AJAX-Rezepte für Web Services mit APEX-Bordmitteln, jQuery und JSON

Andreas Wismann
MT AG
Ratingen

Schlüsselworte:

APEX, Oracle Application Express, AJAX, Web Service, jQuery, JSON

AJAX-Rezepte

Der Schwerpunkt dieses Manuskripts (und auch der Präsentation am 15. November auf der DOAG 2011) liegt im Einsatz von AJAX-Methoden in Oracle Application Express. Dieses Skript legt die theoretischen Grundlagen für das Verständnis der live-Beispiele, die während des DOAG-Vortrags gezeigt werden. Es werden Ansätze präsentiert, wie man in APEX souverän mit Datenquellen umgehen kann, indem man Web Services als Alternative zu Tabellen, Views oder LOVs benutzt. Ein Ziel des Vortrags ist, dem Entwickler AJAX und Web Services „schmackhafter" zu machen – und zwar nicht nur, um entfernte Daten mit fremden Servern auszutauschen, sondern erst recht als architektonisches Gestaltungsmittel innerhalb der eigenen Projektlandschaft!

Immer wieder aufs Neue wird, wenn von Web Services die Rede ist, das Beispiel des Wetterdienstes strapaziert: Da gibt es also irgendwo einen Server, bei dem man, ganz bequem per Web Service, die Wetterdaten aller Großstädte dieser Welt abrufen kann. Abgesehen von der Sinnfrage (zumindest taugt das Szenario als Schulbeispiel!) wird man allein deshalb wohl kaum über die Einführung von Web Services in den eigenen Projekten nachdenken – außer man programmiert hauptberuflich Wettervorhersagen.

Hinzu kommt, dass die meisten Lehrbuch-Beispiele SOAP-basierte Web Services aufgreifen, bei denen man zunächst mit einer recht steilen Lernkurve konfrontiert wird (SOAP-Header, WSDL, Erzeugen und Verarbeiten der XML-Schemata und -dokumenten). Zum Kennenlernen sind jedoch RESTful Web Services wesentlich leichter zu beherrschen, und im Rahmen einer selbstdefinierten Architektur sind sie oft auch effizienter als ihre SOAP-Verwandten. APEX bietet sowohl die Generierung als auch den Konsum von RESTful Web Services mit bordeigenen Mitteln an.

Grundlage für die hier gezeigten Beispiele ist die APEX-Version 4.1[1]. Das Vorgehen ist prinzipiell zur Version 4.0 identisch. Zur besseren Nachvollziehbarkeit wird die Tabelle DEMO_PRODUCT_INFO aus der Anwendung „Product Portal" verwendet – Oracle hat darin eine Art Online-Shop nachempfunden. Diese Sample Application wird von APEX mitgeliefert und kann bei Bedarf im Application Builder nachinstalliert und später wieder gefahrlos entfernt werden, ohne die bereits installierten Anwendungen im Geringsten zu beeinflussen.

[1] für die APEX-Version 4.1.0.00.32 (Stand 09/2011) existiert ein Bugfix für die deklarative Arbeit mit Web Services, siehe `https://forums.oracle.com/forums/thread.jspa?threadID=2276357`. Die APEX-Version 4.0 ist nicht betroffen.

Mit APEX einen Web Service als Datenquelle zur Verfügung stellen

Das Ziel ist zunächst: aus einem APEX-Report einen Web Service generieren.

1. Erstellen Sie eine Seite mit einem „Classic Report“ über die Tabelle DEMO_PRDUCT_INFO (SQL-Statement siehe Screenshot; die Items für die WHERE-Klausel folgen weiter unten).

