IBM Cúram Social Program Management

Working with Intelligent Evidence Gathering (IEG)

Version 6.0.5

IBM Cúram Social Program Management

Working with Intelligent Evidence Gathering (IEG)

Version 6.0.5

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen in „Bemerkungen" auf Seite 67 gelesen werden.

Ausgabe: Mai 2013

Diese Ausgabe bezieht sich auf IBM Cúram Social Program Management v6.0 5 und alle nachfolgenden Releases und Modifikationen, bis dieser Hinweis in einer Neuausgabe geändert wird.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellen

Kapitel 1. Einführung

1.1 Zielgruppe

Dieses Handbuch richtet sich an Scriptautoren, die sich zum ersten Mal mit Intelligent Evidence Gathering (IEG) befassen und seine Funktionen dazu benutzen wollen, Daten als Teil einer internen oder externen Anwendung intelligent zu erfassen. Fachlich gesehen, können dies beliebige Daten für beliebige Zwecke sein. Gewöhnlich handelt es sich jedoch um kundenbezogene Daten, die als Teil einer Anwendung für ein Sozialprogramm oder zum Bestimmen möglicher Anspruchsberechtigungen erforderlich sind. In Cúram fallen alle Informationen dieser Art generell unter die Kategorie "Angaben". Mit seinen Anweisungen richtet sich dieses Handbuch direkt an den Scriptentwickler.

1.2 Zweck

Dieses Handbuch verfolgt den Zweck, Scriptautoren mit den notwendigen Informationen darüber auszustatten, wie IEG-Scripts und verknüpfte Datenspeicherschemata für die Verwendung in internen oder externen Anwendungen definiert und gewartet werden.

Die IEG-Technologie wird als Teil der Cúram Application Suite bereitgestellt, die es dem Kunden ermöglicht, dynamische Scripts zur Datenerfassung auf unterschiedliche Art zu erstellen. Einige Dinge sind jedoch bei der Erstellung eines IEG-Scripts und Datenspeicherschemas zu beachten. Diese werden in diesem Handbuch dargestellt, zusammen mit Informationen bezüglich der Wartung von Scripts.

1.3 Zusätzliche Literatur

Es gibt noch weitere Dokumente, die vor der Erstellung und Freigabe eines IEG-Scripts gelesen werden sollten. Zunächst gibt das Konformitätshandbuch `Cúram Development Compliancy Guide` einen Überblick über die geltenden Einschränkungen für das Entwickeln von Anwendungen, die IEG verwenden. Diese sollte man kennen, bevor man eine Implementierung startet. Das zweite Dokument, das man gelesen haben sollte, ist das Handbuch `Authoring Scripts using Intelligent Evidence Gathering (IEG)` (Scripts mithilfe von IEG verfassen). Dieses Dokument, das als Referenzhandbuch verwendet werden kann, enthält Detailinformationen zu allen in IEG verfügbaren Funktionen sowie Anweisungen zur Verwendung dieser Funktionen. Im Handbuch `Creating Datastore Schemas` (Datenspeicherschemata erstellen) wird die Erstellung und Verwaltung von Datenspeicherschemata für die Verwendung mit IEG erläutert.

Kapitel 2. Erste Schritte

2.1 Einführung

In diesem Kapitel werden die Grundprinzipien von IEG und seiner Abhängigkeit vom Datenspeicher (DS) und dem Ressourcenspeicher (RS) erklärt. Es führt den Leser durch den Erstellungsprozess eines einfachen IEG-Scripts, mit dem Informationen zu einem Kunden erfasst werden sollen.

2.2 Produktinfo zu IEG

IEG stellt eine effiziente Alternative zu den herkömmlichen Informationsbeschaffungsprozessen dar. Mithilfe von IEG werden die Informationen interaktiv gesammelt, indem ein Fragenscript angezeigt wird, zu dem ein Benutzer Antworten angeben kann. Die Fragen werden nur dann angezeigt, wenn sie in keinem Widerspruch zu vorherigen Benutzerantworten stehen, sodass der Benutzer nur Antworten angeben muss, die für seine Bedürfnisse und Situation relevant sind. Dadurch entsteht eine benutzerfreundliche Umgebung, die sich für eine Reihe von Prozessen effektiv einsetzen lässt, darunter das Erfassen von Kundeninformationen, Triage für die Einschätzung von Leistungsbezügen, Onlinebewertung der Anspruchsberechtigung u.ä.

Anders als die traditionellen Informationsbeschaffungsprozesse reduziert IEG den Verwaltungsaufwand einer Organisation, indem sie das Potenzial für mehrere Routen durch dasselbe Fragenscript herstellt. Dadurch wird das Entwickeln einer Vielzahl von Scripts für die Informationsbeschaffung aus unterschiedlichen Benutzertypen unnötig.

Ein weiterer Vorteil von IEG ist die Flexibilität seiner Implementierung und das große Spektrum seiner potenziellen Benutzer. Die IEG-Laufzeitumgebung kann für den Zugriff von jeder beliebigen UIM-Seite aus eingerichtet werden. Das bedeutet, dass von einer Organisationsanwendung aus direkt oder durch einen Onlinebenutzer über Fernzugriff auf IEG zugegriffen werden kann.

Die zwei Hauptkomponenten von IEG sind die Engine und der Player. Die IEG-Scripts sind in XML definiert. Die Engine interpretiert die Scriptdefinitionen zur Laufzeit und wertet die Antworten des Benutzers aus, um den Ausführungsablauf bestimmen zu können. Die Engine legt fest, welche Seiten und mit welcher Häufigkeit sie dem Benutzer angezeigt werden. Der Player stellt dem Benutzer die Seiten, Fragen und anderen Steuerelemente dar. IEG baut auch auf anderen Elementen der Cúram Application Suite auf, wie dem Datenspeicher und dem Ressourcenspeicher.

2.2.1 Datenspeicher (DS)

Die vom Benutzer während einer Scriptausführung angegebenen Daten werden nicht direkt von IEG selbst persistent gespeichert. Diese Aufgabe wird an den DS delegiert. Beim DS handelt es sich um eine konfigurierbare Datenbank. So wie ein IEG-Script festlegt, welche Fragen und Fragenseiten dem Benutzer angezeigt werden sollen, kann von einem XML-Schema dynamisch festgelegt werden, welche Daten im DS gespeichert werden sollen. Das Schema beschreibt die Struktur der Informationen, die gespeichert werden sollen, sowie eventuell vorhandene Beziehungen zwischen den Daten. Die Daten werden im DS im XML-Format gespeichert und entsprechen der W3C XML Schema Definition Language. Weitere Informationen zum DS und seiner Arbeitsweise finden Sie im Handbuch `Creating Datastore Schemas` (Datenspeicherschemata erstellen).

Ein IEG-Script ist sehr eng mit einem DS-Schema verknüpft. Es ist mit Verweisen auf die in einem Schema enthaltenen Elemente definiert. Aus diesem Grund muss zum Bearbeiten eines Scripts ein Schema bereitgestellt werden. Dasselbe Schema ist erforderlich, wenn ein Script ausgeführt wird. Schemata können wiederverwendet werden, um mehrere Scripts zu bearbeiten und auszuführen, so dass man dieselben Datenstrukturen unter unterschiedlichen Bedingungen verwenden kann.

2.2.2 Ressourcenspeicher (RS)

Ein IEG-Script kann Verweise auf Bilder enthalten, die dem Benutzer während der Ausführung eines Scripts angezeigt werden, z.B. Bilder, die Sektionen oder Fragenseiten darstellen. Diese Bilder werden im Ressourcenspeicher (RS) gespeichert. Außerdem enthält ein IEG-Script eine Anzahl unterschiedlicher Textelemente, z.B. Seitenüberschriften, Fragenbeschriftungen oder Hilfetexte. IEG ermöglicht es, den gesamten Text für Ihr Script für die Standard-Ländereinstellung direkt in die Scriptdefinition einzugeben.

Wenn ein IEG-Script über die IEG-Administrationsfenster in das System hochgeladen wird, wird der ganze in ihm enthaltene Text automatisch in entsprechend benannte Eigenschaftendateien für das Script extrahiert. Diese Eigenschaftendateien werden ebenfalls im RS gespeichert. Die Eigenschaftendateien werden ohne verknüpfte Ländereinstellung gespeichert (so dass diese als Ausweicheigenschaften agieren können, wenn für die von Ihnen aktivierte Ländereinstellung keine Eigenschaften vorhanden sind). Der RS ermöglicht es, Eigenschaftendateien für mehrere Ländereinstellungen hochzuladen, was die Lokalisierung von Scripts zu einer einfachen Aufgabe macht. Zur Laufzeit werden die Eigenschaftendateien für die entsprechende Ländereinstellung abgerufen und dem Benutzer im IEG-Player dargestellt.

2.2.3 Scriptstruktur

In seiner einfachsten Form besteht ein IEG-Script aus Seiten mit eingeschlossenen Fragen, die den IEG-Benutzern vorgelegt werden sollen. Die Struktur des IEG-Scripts besteht in einer logischen Gruppierung dieser Seiten, so dass die Antworten auf die Fragen effektiv erfasst werden können. Sequenzen von Seiten lassen sich in logische Sektionen gruppieren. Der Zweck dieser Sektionen besteht darin, dem Benutzer einen Überblick über die Art der Informationen zu verschaffen, die vom IEG-Script erfasst werden.

Zusätzlich zum Einschließen einer variablen Anzahl an Seiten sollte jede Sektion genau eine Zusammenfassungsseite enthalten. Diese Seite wird dazu verwendet, dem Benutzer eine Rückmeldung zu den Informationen zu geben, die auf den Seiten einer Sektion eingegeben wurden. Zusammenfassungsseiten enthalten gewöhnlich Cluster und Listen, die schreibgeschützte Versionen der Antworten auf die gestellten Fragen anzeigen. Die Zusammenfassungsseite ist immer die letzte Seite, die innerhalb einer Sektion angezeigt wird. Sie wird jedes Mal mit angezeigt, wenn ein Benutzer auf den Link für diese Sektion in der Seitenleiste des IEG-Players klickt.

Zusammengefasst bestehen IEG-Scripts aus einer Hierarchie von Elementen, die etwa folgendermaßen strukturiert sind:

- Script
  - Sektion
    - Seite
      - Cluster
        - Frage
    - Zusammenfassungsseite

2.3 Verwendung von IEG bewerten

Die Verwendung von IEG wird in einer beliebigen Anwendung mittels einiger Schlüsselfragen bewertet:

- Welche Informationen werden erfasst?
- Was ist die Quelle für diese Informationen?
- Wie werden diese Informationen verwendet?
- Wie ist die Lebensdauer dieser Informationen in der Anwendung?

Viele der gegenwärtigen Verwendungen von IEG entspringen dem Bedarf an einer Anwendung für Produkte und Services, die von Behörden extern oder intern angeboten werden. Bei den erfassten Informationen handelt es sich im Allgemeinen um kundenbezogene Informationen wie persönliche Angaben zum Kunden, Angaben zu seiner Familie und seinem Haushalt sowie Angaben zu seinen Bedürfnissen.

Oft verfügen die Behörden bereits über Daten zu dem Kunden und sind deshalb in der Lage, Informationen aus einem anderen System zu beschaffen, indem sie wichtige Einzelinformationen wie eine Sozialversicherungsnummer verwenden. Auf diese Weise können sie die eingegebenen Kundeninformationen überprüfen oder abrufen, um den Benutzer bei der Anwendung zu unterstützen.

Einige der Anwendungen sind sehr komplex und erfordern Informationen aus unterschiedlichen Quellen. Der Kunde muss möglicherweise Informationen eingeben, die ihm nicht unmittelbar zur Verfügung stehen. Das können beispielsweise erforderliche Informationen sein, die sich beim Arbeitgeber befinden. Deshalb muss es eine Möglichkeit geben, die gemachten Angaben zu speichern und zu einem späteren Zeitpunkt auf die Anwendung zurückzukommen, nachdem er die erforderlichen Daten zusammengestellt hat.

Möglicherweise hat es der Kunde auch mit einer einfachen Screening-Anwendung zu tun, die ihn über seine Anspruchsberechtigungen unter der gegenwärtigen oder neuen Gesetzgebung informiert. Solche Informationen sind oft unverlässlich. Temporäre Daten müssen nach der Abmeldung des Kunden oder innerhalb einer festgesetzten Zeitspanne aus dem System entfernt werden.

Alle diese Erfordernisse veranlassen die Verwendung von IEG und bieten wichtige Informationen für die Verwendung der Daten im Laufe ihrer Lebensdauer.

Die im Folgenden beschriebenen Grundlagen bestehen im Erfassen und Speichern von Informationen zu einem Kunden.

2.4 Grundlegende Informationen

2.4.1 Schema erstellen

Der erste Schritt zum Erfassen von Kundendaten ist das Erstellen eines DS-Schemas. In diesem Abschnitt wird ein Beispiel für das Erstellen eines Basisschemas gegeben, mit dem man das Erfassen einiger allgemeiner Kundendaten definieren kann.

Der DS speichert Benutzerdaten, die während Online-Screenings oder durch die Annahme von Anträgen erfasst werden. Die Inhalte des DS lassen sich mittels einer Schemadefinition dynamisch definieren. Die Anforderungen für das Erfassen und Speichern von Kundendaten können komplex sein, aber mit einem angemessenen Schemadesign lassen sich diese Daten während ihrer gesamten Lebensdauer effizient verwalten.

Für dieses Beispiel soll die Anforderung darin bestehen, folgende Daten zu erfassen:

Tabelle 1. Zu erfassende Kundendaten

Attribute	**Typ**
Vorname	Zeichenfolge
Zweiter Vorname	Zeichenfolge
Nachname/Familienname	Zeichenfolge
Geschlecht	Männlich/Weiblich
Geburtsdatum	Datum

Ein Schema erfordert einen Mindestsatz an Definitionen. Damit ein Schema in IEG verwendet werden kann, ist Folgendes erforderlich:

- Einschluss von Basisdomänen
- Einschluss von IEG-Domänen
- Eine Stammentität

Weitere Informationen zum Mindestsatz an erforderlichen Definitionen finden Sie im Handbuch `Creating Datastore Schemas` (Datenspeicherschemata erstellen).

Bevor neuer Inhalt wie beispielsweise die oben beschriebene Personenentität hinzugefügt wird, sieht das Schema etwa folgendermaßen aus:

```
<xsd:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:d="http://www.curamsoftware.com/BaseDomains">
  <xsd:import namespace="http://www.curamsoftware.com/BaseDomains"/>
  <xsd:include schemaLocation="IEGDomains"/>
  <xsd:element name="Application">
    <xsd:complexType>
      <xsd:sequence minOccurs="0">
    </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Abbildung 1. Anfangsschema

Der Inhalt des Schemas zeigt an, dass es ein XML-Schema ist, welches das Basisdomänenschema ("Base-Domains") importiert und das IEG-Domänenschema ("IEGDomains") einschließt. Das erste Element mit Namen `Application` stellt die Stammentität für das Schema dar. In IEG ist es erforderlich, dass die Stammentität stets mit `Application` benannt ist.

Das IEG-Domänenschema enthält die Domänen, die erforderlich sind, um die Attribute von Entitäten zu definieren, die mit IEG verwendet werden sollen. Die Typen der Attribute müssen von den IEG-Domänen abgeleitet sein, nicht von den Basisdomänen. Eine Personenentität kann so definiert sein, dass sie einen Kunden wie folgt darstellt:

```
<xsd:element name="Person">
  <xsd:complexType>
    <xsd:attribute name="firstName" type="IEG_STRING"/>
    <xsd:attribute name="middleName" type="IEG_STRING"/>
    <xsd:attribute name="lastName" type="IEG_STRING"/>
    <xsd:attribute name="gender" type="IEG_GENDER"/>
    <xsd:attribute name="dateOfBirth" type="IEG_DATE"/>
  </xsd:complexType>
</xsd:element>
```

Abbildung 2. Personenentität

Im Zusammenhang mit dem oben beschriebenen Hinzufügen einer Entität, beispielsweise einer Personenentität, sind einige Dinge zu beachten:

- Wie bei relationalen Datenbanktabellen ist ein ID-Feld erforderlich und es ist ein Schlüssel für diese Tabelle mit dieser eindeutigen ID definiert.
- Die Personenentität wird als der Stammentität untergeordnete Entität hinzugefügt.

Das Schema zum Erfassen grundlegender Informationen zu einer Person kann wie folgt definiert werden:

```
<xsd:element name="Application">
  <xsd:complexType>
    <xsd:sequence minOccurs="0">
       <xsd:element ref="Person" minOccurs="0"
                                 maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="Person">
  <xsd:complexType>
    <xsd:attribute name="personID" type="d:SVR_KEY"/>
    <xsd:attribute name="firstName" type="IEG_STRING"/>
    <xsd:attribute name="middleName" type="IEG_STRING"/>
    <xsd:attribute name="lastName" type="IEG_STRING"/>
    <xsd:attribute name="gender" type="IEG_GENDER"/>
    <xsd:attribute name="dateOfBirth" type="IEG_DATE"/>
  </xsd:complexType>
  <xsd:key name="Person_Key">
    <xsd:selector xpath="./Person"/>
    <xsd:field xpath="@personID"/>
  </xsd:key>
</xsd:element>
```

Abbildung 3. Basisschema

Nachdem das Schema definiert ist, kann ein Script erstellt werden, mit dem man das Schema verwenden kann.

2.4.2 Script erstellen

Mit IEG können dynamische Scripts zur Datenerfassung erstellt werden. IEG-Scripts enthalten Sektionen, Fragenseiten, Fragen und Bedingungslogik, die bei der Entscheidung helfen, welche Informationen erfasst, welche Seiten angezeigt und wie oft sie angezeigt werden sollen.

Weitere Informationen zum Definieren der einzelnen Elemente eines IEG-Scripts finden Sie im Handbuch `Authoring Scripts using Intelligent Evidence Gathering (IEG)`.

Um den oben beschriebenen Anforderungen zu entsprechen, muss dort, wo Informationen zu einer Person erfasst werden müssen, das Script definiert sowie entschieden werden, wie die Seiten zur Informationserfassung angeordnet werden sollen.

Ein neues Script kann in der Verwaltungsanwendung erstellt werden. Der Editor kann dazu verwendet werden, diesem Script Elemente hinzuzufügen. Der Inhalt des neu erstellten Scripts sieht dann etwa folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>
<ieg-script xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="ieg-schema.xsd">
  <identifier id="WorkingWithIEG" scriptversionnumber="V1"
              type="Intake" />
</ieg-script>
```

Abbildung 4. Neues Script

Die bei der Erstellung des Scripts zur Verfügung gestellten Eigenschaften `ID`, `Typ` und `Version` werden zu einer Scriptkennung kombiniert, mit der die Scriptdefinition eindeutig angegeben wird.

Nachdem ein neues Script erstellt ist, können ihm Elemente wie Sektionen, Fragenseiten und Zusammenfassungsseiten hinzugefügt werden. Die Beispiele der zwei folgenden Abschnitte zeigen, wie man einem Script eine Sektion und eine Fragenseite hinzufügt und wie eine Zusammenfassungsseite hinzugefügt

wird, die dem Benutzer die Informationen anzeigt. Zusammenfassungsseiten bieten dem Benutzer die Möglichkeit zu bestätigen, dass die eingegebenen Daten korrekt sind, bevor er fortfährt. Außerdem hat er hier die Möglichkeit, Daten zu ändern.

