. 96)

Impromptu-Ausdrücke in PowerPrompts

Durch Impromptu-Ausdrücke werden Abfragen modifiziert. Sie werden in Berechnungen, Filtern und Bedingungen verwendet. Impromptu-Ausdrücke in PowerPrompts und Impromptu sind ähnlich, jedoch nicht identisch.

In PowerPrompts geben Sie Impromptu-Ausdrücke direkt in das Dialogfeld *Skript-Editor* ein, der Code ist jedoch JavaScript. Komplexe Impromptu-Ausdrücke erstellen Sie zunächst in Impromptu (unter Verwendung eines Ausdruckseditors) und anschließend mit den folgenden JavaScript-Änderungen erneut in PowerPrompts:

- Entfernen Sie Leerzeichen aus Funktionsnamen, beispielsweise wird aus „starts with" die Zeichenfolge „startswith".
- Verwenden Sie Unterstriche statt Bindestrichen für Funktionsnamen, aus DATUM-IN-ZEICHENKETTE wird DATUM_IN_ZEICHENKETTE.
- Schließen Sie Katalogverweise in eckige Klammern ein, und verwenden Sie einen doppelten umgekehrten Schrägstrich am Anfang, beispielsweise: [\\Admin.\\Land\\LänderCd].
- Unterscheiden Sie Abfrageverweise von Katalogverweisen, indem Sie sie nicht mit einem Schrägstrich beginnen, beispielsweise [Spal 1].
- Schließen Sie Eingabeaufforderungen im Bericht mit Fragezeichen ein, beispielsweise ?Aufforderungsname?.
- Verweisen Sie auf im Katalog gespeicherte Eingabeaufforderungen wie auf alle anderen Katalogspalten.

Die folgende Tabelle zeigt zwei Impromptu-Ausdrücke und ihre Entsprechungen in PowerPrompts:

PowerPrompts-Ausdruck	Impromptu-Ausdruck
[\Bestellungen\Bestelldatum] >= 1996-01-01	Bestelldatum >= 1996-01-01
[\Datum] startswith '1999'	Datum starts with '1996'

Wenn Sie eines der folgenden Zeichen in einen Impromptu-Ausdruck einfügen, müssen sie ihm einen umgekehrten Schrägstrich voranstellen:

- ein einfaches Anführungszeichen (oder Apostroph) in einem Zeichenkettenliteral
- ein umgekehrter Schrägstrich in einem Zeichenkettenliteral
- eine eckige Klammer in einem Katalogverweis

Hinweise

- Namen unterscheiden nicht zwischen Groß- und Kleinschreibung, „startswith" ist beispielsweise identisch mit „StartsWith".
- Zwischen Operatoren sind keine Leerzeichen erforderlich „x+3*abs(y)" ist beispielsweise identisch mit „x + 3 * abs (y)".
- Numerische Konstanten müssen mit einer Zahl beginnen, aus „,3" wird beispielsweise „0,3".

Weitere Informationen über Impromptu-Ausdrücke finden Sie im Referenz-Abschnitt der Impromptu-Online-Hilfe.

JavaScript-Clientmethoden

JavaScript-Clientmethoden werden mit der JavaScript-Clientbibliothek geliefert.

Die folgenden Methoden können nur auf Ihren HTML-Seiten verwendet werden:

- FormatUserVar-Methode
- GetUserVar-Methode
- GetUserVarValues-Methode

Schließen Sie diese Methoden nicht in spitze Klammern (<%...%>) ein, da es sich dabei um clientseitige JavaScript-Methoden handelt.