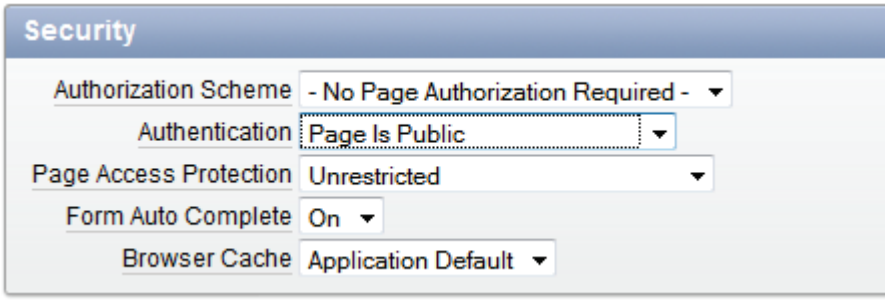
Source

Region Source

```
select PRODUCT_ID
      ,PRODUCT_NAME
      ,PRODUCT_DESCRIPTION
      ,CATEGORY
      ,PRODUCT_AVAIL
      ,LIST_PRICE
   -- ,PRODUCT_IMAGE -- das Bild ist nicht nötig
      ,MIMETYPE
      ,FILENAME
      ,IMAGE_LAST_UPDATE
  from demo_product_info
 where product_id = nvl(:P1000_SELECT_PRODUCT_ID, product_id)
   and category = nvl(:P1000_SELECT_CATEGORY, category)
```

2. Stellen Sie in den Eigenschaften der Seite den Wert für „Authentication“ auf „Page Is Public, damit APEX die Seite im übernächsten Schritt als Web Service-Quelle akzeptiert

.



3. Zum Steuern des Reports programmieren Sie nun zwei Auswahllisten, die den Report jeweils neu laden („Redirect and Set Value“). Auf diese Items verweisen Sie in der WHERE-Klausel des Reports. Ziel der Übung: Diese Bindevariablen übernehmen später die Rolle von Web Service-Steuerungsparametern. Den Auswahllisten weisen Sie passende SELECT-Statements zu, auf deren Rückgabewerte der Report jeweils reagiert.
Für P1000_SELECT_PRODUCT_ID:

List of values definition

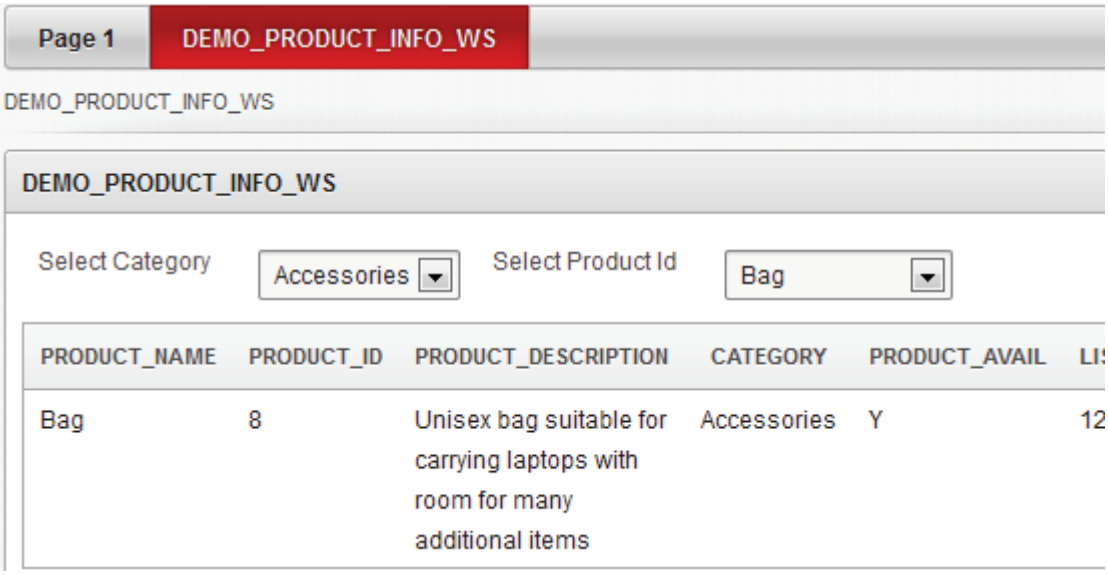
```
select product_name display_value
     , product_id   return_value
  from demo_product_info
 order by product_name
```

für P1000_SELECT_CATEGORY:

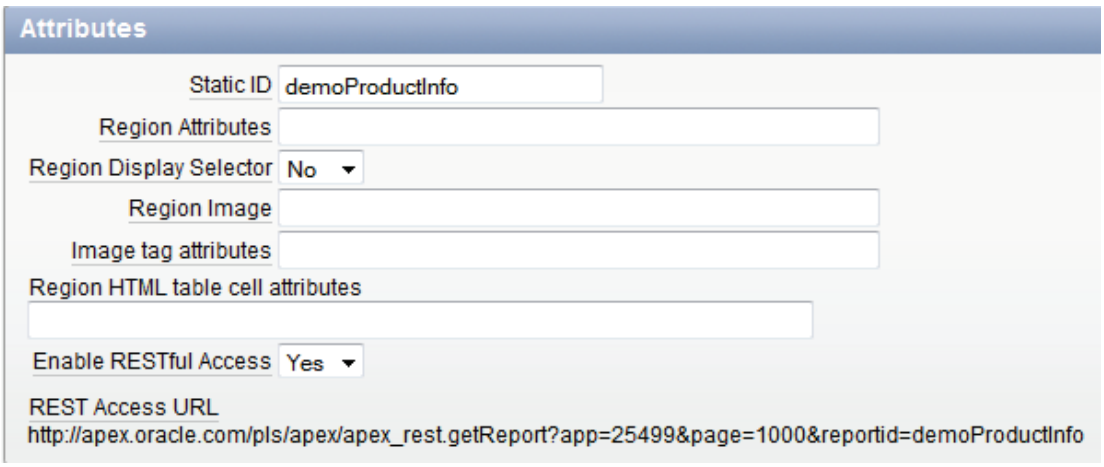
List of values definition

```
select distinct
       category display_value
     , category return_value
  from demo_product_info
 order by 1
```

4. Testen Sie den Report, indem Sie die Seite ausführen. Er sollte je nach Einstellung der Auswahlliste einen bzw. mehrere Datensätze anzeigen:



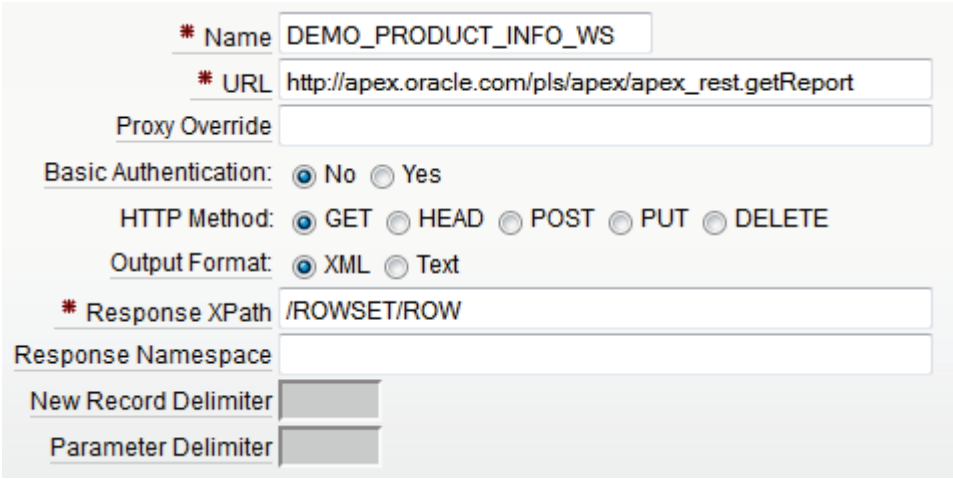
5. Geben Sie in den „Attributes“-Eigenschaften des Reports als Static ID den Wert „demoProductInfo“ ein und ändern die Eigenschaft „Enable RESTful Access“ auf „Yes“. Dadurch stellt APEX den Report (auch) als Web Service-Datenquelle zur Verfügung.



6. Speichern Sie Ihre Einstellungen und kopieren Sie anschließend die URL, die nun unterhalb des Items „Enable RESTful Access“ eingeblendet wird, in die Zwischenablage.