2.4.2.1 Dem IEG-Script eine Sektion und eine Fragenseite hinzufügen

Es sollen eine Sektion und eine Fragenseite hinzugefügt werden. Mit einer Sektion werden verwandte Seiten gruppiert, so dass der Benutzer auf logische Weise durch die Anzeigen blättern kann. Zudem helfen Sektionen, dem Benutzer seinen Verarbeitungsfortschritt durch das Script sichtbar zu machen. Sowohl die Sektion als auch die Fragenseite können einen Titel erhalten; die Fragenseite optional auch eine Beschreibung.

Im folgenden Codebeispiel wird eine Sektion mit einer Fragenseite gezeigt, die einem Script hinzugefügt wird:

```
<?xml version="1.0" encoding="UTF-8"?>
<ieg-script xmlns:xsi="http://www.w3.org/2001/XMLSchema-in stance"
 xsi:noNamespaceSchemaLocation="ieg-schema.xsd">
  <identifier id="WorkingWithIEG" scriptversionnumber="V1"
              type="Intake" />
  <section>
    <title id="AboutYouSection.Title">
      <![CDATA[About You]]>
    </title>
    <question-page id="AboutYouPage" entity="Person">
      <title id="PrimaryPersonPage.Title">
        <![CDATA[About You]]>
      </title>
      <description id="PrimaryPersonPage.Description">
        <![CDATA[Please enter some information about yourself]]>
      </description>
    </question-page>
  </section>
</ieg-script>
```

Abbildung 5. Neue Sektion

Die Fragenseite erfordert geeignete Fragen, mit denen die Daten erfasst werden können. Alle im DS zu speichernden Daten müssen im DS-Schema mit einem Attribut einer Entität verknüpft sein, um mit diesem Script verwendet werden zu können. Wenn alle Fragen einer Seite mit derselben Entität verwandt sind, kann die Seite diesem Entitätstyp zugeordnet werden. Im obigen Beispiel wird die Seite der Personenentität zugeordnet.

Zum Hinzufügen von Fragen zu einer Seite ist ein Cluster erforderlich. Ein Cluster trägt zur Steuerung des Layouts der Fragen auf der Seite bei. Eine Seite kann viele Cluster enthalten, die es ermöglichen, die Fragen auf der Seite logisch zu gruppieren. Cluster können ebenfalls einen Titel und eine Beschreibung enthalten.

Das untenstehende Beispiel zeigt zwei Cluster, einen nur zum Anzeigen von Informationstexten für den Benutzer sowie einen, der die Fragen für die persönlichen Angaben enthält. Jedem der Cluster können Fragen und Anzeigetext hinzugefügt werden. Fragen müssen eine ID erhalten, die einem der Attribute des Attributtyps entsprechen, dem die Seite zugeordnet ist. Wenn eine Frage eine Pflichtangabe erfordert, kann der Indikator dafür, dass die Frage obligatorisch ist, auf `true` gesetzt werden. Das untenstehende Scriptfragment enthält die Fragen, mit denen die für das genannte Beispiel erforderlichen Daten erfasst werden können.

```
<question-page ...
    <cluster>
    <display-text id="RequiredFields.Text">
      <![CDATA[<span style="color: orange;">
                * indicates a required field</span>]]>
    </display-text>
  </cluster>
  <cluster>
    <title id="DetailsCluster.Title">
      <![CDATA[Personal Details]]>
    </title>
    <description id="DetailsCluster.Description">
      <![CDATA[Enter your details here]]>
    </description>
    <question id="firstName" mandatory="true">
      <label id="FirstName.Label">
        <![CDATA[First Name:]]>
      </label>
    </question>
    <question id="middleName">
      <label id="MiddleName.Label">
        <![CDATA[Middle Name:]]>
      </label>
    </question>
    <question id="lastName">
      <label id="lastName.Label">
        <![CDATA[Last Name:]]>
      </label>
    </question>
    <question id="gender" mandatory="true">
      <label id="Gender.Label">
        <![CDATA[Gender:]]>
      </label>
    </question>
    <question id="dateOfBirth" mandatory="true">
      <label id="DateOfBirth.Label">
        <![CDATA[Date Of Birth:]]>
      </label>
    </question>
  </cluster>
</question-page>
```

Abbildung 6. Cluster, Fragen und Anzeigetexte

Beachten Sie, dass es mehr Eigenschaften von Scripts, Sektionen, Fragenseiten, Clustern, Fragen und Anzeigetexten gibt, als hier behandelt werden können. Solche Eigenschaften werden im Handbuch `Authoring Scripts using Intelligent Evidence Gathering (IEG)` beschrieben; einige von ihnen auch im Verlaufe dieses Handbuchs.

2.4.3 Einem IEG-Script eine Zusammenfassungsseite hinzufügen

Der letzte Schritt dieses Basisbeispiels besteht im Anzeigen einer Zusammenfassung der erfassten Informationen. Grundsätzlich besitzt jede Sektion eine Zusammenfassungsseite. Eine Zusammenfassungsseite wird dazu verwendet, dem Benutzer die wichtigsten Daten anzuzeigen, so dass er überprüfen kann, ob sie erfasst oder korrekt berechnet wurden. Eine Zusammenfassungsseite kann Daten anzeigen, die auf mehreren Fragenseiten erfasst wurden. Sie muss nicht alle in der Sektion erfassten Informationen beinhalten. Das würde sie unnötig vergrößern und ihren Nutzen verringern.

Sind die auf der Zusammenfassungsseite angezeigten Daten falsch, wird der Benutzer den Wunsch haben, sie zu ändern. Dazu kann er in der Scriptausführung zurücknavigieren, indem er im IEG-Player die Schaltfläche "Zurück" drückt, bis er die Seite erreicht hat, auf der die Daten eingegeben wurden, dort die Daten aktualisieren und anschließend wieder vorwärts durch das Script gehen. Alternativ dazu kann man den Clustern auf der Zusammenfassungsseite Bearbeitungslinks hinzufügen. Wenn der Benutzer einen Bearbeitungslink auf der Zusammenfassungsseite anklickt, wird ihm die im Bearbeitungslink angegebene

Fragenseite im IEG-Player angezeigt. Anschließend kann der Benutzer die Daten ändern. Je nachdem, ob die geänderten Daten an anderer Stelle im Script referenziert werden, wird die Zusammenfassungsseite wieder angezeigt, sobald der Benutzer im IEG-Player die Schaltfläche "Weiter" drückt.

In diesem Fall ist die Zusammenfassungsseite sehr einfach und ähnelt der zuvor hinzugefügten Fragenseite. Wie bei der Fragenseite, so gilt auch hier: Wenn alle die Attribute, auf die auf der Seite verwiesen wird, mit derselben Entität verwandt sind, kann die Zusammenfassungsseite diesem Entitätstyp zugeordnet werden. Dazu geht man wie folgt vor:

```
<section>
...
  <summary-page id="AboutYouSummary" entity="Person">
    <title id="AboutYouSummary.Title">
      <![CDATA[Information about you]]>
    </title>
    <description id="AboutYouSummary.Description">
      <![CDATA
        [Here's the information you've entered about yourself]]>
    </description>
    <cluster>
      <title id="DetailsCluster.Title">
        <![CDATA[Person Details]]>
      </title>
      <description id="DetailsCluster.Description">
        <![CDATA[Enter the details for this person here]]>
      </description>
      <edit-link start-page="AboutYouPage" />
      <question id="firstName">
        <label id="FirstName.Label">
          <![CDATA[First Name:]]>
        </label>
      </question>
      <question id="middleName">
        <label id="MiddleName.Label">
          <![CDATA[Middle Name:]]>
        </label>
      </question>
      <question id="lastName">
        <label id="lastName.Label">
          <![CDATA[Last Name:]]>
        </label>
      </question>
      <question id="gender">
        <label id="Gender.Label">
          <![CDATA[Gender:]]>
        </label>
      </question>
      <question id="dateOfBirth">
        <label id="DateOfBirth.Label">
          <![CDATA[Date Of Birth:]]>
        </label>
      </question>
    </cluster>
  </summary-page>
</section>
```

Abbildung 7. Zusammenfassungsseite

Damit ist dieses Basisscript und Schema zum Erfassen von Informationen zu einer Person und zum Anzeigen einer Zusammenfassungsseite vollständig und kann ausgeführt werden.

2.4.4 Script ausführen

Um ein IEG-Script auszuführen, müssen die Scriptdefinition und die verknüpfte Schemadefinition in das System hochgeladen werden. Es gibt dafür eine Anzahl verschiedener Methoden, die weiter unten in diesem Handbuch behandelt werden. Die einfachste Art, Definitionen hochzuladen, ist über die Administrationsfenster in der IEG-Sektion des Administrationsarbeitsbereichs.

Für den Zugriff auf die Administrationsfenster müssen Sie sich als Benutzer mit Administratorberechtigung anmelden. Nach der Anmeldung sehen Sie eine Sektion in Ihrem Verknüpfungsfenster, das mit "Intelligent Evidence Gathering" (IEG) benannt ist. Beim Klicken auf diese Sektion erscheint ein Menü für IEG, das den Link "Scripts" enthält. Mit dem Klicken auf diesen Link erscheint eine Seite, die eine Liste von aktuell im System vorhandenen IEG-Scripts enthält, sowie verschiedene Links, die es ermöglichen, Aktivitäten für diese Scripts auszuführen.

Am Anfang der Seite "Scripts" befindet sich ein Link zum Importieren, mit dem man eine neue Scriptdefinition hochladen oder importieren kann.

Desgleichen erscheint, wenn Sie auf den Link "Datenspeicherschemata" des Menüs "IEG" klicken, eine Seite, die eine Liste mit den aktuell im System vorhandenen DS-Schemata enthält. Am Anfang der Seite "Datenspeicherschemata" befindet sich ein Link zum Importieren, mit dem man eine neue Schemadefinition hochladen oder importieren kann.

Aus Komfortgründen stellt IEG einen Typ von Testharness bereit, mit dem IEG-Scripts getestet werden können, ohne sie vorher in die Cúram-Anwendung integrieren zu müssen. Der Testharness besitzt zwar einige Einschränkungen, ermöglicht aber das Testen der meisten Scripts, sobald sie ins System hochgeladen werden. IEG-Scripts können entweder in einer Registerkarte oder in einem Modalfenster über die Administrationsfenster ausgeführt werden.

Ein Script kann entweder über die Option "Ausführen" oder die Option "Im Modalmodus ausführen" für das Script aus der Seite "Scripts" ausgeführt werden. Da zwischen einem IEG-Script und einem DS-Schema keine explizite Verknüpfung besteht, werden Sie beim Auswählen der Option, ein Script auszuführen, mittels einer Dropdown-Liste zum Auswählen eines Schemas aufgefordert, mit dem das Script ausgeführt werden soll. Durch Klicken auf die Schaltfläche "Script ausführen" wird der IEG-Player gestartet und die erste Seite des Scripts angezeigt.

2.4.4.1 Script validieren

Wenn ein Script auf diese Weise über die Administrationsfenster ausgeführt werden soll, wird es zuvor validiert. Dazu kann man für das Script auch die Option "Validieren" aus der Seite "Scripts" auswählen. Jedes Script sollte vor seiner Ausführung validiert werden. Bei einem Fehlschlagen der Validierung wird eine Liste der Validierungsfehler angezeigt. Diese Validierungsfehler müssen behandelt werden, bevor das Script von der Seite "Scripts" aus ausgeführt werden kann.

Geben Sie auf der ersten Seite des Scripts einige Beispieldaten an und klicken Sie auf die Schaltfläche "Weiter". Nun sollten diese Beispieldaten auf der Zusammenfassungsseite angezeigt werden. Die Antworten sind nicht veränderbar, aber es wird ein Bearbeitungslink zur Verfügung gestellt, mit dem man auf die Seite zurückspringen kann, wo diese Daten eingegeben wurden.

Beachten Sie, dass durch Drücken der Schaltfläche "Weiter" im IEG-Player auf der Zusammenfassungsseite des in diesem Beispiel implementierten Scripts eine Fehlermeldung verursacht wird. Das liegt daran, dass noch nicht alle Eigenschaften des Scripts definiert worden sind. Die erforderlichen Eigenschaften werden an späterer Stelle in diesem Handbuch erläutert.

Kapitel 3. Kundeninformationen erfassen

3.1 Einführung

Im vorhergehenden Kapitel wurde ein Basisbeispiel dafür entworfen, wie IEG zum Erfassen von Daten eines Kunden verwendet werden kann. Die Antragsformulare für Leistungsbezüge und Services können komplex sein und die erforderlichen Informationen zu den Antragstellern sehr ins Detail gehen.

Ausgehend von dem ursprünglichen Beispiel im vorigen Kapitel, wo Anfangsdaten zu einem Primärmitglied des Haushalts erfasst wurden, soll nun gezeigt werden, wie man weitere Angaben für die übrigen Haushaltsmitglieder hinzufügen kann.

3.2 Familien und Haushalte

Unser aktuelles Script ist unkompliziert, da es sich auf nur eine Person bezieht. Oft benötigen Anwendungen mehr Informationen zur Situation des Kunden, angefangen mit seinen Lebensbedingungen.

Im Allgemeinen werden Informationen zur Primärperson abgefragt, gefolgt von einer einfachen Frage, die es dem Kunden ermöglicht, in einen anderen Bereich der Anwendung zu springen. Beispiel: Nach dem Eingeben der persönlichen Angaben wird der Kunde gefragt, ob er allein lebt. Ist die Antwort "Ja", kann die Person als einzelne Person behandelt werden, die nicht in einem Haushalt mit der Familie oder anderen Personen zusammenlebt. Da die meisten Kunden den Anwendungsprozess so schnell wie möglich hinter sich bringen wollen, bilden solche Fragen eine effiziente Methode, um zu den relevanten Teilen der Anwendung vorzudringen.

Lebt der Kunde mit anderen Personen zusammen, müssen möglicherweise Fragen zu jeder dieser Personen gestellt werden. Für das Erfassen der Informationen zu jeder Person werden Schleifen verwendet. Je nachdem, wie der Scriptautor diese Fragen darstellen will, kann er einen der folgenden Schleifentypen verwenden: For-, While- und Foreach-Schleifen.

IEG bietet auch eine Personenregisterkarte, die dem Kunden beim Eingeben der Daten anzeigt, auf wen sich diese Fragen beziehen. Diese erscheint für eine Personenentität automatisch im DS. Jede Person wird durch ein Symbol, das auf Alter und Geschlecht beruht, und einen Namen dargestellt. Die aktuelle Person wird hervorgehoben.

Hier ein Beispielszenario zur Handhabung von Familien- bzw. Haushaltsdaten als Erweiterung der Erfordernisse im Basisbeispiel. Der Kunde wird gefragt, wie viele Personen, inklusive seiner selbst, im Haushalt leben. Um diese Information zu erfassen, müssen einige neue Fragenseiten hinzugefügt werden.

Die erste Frage bezieht sich auf die Lebensbedingungen. In diesem Beispiel muss nur eine Frage gestellt werden: Wie viele Personen gehören außer Ihnen zur Familie?

```
<question-page id="HouseholdPage" progress="10">
    <title id="LoopControlPage.Title">
        <![CDATA[Household Details]]>
    </title>
    <description id="LoopControlPage.Description">
        <![CDATA[Please tell us some information about your
        household]]>
    </description>
    <icon image="sample_title_household" />
    <cluster>
        <title id="DetailsCluster.Title">
            <![CDATA[Details]]>
        </title>
        <question id="numPeople" control-question="true"
            control-question-type="IEG_INT32"
            mandatory="true">
            <label id="NumPeople.Label">
                <![CDATA[How many other people are in your
                household?]]>
            </label>
        </question>
    </cluster>
</question-page>
```

Abbildung 8. Haushaltsgröße erfassen

Diese Frage ist eine Kontrollfrage, d.h. eine Frage, mit der die Größe einer Schleife gesteuert wird. Sie dient nicht der Datenerfassung. Kontrollfragen werden nicht im DS-Schema gespeichert. Die Kontrollfrage wird im Schleifenausdruck der For-Schleife in der nächsten Fragenseite verwendet.

Die Fragenseite zu den Familienmitgliedern befindet sich innerhalb einer For-Schleife, die über die Anzahl der Familienmitglieder iteriert.

```
<loop loop-type="for" loop-expression="numPeople"
    entity="Person" criteria="isPrimary==false">
    <question-page id="PersonDetailsPage"
        show-person-tabs="true"
        progress="20">
        <title id="PersonDetailsPage.Title">
            <![CDATA[Household Member Details]]>
        </title>
        <description id="PersonDetailsPage.Description">
            <![CDATA[Please enter the details for the
                next person in your household]]>
        </description>
        <icon image="sample_title_household" />
        <cluster>
            <title id="DetailsCluster.Title">
                <![CDATA[Person Details]]>
            </title>
            <description id="DetailsCluster.Description">
                <![CDATA[Enter the details for this person
                below]]>
            </description>
            <question id="firstName" mandatory="true">
                <label id="FirstName.Label">
                    <![CDATA[First Name:]]>
                </label>
            </question>
            <question id="lastName">
                <label id="lastName.Label">
                    <![CDATA[Last Name:]]>
                </label>
            </question>
            <question id="gender" mandatory="true">
                <label id="Gender.Label">
                    <![CDATA[Gender:]]>
                </label>
            </question>
        </cluster>
    </question-page>
</loop>
```

Abbildung 9. For-Schleife zum Erfassen der Haushaltsmitglieder verwenden

Das obige Beispiel zeigt, wie der Kunde die Anzahl an Familienmitgliedern eingibt. Die Frage hätte aber auch anders gestellt werden können, z.B.: "Leben Sie mit Ihrer Familie zusammen?" In diesem Fall kann im Script ein Bedingungselement verwendet werden, um den Wert dieser Frage zu überprüfen. Dadurch würde in dem Fall, dass der Kunde tatsächlich mit seiner Familie in einem Haushalt lebt, die Seite mit den Familienmitgliedern angezeigt werden. Auf dieser Fragenseite wird eine Kontrollfrage gestellt, um festzustellen, ob der Kunde Angaben zu einem weiteren Familienmitglied erfassen möchte.