Vorgehensweise

- Wenn Sie diese Methoden verwenden möchten, müssen Sie Folgendes in Ihre HTML-Datei eingeben:

```
<script language= "javascript" src="/cognos/PowerPrompts/PowerPromptLib.js"></
script>
```

FormatUserVar-Methode

Verwenden Sie diese Methode, um die Werte einer Variable entsprechend der angegebenen Formatzeichenkette und dem Listentrennzeichen zu formatieren. Der Benutzer legt diese Variable zum Abschluss der PowerPrompts-Anwendung fest. Das Prozentzeichen (%) fungiert als Platzhalterzeichen. Geben Sie das Prozentzeichen zweimal ein (%%), wenn Sie ein Prozentzeichen (%) darstellen möchten.

Syntax

```
FormatUserVar(varName,Format,Trennzeichen)
```

Parameter

Variablenname: Zeichenkette

Format: Zeichenkette

Trennzeichen: Zeichenkette

Beispiel

```
<html>
<head>
<script language="javascript" src="/cognos/PowerPrompts/PowerPromptLib.js"></script>
</head>
<body>
<Form>
</form>
Show me values in Country:
<script language="javascript">
document.write(FormatUserVar("'Länderliste","state = '%'"," AND "));
</script>
</body>
</html>
```

Rückgabe:

```
state = 'AK' AND state = 'VT' AND state = 'AL'
```

Verwandte Themen

- "GetUserVar-Methode" (S. 100)
- "GetUserVarValues-Methode" (S. 101)
- "JavaScript-Clientmethoden" (S. 100)

GetUserVar-Methode

Gibt den Wert einer Variable zurück, die der Benutzer als Teil des Abschlusses der PowerPrompts-Anwendung festlegt. Wenn mit der Variable mehrere Werte verbunden sind, werden diese durch Komma getrennt.

Syntax

```
GetUserVar(Variablenname)
```

Parameter

Variablenname: Zeichenkette

Beispiel

```
<html>
<head>
<script language="javascript" src="/cognos/PowerPrompts/PowerPromptLib.js"></script>
</head>
<body>
<Form>
</form>
Show me values in Country:
<script language="javascript">
document.write(GetUserVar("Land"));
</script>
</body>
</html>
```

Verwandte Themen

- "FormatUserVar-Methode" (S. 100)
- "GetUserVarValues-Methode" (S. 101)
- "JavaScript-Clientmethoden" (S. 100)

GetUserVarValues-Methode

Gibt eine Reihe der Werte für die benannte Variable zurück.

Syntax

```
GetUserVarValues(Variablenname)
```

Parameter

Variablenname: Zeichenkette

Beispiel

```
<html>
<head>
<script language="javascript" src="/cognos/PowerPrompts/PowerPromptLib.js"></script>
</head>
<body>
<Form>
</form>
Show me values in Country:
<script language="javascript">
document.write(GetUserVarValues("Ländercode"));
</script>
</body>
</html>
```

Verwandte Themen

- "FormatUserVar-Methode" (S. 100)
- "GetUserVar-Methode" (S. 100)
- "JavaScript-Clientmethoden" (S. 100)

Index

Symbole

#else-Anweisung, 40
#endif -Anweisung, 40
#ifdef-Anweisung, 40
#include-Anweisung, 40

A

Abfragemethoden
  AddColumn, 94
  AddNamedCondition, 95
  AndFilterBy, 95
  AndSummaryFilterBy, 95
  AssociateColumn, 96
  GroupBy, 96
  OrFilterBy, 96
  OrSummaryFilterBy, 97
  RemoveColumn, 97
  SetMaxRows, 98
  SetPromptValue, 98
  SortBy, 98
Abrufen
  Benutzerkennung und Kennwort, 34
Access Manager
  Benutzerkennung und Kennwort abrufen, 34
AddColumn, 94
AddConditionalFormat, 90
AddDataItem, 88
AddNamedCondition, 95
ADO, 34
Ändern
  Standard-HTML-Editor, 11
AndFilterBy, 95
AndSummaryFilterBy, 95
Anforderungen für PowerPrompts, 8
Anwendung
  testen, 25
  überprüfen, 24
Anzeigen eines generierten Skripts, 26
Apostroph, 99
App.BackURL, 42
App.CurrentPage, 43
App.Errors, 43
App.FinalURL, 44
App.IsTestMode, 44
App.Path, 45
App.ReportScript, 45
App.RunDynamo, 46
App.Variables, 45
AppendCookie-Methode, 69
AppendHeader-Methode, 69
ApplyStyle, 91, 93
ApplyTemplate, 88
App-Objekt, 41
AssociateColumn, 96
Ausführen der Anwendung, 25
Auswählen eines Berichts, 24
Auswahllisten
  Impromptu-Benutzerklassenfilter hinzufügen, 31
  kaskadierend erstellen, 32
  mit Direkteingabe erstellen, 32