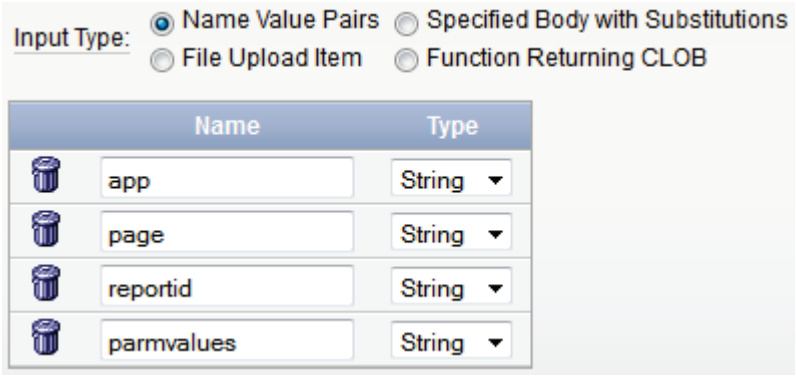
7. Damit man diesen Web Service von einer anderen APEX-Seite aus „anzapfen“ kann, muss er noch bei als Gemeinsame Komponente bei APEX registriert werden. Dazu wechseln Sie zu den „Shared Components > Web Service References“ und folgen dem Assistenten durch Anklicken der Schaltfläche „Create“. Wählen Sie für „What type of Web reference would you like to create?“ den Wert “REST”.

8. Im nächsten Dialog geben Sie Ihrem Service einen Namen und fügen dann die URL aus der Zwischenablage ein, allerdings löschen Sie dort den Parameter-String, also alle Zeichen ab dem „?“. Übernehmen Sie auch den hier gezeigten „Response Xpath“.



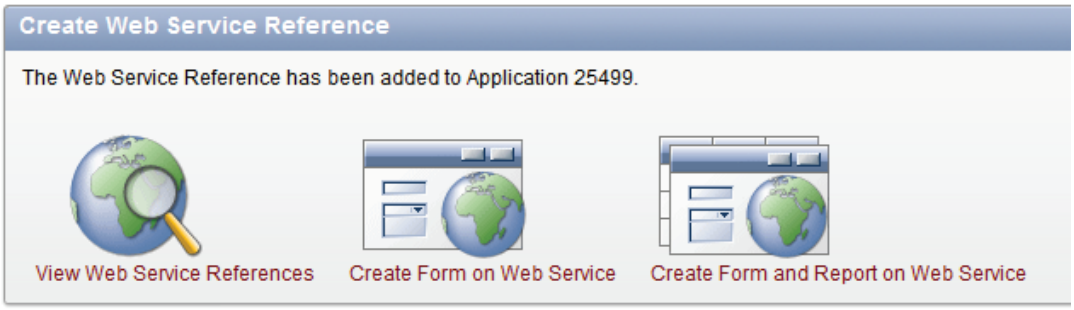
Übrigens, unter der hier verwendeten Domain `http://apex.oracle.com` findet der interessierte Entwickler eine kostenlose APEX-Entwicklungsumgebung, die Oracle freundlicherweise jedermann für Evaluierungszwecke zur Verfügung stellt.

9. Für die REST Input Parameter sind folgende Angaben einzutragen:



Die Namen aller vier Parameter sind vorgegeben.

10. Damit ist der Web Service „scharf“ und wartet darauf, abgefragt zu werden. Nun könnten Sie also mit Hilfe der Formular- und Report-Assistenten Seiten erstellen, die auf diesem Web Service basieren[2].



Die Beschreibung des Web Services

Es gibt einen Mechanismus, der über die von APEX erzeugten Web Services einer bestimmten Anwendung Auskunft gibt. Über die Methode *apex_rest.getServiceDescription* erhält man das Antwortdokument zurück (hier in der XML-Darstellung des Internet Explorers).

`http://apex.oracle.com/pls/apex/apex_rest.getServiceDescription?app=25499`

```
<?xml version="1.0" ?>
- <urn:getServiceDescriptionResponse
    xmlns:urn="urn:oasis:names:tc:wsrp:v1:types">
    <urn:requiresRegistration>false</urn:requiresRegistration>
  - <urn:offeredPortlets>
    - <urn:PortletDescription>
        <urn:portletHandle>demoProductInfo</urn:portletHandle>
      - <urn:markupTypes>
          <urn:mimeType>application/xml</urn:mimeType>
          <urn:mimeType>application/json</urn:mimeType>
        </urn:markupTypes>
        <urn:groupID>1000</urn:groupID>
        <urn:description />
        <urn:title>DEMO_PRODUCT_INFO_WS</urn:title>
      - <urn:keywords>
          <urn:value>P1000_SELECT_PRODUCT_ID</urn:value>
          <urn:value>P1000_SELECT_CATEGORY</urn:value>
        </urn:keywords>
      </urn:PortletDescription>
    </urn:offeredPortlets>
</urn:getServiceDescriptionResponse>
```