Diese Kontrollfrage würde wie folgt in einer While-Schleife um die Fragenseite mit den Familienmitgliedern herum verwendet werden:

```
<question-page id="HouseholdPage" progress="10">
    <title id="LoopControlPage.Title">
        <![CDATA[Household Details]]>
    </title>
    <description id="LoopControlPage.Description">
        <![CDATA[Please tell us some information about your
        household]]>
    </description>
    <icon image="sample_title_household" />
    <cluster>
        <title id="DetailsCluster.Title">
            <![CDATA[Details]]>
        </title>
        <question id="livesWithFamily" control-question="true"
            control-question-type="IEG_BOOLEAN"
            mandatory="true">
            <label id="NumPeople.Label">
                <![CDATA[Do you live with your family?]]>
            </label>
        </question>
    </cluster>
</question-page>
```

Abbildung 10. While-Schleife zum Erfassen der Haushaltsmitglieder verwenden

Bei dieser Methode ist die Kontrollfrage ein boolescher Typ, da sie in einem Bedingungsausdruck verwendet wird, der anzeigt, ob eine While-Schleife eingegeben werden sollte oder nicht. Nachdem die Schleife eingegeben ist, iteriert sie, bis die Angaben zu allen Haushaltsmitgliedern erfasst sind, wie folgt:

```
<condition expression="livesWithFamily==true">
  <loop loop-type="while" loop-expression="
    anotherMember==true"
    entity="Person">
      <question-page id="PersonDetailsPage"
       show-person-tabs="true"
         progress="20">
           <title id="PersonDetailsPage.Title">
               <![CDATA[Household Member Details]]>
           </title>
           <description id="PersonDetailsPage.Description">
               <![CDATA[Please enter the details for
                 the next person in your household]]>
           </description>
           <icon image="sample_title_household" />
           <cluster>
               <title id="DetailsCluster.Title">
                   <![CDATA[Person Details]]>
               </title>
               <description id="DetailsCluster.Description">
                   <![CDATA[Enter the details for this
                   person below]]>
               </description>
           <question id="firstName" mandatory="true">
               <label id="FirstName.Label">
                   <![CDATA[First Name:]]>
               </label>
           </question>
           <question id="lastName">
               <label id="lastName.Label">
                   <![CDATA[Last Name:]]>
               </label>
           </question>
           <question id="gender" mandatory="true">
               <label id="Gender.Label">
                   <![CDATA[Gender:]]>
               </label>
           </question>
           </cluster>
           <cluster>
               <question id="anotherMember"
               control-question="true"
                 control-question-type="IEG_BOOLEAN">
                   <label id="AnotherMember.Label">
                       <![CDATA[Is there another
                       household member?]]>
                   </label>
               </question>
           </cluster>
      </question-page>
  </loop>
</condition>
```

Abbildung 11. While-Schleife zum Erfassen der Haushaltsmitglieder verwenden

3.3 Beziehungen im Haushalt

Für das Erfassen von Informationen zu einer Gruppe von Personen innerhalb eines Haushalts kann es notwendig sein zu ermitteln, inwiefern diese Personen zusammengehören. Mittels der Verwendung der Beziehungsseiten und einer bestimmten DS-Schemastruktur stellt IEG einen Mechanismus zum Erfassen von Beziehungen zur Verfügung.

Eine Beziehungsentität sollte in folgender Form im DS-Schema definiert sein:

```
<xsd:element name="Person">
  <xsd:complexType>
    <xsd:sequence minOccurs="0">
      <xsd:element ref="Relationship" minOccurs="0"
        maxOccurs="unbounded"/>
    </xsd:sequence>
    ...
</xsd:element>
<xsd:element name="Relationship">
  <xsd:complexType>
    <xsd:attribute name="relationshipType"
    type="IEG_STRING"/>
    <xsd:attribute name="isNonParentPrimaryCaretaker"
        type="IEG_BOOLEAN" default="false"/>
    <xsd:attribute name="personID" type="D:SVR_KEY"/>
  </xsd:complexType>
</xsd:element>
```

Abbildung 12. Beziehungsentität im DS-Schema

Unter der Voraussetzung, dass die Beziehungsentität der Personenentität untergeordnet ist, kann eine Beziehungsseite für den Haushalt folgendermaßen definiert werden:

```
<relationship-page id="RelationshipPage" show-person-tabs="true"
    progress="40">
    <title id="RelationshipPage.Title">
        <![CDATA[Household Relationships]]>
    </title>
    <description id="RelationshipPage.Description">
    <![CDATA[Please enter the relationships for %1s below]]>
      <argument id="Person.firstName" />
    </description>
    <icon image="sample_title_household" />
    <question id="caretakerInd">
        <label id="CaretakerInd.Label">
            <![CDATA[Is this a non-parent caretaker
            relationship?]]>
        </label>
    </question>
</relationship-page>
```

Abbildung 13. Beziehungsseite

Die Beziehungsseite muss nur einmal definiert werden. Die Seite wird von IEG später so viele Male angezeigt, wie es notwendig ist, um die Beziehungen Person für Person zu erfassen. Das entspricht der Anzahl der Personen im Haushalt minus eine Person, da die Beziehungen der letzten Person im Laufe des Prozesses in ihrer Gesamtheit erfasst worden sind.

Standardmäßig wird das Feld für den Beziehungstyp als Dropdown-Liste dargestellt, die aus einer Codetabelle gefüllt wird (zu konfigurieren über die Eigenschaft `relationship.type.domain.name`):

Am Anfang der Beziehungsseite wird eine Personenregisterkarte mit der Liste der Haushaltsmitglieder angezeigt, wobei die aktuelle Person hervorgehoben ist. Zusätzlich werden die Beziehungen zwischen der aktuellen Person und den übrigen Mitgliedern angezeigt.

Der Indikator für die Betreuungsperson ist die einzige Frage, die der Beziehungsseite direkt hinzugefügt werden kann. Fragen bezüglich der übrigen Attribute einer Beziehungsentität müssen Clustern hinzugefügt werden, die ihrerseits der Beziehungsseite hinzugefügt worden sind.

3.4 Kundeninformationen zusammenfassen

Auf den Zusammenfassungsseiten werden Listen verwendet, die die in Schleifen erfassten Informationen anzeigen. Die Struktur einer solchen Liste sollte die Struktur der Schleife oder Schleifenhierarchie widerspiegeln, mit der die Daten erfasst wurden. Das heißt, die Entität und die Kriterien auf der Liste sollten mit der Entität und den Kriterien in der Schleife übereinstimmen. Zum Erfassen der Mitglieder der Familie, die unter 3.2, „Familien und Haushalte”, auf Seite 13 beschrieben wurde, verwendet man beispielsweise eine For-Schleife:

```
<loop loop-type="for" loop-expression="numPeople"
    entity="Person" criteria="isPrimary==false">
  ...
</loop>
```

Abbildung 14. For-Schleife zum Erfassen von Informationen zu den Haushaltsmitgliedern verwenden

Auf der Zusammenfassungsseite der Sektion sind die mit dieser Schleife erfassten Informationen in Form einer Liste aufgeführt. Wie die Schleife, so hat auch die Liste "Person" zur Entität und "isPrimary==false" zum Kriterium:

```
<list entity="Person" criteria="isPrimary==false">
  ...
</list>
```

Abbildung 15. Liste von Personen

Die mithilfe der Beziehungsseite erfassten Beziehungsinformationen können auf den Zusammenfassungsseiten in Form von Beziehungszusammenfassungslisten angezeigt werden:

```
<relationship-summary-list>
    <title id="RelationshipSummaryList.Title">
        <![CDATA[Person Relationships Summary]]>
    </title>
    <description id="PersonRelationshipSummaryList.Description">
        <![CDATA[Person Relationship Summary Details]]>
    </description>
    <column id="caretakerInd">
        <title id="CaretakerInd.Title">
            <![CDATA[NPCR]]>
        </title>
    </column>
    <edit-link start-page="RelationshipPage" />
</relationship-summary-list>
```

Abbildung 16. Beziehungszusammenfassungsliste

Kapitel 4. Verwandte Daten erfassen

4.1 Einführung

Nachdem Informationen zu den Mitgliedern des Haushalts erfasst worden sind, wie beispielsweise persönliche Angaben und Beziehungen, sollen nun möglicherweise verwandte Daten erfasst werden. Dies wird durch Komposition (das Verwenden von verschachtelten DS-Entitäten) oder durch Verknüpfung (Verwenden von verwandten, nichtverschachtelten DS-Entitäten) erreicht.

4.2 Verbundene Daten erfassen

Es wurde bereits die Möglichkeit erwähnt, in IEG Beziehungen zu erfassen. Das Kombinieren von Beziehungsentität und Beziehungsseite bietet einen bequemen Mechanismus zur Erfassung der Beziehungen zwischen den Personen eines Haushalts. Die Beziehung zwischen den Personen eines Haushalts ist nur eine Form der Beziehung. Es werden von IEG auch andere Typen von Beziehungen unterstützt. IEG und der DS ermöglichen eine Verschachtelung von Entitäten durch Erstellen einer Beziehung zwischen übergeordneter und untergeordneter Entität. Beispiel: Es müssen Informationen zu den Einkommen der Personen eines Haushalts erfasst werden. Die Entität "Einkommen" wird definiert wie jede andere Entität auch. Sie wird in der Personenentität verschachtelt, indem man in einer Sequenz auf sie verweist, wie das folgende Fragment aus einem Beispielcode zeigt:

```
<xsd:element name="Person">
  <xsd:complexType>
    <xsd:sequence minOccurs="0">
      <xsd:element ref="Income" minOccurs="0"
        maxOccurs="unbounded" />
    </xsd:sequence>
    ...
    <xsd:attribute name="hasIncome" type="IEG_BOOLEAN"
      default="false"/>
  </xsd:complexType>
  ...
</xsd:element>
<xsd:element name="Income">
  <xsd:complexType>
    <xsd:attribute name="type" type="IEG_STRING" />
    <xsd:attribute name="amount" type="IEG_MONEY" />
  </xsd:complexType>
</xsd:element>
```

Abbildung 17. Schema einer Beziehung zwischen übergeordneter und untergeordneter Entität

Anschließend können durch Anwendung von Schleifen für jede Person, die über ein Einkommen verfügt, Informationen zum Einkommen der Personen eines Haushalts erfasst werden. Das Schleifenkriterium verwendet die boolesche Frage "hasIncome", die beim Erfassen der Einzeldaten für jede Person abgefragt wird. Eine Seite innerhalb der Schleife kann der Entität "Einkommen" zugeordnet werden, wodurch die verschachtelte Beziehung entsteht, wie folgendes Beispiel zeigt:

```
<loop loop-type="for-each" entity="Person"
        criteria="hasIncome==true">
  <loop loop-type="while" loop-expression="hasMoreIncome"
          entity="Income">
    <question-page id="IncomePage" entity="Income"
    ...
```

Abbildung 18. Verschachtelte Entitäten erstellen

4.3 Verbundene Daten in einer Zusammenfassung anzeigen

Die für verschachtelte Entitäten erfassten Informationen können mithilfe einer verschachtelten Liste auf einer Zusammenfassungsseite angezeigt werden. Ähnlich den regulären Listen müssen verschachtelte Listen mit den Entitäten und Kriterien übereinstimmen, die in den verschachtelten Schleifen verwendet werden, die die Daten erfasst haben.

```
<list entity="Person" show-icons="true" criteria="hasIncome==true">
  <title id="IncomeList.Title">
      <![CDATA[Income]]>
  </title>
  <description id="IncomeList.Description">
    <![CDATA[Here's the income details you've entered for all the
                  people in your household]]>
  </description>
  <column id="firstName">
    <title id="FirstName.Title">
      <![CDATA[First Name]]>
    </title>
  </column>
  <list entity="Income">
    <column id="type">
      <title id="IncomeType.Title">
        <![CDATA[Income Type]]>
      </title>
    </column>
    <column id="amount">
      <title id="IncomeAmount.Title">
        <![CDATA[Income Amount]]>
      </title>
    </column>
  </list>
</list>
```

Abbildung 19. Verschachtelte Entitäten auf Zusammenfassungsseiten anzeigen

Das obige Fragment aus einem Beispielcode für eine zusammenfassende Einkommensliste wird im IEG-Player als eine reguläre Liste mit den nach Personen gruppierten Einkommen angezeigt. Sie enthält auch Links zum Bearbeiten und Löschen für jedes Einkommen sowie einen Link zum Hinzufügen mit einer Dropdown-Liste, in der alle Personen aufgeführt sind.

4.4 Verknüpfte Daten erfassen

IEG ermöglicht das Erstellen von Verknüpfungsbeziehungen zwischen Entitäten. Da für verschachtelte Entitäten und Listen die Einschränkung gilt, dass sie nur auf zwei Ebenen verschachtelt werden können, ist diese Funktionalität hilfreich. Die Verwendung von Verknüpfungsbeziehungen bietet eine wirksame Alternative dazu, Entitäten auf drei Ebenen zu verschachteln.

Beispiel: Es müssen Informationen über die Beschäftigung der Personen eines Haushalts erfasst werden. Informationen über die Beschäftigung können unabhängig von den Informationen zum Einkommen erfasst werden, da es für eine Beschäftigung mehrere Einkommen geben kann.

Nachdem die Informationen zum Einkommen und zur Beschäftigung erfasst und die Entitäten erstellt worden sind, kann die Verknüpfung zwischen den Entitäten erfolgen. Das geschieht durch Erstellen der Entität "Beziehung". Eigner der Beziehungsentität ist gewöhnlich eine der Entitäten, die an der Beziehungen teilhaben. Die Beziehungsentität wird wie andere Beziehungstypen als eine Sequenz dargestellt.

Das Definieren einer Beziehungsentität erfordert die Möglichkeit, die verwandte Entität zu ermitteln, weshalb in der verwandten Entität ein Schlüssel definiert sein muss. Übertragen auf das Beispiel mit dem Einkommen und der Beschäftigung bedeutet dies, dass die Entität "Beschäftigung" einen Schlüssel enthält, dass für den Entitätstyp "EmploymentRelationship" (Beschäftigungsbeziehung) definiert ist und dass

die Entität "Einkommen" Eigner einer Sequenz von "EmploymentRelationships" ist, wie im Folgenden aufgeführt:

```
<xsd:element name="Employment">
  <xsd:complexType>
    <xsd:attribute name="employmentID" type="d:SVR_KEY" />
    <xsd:attribute name="employer" type="IEG_STRING" />
    <xsd:attribute name="employmentType" type="IEG_STRING" />
  </xsd:complexType>
  <xsd:key name="Employment_Key">
    <xsd:selector xpath="./Employment" />
    <xsd:field xpath="@employmentID" />
  </xsd:key>
</xsd:element>
<xsd:element name="Income">
  <xsd:complexType>
    <xsd:sequence minOccurs="0">
      <xsd:element ref="EmploymentRelationship" minOccurs="0"
                maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="type" type="IEG_STRING" />
    <xsd:attribute name="amount" type="IEG_MONEY" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="EmploymentRelationship">
  <xsd:complexType>
    <xsd:attribute name="employmentID" type="d:SVR_KEY" />
  </xsd:complexType>
</xsd:element>
```

Abbildung 20. Schema für verknüpfte Entitäten

Die Verknüpfung kann anschließend im Script erfasst werden, indem eine Listenfrage definiert und ein Attribut zum Verlinken der Entität angegeben wird, das auf den Schlüssel der verwandten Entität verweist. Um bei unserem Beispiel zu bleiben, kann auf einer Seite, die der Entität "Einkommen" zugeordnet ist, eine Listenfrage definiert werden, die den Schlüssel aus "EmploymentRelationship" angibt, mit dessen Hilfe die Entität "Beschäftigung" ermittelt wird.

Listenfragen sind Konstrukte, mit deren Hilfe der Benutzer aus einer Liste von Entitäten wählen kann. Weitere Informationen finden Sie unter 5.2, „Listenfragen”, auf Seite 27.

```
<question-page id="IncomePage" entity="Income"  ...
  <cluster>
    <layout>
      <label-width>0</label-width>
    </layout>
    <list-question link-entity="EmploymentRelationship.employmentID"
        entity="Employment" single-select="true">
      <label id="SelectEmployer.Label">
        <![CDATA[Select Employer]]>
      </label>
      <item-label>
        <label-element attribute-id="employer" />
      </item-label>
    </list-question>
  </cluster>
</question-page>
```

Abbildung 21. Verknüpfungsbeziehungen erstellen

4.5 Verknüpfte Daten in einer Zusammenfassung anzeigen

Die Verknüpfung zwischen Entitäten kann auf einer Zusammenfassungsseite angezeigt werden, indem der Liste von Entitäten eines Typs eine Spalte hinzugefügt wird, so dass die Details einer verwandten Entität angezeigt werden können. Um die verwandte Entität zu ermitteln, muss für diese Spalte ein Attribut zum Verlinken von Entitäten angegeben werden.

Das folgende Beispiel zeigt, wie die Einkommen einer Person auf der Zusammenfassungsseite aufgelistet werden können und gleichzeitig für jedes Einkommen der Name des verknüpften Arbeitgebers angezeigt werden kann:

```
<summary-page id="IncomeSummary"
  ...
  <list entity="Person" criteria="hasIncome==true"
                        show-icons="true">
    <title id="IncomeList.Title">Income</title>
    <description id="IncomeList.Description">Here's the income
        details you've entered for all the people in your
        household</description>
    <column id="firstName">
      <title id="FirstName.Title">First Name</title>
    </column>
    <list entity="Income" show-icons="false">
      <column id="type">
        <title id="IncomeType.Title">Income Type</title>
      </column>
      <column id="amount">
        <title id="IncomeAmount.Title">Income Amount</title>
      </column>
      <column id="employer"
              link-entity="EmploymentRelationship.employmentID"
              entity="Employment">
        <title id="Employer.Title">Employer</title>
      </column>
    </list>
  </list>
</summary-page>
```

Abbildung 22. Zusammenfassungsseite für das Verknüpfen von Entitäten

4.6 Verknüpfte Daten löschen

Wenn in einer Beziehung zwischen übergeordneter und untergeordneter Entität die übergeordnete Entität gelöscht wird, so werden auch alle ihre untergeordneten Entitäten gelöscht. Wird eine Entität gelöscht, die an einer Beziehung teilhat, so werden standardmäßig die Beziehungen für diese Entität gelöscht, nicht aber die verwandten Entitäten.

Beispiel: Die Details für alle Personen eines Haushalts wurden erfasst, Personenentitäten erstellt, die Beziehungen zwischen den Personen des Haushalts erfasst und die Beziehungsentitäten erstellt. Entscheidet sich nun der Benutzer, eine Person zu entfernen, so werden auch die Beziehungen entfernt, an denen diese Person teilhat, nicht aber die übrigen Personen des Haushalts.

Dieses Standardverhalten ist auch auf das Beispiel mit dem Einkommen und der Beschäftigung anwendbar: Entscheidet sich der Benutzer, ein Einkommen zu entfernen, so werden alle Beschäftigungsbeziehungen für dieses Einkommen entfernt, aber keine der Beschäftigungsentitäten.

Es besteht die Möglichkeit, das Standardverhalten für das Löschen verknüpfter Entitäten zu ändern, so dass alle Entitäten, die mit der entfernten Entität verwandt sind, ebenfalls entfernt werden.

Um dieses Standardverhalten zu ändern, kann der Definition einer Beziehungsentität im DS-Schema eine Anmerkung hinzugefügt werden, die ein Dokumentationselement enthält. Ein Dokumentationselement, das den Text "@curam.ieg.cascading.delete=true" enthält, zeigt an, dass beim Löschen der Beziehung auch verwandte Entitäten gelöscht werden sollten.

```
<xsd:element name="EmploymentRelationship">
  <xsd:annotation>
    <xsd:documentation>@curam.ieg.cascading.delete=true
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:attribute name="employmentID" type="d:SVR_KEY" />
  </xsd:complexType>
</xsd:element>
```

Abbildung 23. Schema für kaskadierende Löschoperationen

Für das Beispiel mit dem Einkommen und der Beschäftigung heißt das: Wenn während des Löschens einer Einkommensentität die Eigenschaft curam.ieg.cascading.delete für die Beschäftigungsbeziehung auf "true" gesetzt ist, wird auch jede verknüpfte Beschäftigungsentität gelöscht. Das Entfernen der Beschäftigungsentitäten auf diesem Wege bewirkt nicht das Entfernen anderer Einkommensentitäten.