B

BackURL-Eigenschaft, 42
Bearbeiten von Seiten, 15
Bedingungen, 18
Beheben von Skriptfehlern, 25
Benutzerkennung, 34
Benutzeroptionen
  Auswertung erstellen, 30
  zwischen Sitzungen speichern, 30
Berichtsmethoden
  AddDataItem, 88
  ApplyTemplate, 88
  RemoveReportObject, 89
  SetListInsertCursor, 89
  SetPrimaryFrame, 90
Berichtsobjektmethoden
  AddConditionalFormat, 90
  ApplyStyle, 91
  HorizontalAlign, 91
  HorizontalAlignToColumn, 92
  SetText, 92
  SetTextJustification, 93
  VerticalAlign, 93

C

ClearContent-Methode, 70
ClearHeaders-Methode, 70
Clear-Methode, 69
Close-Methode, 63
Connection.Execute, 47
Connection-Objekt, 47
Contains-Methode, 77
ContentType-Eigenschaft, 68
Cookies-Eigenschaft, 66
Copyright, 2
Count-Eigenschaft, 77
CreateObject-Methode, 74
CurrentPage-Eigenschaft, 43
CurrentRecordIndex-Eigenschaft, 60

D

Datenbankdefinition, 19
Datenbanken
  Zugriff unter Verwendung von ADO, 34
Datenquelle, 19, 21
Description-Eigenschaft, 83
Dokument
  Version, 2
Dynamos, 19, 20, 21

E

Eigenschaften
  BackURL, 42
  ContentType, 68
  Cookies, 66
  Count, 77
  CurrentPage, 43
  CurrentRecordIndex, 60
  Description, 83
  Email, 84
  EOF, 61
  Errors, 43
  Fields, 62
  FinalURL, 44
  HTMLEncodedValue, 48
  Index, 48
  IsTestMode, 44
  Language, 81
  Locale, 82
  MaxRecords, 62
  Name, 49
  Path, 45
  ReportScript, 45
  ScriptTimeOut, 73
  ServerVariables, 66
  SQL, 54
  Telephone, 84
  Theme, 82
  Type, 50
  UserClass, 85
  UserClasses, 85
  UserName, 86
  Value, 51
  Variables, 45, 67
Einfaches Anführungszeichen, 99
Einführung in PowerPrompts, 7
Einstellung
  HTML-Vorlage, 11
  Standard-HTML-Editor, 11
Email-Eigenschaft, 84
EOF-Eigenschaft, 61
Errors-Eigenschaft, 43
Erstellen
  Auswahllliste mit Direkteingabe, 32
  Auswertung der vom Benutzer gewählten Optionen, 30
  Dynamos, 20, 21
  kaskadierende Auswahlliste, 32
  neue Anwendung, 12
  unterschiedliches Design für Upfront-Themen, 35
ExecuteCommand-Methode, 82
Execute-Methode, 47

F

Fehler
  Anwendung, 24
  Skript, 25
Fehlerseite, 13
Field-Objekt, 47
Fields.HTMLEncodedValue, 48
Fields.Index, 48
Fields.Name, 49
Fields.Type, 50
Fields.Value, 51
Fields-Eigenschaft, 62
Filtern
  Berichte nach mehreren Werten, 29
FinalURL-Eigenschaft, 44
FormatNumber-Methode, 74
FormatUserVar (JavaScript), 100