[2] eine ausführliche Anleitung zum Umgang mit den Web Service-Assistenten in APEX finden Sie auf der deutschen APEX-Community-Seite unter
`www.oracle.com/webfolder/technetwork/de/community/apex/tipps/HowToRESTfulServices/index.html`

Wie man dieser Reponse entnimmt, bietet der Web Service demoProductInfo die Antwortformate XML und JSON an („mimeType“). Neben der Seitennummer 1000 („groupID“) erfährt man, welche fachlichen Parameter („keywords“) bedient werden müssen, hier P1000_SELECT_PRODUCT_ID und P1000_SELECT_CATEGORY.

Testen des Web Services im Browser

Die Abfrage des Web Service mit XML als Output-Format erfolgt über *apex_rest.getReport*. Die Steuerungsvariablen (hier „8“ und „Accessories“) sind dabei als kommaseparierte Liste an den Parameter *parmvalues* anzuhängen. Geben Sie die entsprechende URL in die Adresszeile des Browsers ein.

```
http://apex.oracle.com/pls/apex/apex_rest.getReport?app=25499&page=1000
&reportid=demoProductInfo&parmvalues=8,Accessories&output=xml
```

Sie erhalten folgendes Ergebnisdokument angezeigt:

```
<?xml version="1.0" ?>
- <ROWSET>
  - <ROW>
      <PRODUCT_ID>8</PRODUCT_ID>
      <PRODUCT_NAME>Bag</PRODUCT_NAME>
      <PRODUCT_DESCRIPTION>Unisex bag suitable for carrying
        laptops with room for many additional
        items</PRODUCT_DESCRIPTION>
      <CATEGORY>Accessories</CATEGORY>
      <PRODUCT_AVAIL>Y</PRODUCT_AVAIL>
      <LIST_PRICE>125</LIST_PRICE>
      <MIMETYPE>image/jpeg</MIMETYPE>
      <FILENAME>bag.jpg</FILENAME>
      <IMAGE_LAST_UPDATE>03.04.2011</IMAGE_LAST_UPDATE>
    </ROW>
  </ROWSET>
```

Dieser Web Service besitzt zwei fachliche Parameter, damit man ihn wahlweise mit der Product_ID oder der Kategorie abfragen kann. Dem aufmerksamen Leser ist nicht entgangen, dass `parmvalues=8,Accessories` eine ungeschickt formulierte Abfrage darstellt, die man in einer WHERE-Klausel normalerweise vermeidet (`... where product_id=8 and category='Accessories'`). Zwecks Übermittlung von NULL an den Web Service muss man experimentieren, denn keine der folgenden Varianten bringt den gewünschten Erfolg:

`&parmvalues=,Accessories`	(Resultset leer)
`&parmvalues=NULL,Accessories`	(Resultset leer)

Stattdessen lässt man entweder den Parameter parmvalues vollständig weg (dann werden beide Variablen implizit auf NULL gesetzt), oder man ersetzt eine bzw. beide ausdrücklich mit dem Wert %20 (das entspricht in URL-Notation einem Leerzeichen).

`&parmvalues=%20,Accessories`	(liefert alle Zubehörartikel)
`&parmvalues=8,%20`	(liefert einen genau bestimmten Artikel)

Wahlweise kann man die Angabe `parmvalues` der besseren Übersicht halber auch ans Ende des URL-Aufrufs stellen (die Abfrage liefert alle Tabellendaten):

```
http://apex.oracle.com/pls/apex/apex_rest.getReport?app=25499&page=1000
&reportid=demoProductInfo&output=xml&parmvalues=%20,%20
```

Web Service im JSON-Gewand

JSON („JavaScript Object Notation“) ist ein schlankes, gut lesbares Daten-Transportformat, das validen JavaScript-Code darstellt – eine Datenstruktur vom Typ Object. Es stellt eine adäquate Alternative zu XML mit gewissen Vor- und Nachteilen dar. Verwendet man den output-Parameter wie folgt:

```
http://apex.oracle.com/pls/apex/apex_rest.getReport?app=25499&page=1000&rep
ortid=demoProductInfo&parmvalues=8,&output=json
```

so liefert nun der APEX Web Service pro Datensatz die folgende JSON-Struktur zurück:

```
{"row":[{"PRODUCT_ID":8,"PRODUCT_NAME":"Bag","PRODUCT_DESCRIPTION":"Unisex
bag suitable for carrying laptops with room for many additional
items","CATEGORY":"Accessories","PRODUCT_AVAIL":"Y","LIST_PRICE":125,"MIMET
YPE":"image\/jpeg","FILENAME":"bag.jpg","IMAGE_LAST_UPDATE":"03.04.2011"}]}
```