Kapitel 5. Effiziente Wege der Datenerfassung

5.1 Einführung

In diesem Kapitel werden einige IEG-Funktionen genauer beschrieben, die eine effektivere und intuitivere Art der Erfassung von Informationen ermöglichen.

5.2 Listenfragen

In einem früheren Beispiel wurde die Anforderung untersucht, Informationen zum Einkommen der Personen eines Haushalts zu erfassen. Zum Erfassen der Einkommensinformationen nur der Personen, die tatsächlich ein Einkommen haben, wurde der Seite "Details zu den Haushaltsmitgliedern" eine Frage hinzugefügt, um anzuzeigen, ob die jeweilige Person über ein Einkommen verfügt oder nicht.

IEG bietet eine Alternative zum Abfragen derselben booleschen Frage für eine Anzahl an Entitäten an. Mithilfe einer Listenfrage können alle Antworten gleichzeitig erfasst werden.

Als Weiterführung des vorhergehenden Beispiels, wo Informationen zu den Personen eines Haushalts erfasst wurden, wird jetzt das Attribut `hasIncome` hinzugefügt, um anzuzeigen, ob für die jeweilige Person Einkommensinformationen erfasst werden sollen. Das geschieht wie folgt:

```
<xs:element name="Person">
  <xs:complexType>
    ...
    <xs:attribute name="hasIncome" type="IEG_BOOLEAN"/>
```

Abbildung 24. Personenschema mit "hasIncome"

Wie andere Fragen auch, müssen Listenfragen einem Cluster hinzugefügt werden. Wo Listenfragen differieren, muss der Typ der Entitäten angegeben werden, die in der Liste angezeigt werden sollen. Die ID der Listenfrage entspricht dem Namen des booleschen Attributs, das gesetzt werden sollte, wenn der Benutzer einen Eintrag aus der Liste auswählt. Wie andere Fragen auch, sollte eine Listenfrage mit einer Beschriftung versehen sein, die den Zweck der Frage angibt. Listenfragen sollten auch ein Element zur Elementbeschriftung besitzen. Die Elementbeschriftung gibt an, welches Attribut aus den Entitäten zur Ermittlung der Entitäten in der Liste verwendet werden sollte. Im folgenden Beispiel werden die Vornamen der Haushaltsmitglieder angezeigt, um sie zu ermitteln.

```
<question-page id="AnyoneHaveIncome">
...
  <cluster>
    <list-question id="hasIncome" entity="Person">
      <label id="HasIncome.Label">
             <![CDATA[Which people have income?]]>
      </label>
      <item-label>
        <label-element attribute-id="firstName"/>
      </item-label>
    </list-question>
  </cluster>
</question-page>
```

Abbildung 25. Listenfrage

Statt der Schleife, mit der die Angaben zu den Haushaltsmitgliedern erfasst werden, eine Frage hinzuzufügen, kann nach dem Erfassen der Angaben zu den Haushaltsmitgliedern eine Liste der Haushaltsmitglieder angezeigt werden. Daraufhin kann der Benutzer die Mitglieder auswählen, die über ein Einkommen verfügen.

Listenfragen sind besonders dann hilfreich, wenn sie in Verknüpfung mit einer Foreach-Schleife verwendet werden und auf die Frage verweisen, die in der Listenfrage im Kriterienausdruck der Schleife gesetzt wurde. Listenfragen können auch mit anderen Entitätstypen als den Personentypen verwendet werden.

5.2.1 Einzelauswahl

Listenfragen können auch verwendet werden, wenn die Auswahlmöglichkeiten sich gegenseitig ausschließen sollen. Wenn das Attribut `single-select` einer Listenfrage auf `true` gesetzt ist, kann nur einer der Einträge in der Liste ausgewählt werden.

Soll zum Beispiel angezeigt werden, welches Haushaltsmitglied die primäre Betreuungsperson ist, kann der Personenentität ein Attribut und dem Script eine Listenfrage mit Einfachauswahl hinzugefügt werden:

```
<xsd:element name="Person">
  <xsd:complexType>
    ...
    <xsd:attribute name="primaryCareGiver" type="IEG_BOOLEAN"/>
```

Abbildung 26. Personenschema für primäre Betreuungsperson

```
<question-page id="PrimaryCareGiver" ...>
...
  <cluster>
    <list-question id="primaryCareGiver" entity="Person"
          single-select="true" criteria="age > 14">
      <label id="PrimaryCareGiver.Label">
        <![CDATA[Which person is the primary care giver?]]>
      </label>
      <item-label>
        <label-element attribute-id="firstName" />
      </item-label>
    </list-question>
  </cluster>
```

Abbildung 27. Listenfrage "Single-select"

Die obige Listenfrage führt dazu, dass eine Liste der über 14jährigen Haushaltsmitglieder mit einem Optionsfeld neben jeder Person angezeigt wird, wodurch nur eine Person ausgewählt werden kann.

5.3 Codetabellenfragen

Ist ein Attribut in einem DS-Schema als Codetabelle definiert, so wird die entsprechende Frage standardmäßig in einer Dropdown-Liste angezeigt. Aus dieser Dropdown-Liste kann nur eine Antwort ausgewählt werden.

Wenn beispielsweise der Herkunftsbundesstaat eines Haushaltsmitglieds erfasst werden soll, kann eine neue Domänendefinition hinzugefügt werden, die die Codetabelle `AddressState` darstellt. Zudem kann ein Attribut zum Speichern des Herkunftsbundesstaats zur Personenentität hinzugefügt werden. Dazu geht man wie folgt vor:

```
...
<xsd:simpleType name="IEG_STATE_ADDRESS">
  <xsd:annotation>
    <xsd:appinfo>
      <D:options>
        <D:option name="code-table-name">AddressState</D:option>
      </D:options>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:restriction base="IEG_CODETABLE_CODE" />
</xsd:simpleType>
...

<xsd:element name="Person">
...
  <xsd:attribute name="homeState" type="IEG_STATE_ADDRESS" />
```

Abbildung 28. Codetabelle für Bundesstaaten sowie Attribut

Anschließend kann dem Script eine Frage zum Erfassen von Informationen zum Herkunftsbundesstaat hinzugefügt werden:

```
<question-page id="AboutYouPage" entity="Person">
...

  <cluster>
    <question id="homeState">
      <label id="State.Label">
        <![CDATA[Please select your home state:]]>
      </label>
    </question>
  </cluster>
```

Abbildung 29. Frage für Codetabelle zum Bundesstaat

Während der Ausführung des Scripts wird dem Benutzer die Frage in Form einer Dropdown-Liste angezeigt.

Das Definieren von Codetabellenfragen wird von IEG ebenfalls so unterstützt, dass der Benutzer eine Mehrfachauswahl treffen kann.

Wenn die Codetabellenfrage nur eine einzige Auswahl zulässt, kann die Antwort auf die Frage in einem einzelnen Attribut einer Entität gespeichert werden. Da in einer Codetabellenfrage mit Mehrfachauswahl mehrere Antworten möglich sind, muss eine Sequenz hinzugefügt werden, die alle Antworten speichert. Zudem muss ein neuer Entitätstyp definiert werden, der die Antworten in der Sequenz darstellt.

```
<xsd:element name="Person">

  <xsd:complexType>
    <xsd:sequence minOccurs="0">
      <xsd:element ref="State" minOccurs="0"
          maxOccurs="unbounded" />
      </xsd:sequence>
...
  </xsd:complexType>
</xsd:element>

<xsd:element name="State">
  <xsd:complexType>
    <xsd:attribute name="stateCode" type="IEG_STATE_ADDRESS" />
  </xsd:complexType>
</xsd:element>
```

Abbildung 30. Entität "Bundesstaat"

Eine Codetabellenfrage wird auf Mehrfachauswahl eingestellt, indem man das Attribut `multi-select` der Frage auf `true` setzt. Wenn eine Codetabelle mit Mehrfachauswahl hinzugefügt wird, muss der Cluster, dem die Frage hinzugefügt wird, dem neuen Entitätstyp zugeordnet werden, der die Antworten auf die Frage darstellt. Im genannten Beispiel muss der Cluster der Entität `State` zugeordnet werden. Die Seite, die die Frage mit der Mehrfachauswahl enthält, muss der Entität zugeordnet sein, die die Sequenz enthält. In diesem Fall muss die Seite der Personenentität zugeordnet werden. Als letzter Schritt muss zum Sichtbarmachen einer Anzahl von Optionen in einer Codetabellenfrage mit Mehrfachauswahl ein Layout zur Frage hinzugefügt werden. In diesem Layout sollte die Anzahl sichtbarer Zeilen für die Frage angegeben sein. Übersteigt die Anzahl der verfügbaren Optionen für die Frage die Anzahl an Zeilen, die im Layout angegeben sind, wird der Frage eine Bildlaufleiste hinzugefügt.

```
<question-page id="AboutYouPage" entity="Person">
...

  <cluster entity="State">
    <question id="stateCode" multi-select="true">
      <label id="State.Label">
        <![CDATA[Please select the states you lived in:]]>
      </label>
      <layout>
        <num-rows>4</num-rows>
      </layout>
    </question>
  </cluster>
```

Abbildung 31. Codetabellenfrage mit Mehrfachauswahl

Während der Ausführung des Scripts wird dem Benutzer die Frage in Form einer Liste von Codetabellenbeschreibungen mit einem Kontrollkästchen für jeden Eintrag angezeigt.

5.4 Bedingte Elemente

IEG-Scripts können über mehrere unterschiedliche bedingte Elemente verfügen: Sektionen, Seiten oder Cluster. Bedingte Elemente können ausgehend von Fragen vorhergehender Seiten oder vorab im DS eingegebenen Daten angezeigt oder ausgeblendet werden.

5.4.1 Bedingte Sektionen

Es ist möglich, Sektionen aus einer Scriptausführung zu entfernen, indem man einen Ausdruck zu Beginn der Ausführung bewertet: Ist die Sektion nicht sichtbar, so wird sie im Sektionsfenster nicht aufgeführt und der Ausdruck während der Scriptausführung nicht erneut bewertet.

Mithilfe eines vorab gefüllten DS wie unter 8.3, „Scripts vorab mit erfassten Daten füllen”, auf Seite 55 beschrieben kann man, abhängig von den Bedingungen außerhalb des Scripts, eine Markierung für eine Entität setzen. Beispiel: Es ist eine Entität "IntakeInformation" mit dem booleschen Attribut "collectIncomeInformation" vorhanden. In diesem Fall kann im Script eine Sektion zum Einkommen angegeben werden:

```
...
<section visible="IntakeInformation.collectIncomeInformation==true">
   ...
</section>
...
```

Abbildung 32. Sichtbares Attribut einer Sektion

Damit wird, wenn das Attribut "collectIncomeInformation" den Wert "false" hat, die Sektion zum Einkommen ausgeblendet, so als wäre sie in der Scriptdefinition nicht vorhanden.

Muss eine Sektion abhängig von den Antworten aus vorhergehenden Sektionen aktiviert oder inaktiviert werden, so ist es möglich, alle Seiten einer Sektion in einer einzigen Bedingung einzuschließen. Anders als das sichtbare Attribut kann diese Bedingung bei jedem Vorkommen der Sektion bewertet werden, weshalb es möglich ist, zurückzugehen und eine Antwort zu ändern, die die Navigierbarkeit einer Sektion beeinflusst. Die Sektion erscheint weiterhin im Sektionsfenster, jedoch abgeblendet, so dass der Benutzer dort nichts anklicken kann.

Das vorhergehende Beispiel kann so modifiziert werden, dass die Frage "collectIncomeInformation" gleich zu Anfang des Scripts gestellt wird. Die Sektion zum Einkommen kann daraufhin wie folgt geändert werden:

```
...
<section>
   <condition
     expression="IntakeInformation.collectIncomeInformation">
     ...
   </condition>
</section>
...
```

Abbildung 33. Bedingte Sektion

5.4.2 Bedingte Seiten

Je nach dem Wert des Bedingungsausdrucks können Seiten angezeigt oder ausgeblendet werden. Auch Schleifen können in solche Bedingungen eingeschlossen werden.

Die bereits erwähnte bedingte Sektion, in der eine Bedingung den gesamten Inhalt der Sektion einschließt, bildet ein Beispiel für bedingte Seiten.

5.4.3 Bedingte Cluster

Cluster können auch in einem Bedingungselement eingeschlossen sein. Wenn der Ausdruck eines Bedingungselements nicht auf eine der Fragen derselben Seite verweist, ist dieser Cluster ein statischer bedingter Cluster. Das liegt daran, dass vor dem Anzeigen der Seite bestimmt werden kann, ob der Cluster angezeigt wird oder nicht.

Nachdem beispielsweise Informationen zu den Haushaltsmitgliedern erfasst worden sind, soll vielleicht eine weitere Seite hinzugefügt werden, auf der weitere persönliche Angaben abgefragt werden, darunter die Angabe, ob die betreffende Person schwanger ist. Dazu muss das neue Attribut `isPregnant` zur Personenentität hinzugefügt werden, mit dem diese Information gespeichert wird:

```
<xsd:element name="Person">
  <xsd:complexType>
    ...
    <xsd:attribute name="isPregnant" type="IEG_BOOLEAN"/>
```

Abbildung 34. Zusätzliches Personenattribut

Diese Frage ist natürlich nur dann anwendbar, wenn die Geschlechtsangabe "weiblich" lautet. Deshalb kann der Cluster in eine Bedingung eingeschlossen werden, so dass er nur dann angezeigt wird, wenn der Bedingungsausdruck mit "true" bewertet wird. Die neue zusätzliche Seite mit Angaben zur Person kann folgendermaßen definiert werden:

```
<question-page id="AboutTheClientContinued" entity="Person" ...>
  <condition expression="Person.gender==&quot;SX2&quot;">
    <cluster>
      <question id="isPregnant" mandatory="true">
          <label id="IsPregnant.Label">
              Are you pregnant?
          </label>
          <help-text id="IsPregnant.HelpText">
              Are you pregnant?
          </help-text>
      </question>
    </cluster>
  </condition>
</question-page>
```

Abbildung 35. Statischer bedingter Cluster

Befindet sich hingegen eine der Fragen, auf die im Bedingungsausdruck verwiesen wird, auf derselben Seite, so handelt es sich um einen dynamischen bedingten Cluster. Das heißt, dass der Cluster angezeigt oder ausgeblendet wird, je nachdem, wie der Benutzer die Antworten auf die Fragen dieser Seite ändert. Dieses dynamische IEG-Feature erfordert, dass JavaScript im Browser aktiviert ist. Da die Ausdrücke ohne Serveraufruf bewertet werden, kann es vorkommen, dass die Ausdrücke eines dynamischen bedingten Clusters nicht auf angepasste Funktionen verweisen.

Wenn das obige Beispiel bei gleichbleibendem DS-Schema dahingehend geändert wird, dass der bedingte Cluster auf derselben Seite definiert ist wie die Frage nach dem Geschlecht, so handelt es sich um einen dynamischen bedingten Cluster.

```
<question-page id="AboutTheClient" entity="Person" ...>
...
  <cluster>
    <title id="DetailsCluster.Title">
      <![CDATA[Personal Details]]>
    </title>
...
    <question id="gender" mandatory="true">
      <label id="Gender.Label">
        <![CDATA[Gender:]]>
      </label>
    </question>
...
  <condition expression="Person.gender==&quot;SX2&quot;">
    <cluster>
      <question id="isPregnant" mandatory="true">
        <label id="IsPregnant.Label">
        <![CDATA[Are you pregnant?]]>
        </label>
      </question>
    </cluster>
  </condition>
</question-page>
```

Abbildung 36. Dynamischer bedingter Cluster

Die Frage nach der Schwangerschaft wird dynamisch angezeigt bzw. ausgeblendet, sobald sich der ausgewählte Wert für das Geschlecht ändert. Das dynamische Verhalten einer Seite kann über Textfelder, Datumsfelder, Kontrollkästchen, Optionsfelder oder auszuwählende Elemente ausgelöst werden. Es wird jedoch nicht über die Antwort auf eine Mehrfachauswahlfrage oder eine Fragenmatrix ausgelöst, was mit Einschränkungen in der Ausdruckssyntax zusammenhängt.

Es wird darauf hingewiesen, dass um einen Cluster herum nur eine Bedingungsebene zulässig ist, d.h. bedingte Cluster können nicht in weitere Bedingungen verschachtelt werden. Der Bedingungsausdruck für einen dynamischen bedingten Cluster kann auf Fragen derselben Seite verweisen, die ihrerseits in dynamischen bedingten Clustern definiert sind. Dies bewirkt eine kaskadierende Abhängigkeit zwischen den Clustern.

5.5 Fragenmatrizen

Die in 5.2, „Listenfragen”, auf Seite 27 dargestellten Listenfragen stellen dieselbe boolesche Frage zu einer Gruppe von Entitäten. Es ist möglich, dieselbe Codetabellenfrage zu einer Entitätengruppe mithilfe von Fragenmatrizes zu stellen.

Mit einer Fragenmatrix wird anhand einer Codetabelle eine Liste von Fragen angezeigt. Für jeden aufgeführten Codetabellenwert und jede Entität werden Kontrollkästchen angezeigt, mit denen der Benutzer alle die Werte auswählen kann, die auf eine bestimmte Entität zutreffen.

Wenn beispielsweise für jedes Haushaltsmitglied mögliche Abstufungen von Drogenmissbrauch erfasst werden sollen, kann eine neue Domänendefinition hinzugefügt werden, die die Codetabelle `SubstanceAbuse` (Drogenmissbrauch) darstellt. Zudem kann ein Attribut zum Speichern der Abstufung des Drogenmissbrauchs hinzugefügt werden. Dazu geht man wie folgt vor:

```
<xsd:simpleType name="IEG_SUBSTANCEABUSE">
  <xsd:annotation>
    <xsd:appinfo>
      <D:options>
        <D:option name="code-table-name">SubstanceAbuse</D:option>
      </D:options>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:restriction base="IEG_CODETABLE_CODE" />
</xsd:simpleType>

<xsd:element name="Person">
  <xsd:complexType>
    ...
    <xsd:attribute name="substanceAbuse"
                   type="IEG_SUBSTANCEABUSE" />
```

Abbildung 37. Attribut "SubstanceAbuse"

Anschließend wird die Fragenmatrix als eine reguläre Listenfrage definiert, nur dass sie aufgrund der Tatsache, dass sie auf einer Codetabelle basiert statt auf einem booleschen Wert, anders angezeigt wird.

```
...
<list-question entity="Person" id="substanceAbuse"
      criteria="age > 14">
  <label id="SubstanceAbuse.Label">
    <![CDATA[Substance Abuse:]]>
  </label>
  <item-label>
    <label-element attribute-id="firstName" />
  </item-label>
</list-question>
```

Abbildung 38. Codebeispiel für Fragenmatrix

Die Fragenmatrix aus dem obigen Beispiel, mit der Informationen zum Drogenmissbrauch mehrerer Mitglieder eines Haushalts erfasst wurden, wird im IEG-Player als eine Matrix angezeigt, in der jede Zeile einer Codetabellenbeschreibung und jede Spalte einer Person entspricht.