G

Gemeinsam verwenden
  JavaScript-Code, 40
GetColumnTitle, 86
GetPageFragment-Methode, 83
GetQuery, 87
GetReport, 87
GetReportObject, 88
GetUserVar, 100
GetUserVarValues, 101
GroupBy, 96

H

Hinzufügen
  Grafiken, 11
  Impromptu-Benutzerklassenfilter, 31
  Seiten, 14
  Skript, 23
  Verknüpfungen, 17
  Verknüpfungsbedingungen, 18
HorizontalAlign, 91
HorizontalAlignToColumn, 92
HTML-Editor, 11
HTMLEncodedValue-Eigenschaft, 48
HTMLEncode-Methode, 75
HTML-Vorlage, 11

I

Importieren von Seiten, 14
Impromptu-Ausdrücke, 99
Impromptu-Benutzerklassenfilter, 31
include-Anweisung, 40
Index-Eigenschaft, 48
Indexoperator, 52, 79
Informationen für Benutzer von PowerPrompts 6.0, 8
Installieren von PowerPrompts, 11
IsTestMode-Eigenschaft, 44
Item-Methode, 78

J

JavaScript-Clientmethoden
  FormatUserVar, 100
Join-Methode, 79

K

Kataloganwendung, 25
Kennwort, 34
Konfigurieren
  Microsoft Web-Server, 11

L

Language-Eigenschaft, 81
Locale-Eigenschaft, 82
Logische Cognos-Datenbank, 19, 21
Logische Datenbank, 19, 21
Löschen
  Seiten, 15
  Verknüpfungen, 19

M

MaxRecords-Eigenschaft, 62
Mehrere Werte
  Berichte filtern nach, 29
Methoden
  AddColumn, 54
  AndFilterBy, 55
  AndSummaryFilterBy, 55
  AppendCookie, 69
  AppendHeader, 69
  AssociateColumn, 56
  Clear, 69
  ClearContent, 70
  ClearHeaders, 70
  Close, 63
  Contains, 77
  CreateObject, 74
  Execute, 47
  ExecuteCommand, 82
  FormatNumber, 74
  GetColumnTitle, 86
  GetPageFragment, 83
  GetQuery, 87
  GetReport, 87
  GetReportObject, 88
  GetUserVar, 100
  GetUserVarValues, 101
  GroupBy, 56
  HTMLEncode, 75
  Item, 78
  Join, 79
  MoveNext, 64
  Open, 64
  Operator, 52, 79
  OrFilterBy, 57
  OrSummaryFilterBy, 57
  Redirect, 70
  RemoveColumn, 58
  RunDynamo, 46
  SetDistinct, 58
  SetPromptValue, 59
  SortBy, 59
  toString, 80
  URLDecode, 75
  URLEncode, 76
  Write, 70
  WriteFile, 71
  Writeln, 71
MoveNext-Methode, 64

N

Name-Eigenschaft, 49
Neue Anwendung, 12

O

Objekte
  App, 41
  Connection, 47
  Field, 47
  Query, 52
  Recordset, 60
  Request, 66
  Response, 68
  Server, 73
  StringList, 76
  Upfront, 81
Objekte (Fortsetzung)
  User, 83
ODBC, 19, 21
Open-Methode, 64
Operator-Methode, 52, 79
OrFilterBy, 96
OrSummaryFilterBy, 97

P

Path-Eigenschaft, 45
PowerPrompts, 7, 8, 11
PowerPrompts 6.0, 8
Präprozessor-Anweisungen, 40
Produkt
  Version, 2

Q

Query, 52
Query.AddColumn, 54
Query.AndFilterBy, 55
Query.AndSummaryFilterBy, 55
Query.AssociateColumn, 56
Query.GroupBy, 56
Query.OrFilterBy, 57
Query.OrSummaryFilterBy, 57
Query.RemoveColumn, 58
Query.SetDistinct, 58
Query.SetPromptValue, 59
Query.SortBy, 59
Query.SQL, 54
Query-Objekt, 52