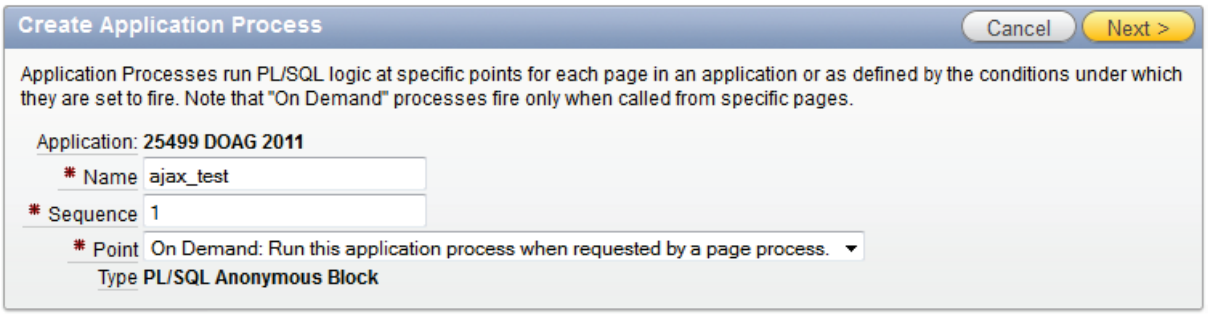
AJAX

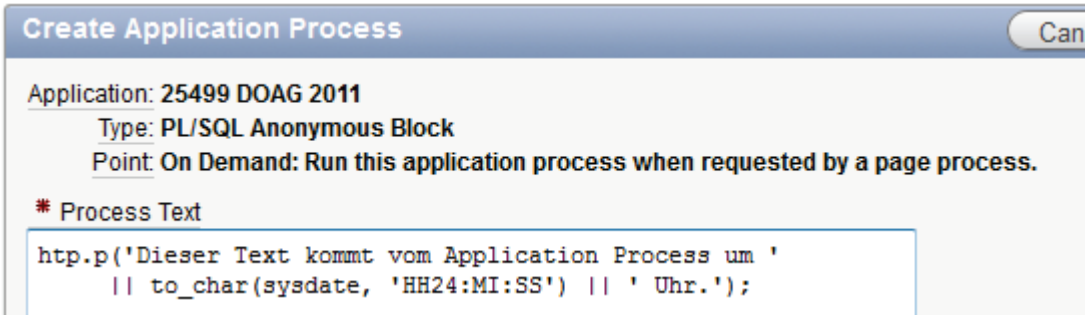
JSON ist ideal geeignet, um Daten zwischen JavaScript-Programmen und einem Webserver hin- und her zu senden. Hier kommen die Entwurfsmuster von AJAX („Asynchronous JavaScript and XML“) ins Spiel. Streng genommen steht das „X“ zwar für XML, jedoch wendet man den Begriff AJAX auch im Zusammenhang mit dem JSON-Format an. Das Prinzip ist jedenfalls gleich: Mittels JavaScript gesteuert, tauscht der Browser Daten mit dem Webserver aus, während die geladene Webseite weiterhin offen zur Bearbeitung durch den Benutzer bleibt. JavaScript-Methoden, die das HTML-Dokument manipulieren können, erlauben dann die (fast) beliebige dynamische Änderung der Inhalte der Webseite, und zwar entweder vollständig oder – typischerweise – partiell.