5.6 Fast-Path-Navigation

Standardmäßig werden beim erneuten Iterieren des Benutzers durch ein Script all jene Seiten erneut angezeigt, deren Bearbeitung besonders bei großen Haushalten mühselig werden könnte. Durch die Fast-Path-Navigation ist der Benutzer in der Lage, IEG-Scripts schneller durchzuarbeiten, indem er Schleifen oder bedingte Seiten, die bereits beantwortet wurden, automatisch überspringen lässt.

Diese Funktion ist optional und standardmäßig ausgeschaltet. Für Schleifen und Bedingungen kann sie aktiviert werden. (Für die Aktivierung der Fast-Path-Navigation ziehen Sie das Handbuch `Authoring Scripts using Intelligent Evidence Gathering (IEG)` zu Rate.

Beim ersten Vorkommen eines Fast-Path-Elements ist sein Verhalten normal. Bei darauffolgenden Navigationen durch das Script werden nur die neuen Seiten innerhalb dieser Fast-Path-Elemente angezeigt. Die zuvor angezeigten Seiten werden nun übersprungen. Diese Funktion verhindert nicht die Bearbeitung von Daten über die Bearbeitungslinks auf der Zusammenfassungsseite, sofern notwendig.

Fast Path kann für die folgenden Szenarien verwendet werden:

- Listenfrage mit Schleife
- Kriterien für die Anspruchsberechtigung
- Fast-Path-Bedingungen
- Bedingung in einer Fast-Path-Schleife

5.6.1 Listenfrage mit Schleife

Unter Verwendung derselben Listenfrage wie unter 5.2, „Listenfragen”, auf Seite 27 beschrieben, sollen nun Informationen zu den Einkommen der Personen eines Haushalts gesammelt werden. Dazu verwendet man eine verschachtelte Fast-Path-Schleife, wie sie im folgenden Beispiel beschrieben wird:

```
...
<loop loop-type="for-each" entity="Person"
  criteria="hasIncome==true" fast-path="true">
  <loop loop-type="while" loop-expression="hasMoreIncome"
    entity="Income">
    <question-page id="IncomePage" entity="Income"
      show-person-tabs="true">
      <title id="IncomePage.Title">
        <![CDATA[Income Details]]>
      </title>
      <cluster>
        <title id="IncomeDetails.Title">
          <![CDATA[Income Details]]>
        </title>
        <question id="type">
          <label id="Type.Label">
            <![CDATA[Type:]]>
          </label>
        </question>
        <question id="amount">
          <label id="Amount.Label">
            <![CDATA[Amount:]]>
          </label>
        </question>
        <question id="hasMoreIncome"
          control-question="true"
          control-question-type="IEG_BOOLEAN">
          <label id="ContinueQuestion.Label">
            <![CDATA[Does %1s have any more income?]]>
            <argument id="Person.firstName" />
          </label>
        </question>
      </cluster>
    </question-page>
  </loop>
</loop>
```

Abbildung 39. Codebeispiel für Fast-Path-Listenfrage mit Schleife

Beim ersten Vorkommen einer Listenfrage erfassen die Seiten, die auf die Schleife folgen, das Einkommen der ausgewählten Personen. Beim erneuten Besuch der Seite, die die Listenfrage enthält, kann Folgendes geschehen:

- Bei unveränderter Einstellung der Kontrollkästchen kann man durch Klicken auf "Weiter" die Einkommensschleife überspringen und die Seite anzeigen, die auf die Schleife folgt.
- Sind einige der Kontrollkästchen inaktiviert, so werden durch Klicken auf "Weiter" die Einkommen der nicht ausgewählten Personen gelöscht; die Einkommensschleife wird übersprungen und die auf die Schleife folgende Seite angezeigt.
- Werden neue Kontrollkästchen aktiviert, bewirkt das Klicken auf "Weiter" das Überspringen der vorhandenen Einkommensseiten. Es werden die neuen Einkommensseiten für die neu ausgewählten Personen und anschließend die der Schleife folgende Seite angezeigt.
- Werden neue Kontrollkästchen aktiviert und andere bleiben inaktiviert, bewirkt man durch Klicken auf "Weiter" das Löschen der Einkommen der nicht ausgewählten Personen. Die vorhandenen Einkommensseiten werden übersprungen; die neuen Einkommensseiten für die neu ausgewählten Personen und anschließend die auf die Schleife folgende Seite werden angezeigt.

5.6.2 Kriterien für die Anspruchsberechtigung

Auf dem bisherigen Szenario aufbauend, können nun die Personen gefiltert werden, die in der Listenfrage angezeigt werden sollen. Die Schleife muss dafür nicht geändert werden. Da nur Personen über 18 Jahren berechtigt sind, ein Einkommen einzugeben, wird der Listenfrage ein Kriterium hinzugefügt. Beim erneuten Iterieren durch das Script stimmen einige Personen nicht mehr mit den Kriterien überein und erscheinen deshalb nicht in der Liste.

```
...
<list-question id="hasIncome" entity="Person" criteria="age > 18">
  <label id="HasIncome.Label">
    <![CDATA[Which people have income?]]>
  </label>
  <item-label>
    <label-element attribute-id="firstName" />
  </item-label>
</list-question>
```

Abbildung 40. Codebeispiel für Fast-Path-Listenfrage mit Kriterium zur Anspruchsberechtigung mittels Schleife

Dies ergibt ein Verhalten wie im vorherigen Szenario beschrieben. Sobald aber das Geburtsdatum einer Person geändert wird, geschieht Folgendes:

- Ist die Person nicht anspruchsberechtigt (unter 18) und ein Einkommen ist eingegeben worden, dann wird das entsprechende Einkommen automatisch gelöscht, sobald das neue Geburtsdatum abgeschickt wird.
- Wird die Person anspruchsberechtigt (über 18), so wird sie (nicht ausgewählt) beim nächsten Anzeigen der Seite mit Listenfragen in der Listenfrage angezeigt.

5.6.3 Fast-Path-Bedingungen

Angaben zur Schwangerschaft von weiblichen Haushaltsmitgliedern können mithilfe einer Bedingungsseite abgefragt werden. Wenn die Bedingung als Fast Path definiert ist, werden die Details zur Schwangerschaft beim erneuten Iterieren über die Haushaltsmitglieder ausgeblendet, da die Seiten in der Bedingung nur dann beim erneuten Iterieren durch das Script angezeigt werden, wenn die zuvor als "false" bewertete Bedingung nun infolge einer Änderung als "true" bewertet wird.

```
...
<question-page id="AboutYouPage" entity="Person">
  <title id="PrimaryPersonPage.Title">
    <![CDATA[About You]]>
  </title>
  <cluster>
    <title id="DetailsCluster.Title">
      <![CDATA[Personal Details]]>
    </title>
    <question id="firstName" mandatory="true">
      <label id="FirstName.Label">
        <![CDATA[First Name:]]>
      </label>
    </question>
    <question id="middleName">
      <label id="MiddleName.Label">
        <![CDATA[Middle Name:]]>
      </label>
    </question>
    <question id="lastName">
      <label id="lastName.Label">
        <![CDATA[Last Name:]]>
      </label>
    </question>
    <question id="gender" mandatory="true">
      <label id="Gender.Label">
        <![CDATA[Gender:]]>
      </label>
    </question>
    <question id="dateOfBirth" mandatory="true">
      <label id="DateOfBirth.Label">
        <![CDATA[Date Of Birth:]]>
      </label>
    </question>
  </cluster>
</question-page>
<condition expression="Person.gender==&quot;SX2&quot;"
  fast-path="true">
  <question-page id="PregnancyPage" entity="Person">
    <title id="PregnancyPage.Title">
      <![CDATA[About You: pregnancy]]>
    </title>
    <cluster>
      <title id="DetailsCluster.Title">
        <![CDATA[Personal Details About Your Pregnancy]]>
      </title>
      <question id="isPregnant" >
        <label id="IsPregnant.Label">
          <![CDATA[Are you pregnant?]]>
        </label>
      </question>
    </cluster>
  </question-page>
</condition>
```

Abbildung 41. Codebeispiel für Fast-Path-Bedingungen

Beim Bearbeiten der persönlichen Angaben kann Folgendes vorkommen:

- Sofern an der Geschlechtsangabe keine Änderung vorgenommen wurde, kann man durch Klicken auf "Weiter" die Bedingung überspringen, selbst wenn sie zum ersten Mal angezeigt wird.
- Hat die Geschlechtsangabe von männlich zu weiblich gewechselt, wird durch Klicken auf "Weiter" die Bedingungsseite angezeigt, auf der die Details zur Schwangerschaft angegeben werden können.
- Hat die Geschlechtsangabe von weiblich zu männlich gewechselt, werden durch Klicken auf "Weiter" die Details zur Schwangerschaft gelöscht und die Seite angezeigt, die der Bedingung folgt.

5.6.4 Bedingung in einer Fast-Path-Schleife

Wenn innerhalb einer Fast-Path-Schleife eine Bedingung definiert ist, verhält sich diese, als ob für die Schleife ein Kriterium verwendet wird, anstatt eine Bedingung zu verschachteln, jedoch mit folgender Ausnahme: Erhält die Bedingung den Wert "true", kann die in der Bedingung enthaltene Seite nicht angezeigt werden, da die Schleife keine neue Iteration aufweist und deshalb übersprungen wird. Hat die Bedingung den Wert "false", so werden die Seite und die verknüpften Daten nicht gelöscht, da die Bedingung nicht erneut bewertet wird. Aus diesem Grund wird empfohlen, ein Kriterium für die Schleife statt einer Bedingung zu verwenden.

```
...
<loop loop-type="for-each" entity="Person"
  fast-path="true">
  <condition expression="Person.hasIncome==true">
    <loop loop-type="while" loop-expression="hasMoreIncome"
      entity="Income">
      <question-page id="IncomePage" entity="Income"
        show-person-tabs="true">
        <title id="IncomePage.Title">
          <![CDATA[Income Details]]>
        </title>
        <cluster>
          <title id="IncomeDetails.Title">
            <![CDATA[Income Details]]>
          </title>
          <question id="type">
            <label id="Type.Label">
              <![CDATA[Type:]]>
            </label>
          </question>
          <question id="amount">
            <label id="Amount.Label">
              <![CDATA[Amount:]]>
            </label>
          </question>
          <question id="hasMoreIncome"
            control-question="true"
            control-question-type="IEG_BOOLEAN">
            <label id="ContinueQuestion.Label">
              <![CDATA[Does %1s have any more income?]]>
              <argument id="Person.firstName" />
            </label>
          </question>
        </cluster>
      </question-page>
    </loop>
  </condition>
</loop>
```

Abbildung 42. Codebeispiel für eine Bedingung in einer Fast-Path-Schleife

5.7 Implizit löschen

Wo immer es möglich ist, versucht die IEG-Engine Daten zu löschen, sobald sie feststellt, dass diese nicht mehr relevant sind.

Wenn eine Antwort durch den Benutzer explizit verändert wird (durch eine reguläre Frage, eine Listenfrage oder ein festgelegtes Attribut, jedoch nicht durch einen angepassten Funktionsaufruf), stellt die Engine fest, ob diese Antwort in einem Bedingungsausdruck, einem Listenfragenkriterium oder einem Schleifenkriterium verwendet wird. Ist dies der Fall, so wird der Ausdruck oder das Kriterium erneut bewertet. Wird der Wert "false" zurückgegeben, werden die entsprechenden Seiten entfernt und die verknüpften Daten gelöscht, ohne dass das Script erneut durchgegangen werden muss, um den Ausdruck oder das Kriterium aufzufinden.

Kapitel 6. Weitere Informationen zur Scriptentwicklung

6.1 Einführung

Die verschiedenen Konstrukte, die bisher vorgestellt wurden, decken bereits eine Bandbreite von Erfordernissen bei der Erfassung von Angaben ab. Einige Situationen erfordern jedoch zusätzliche Funktionen wie die Möglichkeit, Daten im schreibgeschützten Modus anzuzeigen oder externe Funktionen aufzurufen. In diesem Kapitel werden solche Fälle beschrieben.

Zudem werden in diesem Kapitel einige Dinge erwähnt, die beim Pflegen von IEG-Scripts, beim Stellen von Scripts unter Quellcodeverwaltung und beim Laden von Scripts in die Datenbank zu beachten sind.

6.2 Daten schreibgeschützt anzeigen

Es kommt vor, dass die Antwort auf eine Frage dem Benutzer in einer Form angezeigt werden muss, die sich nicht verändern lässt. Auf den Zusammenfassungsseiten, wo der Benutzer seine Antworten überprüfen und die Schaltfläche "Zurück" bzw. Bearbeitungslinks zum Ändern verwenden kann, ist dies bereits der Fall.

Auf einer Fragenseite kann ein schreibgeschütztes boolesches Attribut auf "true" gesetzt werden, um anzuzeigen, dass alle auf der Seite angezeigten Fragen nicht bearbeitbar sind.

Es existiert auch ein weiterführender Mechanismus: Attribute für Read-only-Ausdrücke können für verschiedene Scriptelemente (Sektionen, alle Seitentypen, Cluster, Fragen und Listenfragen) verwendet werden. Wird der Ausdruck mit "true" bewertet, so wird er auf alle im Element enthaltenen Fragen angewendet. Im einfachsten Fall ist der Ausdruck "true", wenn das Element in jedem Fall schreibgeschützt sein muss. Für eine Zusammenfassungsseite bedeutet das, dass die Links zum Hinzufügen, Bearbeiten und Löschen nicht angezeigt werden.

Wenn für die Scriptelemente Cluster, Fragen und Listenfragen ein Read-only-Ausdruck definiert ist und sich eine der im Ausdruck referenzierten Fragen auf derselben Seite wie das Scriptelement befindet, wird das Scriptelement dynamisch aktiviert oder inaktiviert und ist nicht mehr nur schreibgeschützt. Das heißt, dass Fragen aktiviert oder inaktiviert werden, je nachdem, wie der Benutzer Antworten auf andere Fragen dieser Seite verändert. An Stellen, wo der Read-only-Ausdruck eines Clusters eine Frage auf derselben Seite referenziert, werden alle im Cluster enthaltenen Fragen aktiviert bzw. inaktiviert. Dieses dynamische IEG-Feature erfordert, dass JavaScript im Browser aktiviert ist. Da die Ausdrücke ohne Serveraufruf bewertet werden, kann es vorkommen, dass die Ausdrücke zum dynamischen Aktivieren und Inaktivieren von Fragen nicht auf angepasste Funktionen verweisen.

Dynamische Read-only-Ausdrücke können auf Fragen derselben Seite verweisen, die ihrerseits dynamisch aktiviert und inaktiviert werden. Dies bewirkt eine kaskadierende Abhängigkeit zwischen den Fragen. Beim Definieren von Ausdrücken mit kaskadierenden Abhängigkeiten sollte man vorsichtig sein, denn in IEG wird nicht berücksichtigt, ob die Fragen, auf die im Read-only-Ausdruck verwiesen wird, aktiviert sind oder nicht, sondern nur der Wert der Frage. Das kann für den Benutzer verwirrend sein, weil nicht immer ersichtlich ist, wodurch das Aktivieren und Inaktivieren einer Frage gesteuert wird.

Wenn beim Anzeigen einer Frage das entsprechende DS-Attribut einen Wert besitzt, so wird dieser angezeigt, selbst wenn die Frage anfangs inaktiviert ist. Daraufhin kann der Benutzer die Frage aktivieren und Änderungen an der Antwort vornehmen. Wird die Frage inaktiviert, fällt ihr Wert zurück auf den Wert, der anfänglich für die Frage angezeigt wurde. Nachdem eine Seite abgeschickt ist, wird das DS-Attribut erst dann aktualisiert, wenn die Frage aktiviert wird. Deshalb wird beim wiederholten Anzeigen der Seite wieder der ursprüngliche Wert des DS-Attributs angezeigt.

Wenn die Frage für sich selbst oder eines ihrer übergeordneten Elemente einen dynamischen Read-only-Ausdruck besitzt, ist es nicht möglich, sie als obligatorisch zu markieren.

Das dynamische Aktivieren oder Inaktivieren von Scriptelementen wird auf Beziehungsseiten nicht unterstützt.

Die in Schleifen erfassten Informationen können in Form von Listen auf den Zusammenfassungsseiten angezeigt werden. Es ist aber auch möglich, dieses Listenkonstrukt auf regulären Seiten zu verwenden, ohne einen Read-only-Ausdruck in einem der Elemente angeben zu müssen, die die Liste umgeben. Der einzige Unterschied bei Zusammenfassungslisten besteht darin, dass keine Links zulässig sind.

Eine andere Möglichkeit besteht darin, das gesamte Script auf schreibgeschützt zu setzen. Das ist zum Beispiel dann hilfreich, wenn ein Fallbearbeiter ein Script überprüfen muss, ohne eine der Antworten verändern zu können. Das Script wird mithilfe der IEG-Laufzeit-API durch Setzen eines Schreibschutzanzeigers in der Scriptausführung gesetzt, wie im Folgenden beschrieben:

```
...
//Set read only flag.
IEGRuntime runtimeAPI = new IEGRuntime();
IEGScriptExecutionID runtimeExecID = new IEGScriptExecutionID();
runtimeExecID.executionID = execution.getExecutionID();
IEGReadOnlyFlag readOnlyFlag = new IEGReadOnlyFlag();
readOnlyFlag.readOnlyFlag = true;
runtimeAPI.setReadOnlyFlag(runtimeExecID, readOnlyFlag);
...
```

Abbildung 43. Schreibschutzanzeiger auf eine Scriptausführung setzen

6.3 Externe Funktionen mithilfe von Ausdrücken aufrufen

Ausdrücke finden sich an vielen Stellen eines Scripts, wo sie das Verhalten von Schleifen, Bedingungen u.ä. definieren. Informationen hierzu finden Sie im Anhang zur Ausdruckssyntax des Handbuchs `Authoring Scripts using Intelligent Evidence Gathering(IEG)`.

Diese Ausdrücke können auf Antworten verweisen, sie mithilfe verschiedener Operatoren kombinieren und auch Funktionen aufrufen (außer bei Verwendung für dynamische bedingte Cluster, da diese Ausdrücke im Browser bewertet werden).

Die oben beschriebenen Funktionen werden als angepasste Funktionen bezeichnet und unter Verwendung von Java™-Code definiert. Je nach ihrer Verwendung können sie einen der folgenden zwei Typen aufweisen:

- Angepasste Funktionen, die Parameter aufnehmen (beispielsweise durch Aufruf einer externen Funktion) und einen Wert zurückgeben. Sie bewirken keine Änderung am Inhalt des DS. Diese Funktionen werden in den meisten Ausdrücken verwendet.
- Wenn es das Ziel ist, den Inhalt des DS zu aktualisieren, kann die angepasste Funktion in einem eigenständigen Element, `callout`, verwendet werden. Der zurückgegebene Wert ist irrelevant, muss allerdings ein boolescher Wert sein. Die angepasste Funktion sollte keine Werte aktualisieren, die vor dem Aufruf beantwortet wurden. Das hat den Grund, dass die IEG-Engine keine Informationen über Aktualisierungen außerhalb des Scriptkontextes besitzt und daher keine Aktionen ausführen kann, die von den Aktualisierungen erfordert werden.

Ein Praxisbeispiel, bei dem der Aufruf einer externen Funktion erforderlich wäre, ist die Validierung einer US-Postleitzahl, die der Benutzer eingegeben hat, und das Füllen des Felds für den Bundesstaat anhand der eingegebenen Postleitzahl. Im Folgenden werden zwei unterschiedliche Verwendungsarten veranschaulicht.