R

Recordset.Close, 63
Recordset.CurrentRecordIndex, 60
Recordset.EOF, 61
Recordset.Fields, 62
Recordset.MaxRecords, 62
Recordset.MoveNext, 64
Recordset.Open, 64
Recordset-Objekt, 60
Redirect-Methode, 70
RemoveColumn, 97
RemoveReportObject, 89
ReportScript-Eigenschaft, 45
Request.Cookies, 66
Request.ServerVariables, 66
Request.Variables, 67
Request-Objekt, 66
Response.AppendCookie, 69
Response.AppendHeader, 69
Response.Clear, 69
Response.ClearContent, 70
Response.ClearHeaders, 70
Response.ContentType, 68
Response.Redirect, 70
Response.Write, 70
Response.WriteFile, 71
Response.Writeln, 71
Response-Objekt, 68
RunDynamo-Methode, 46

S

Schlussseite, 13

ScriptTimeOut-Eigenschaft, 73
Seiten
  bearbeiten, 15
  hinzufügen, 14
  importieren, 14
  löschen, 15
Server.CreateObject, 74
Server.FormatNumber, 74
Server.HTMLEncode, 75
Server.ScriptTimeOut, 73
Server.URLDecode, 75
Server.URLEncode, 76
Server-Objekt, 73
ServerVariables-Eigenschaft, 66
SetListInsertCursor, 89
SetMaxRows, 98
SetPrimaryFrame, 90
SetPromptValue, 98
SetText, 92, 94
SetTextJustification, 93, 94
Skript, 26
  hinzufügen, 23
  Überblick, 23
Skriptfehler, 25
SortBy, 59, 98
Spaltentitelmethoden
  ApplyStyle, 93
  SetText, 94
  SetTextJustification, 94
Speichern
  Optionen zwischen Sitzungen, 30
SQL-Eigenschaft, 54
Standard-HTML-Editor, 11
Standardverknüpfung, 15
Starten von PowerPrompts, 11
Startseite, 13
StringList.Contains, 77
StringList.Count, 77
StringList.Item, 78
StringList.Join, 79
StringList.toString, 80
StringList-Objekt, 76

T

Telephone-Eigenschaft, 84
Testen der Anwendung, 25
Theme-Eigenschaft, 82
toString-Methode, 80
Type-Eigenschaft, 50

U

Überblick
  Dynamos, 19
  Skript, 23
  Verknüpfungen, 15
Überprüfen der Anwendung, 24
UNIX, 19
Upfront.ExecuteCommand, 82
Upfront.GetPageFragment, 83
Upfront.Language, 81
Upfront.Locale, 82
Upfront.Theme, 82
Upfront-Objekt, 81
Upfront-Themen
  unterschiedliches Design erstellen, 35
URLDecode-Methode, 75
URLEncode-Methode, 76
User.Description, 83
User.Email, 84
User.Telephone, 84
User.UserClass, 85
User.UserClasses, 85
User.UserName, 86
UserClass-Eigenschaft, 85
UserClasses-Eigenschaft, 85
UserName-Eigenschaft, 86
User-Objekt, 83

V

Value-Eigenschaft, 51
Variables-Eigenschaft, 45, 67
Verbinden
  mit dem Katalog, 25
Verknüpfungen, 17
  löschen, 19
  Überblick, 15
Verknüpfungsbedingungen, 18
Version
  Produkt, 2
VerticalAlign, 93
Verwenden
  ADO, 34
Virtuelle Verzeichnisse, 11
Vordefinierte Konstanten, 40

W

Web-Server, 11
WriteFile-Methode, 71
Writeln-Methode, 71
Write-Methode, 70

Z

Zeichenkettenliteral
  Apostroph, 99
  Einfaches Anführungszeichen, 99
Zeichnungsbereich, 12