htmldb_Get und APPLICATION_PROCESS

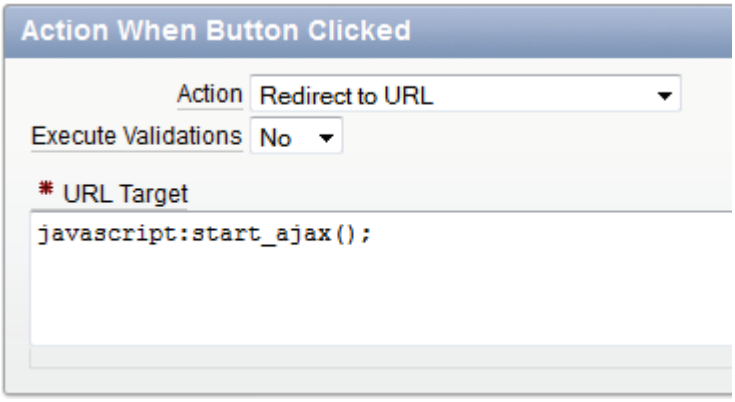
APEX verwendet unter der Haube recht häufig AJAX-Verfahren, beispielsweise zum Refreshen von Interactive Reports. Oracle schlägt vor, für entsprechende eigene AJAX-Skripte den Befehl `htmldb_Get` zu verwenden. Die Mehrzahl der Codebeispiele in der APEX-Dokumentation verfolgt diesen Ansatz. Sehr praktisch ist die Möglichkeit, mit diesem Befehl einen Anwendungsprozess („Application Process“) abzufragen, zu finden unter den Shared Components.

Ein solcher Prozess muss vom Typ „On Demand“ sein. Er besteht aus einem beliebig komplexen PL/SQL-Skript und schickt letztendlich ein Ergebnis mit dem Befehl `htp.p` an den Browser zurück. Im einfachsten Fall ist das „Ergebnis“ eine Zeichenkette (varchar2), es kann aber auch ein XML- oder JSON-Dokument zurückgeschickt werden.





Zum Testen der Prozesskommunikation wird in APEX eine Schaltfläche erzeugt, die eine (hier selbst definierte) JavaScript-Funktion aufruft.



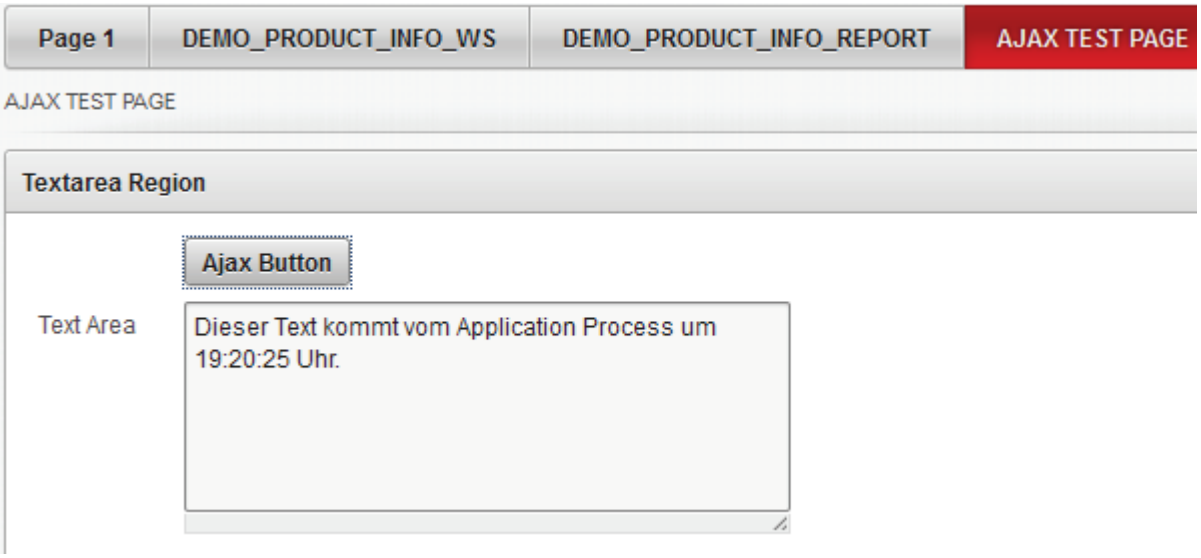
Die aufzurufende Funktion kann der Einfachheit halber zunächst im Seitenkopf untergebracht werden. Dazu gibt es bekanntermaßen seit APEX 4.0 ein Eingabefeld „JavaScript“, wo der Code hinterlegt wird, der beim Rendern im Header der HTML-Seite platziert werden soll:

JavaScript

Function and Global Variable Declaration

```
function start_ajax() {
   var get = new htmldb_Get(
      null,
      $v('pFlowId'),
      'APPLICATION_PROCESS=ajax_test',
      $v('pFlowStepId')
   );
   var ajax_return_value = get.get();
   // warten, bis das Ergebnis vom Server zurückkommt...
   get = null;
   $('#P2001_TEXT_AREA').val(ajax_return_value);
}
```

Der Vorteil dieser Architektur: Es ist kein Submit des gesamten Formulars nötig. Der klassische Bearbeitungsstrang „Seite laden – Seite bearbeiten – Seite abschicken“ kann durchbrochen werden. Im Internet existieren mittlerweile Anwendungen, bei denen man eher das Gefühl hat, mit einer Desktopanwendung zu arbeiten als auf einer Webseite zu sein. Manche davon kommen vollständig ohne jegliches Neuladen der HTML-Seite aus.



Im Hinblick auf gutes Anwendungsdesign mit AJAX-Mitteln ist jedoch das Untypische an htmldb_Get, dass der Befehl die weitere Abarbeitung des Skripts so lange unterbricht, bis die Antwort vom Server zurückkommt (synchrones Verfahren). Derweil ist der Browser also ausschließlich mit Warten beschäftigt und dadurch komplett blockiert – auch für Benutzereingaben. Immerhin funktioniert der Vorgang selbst tadellos und ist bei performanten Abfragen meist auch schnell genug. Dennoch: Bei längerer Antwortzeit des Servers, nennenswerter Netzwerklatenz und größerem Datenvolumen der Übertragung könnte der Benutzer diese Wartezeit mitunter als Fehlverhalten der Anwendung wahrnehmen.

„Echte“ AJAX-Programmierung verlangt, dass asynchron gearbeitet wird, der Benutzer also jederzeit weiterarbeiten kann. Hierzu bieten sich zwei Lösungen an:

1. auf den von Oracle bereitgestellten Befehl htmldb_GetAsync() auszuweichen
2. mit den AJAX-Methoden von jQuery zu arbeiten.

jQuery als treibende AJAX-Kraft

jQuery ist zum „Tabellenführer“ unter den JavaScript-Frameworks aufgestiegen und wird in der Version 1.6.2 (APEX 4.1) komplett vorkonfiguriert für Oracle Application Express ausgeliefert (Version 1.4.2 unter APEX 4.0). Mit jQuery’s AJAX-Methoden lässt sich äußerst komfortabel arbeiten. Der Ablauf stellt sich prinzipiell so dar:

1. zunächst eine Funktion schreiben, die aufgerufen wird, sobald die angeforderten Daten vom Server eintreffen (sogenannte callback-Funktion). Sie verändert beispielsweise den Inhalt eines Items mit Hilfe der zurückerhaltenen Daten.
2. Ein Ereignis festlegen, das die Daten mittels des jQuery.ajax-Befehls vom Server anfordert (etwa, wenn der Benutzer eine Schaltfläche klickt). Dabei wird die callback-Funktion an das success-Ereignis gebunden.

Wie dieser Mechanismus genau zu implementieren ist und welche überzeugenden Resultate im Hinblick auf die „Usability“ und Akzeptanz einer Web-Anwendung damit erzielbar sind, wird im DOAG-Vortrag in Form von live-Demos gezeigt. Der rote Faden der Präsentation ist ein Szenario, in dem die Formulareingaben bereits während des Ausfüllens benutzfreundlich geprüft werden sollen. Dabei wird nicht nur der Datentyp eines Items überwacht (das wäre eine leichte Übung auch ohne AJAX), sondern es können durchaus komplexe Plausibilitätsprüfungen und Geschäftslogiken eingebunden werden. Und ganz nebenbei fallen, wenn man sich bestimmter User-Interface-Plugins bedient, sogar diverse optische „Leckerbissen“ als Nebenprodukte an.

So mancher Entscheider gewinnt deshalb den charmanten Eindruck:

„AJAX ist, wenn es gut aussieht.“

Kontaktadresse:

Andreas Wismann
MT AG
Balcke-Dürr-Allee 9
D-40882 Ratingen

Telefon: +49 (0) 2102 30961-0
Fax: +49 (0) 2102 30961-50
E-Mail andreas.wismann@mt-ag.com
Internet: www.mt-ag.com/apex