Das Datenbankschema muss wie folgt erweitert werden, um die folgenden zwei Attribute zur Personenentität hinzuzufügen:

```
<xsd:attribute name="state" type="IEG_STRING"/>
<xsd:attribute name="zipCode" type="IEG_STRING"/>
```

Abbildung 44. Personenattribute zum DS-Schema hinzufügen

Zunächst soll die Postleitzahl anhand der Angabe für den Bundesstaat validiert werden, eine sehr simple Anwendung. Die Postleitzahl muss fünf Ziffern enthalten, von denen die ersten drei für den Bundesstaat stehen.

Die schon erwähnte Seite mit den persönlichen Angaben und die entsprechende Zusammenfassungsseite können mit zwei zusätzlichen obligatorischen Fragen modifiziert werden: "state" und "zipCode" ("Bundesstaat" und "Postleitzahl"):

```
<question id="state" mandatory="true">
    <label id="State.Label">
        State:
    </label>
    <help-text id="State.HelpText">
        The state you live in
    </help-text>
</question>
<question id="zipCode" mandatory="true">
    <label id="ZipCode.Label">
        ZIP Code:
    </label>
    <help-text id="ZipCode.HelpText">
        Your ZIP code
    </help-text>
</question>
```

Abbildung 45. Die Fragen "state" und "zipCode" in der Scriptdefinition

Anschließend muss die angepasste Funktion, die die Validierung durchführen soll, als Java-Klasse in dem Paket `curam.rules.functions` erstellt werden:

```
...
public class CustomFunctionValidateZipCode extends CustomFunctor {

  public Adaptor getAdaptorValue(final RulesParameters rp)
  throws AppException, InformationalException {

    final List<Adaptor> parameters = getParameters();
    final String zipCode =
      ((StringAdaptor) parameters.get(0)).getStringValue(rp);
    final String state =
      ((StringAdaptor) parameters.get(1)).getStringValue(rp);
    boolean valid = false;

    if (zipCode.length() == 5) {
      final String prefix = zipCode.substring(0, 3);
      //lookup the state prefixes
      if (prefix.equals("100")
        && state.equalsIgnoreCase("New York")) {
        valid = true;
      }
      if (prefix.equals("900")
        && state.equalsIgnoreCase("California")) {
        valid = true;
      }
    }

    return AdaptorFactory.getBooleanAdaptor(Boolean.valueOf(valid));
  }

}
```

Abbildung 46. Angepasste Funktion zum Validieren der Postleitzahl

In `<yourcomponent>/rulesets/functions/CustomFunctionMetaData.xml` müssen die folgenden Metadaten für die angepasste Funktion eingefügt werden:

```
<CustomFunctor name="CustomFunctionValidateZipCode">
  <parameters>
    <parameter>
      curam.util.rules.functor.Adaptor$StringAdaptor
    </parameter>
    <parameter>
      curam.util.rules.functor.Adaptor$StringAdaptor
    </parameter>
  </parameters>
  <returns>curam.util.rules.functor.Adaptor$BooleanAdaptor</returns>
</CustomFunctor>
```

Abbildung 47. Metadaten für eine angepasste Funktion

Weitere Informationen zum Definieren von angepassten Funktionen finden Sie im Handbuch Cúram Rules Codification Guide.

Im genannten Beispiel greift die angepasste Funktion "ValidateZipCode" nicht auf eine externe Datenbank zu, um den entsprechenden Bundesstaat zu suchen. Im Idealfall sollte sie diese Suche ausführen und anschließend den wiedergegebenen Bundesstaat mit der Angabe für den Bundesstaat vergleichen, die eingegeben wurde. Aus Gründen der Vereinfachung sind im obigen Beispiel nur zwei Postleitzahlenpräfixe fest codiert.

Die Validierung wird nun in die Seite mit den persönlichen Angaben eingefügt:

```
<validation
  expression="ValidateZipCode(Person.zipCode, Person.state)">
    <message id="InvalidZipCode">
        The ZIP code is invalid.
    </message>
</validation>
```

Abbildung 48. Validierung der Postleitzahl in der Scriptdefinition

Mit dem Klicken auf "Weiter" werden die Antworten auf die Fragen "zipCode" und "state" an die angepasste Funktion übergeben, die im Falle gültiger Antworten den Wert "true" zurückgibt. Anschließend wird die nächste Seite angezeigt.

Gibt die angepasste Funktion den Wert "false" zurück, wird die in der Validierung angegebene Nachricht am Anfang der Seite mit den persönlichen Angaben angezeigt und der Zugriff auf die nächste Seite so lange blockiert, bis gültige Antworten übergeben werden.

Da sie keine Veränderungen bewirkt, hat die angepasste Funktion keine Nebeneffekte. Sie führt lediglich anhand der Parameter eine Operation durch und gibt ein Ergebnis zurück.

Es besteht auch die Möglichkeit, die obligatorische Markierung für die zwei neuen Fragen zu entfernen und die Antworten nur dann zu validieren, wenn beide angegeben worden sind. In dem Fall müsste der Prüfausdruck mithilfe der sofort einsatzfähigen angepassten Funktion "isNotNull", die überprüft, ob der angegebene Parameter Null ist, folgendermaßen geändert werden:

```
"not(isNotNull(Person.zipCode) and isNotNull(Person.state))
    or ValidateZipCode(Person.zipCode, Person.state)"
```

Abbildung 49. Alternativer Prüfausdruck

Alternativ dazu besteht die Möglichkeit, die Frage "state" mit der angegebenen Postleitzahl zu füllen. Zu diesem Zweck wird auf der Seite mit den persönlichen Angaben nur nach "zipCode" (mit der obligatorischen Markierung) gefragt, und die Zusammenfassungsseite zeigt sowohl Bundesstaat als auch Postleitzahl an.

Folgende angepasste Funktion sollte definiert werden:

```
...
public class CustomFunctionpopulateState extends CustomFunctor {

  public Adaptor getAdaptorValue(final RulesParameters rp)
  throws AppException, InformationalException {

    final IEG2Context ieg2Context = (IEG2Context) rp;
    final long rootEntityID = ieg2Context.getRootEntityID();
    String schemaName;
    //schemaName has to be hard-coded or retrieved outside of IEG
    Datastore ds = null;
    try {
      ds =
        DatastoreFactory.newInstance().openDatastore(
            schemaName);
    } catch (NoSuchSchemaException e) {
      throw new AppException(IEG.ID_SCHEMA_NOT_FOUND);
    }

    Entity applicationEntity = ds.readEntity(rootEntityID);

    Entity personEntity =
      applicationEntity.getChildEntities(
        ds.getEntityType("Person"))[0];
    String zipCode = personEntity.getAttribute("zipCode");
    String state = "Unknown";
    final String prefix = zipCode.substring(0, 3);
    //lookup the state prefixes
    if (prefix.equals("100")) {
      state = "New York";
    }
    if (prefix.equals("900")) {
      state = "California";
    }
    personEntity.setAttribute("state", state);
    personEntity.update();
    return AdaptorFactory.getBooleanAdaptor(new Boolean(true));
  }

}
```

Abbildung 50. Angepasste Funktion zum Füllen des Felds "Bundesstaat"

Dazu die Metadaten:

```
<CustomFunctor name="CustomFunctionpopulateState">
  <returns>curam.util.rules.functor.Adaptor$BooleanAdaptor</returns>
</CustomFunctor>
```

Abbildung 51. Metadaten der angepassten Funktion

Zwischen der Seite mit den persönlichen Angaben und der Zusammenfassungsseite muss ein Aufrufelement eingefügt werden, das diese angepasste Funktion aufruft:

```
<callout id="populateAddress" expression="populateState()"/>
```

Abbildung 52. Aufruf zum Füllen des Felds "Bundesstaat" in der Scriptdefinition

In diesem Fall bewirkt die angepasste Funktion eine Veränderung im DS, indem sie das Feld "Bundesstaat" zur Personenentität füllt. Der Kontext enthält die Stammebenen-Entitäts-ID und die Ausführungs-ID, wodurch das Aktualisieren des DS erleichtert wird. Befindet sich der Aufruf in einer Schleife, enthält der Kontext ebenfalls die aktuelle Entitäts-ID.

6.4 Scripts wiederverwenden

Eine Scriptdefinition kann in mehrere Dateien aufgegliedert werden, wodurch sie wiederverwendbar wird.

Um das zu erreichen, muss eine Scriptdefinition auf untergeordnete Scripts verweisen. Jedes dieser untergeordneten Scripts wird dann zu einem eigenständigen Script, das unabhängig von den anderen ausgeführt werden kann.

Es folgt ein Beispiel für ein Script, das als untergeordnetes Script verwendet werden kann:

```
<?xml version="1.0" encoding="UTF-8"?>
<ieg-script ...>
  <identifier id="Subscript" scriptversionnumber="V1" type="Test" />
  <question-page ...>
    ...
  </question-page>
  ...
</ieg-script>
```

Abbildung 53. Untergeordnetes Script mit enthaltenen Seiten

Das Script im obigen Beispielcodefragment kann wie folgt als untergeordnetes Script in ein anderes Script eingeschlossen werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<ieg-script ...>
  <identifier id="Script" scriptversionnumber="V1" type="Test" />
  <section>
    <ieg-sub-script>
         <identifier id="Subscript"
         scriptversionnumber="V1" type="Test" />
    </ieg-sub-script>
  </section>
  <section>
    ...
  </section>
  ...
</ieg-script>
```

Abbildung 54. Untergeordnetes Script in ein Script einschließen

Ein untergeordnetes Script kann an folgenden Positionen in ein Script eingefügt werden:

- Enthält das Script Sektionen und das untergeordnete Script ebenfalls, so muss das untergeordnete Script auf der höchsten Ebene unterhalb des übergeordneten IEG-Scriptelements eingefügt werden.
- Enthält das Script Sektionen, das untergeordnete Script jedoch nicht, muss das untergeordnete Script in eine Sektion des übergeordneten Scripts eingefügt werden.
- Enthält das Script keine Sektionen, kann auch das untergeordnete Script keine Sektionen enthalten. Es wird auf höchster Ebene unterhalb des IEG-Scriptelements eingefügt.

Zu beachten ist auch die Einschränkung, dass ein untergeordnetes Script nur einmal in einem Script erscheinen darf, da die Seiten-IDs innerhalb des neuen Scripts eindeutig sein müssen.

Zudem könnte ein Script auch an anderen Stellen als untergeordnetes Script verwendet werden. Sorgen Sie beim Ändern von Scripts dafür, dass jegliche auf sie verweisenden Scripts erneut getestet werden, so dass die Änderungen keine unerwünschten Folgen nach sich ziehen.

6.5 Quellcodeverwaltung und Versionskontrolle

IEG-Scriptdefinitionen werden in der Datenbank gespeichert. Beim Bearbeiten eines IEG-Scripts mithilfe des IEG-Editors wird das Script an seinem Ort bearbeitet und direkt in der Datenbank aktualisiert. IEG-Scriptdefinitionen sind Entwicklungsartefakte. Aus Sicht eines Softwarekonfigurationsmanagements ist es wichtig, diese Artefakte unter Quellcodeverwaltung zu stellen, so wie man es auch mit anderen Artefakten tut.

Von den IEG-Scriptadministrationsfenstern aus kann eine Scriptdefinition heruntergeladen werden. Nachdem die Option zum Herunterladen eines Scripts ausgewählt ist, wird zunächst das Script aus der Datenbank abgerufen. Daraufhin werden die Eigenschaftendateien, die mit der Scriptdefinition verknüpft sind, aus dem RS abgerufen und schließlich die Texteigenschaften in die Scriptdefinition eingefügt, bevor diese verfügbar gemacht wird. Wenn das Script auf diese Weise heruntergeladen wird, werden jedoch nicht alle Ressourcen bereitgestellt, die mit einer Scriptdefinition verknüpft sind. Es werden beispielsweise Eigenschaftendateien nicht in mehreren Ländereinstellungen bereitgestellt, ebensowenig wie Bilder und Symbole. Weitere Informationen zur Datenbankdarstellung in IEG-Scripts finden Sie im Anhang `Konformität` des Entwicklerhandbuchs `Authoring Scripts using Intelligent Evidence Gathering (IEG)`.

Beim Füllen der Datenbank mit Scriptdefinitionen ist es wichtig, die Funktionsunterschiede zwischen dem Importieren eines Scripts über die IEG-Scriptadministrationsfenster und dem Laden von Scriptdefinitionen über DMX-Dateien zu beachten.

Kapitel 7. IEG in eine Cúram-Anwendung integrieren

7.1 Einführung

In diesem Kapitel wird ein Überblick darüber gegeben, wie IEG in eine Anwendung integriert werden kann. IEG lässt sich auf zwei Weisen integrieren: durch Öffnen des Players in einer Registerkarte oder in einem modalen Dialogfenster. Zu den hier behandelten Integrationsaufgaben gehören das Erstellen der Scriptausführung, Setzen von Beenden- und Verlassen-Seiten, Ausführen in einer Registerkarte, Ausführen in einem modalen Fenster, Bereinigen von Anwendungsdaten sowie das Fortsetzen von Scripts.

7.2 Scriptausführung erstellen

Es wird empfohlen, vor dem Öffnen des IEG-Players aus einer Anwendung die Scriptausführung mithilfe der öffentlich zugänglichen API zu erstellen. Daraufhin kann die Ausführungs-ID an den Player übermittelt werden.

Mit dem folgenden Beispielcodefragment wird das Erstellen einer Scriptausführung mithilfe der öffentlich zugänglichen API gezeigt:

```
...

// create the script execution
final IEGRuntime runtimeAPI = new IEGRuntime();
final IEGScriptExecutionIdentifier executionIdentifier =
  runtimeAPI.createScriptExecution(iegScriptID, schemaName);
```

Abbildung 55. Scriptausführung erstellen

7.3 Umleitungs-URL angeben

Die Attribute `finish-page` und `quit-page` (Seite beenden/Seite verlassen) eines IEG-Scripts geben an, zu welcher URL beim Verlassen des IEG-Players umgeleitet werden soll. Auf diese Weise sorgen sie für eine Verbindung zwischen dem IEG-Player und der Anwendung. Diese Attribute werden im Kapitel `IEG Script Element Reference` (Referenz für IEG-Scriptelemente) des Handbuchs `Authoring Scripts using Intelligent Evidence Gathering (IEG)` näher beschrieben.

Ändern Sie das Beispielscript dahingehend, dass es die untenstehenden Attribute einschließt:

```
<ieg-script
  finish-page="IEG2_listAllIEG2Scripts"
  quit-page="IEG2_listAllIEG2Scripts"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ieg-schema.xsd">
    ...
<ieg-script>
```

Abbildung 56. Script mit definierter Beenden-Seite und Verlassen-Seite

Im obigen Beispiel bewirkt das Beenden oder Verlassen des Scripts die Umleitung zur Liste aller IEG-Scripts, die im Administrationsfenster bereitgestellt wird.

7.4 IEG-Player in einer Registerkarte ausführen

Es ist weniger aufwendig, den IEG-Player in einer Registerkarte auszuführen als in einem modalen Dialogfenster. Voraussetzung ist, dass der Link zum Öffnen auf `ieg/Screening.do` verweist und die `executionID` übergibt. Mit `Screening.do` wird der IEG-Player aufgerufen. Die Parameter dazu sind unten aufgeführt.

Hier ein Beispiel eines Auflöse-UIM, mit dem der IEG-Player in einer Registerkarte geöffnet wird:

```
<?xml version="1.0" encoding="UTF-8"?>
  <PAGE PAGE_ID="System_IEGResolver">
    <JSP_SCRIPTLET>
    <![CDATA[

      String scriptID = request.getParameter("scriptID");
      String scriptType = request.getParameter("scriptType");
      String scriptVersion = request.getParameter(
          "scriptVersion");
      String schemaName = request.getParameter("schemaName");
      String name = request.getParameter("name");

      String executionIDParam =
          request.getParameter("executionIDParam");
      String url = null;

      curam.omega3.request.RequestHandler
          rh = curam.omega3.request.
      RequestHandlerFactory.getRequestHandler(request);

      String context = request.getContextPath() + "/";

      if (executionIDParam == null) {
        // Need to check to see if there are any script validation
        // errors before running the script.

        String contextWithUserPreferences = context +
          curam.omega3.user.UserPreferencesFactory
          .getUserPreferences(
              pageContext.getSession()).getLocale() + "/";

        curam.interfaces.IEGScriptAdminPkg.
        IEGScriptAdmin_checkForScriptErrors_TH
            iegScriptAdminCheckForErrors
                = new curam.interfaces.IEGScriptAdminPkg.
        IEGScriptAdmin_checkForScriptErrors_TH();

        iegScriptAdminCheckForErrors.setFieldValue(
        iegScriptAdminCheckForErrors.key$scriptID_idx, scriptID);
        iegScriptAdminCheckForErrors.setFieldValue(
        iegScriptAdminCheckForErrors.key$scriptType_idx,
            scriptType);
        iegScriptAdminCheckForErrors.setFieldValue(
        iegScriptAdminCheckForErrors.key$scriptVersion_idx,
            scriptVersion);
        iegScriptAdminCheckForErrors.setFieldValue(
        iegScriptAdminCheckForErrors.key$schemaName_idx,
            schemaName);
        //Call the method.
        iegScriptAdminCheckForErrors.callServer();

        String errorsPresentInScript =
            iegScriptAdminCheckForErrors.getFieldValue(
                iegScriptAdminCheckForErrors
                    .result$errorsExist_idx);
        boolean errorsPresent =
            Boolean.valueOf(errorsPresentInScript).booleanValue();

        if (errorsPresent) {

          // If there are errors, redirect to the validation error
          // page.
          String redirectTo = contextWithUserPreferences
          + "System_listValidationErrorsForRunPage.do"
          + "?name=" + name
          + "&scriptID=" + scriptID
          + "&scriptType=" + scriptType
          + "&scriptVersion=" + scriptVersion
          + "&schemaName=" + schemaName;
          url = redirectTo + "&amp;" + rh.getSystemParameters();

        } else {
```

7.5 IEG-Player in modalem Dialogfenster ausführen

Der IEG-Player kann in einem modalen Dialogfenster geöffnet werden. Es gibt bestimmte Aspekte, die ein Scriptentwickler in diesem Bereich berücksichtigen muss.

7.5.1 IEG-Player in modalem Dialogfenster öffnen

Um den IEG-Player in einem modalen Dialogfenster zu öffnen, öffnen Sie `Screening.do` im modalen Dialogfenster und übergeben Sie den Parameter `executionID` und die Systemparameter mithilfe eines Auflöse-UIM. `System_IEGResolverModal.uim` wird ohne Vorbereitungs- oder Anpassungsaufwand bereitgestellt, um diese Verarbeitung durchzuführen:

```
<PAGE PAGE_ID="System_IEGResolverModal">
          <JSP_SCRIPTLET>
    <![CDATA[

      String scriptID = request.getParameter("scriptID");
      String scriptType = request.getParameter("scriptType");
      String scriptVersion =
        request.getParameter("scriptVersion");
      String schemaName = request.getParameter("schemaName");
      String name = request.getParameter("name");

      // Need to check to see if there are any script
      // validation errors before running the script.
      curam.omega3.request.RequestHandler
        rh = curam.omega3.request.
          RequestHandlerFactory.getRequestHandler(request);

      String context = request.getContextPath() + "/";
      String contextWithUserPreferences = context +
       curam.omega3.user.UserPreferencesFactory
          .getUserPreferences(
              pageContext.getSession()).getLocale() + "/";

      String url = null;

      curam.interfaces.IEGScriptAdminPkg.
        IEGScriptAdmin_checkForScriptErrors_TH
            iegScriptAdminCheckForErrors
          = new curam.interfaces.IEGScriptAdminPkg.
            IEGScriptAdmin_checkForScriptErrors_TH();

      iegScriptAdminCheckForErrors.setFieldValue(
        iegScriptAdminCheckForErrors.key$scriptID_idx,
            scriptID);
      iegScriptAdminCheckForErrors.setFieldValue(
        iegScriptAdminCheckForErrors.key$scriptType_idx,
            scriptType);
      iegScriptAdminCheckForErrors.setFieldValue(
        iegScriptAdminCheckForErrors.key$scriptVersion_idx,
            scriptVersion);
      iegScriptAdminCheckForErrors.setFieldValue(
        iegScriptAdminCheckForErrors.key$schemaName_idx,
            schemaName);
      //Call the method.
      iegScriptAdminCheckForErrors.callServer();

      String errorsPresentInScript =
         iegScriptAdminCheckForErrors.getFieldValue(
           iegScriptAdminCheckForErrors.result$errorsExist_idx);
      boolean errorsPresent =
          Boolean.valueOf(errorsPresentInScript).
              booleanValue();
```

```
      if (errorsPresent) {

        // If there are errors, redirect to the validation
        // error page.
        String redirectTo = contextWithUserPreferences
          + "System_listValidationErrorsForModalPage.do"
          + "?name=" + name + "&scriptID=" + scriptID
                + "&scriptType=" + scriptType
          + "&scriptVersion=" + scriptVersion
          + "&schemaName=" + schemaName;
        url = redirectTo + "&&" + rh.getSystemParameters();

      } else {

        // Call the run script method and redirect to
        // the IEG player.
        curam.interfaces.IEGScriptAdminPkg.
          IEGScriptAdmin_runScript_TH iegScriptAdminRunScript
            = new curam.interfaces.IEGScriptAdminPkg.
              IEGScriptAdmin_runScript_TH();

        iegScriptAdminRunScript.setFieldValue(
          iegScriptAdminRunScript.key$dtls$scriptID_idx,
              scriptID);
              iegScriptAdminRunScript.setFieldValue(
          iegScriptAdminRunScript.key$dtls$scriptType_idx,
              scriptType);
        iegScriptAdminRunScript.setFieldValue(
          iegScriptAdminRunScript.key$dtls$scriptVersion_idx,
              scriptVersion);
        iegScriptAdminRunScript.setFieldValue(
          iegScriptAdminRunScript.key$schemaName_idx,
              schemaName);
        //Call the method.
        iegScriptAdminRunScript.callServer();

        String executionID =
            iegScriptAdminRunScript.getFieldValue(
          iegScriptAdminRunScript.result$executionID_idx);
        executionID = executionID.replaceAll(",", "");

        url = context + "ieg/Screening.do?"
              + "executionID=" + executionID
              + "&" + rh.getSystemParameters();
      }

      // Redirect to the correct page.
      response.sendRedirect(
          response.encodeRedirectURL(url));
    ]]>
  </JSP_SCRIPTLET>
</PAGE>
```

7.5.2 Scriptausführung in modalem Dialogfenster verlassen

Der Scriptentwickler kann aus zwei Grundlösungen wählen, eine IEG-Scriptausführung in einem modalen Dialogfenster zu beenden oder zu verlassen:

- Direktes Schließen des modalen Dialogfensters sowie Aktualisieren oder Umleiten in die übergeordnete Registerkarte
- Übergehen zu weiteren UIM-Anzeigen im modalen Dialogfenster

7.5.2.1 Modales Dialogfenster nach Scriptbeendigung direkt schließen

Um ein modales Dialogfeld direkt nach der Beendigung oder dem Verlassen ("Beenden", "Speichern" & Austrittsaktionen) einer IEG-Scriptausführung zu schließen, muss der Scriptentwickler ein Auflöse-UIM

als Beenden-Seite bzw. Verlassen-Seite angeben. Dieses Auflöse-UIM muss seinerseits ein angepasstes JSP aufrufen, das die entsprechende JavaScript-Funktion aufruft, um den Dialog zu schließen.

Um zum Beispiel auf das Administrationsfenster `IEG2_listAllIEG2Scripts` umzuleiten, muss das folgende JSP-Scriptlet in Ihre UIM-Datei eingeschlossen werden:

```
<PAGE
   PAGE_ID="IEG2_resolveFinishScript"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="file://Curam/UIMSchema.xsd"
 >
 <JSP_SCRIPTLET>
   <![CDATA[

     curam.omega3.request.RequestHandler
       rh = curam.omega3.request.RequestHandlerFactory
           .getRequestHandler(request);

     String context = request.getContextPath() + "/";
     context += curam.omega3.user.UserPreferencesFactory
         .getUserPreferences(
           pageContext.getSession()).getLocale() + "/";

     String url = "";
     url = context + "IEG2_listAllIEG2ScriptsPage.do";

     String forwardParams =
         request.getParameter("forwardParams");

     if (screenContext != null && screenContext
         .hasContextBits(
         curam.omega3.taglib.ScreenContext.MODAL)) {
       url += "?" +  rh.getSystemParameters();
       String encodeRedirectURL = response.encodeURL(url);
       response.sendRedirect(response.encodeRedirectURL(
           request.getContextPath() +
               "/ieg/CloseAndRedirect.jspx?redirect="
               + encodeRedirectURL));
     } else {
       url += "?" +  rh.getSystemParameters();
       response.sendRedirect(
           response.encodeRedirectURL(url));
     }

     ]]>
 </JSP_SCRIPTLET>
</PAGE>
```

`CloseAndRedirect.jspx` wird als sofort einsatzfähiges Element bereitgestellt, um das modale Dialogfeld zu schließen und zu einem angegebenen UIM (sofern bereitgestellt) im übergeordneten Element umzuleiten.

7.5.2.2 Zu weiteren UIM-Anzeigen im modalen Dialogfenster weitergehen

Um das modale Dialogfenster nach Beendigung der Scriptausführung für die Anzeige weiterer UIM-Anzeigen offenzuhalten, geben Sie die erforderliche UIM-Seite als Beenden-Seite bzw. Verlassen-Seite in der IEG-Scriptdefinition an. Mit dem Laden des UIM haben Sie sich aus IEG herausbewegt. Nun findet die Standard-UIM-Verarbeitung im modalen Dialogfeld Anwendung.

7.6 Anwendungsdaten bereinigen

Das Bereinigen von Anwendungsdaten beinhaltet auch die Entfernung von Daten aus der Datenbanktabelle "IEGEXECUTIONSTATE" und dem DS, wo es erforderlich ist. In diesem Abschnitt werden die manuellen und automatischen Bereinigungstasks aufgeführt, die von Scriptautoren zu berücksichtigen sind, sowie Empfehlungen gegeben, die bei Beachtung zur reibungslosen Bereinigung von Anwendungsdaten beitragen.

Um die Ausführung eines IEG-Scripts zu unterstützen, müssen Informationen zu einzelnen Scriptausführungen von der IEG-Engine verwaltet werden. So muss die IEG-Engine beispielsweise die aktuelle Seite für die Scriptausführung überwachen. Zusätzlich muss die IEG-Engine eine Liste der Seiten verwalten, die dem Benutzer zum Unterstützen der Navigation dargestellt wurden. Die Antworten auf Kontrollfragen werden im DS nicht persistent gespeichert; sie werden ebenfalls von der IEG-Engine überwacht. Alle Informationen zum Unterstützen der Ausführung eines IEG-Scripts werden in der Tabelle "IEGEXECUTIONSTATE" persistent gespeichert. Bei der Erstellung eines neuen "IEGScriptExecution"-Objekts über die API "IEGScriptExecutionFactory" wird ein entsprechender Eintrag in der Tabelle "IEGEXECUTIONSTATE" erstellt. Bei der Tabelle "IEGEXECUTIONSTATE" handelt es sich um eine 'interne' Tabelle, die nur für die Verwendung durch die IEG-Engine gedacht ist und nicht geändert oder erweitert werden sollte. Scriptautoren sollten nichts von den Inhalten dieser Tabelle abhängig machen oder auf ihnen aufbauen, da sie kurzfristigen Änderungen unterliegen können.

IEG kann nicht wissen, ab wann ein Eintrag in der Tabelle "IEGEXECUTIONSTATE" nicht mehr eforderlich ist. Deshalb bleiben die Einträge bestehen, bis sie explizit gelöscht werden. Um ein unnötiges Überladen der Tabelle "IEGEXECUTIONSTATE" zu vermeiden, sollte der Eintrag einer Scriptausführung, wenn sie beendet ist und nicht wieder aufgenommen oder erneut ausgeführt wird, über die Methode "removeScriptExecutionObject" der API "IEGScriptExecutionFactory" entfernt werden.

IEG kann nicht ableiten, welche Daten sich dazu verwenden lassen, eine bestimmte Scriptausführung logisch und eindeutig zu ermitteln, da dies je nach Scriptdefinition unterschiedlich sein kann. Der einzige Weg für IEG, eine Scriptausführung zu ermitteln, ist über die generierte ID, die der Scriptausführung bei ihrer anfänglichen Erstellung zugewiesen wird. Scriptautoren wird dringend empfohlen, einen Mechanismus zum Ermitteln von Scriptausführungen zu implementieren, indem eindeutige Daten mit den Scriptausführungs-IDs verknüpft werden. Es kann eine neue Tabelle erstellt werden, um die Beziehung zwischen den Daten, die die Ausführung identifizieren, und der Ausführungs-ID zu verwalten, so dass Scriptausführungen problemlos fortgesetzt werden können. Sobald sie nicht mehr erforderlich sind, können sie entfernt werden. Durch das Entfernen einer Scriptausführung werden keine der erfassten Daten entfernt, die im DS persistent gespeichert sind.

Ähnlich wie bei der Tabelle "IEGEXECUTIONSTATE" wissen die IEG-Engine und der DS nicht, ab wann die Daten, die während einer Scriptausführung erfasst und im DS persistent gespeichert werden, nicht mehr benötigt werden. Auch hier gilt, dass der DS möglicherweise unnötig mit Entitäten überladen wird, die nicht mehr erforderlich sind. Es ist so gedacht, dass die Entitäten im DS nicht für immer persistent gespeichert sind, sondern dass die erfassten Daten in Anwendungstabellen verschoben und anschließend aus dem DS entfernt werden. Mit dem Löschen einer Entität aus dem DS werden ihre untergeordneten Entitäten ebenfalls gelöscht. Nachdem die während einer Scriptausführung erfassten Daten in Anwendungstabellen verschoben und nicht mehr erforderlich sind, reicht es deshalb aus, die Stammentität für die Ausführung zu löschen.

Mit dem folgenden Beispielcodefragment wird das Löschen der Stammentität demonstriert:

```
final Long applicationID = execution.getRootEntityID();
      final Entity rootEntity = datastore.readEntity(applicationID);
      rootEntity.delete();
```

Abbildung 58. Stammentität löschen

7.7 Ausgeführte Scripts fortsetzen

Es besteht die Möglichkeit, eine Scriptausführung zu stoppen und später fortzusetzen. Zu diesem Zweck muss darauf geachtet werden, dass die Anwendung die Ausführungs-ID in einer angepassten Tabelle speichert und sie mit einer Benutzer-ID verknüpft. Weitere Details finden Sie unter 7.6, „Anwendungsdaten bereinigen”, auf Seite 53.

Vorausgesetzt, dass die Tabelle "IEGEXECUTIONSTATE" nicht bereinigt und die Scriptdefinition nicht geändert worden ist, wird eine Scriptausführung fortgesetzt, indem der Parameter "executionID" an den IEG-Player übergeben wird, so wie es auch geschieht, wenn eine neue Scriptausführung gestartet wird.

Kapitel 8. Erfasste Daten verwalten

8.1 Einführung

Wie zuvor erwähnt, werden die während einer Scriptausführung erfassten Daten persistent im DS gespeichert. In diesem Kapitel wird erläutert, wie die erfassten Daten aus dem DS abgerufen werden. Zudem wird erklärt, wie Daten in den DS eingefügt werden, um während der Ausführung von Scripts für IEG verfügbar zu sein.

8.2 Erfasste Daten abrufen

Der DS besitzt eine öffentlich zugängliche API, die Sie für Ihren Anwendungscode verwenden können. Diese API wird meist zum Abrufen von Informationen aus einem gefüllten Schema verwendet, kann aber auch zum Vorabfüllen eines Schemas verwendet werden. Beispiel: Ein Kunde schickt nach dem Beenden einer Anwendung seine Informationen ab. Nun kann die API dazu verwendet werden, die Daten aus dem Schema zu extrahieren und Tabellen in der relationalen Datenbank zu füllen.

Ein Beispiel zum Vorabfüllen ist der Fall, dass Informationen zum Kunden bereits bekannt sind, bevor er die Anwendung startet. Wenn einige der Informationen erforderlich sind, um durch die Anwendung zu navigieren, kann der DS auch vorab mit diesen Informationen gefüllt werden.

Um Daten aus einem Schema lesen zu können, muss die entsprechende Ausführung des Scripts bekannt sein. Das heißt, Sie rufen die korrekten Anwendungsinformationen für einen Kunden auf. Aus diesem Grund sind die "executionID" und der Schemaname unbedingt notwendig, um Zugriff auf die Daten zu erhalten.

Das folgende Beispielcodefragment zeigt, wie man eine Stammentität anfordert:

```
final IEGRuntime runtimeAPI = new IEGRuntime();
final IEGRootEntityID rootEntityID =
  runtimeAPI.getScriptExecutionRootEntityID(executionID);

Datastore ds = DatastoreFactory.newInstance()
  .openDatastore(kSchemaName);

final Entity rootEntity =
  ds.readEntity(rootEntityID.entityID));
```

Abbildung 59. Stammentität anfordern

Ab hier kann die Stammentität dafür verwendet werden, andere Entitäten unter dieser Stammentität abzurufen.

8.3 Scripts vorab mit erfassten Daten füllen

Es besteht die Möglichkeit, die dem Benutzer angezeigten Werte vorab zu füllen, so dass die Antworten nur noch bestätigt oder geändert werden müssen.

Beispielsweise können Name und Geburtsdatum eines Benutzers auf der Seite "Persönliche Angaben" vorab gefüllt werden, vorausgesetzt, dass der Benutzer sich bereits angemeldet hat und es eine andere Datenbank gibt, in der diese persönlichen Angaben gespeichert sind.

Der DS kann vor dem Start der Scriptausführung wie folgt vorab gefüllt werden:

```
...
Datastore ds = null;

try {
  // open the data store and create the root entity
  ds = DatastoreFactory.newInstance().openDatastore(schemaName);
} catch (NoSuchSchemaException e) {
  throw new AppException(IEG.ID_SCHEMA_NOT_FOUND);
}

final EntityType appType = ds.getEntityType("Application");
final Entity rootElement = ds.newEntity(appType);

ds.addRootEntity(rootElement);

final EntityType personType = ds.getEntityType("Person");
final Entity person = ds.newEntity(personType);

person.setAttribute("firstName", "TestFirstName");
person.setAttribute("lastName", "TestLastName");
person.setAttribute("dateOfBirth", "19700101");
//...

rootElement.addChildEntity(person);
```

Abbildung 60. Fragment eines Codes, mit dem der DS gefüllt wird

Die Stammentität kann anschließend wie folgt zum Erstellen einer neuen Scriptausführung verwendet werden:

```
...

// create the script execution
final IEGRootEntityID rootEntityID = new IEGRootEntityID();
rootEntityID = rootElement.getUniqueID();
final IEGRuntime runtimeAPI = new IEGRuntime();
final IEGScriptExecutionIdentifier executionIdentifier =
  runtimeAPI.createScriptExecutionExistingRootEntity(
    iegScriptID, schemaName, rootEntityID);
```

Abbildung 61. Scriptausführung erstellen

Anschließend kann der IEG-Player wie folgt unter Verwendung dieser neuen Scriptausführung ausgeführt werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<PAGE PAGE_ID="IEGScriptLauncher">
  <JSP_SCRIPTLET>
    <![CDATA[
curam.omega3.request.RequestHandler rh =
  curam.omega3.request.RequestHandlerFactory.getRequestHandler(
    request);

String context = request.getContextPath() + "/";

String url =
  context + "ieg/Screening.do?" + "executionID=" + executionID
    + "&" + rh.getSystemParameters();

// Redirect to the correct page.
response.sendRedirect(response.encodeRedirectURL(url));
    ]]>
  </JSP_SCRIPTLET>
</PAGE>
```

Abbildung 62. IEG-Player starten

Beachten Sie, dass nur der DS vorab gefüllt werden kann, nicht die Kontrollfragen oder andere scriptbezogene Informationen, da sie in der Scriptausführung gespeichert werden und nicht im DS. Das bedeutet, dass es nicht möglich ist, die in der ersten Sektion des Scripts angezeigten Daten vorab zu füllen und bei der zweiten Sektion anzufangen. Es wird die erste Sektion angezeigt, in der der Benutzer die vorab gefüllten Daten bestätigen kann.

Kapitel 9. Ressourcenspeicher verwenden

9.1 Einführung

Der Ressourcenspeicher (RS) ist ein Bereich der Infrastrukturdatenbank, in dem die in einer Live-Anwendung verwendeten Ressourcen gespeichert werden. Es sind beliebige Ressourcentypen möglich, aber von IEG werden am häufigsten Bilder- und Eigenschaftendateiressourcen verwendet.

9.2 Alle Ressourcen auflisten

Für den Zugriff auf die Administrationsfenster für die Ressourcen müssen Sie sich als Benutzer mit Administratorberechtigung anmelden. Nach der Anmeldung sehen Sie in Ihrem Navigationsfenster die IEG-Sektion. Beim Klicken auf diese Sektion erscheint ein Menü mit dem Link "Anwendungsressourcen". Mit dem Klicken auf diesen Link wird eine Liste von Ressourcen mit einem Suchfeld für die Suche anhand von Kategorien angezeigt.

Die Ressourcen sind nach Kategorien zusammengefasst. Vorhandene Ressourcen werden angezeigt, indem man eine Kategorie in den Filterkriterien und anschließend "Suchen" auswählt. Die von IEG verwendeten Ressourcenkategorien sind folgende:

- CSS

  Formatvorlagenschablonen, die modifiziert werden können, um die Darstellung und Funktionsweise des IEG-Players anzupassen.
- Bild

  Bilder, die im IEG-Player und den IEG-Scripts verwendet werden.
- Eigenschaft

  Eigenschaftendateien, die den für die Ländereinstellung bestimmten Text für Scripts und Fragenseiten enthalten.

9.3 Neue Ressource hochladen

Am Anfang der Anzeige, in der alle Ressourcen aufgelistet sind, befindet sich ein Link, mit dessen Hilfe eine neue Ressource hinzugefügt werden kann. Mit dem Klicken auf diesen Link erscheint eine Anzeige, in der die Ressourcendetails eingegeben werden können.

Geben Sie die folgenden Informationen ein:

- Name

  Dies ist ein eindeutiger Name für die Ressource, mit dem sie in einem IEG-Script referenziert werden kann. Je nach Ressourcentyp kann eine Namenskonvention für die Verwendung in einem IEG-Script durchgesetzt werden. In den Abschnitten zu 9.7, „Bilder hinzufügen”, auf Seite 60 und 9.8, „Statischen Text ändern”, auf Seite 60 sind weitere Details enthalten.
- Inhaltstyp

  Beim Hochladen einer Ressource in einem Web-Browser ist ein Inhaltstyp erforderlich, um den Browser anzuweisen, wie mit der Ressource umzugehen ist. Die am häufigsten in einem IEG-Script verwendeten Inhaltstypen sind `image/png` für ein PNG-Bild und `text/plain` für eine Eigenschaftendatei.
- Inhalt

  Die Dateiauswahlfunktion ermöglicht dem Benutzer das Auswählen der hochzuladenden Ressource.

Die folgenden Informationen sind optional:

- Kategorie

Die Kategorie, zu der die neue Ressource hinzugefügt werden soll.

- Inhaltsdisposition

  Für in IEG-Scripts verwendete Ressourcen; kann freigelassen werden.

- Ländereinstellung

  Wenn Sie eine spezielle Version einer Ressource für eine Ländereinstellung festlegen möchten, geben Sie hier den Ländereinstellungscode ein. Beim Suchen des Systems nach einer Ressource verwendet es einen Ausweichmechanismus ähnlich dem von Java. Wenn die aktuelle Ländereinstellung beispielsweise `en_US` ist, versucht das System, die Ressource für die Ländereinstellung `en_US`, dann `en` und schließlich die „Standard"-Ressource zu lokalisieren. Die „Standard"-Ressource wird angegeben, indem das Ländereinstellungsfeld beim Hochladen der Ressource leer gelassen wird.

- Intern

  Hiermit wird angegeben, ob die Ressource nur für die interne Verwendung bestimmt ist und nie in den Web-Browser hochgeladen werden sollte. Für dieses erste IEG-Release kann diese Einstellung ignoriert werden.

- Beschreibung

  Eine Beschreibung der Ressource.

9.4 Vorhandene Ressource entfernen

Um eine vorhandene Ressource zu löschen, wählen Sie in der Ressource den Link "Anzeigen" und auf der Seite mit den angezeigten Ressourcen "Löschen" aus. Damit wird diese Ressource aus dem System entfernt. Mit dem Klicken auf diesen Link erscheint ein Bestätigungsdialog, der zum Bestätigen auffordert, dass diese Ressource aus dem System entfernt werden soll.

9.5 Vorhandene Ressource aktualisieren

Um eine vorhandene Ressource zu aktualisieren, wählen Sie auf der Seite "Anwendungsressourcen" oder der Seite "Ressource anzeigen" den Link "Bearbeiten" aus. Daraufhin können Sie im Feld "Neuer Inhalt" die aktualisierte Ressource in Ihrem Dateisystem suchen.

9.6 Vorhandene Ressource herunterladen

Jeder Eintrag in der Ressourcenliste kann durch Klicken auf den "Download"-Link auf der Seite "Anwendungsressourcen" heruntergeladen werden. Mit diesem Link wird der Browser-Dialog zum Herunterladen von Dateien geöffnet, mit dem der Benutzer die Ressource wahlweise speichern oder direkt öffnen kann.

9.7 Bilder hinzufügen

IEG-Scripts ermöglichen es, Bilder zur Verwendung sowohl in Ihren Sektionen (standardmäßig im Navigationsfenster links auf einer Seite) als auch auf Ihren Seiten (im Bereich der Seitenüberschrift für die jeweilige Seite) anzugeben. Einige Bilder sind bereits im System integriert, wie die verschiedenen Personenbilder, die in Personenregisterkarten u.ä. verwendet werden. Alle diese Bilder müssen im RS abgelegt werden, damit neue Bilder hinzugefügt und vorhandene aktualisiert werden können, ohne dass Ihre Anwendung wiederhergestellt und erneut implementiert werden muss. Zum Hochladen einer Bildressource setzen Sie den „Inhaltstyp" entsprechend dem Bildformat (image/png, image/gif usw.) und lassen Sie das Feld „Inhaltsdisposition" leer.

9.8 Statischen Text ändern

Die IEG-Engine ermöglicht es, den gesamten Text für Ihr Script für die Standard-Ländereinstellung direkt in die Scriptdefinition einzugeben. Der auf dem Bildschirm angezeigte Text wird jedoch nicht von dort gelesen. Vielmehr wird aller aus einem Script referenzierter Text in für Ländereigenschaften bestimmte Eigenschaftendateien innerhalb des RS abgelegt. Für jedes Script gibt es mindestens eine Eigenschaftendatei für das Script selbst und eine Eigenschaftendatei für jede Seite innerhalb des Scripts. Um die Eindeu-

tigkeit dieser Dateien zu gewährleisten, wird die folgende Namenskonvention verwendet (der letzte Teil ist selbstverständlich nur für die seitenspezifischen Eigenschaftendateien anwendbar):

```
scriptID_scriptVersion_scriptType_pageID
```

Wenn zum Hochladen eines neuen Scripts in das System die IEG-Administrationsfenster verwendet werden, wird aller statischer Text, der in ihnen enthalten ist (z.B. alle Beschriftungen, Titel, Beschreibungen usw.), automatisch in die entsprechend benannten Eigenschaftendateien für Ihr Script extrahiert und ohne verknüpfte Ländereinstellung im RS abgelegt (so dass diese als Ausweicheigenschaften agieren können, wenn für die von Ihnen aktivierte Ländereinstellung keine Eigenschaften vorhanden sind). Beliebige Teile des Textes lassen sich einfach ändern, indem man die aktuelle Eigenschaftendatei herunterlädt. Dabei ist die oben beschriebene Namenskonvention zu berücksichtigen, damit die Ressource in der Ressourcenliste lokalisiert werden kann. Nehmen Sie dann die notwendigen Änderungen vor und aktualisieren Sie die Ressource, wie unter 9.5, „Vorhandene Ressource aktualisieren", auf Seite 60 beschrieben. Am Script selbst sind keine Änderungen erforderlich.

Auf gleiche Weise können Versionen für weitere Ländereinstellungen leicht für diese Dateien hinzugefügt werden. Beim nächsten Ausführen des Scripts in dieser Ländereinstellung wird solch eine Version den standardmäßig eingestellten Ländereigenschaften vorgezogen. Setzen Sie zum Hochladen einer Ressource für eine Eigenschaftendatei den „Inhaltstyp" auf `text/plain` und lassen Sie das Feld „Inhaltsdisposition" leer.

9.9 Standarddateicodierung ändern

Beim Herunterladen einer Klartextressource wird die Datei in UTF-8-Codierung erwartet. Wenn für das Hochladen der Datei eine andere Codierung gewünscht ist, kann dies mit dem Feld "Inhaltstyp" angegeben werden, indem der optionale Parameter `charset` gesetzt wird. Beispiel:

```
text/plain; charset=ISO-8859-1
```

Kapitel 10. IBM Rational AppScan zum Scannen von IEG verwenden

10.1 Einführung

In diesem Kapitel werden die Schritte beschrieben, die erforderlich sind, um mithilfe des Tools IBM®Rational AppScan Sicherheitsscans von IEG-Darstellungsanwendungen durchzuführen.

10.2 Vorbereitung

In IEG wird die Kommunikation zwischen dem Player und der Engine mithilfe eines Synchronisationstokens koordiniert. Mit dem Synchronisationstoken wird gewährleistet, dass die vom Browser übergebene Seite mit der von der IEG-Engine erwarteten Seite konsistent ist. So lässt sich leichter erkennen, wann der Benutzer die Navigationsschaltflächen des Browsers verwendet statt der Navigationsschaltflächen im Player selbst. Das Synchronisationstoken wechselt mit jeder Fragenseite, die im IEG-Player angezeigt wird. Das macht es sehr schwierig, IEG-Fragenscripts zu scannen, die im IEG-Player ausgeführt werden.

Aus diesem Grund empfiehlt es sich, die Konfigurationseigenschaft "appscan.mode.enabled" des Scripts vor dem Scannen auf "true" zu setzen. Ist die Eigenschaft auf "true" gesetzt, wird der Wert des vom Player übergebenen Synchronisationstokens nicht durch die Engine überprüft. Für die Durchführung eines Scans ist das Inaktivieren der Überprüfung des Synchronisationstokens annehmbar. In einer Produktionsumgebung sollte sie jedoch immer aktiviert sein.

Um die Menge überflüssiger vom Scan zurückgemeldeter Informationen zu reduzieren, sollte außerdem der Stack-Trace inaktiviert sein. Zum Inaktivieren des Stack-Trace gehen Sie wie folgt vor:

- Gehen Sie zum Ordner "webclient\JavaSource\curam\omega3\"
- Benennen Sie "Initial_ApplicationConfiguration.properties" in "ApplicationConfiguration.properties" um
- Öffnen Sie "ApplicationConfiguration.properties"
- Fügen Sie folgenden Eintrag hinzu: `errorpage.stacktrace.output=false`

10.3 Beziehungsseiten

Beziehungsseiten sind eine besondere Funktion von IEG, mit deren Hilfe Informationen zu den Beziehungen zwischen Personen eines Haushalts zusammengestellt werden können. Im Gegensatz zu den anderen Seiten eines IEG-Scripts sind Beziehungsseiten dynamischer in ihrer Art und enthalten eine variable Anzahl von Feldern. Gegenwärtig variieren die Namen der Felder, die für Beziehungsseiten erstellt werden, je nach ihrer Ausführung. Demzufolge ist es zur Zeit nicht möglich, einen Scan für ein IEG-Fragenscript auszuführen, das eine Beziehungsseite enthält.

10.4 Scankonfiguration

Nachdem AppScan gestartet ist, kann durch Auswählen der Option "Neuen Scan erstellen..." im Willkommensfenster ein neuer Scan erstellt werden.

Wählen Sie aus den vordefinierten Schablonen des folgenden Fensters den regulären Scan aus.

Wählen Sie auf der ersten Seite des Konfigurationsassistenten "Webanwendungsscan" aus und klicken Sie auf "Weiter".

Geben Sie auf der Seite "URL und Server" des Assistenten die Start-URL der Anwendung ein. Die eingegebene URL kann durch Klicken auf das Symbol neben dem Eingabefeld überprüft werden. Daraufhin

wird der AppScan-Browser angezeigt, der versucht, die URL zu öffnen. Bestätigen Sie, dass die URL korrekt und zugänglich ist. (Klicken Sie bei Erscheinen einer Sicherheitswarnung auf "Ja".) Schließen Sie den Browser und klicken Sie im Konfigurationsassistenten auf "Weiter".

Geben Sie die notwendigen Details für das Anmeldemanagement ein. Anwendungen, die mit Eclipse/Tomcat laufen, erfordern keine Anmeldung des Benutzers. Deshalb kann die Option "Keine" ausgewählt werden. Klicken Sie auf "Weiter".

Klicken Sie in der Anzeige "Testrichtlinie" im Fenster "Allgemeine Aufgaben" auf den Link für die "Vollständige Scankonfiguration". Damit wird der Dialog "Scankonfiguration" aufgerufen.

10.5 Testrichtlinie

Stellen Sie sicher, dass im Fenster für die Ansichtsauswahl auf der linken Seite des Konfigurationsdialogs "Testrichtlinie" ausgewählt ist. Die einfachste Methode beim Konfigurieren eines Scans besteht darin, alle Tests zu aktivieren und anschließend die niedrigwertigen Tests zu inaktivieren, die das Ausführen des Scans zeitlich verlängern.

Wählen Sie aus der Dropdown-Liste "Tests sortieren nach" den Eintrag "Aktiviert/Inaktiviert" aus. Aktivieren Sie zunächst das Kontrollkästchen "Teilweise aktiviert" und anschließend das Kontrollkästchen "Inaktiviert". Der einzige angezeigte Eintrag sollte "Aktiviert" sein. Wählen Sie aus der Dropdown-Liste den Eintrag "Schweregrad" aus. Inaktivieren Sie die Kontrollkästchen "Niedrig" und "Information". Zum Scannen von IEG ist es nicht erforderlich, invasive Tests durchzuführen, da solche Tests eher zum Testen der Plattform ausgelegt sind. Wählen Sie aus der Dropdown-Liste den Eintrag "Invasiv" aus. Inaktivieren Sie das Kontrollkästchen "Invasiv".

10.6 Optionen anzeigen

Wählen Sie im Fenster mit der Ansichtsauswahl "Optionen anzeigen" aus. Setzen Sie den "Grenzwert für redundante Pfade" auf 1. Wählen Sie als Anzeigemethode "Flexibilität zuerst" aus.

10.7 Kommunikation und Proxy

Wählen Sie im Fenster mit der Ansichtsauswahl "Kommunikation und Proxy" aus. Setzen Sie die "Anzahl der Threads" auf 1.

10.8 Testoptionen

Wählen Sie im Fenster für die Ansichtsauswahl "Testoptionen" aus. Inaktivieren Sie die Option zum adaptiven Testen anhand des Anwendungsverhaltens.

10.9 Vorgänge in mehreren Schritten

Für bestimmte Parameter erfordert IEG ordnungsgemäß formatierte Daten. So muss auch AppScan die Verwendung der getesteten Anwendung erst "erlernen". Wählen Sie dazu im Fenster für die Ansichtsauswahl die Option für Vorgänge in mehreren Schritten aus. Klicken Sie auf die Schaltfläche zum Erfassen. Daraufhin wird der AppScan-Browser angezeigt, der versucht, die URL zu öffnen, die auf der Seite "URL und Server" des Konfigurationsassistenten angegeben ist. Navigieren Sie nach Bedarf durch die Anwendung, indem Sie die entsprechenden Daten eingeben. AppScan erfasst die eingegebenen Werte und verwendet sie später für jeden Test, den es ausführt. Nachdem dieser Vorgang beendet ist, schließen Sie einfach den Browser. Der Dialog für die Scankonfiguration wird mit der Sequenz aktualisiert, die soeben erfasst wurde. Aktivieren Sie das Kontrollkästchen "Wiedergabe dieser Sequenz aktivieren" und inaktivieren Sie das Kontrollkästchen "Wiedergabeoptimierung zulassen".

Vermerken Sie alle Sequenzschritte, die `Screening.do` enthalten. Diese Sequenzschritte müssen in reguläre Ausdrücke umgewandelt und als Ausnahmepfade zu den AppScan-Optionen für den Pfadausschluss hin-

zugefügt werden. AppScan kann beim Erfassen von Operationen leicht aus der Synchronisierung herausfallen. Deshalb muss dafür gesorgt sein, dass AppScan bei der Ausführung der Tests den falschen Pfad ignoriert und sich an das erfasste Script hält. Dies erreicht man, indem man AppScan anweist, alle Sequenzschritte zu ignorieren, die `screening.do` enthalten, mit Ausnahme derer, die Sie in den regulären Ausdrücken angegeben haben. Vermerken Sie jeden `__u=x`-Wert, der in der Liste von Sequenzschritten gefunden wird.

10.10 Pfade und Dateien ausschließen

Wählen Sie im Fenster mit der Ansichtsauswahl "Pfade und Dateien ausschließen" aus. Klicken Sie zum Hinzufügen von "Pfad ausschließen" auf die entsprechende Schaltfläche. Wählen Sie als "Typ" "Ausschließen" und aus der Dropdown-Liste "Abgleichen" "Regulärer Ausdruck" aus. Geben Sie als Pfad `.*/Curam/ieg/Screening.do.*` ein und klicken Sie auf "OK".

Fügen Sie einen weiteren Ausschlusspfad hinzu. Wählen Sie dazu als "Typ" "Ausnahme" und aus der Dropdown-Liste "Abgleichen" "Regulärer Ausdruck" aus. Geben Sie als Pfad `.*/Curam/ieg/Screening.do?executionID=.\d*` ein und klicken Sie auf "OK".

Es sollte außerdem eine Ausnahme für jeden `__u=x`-Wert hinzugefügt werden, der in der Liste von Sequenzschritten gefunden wird. Wählen Sie wiederum "Regulärer Ausdruck" aus der Dropdown-Liste "Abgleichen" aus und geben Sie für "Pfad" einen Ausdruck in folgendem Format ein: `.*/Curam/ieg/Screening.do?executionID=.*&__u=[in der Zusammenfassungsanzeige angezeigter Wert]`

Klicken Sie auf "OK".

Klicken Sie auf "OK", um zum Konfigurationsassistenten zurückzukehren.

Klicken Sie im Konfigurationsassistenten auf "Weiter".

10.11 Vorgang fertigstellen

An diesem Punkt ist die Konfiguration des Scans vollständig. Wählen Sie die Option zum späteren Starten des Scans, um den konfigurierten Scan zu speichern und nicht zuzulassen, dass AppScan die gesamte Anwendung nach dem Zufallsprinzip durchsucht. Klicken Sie auf "Fertigstellen".

10.12 Scan ausführen

Um den Scan zu starten, wählen Sie im AppScan-Hauptfenster den Menüeintrag "Scannen" und anschließend die Option zum Testen nur mehrschrittiger Operationen. In Abhängigkeit von der zu testenden Anwendung kann ein Scan für die Durchführung einige Tage benötigen. Ist der Scan vollständig ausgeführt, zeigt AppScan die Ergebnisse des Scans in einer zusammenfassenden Anzeige an. Diese Ergebnisse müssen anschließend dahingehend untersucht werden, ob die gemeldeten Probleme echte Risiken darstellen oder falsch-positiv sind.

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM-Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden. Für die in diesem Handbuch beschriebenen Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense Cedex
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden.

Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen. Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar.

Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht. Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation
Dept F6, Bldg 1
294 Route 100
Somers NY 10589-3216
U.S.A.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Bereitstellung des in diesem Dokument beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen.

IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Alle von IBM angegebenen Preise sind empfohlene Richtpreise und können jederzeit ohne weitere Mitteilung geändert werden. Händlerpreise können u. U. von den hier genannten Preisen abweichen.

Diese Veröffentlichung dient nur zu Planungszwecken. Die in dieser Veröffentlichung enthaltenen Informationen können geändert werden, bevor die beschriebenen Produkte verfügbar sind.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufs. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren und können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Musteranwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Musterprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. IBM kann daher die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programm nicht garantieren oder implizieren. Die Musterprogramme werden "WIE BESEHEN", ohne Gewährleistung jeglicher Art bereitgestellt. IBM übernimmt keine Haftung für Schäden, die durch Ihre Verwendung der Musterprogramme entstehen.

Kopien oder Teile der Musterprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (Name Ihres Unternehmens) (Jahr). Teile des vorliegenden Codes wurden aus Musterprogrammen der IBM Corp. abgeleitet.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der International Business Machines Corporation. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Website "Copyright and trademark information" unter http://www.ibm.com/legal/us/en/copytrade.shtml.

Java und alle auf Java basierenden Marken und Logos sind eingetragene Marken von Oracle und/oder Tochterunternehmen.

Andere Namen können Marken der jeweiligen Rechtsinhaber sein. Weitere Firmen-, Produkt- und Servicenamen können Marken oder Servicemarken anderer Unternehmen sein.