Mehr Dynamik in Apex mit Javascript und JQuery

Alexander Scholz
its-people
Frankfurt am Main

Schlüsselworte:

Javascript und JQuery in Apex einbinden, Elemente dynamisch anzeigen, Selectlisten aktualisieren, modalen Dialog öffnen, JQuery Plugin benutzen, Dynamic Action aufrufen, Javascript in interaktiven Reports verwenden

Einleitung

Durch die Einbindung von Javascript in Apex können einzelne Seiten dynamischer dargestellt werden. Werte für Select Listen können zum Beispiel in Abhängigkeit von anderen Elementen auf der Seite aktualisiert werden, oder Elemente können ohne vorheriges Submit auf der Seite abhängig von der jeweiligen Benutzeraktion angezeigt oder versteckt werden. Mit der Verwendung von JQuery und JQuery-UI können neue Elemente wie zum Beispiel ein modales Dialogfenster oder ein Rating Element auf APEX Seiten eingebunden werden.

In einzelnen Beispielen werden die Möglichkeiten des Einsatzes von Javascript und JQuery in Apex demonstriert. Dabei wird zunächst beschrieben, an welchen Stellen Javascript in Apex eingebunden werden kann und welche Vor- und Nachteile es jeweils abzuwägen gilt. Über diese Grundlagen hinaus wird dargestellt, was beim Einsatz von JQuery und JQuery-UI besonderes zu beachten ist, um einzelne Elemente in Apex einzubinden. Insbesondere bei JQuery-UI kann es passieren, dass einige Elemente nicht oder nicht vollständig angezeigt werden.

Mit der Einführung von Apex 4 hat sich sehr viel im Bereich dynamischer Seitengestaltung geändert. Es wird aufgezeigt, wie Javascript mit Dynamic Actions kombiniert werden kann. Außerdem wird beschrieben was in Apex 4 standardmäßig eingebaut ist und wo noch manuelle Anpassungen notwendig sind, um noch mehr Dynamik auf den Seiten zu erhalten. So wird z.B. erklärt, warum das Seitentemplate unter Umständen verändert werden muss, um Javascript zu nutzen, oder wie Javascript in interaktiven Reports verwendet werden kann.

Einbinden von Javascript in Apex

Um eigene Javascript Bibliotheken in Apex nutzen zu können, müssen diese zunächst in Apex eingebunden werden. Die erste Möglichkeit hierzu ist das Kopieren der Javascript Datei in das Images Verzeichnis des HTTP Servers. Eine weitere Möglichkeit ist das hochladen der Datei in Apex als Statische Datei. Anstatt das Javascript als Datei einzubinden, kann auch der Quellcode direkt in einer Apex Seite eingetragen werden. Hierzu gibt es die Möglichkeit den Quellcode in den Seitenheader zu schreiben, oder eine HTML Region zu erstellen und dort den Quellcode einzutragen.

Kopieren der Javascript Datei in das Image Verzeichnis

Kopieren Sie ihre Javascript Datei in das Images Verzeichnis des HTTP Servers und referenzieren Sie sie auf der jeweiligen Apex Seite durch einen Eintrag im Seitenheader. Der Vorteil hierbei ist, dass

die Javascript Datei schneller geladen wird. Sie liegt direkt im HTTP Server vor und muss beim Seitenaufbau nicht erst über die Apex geladen werden. Der Quellcode des Javascripts kann vom Benutzer nicht eingesehen werden. Der Nachteil hierbei ist, dass der Entwickler Zugriff auf den Webserver beziehungsweise auf das Image Verzeichnis benötigt. Jede Änderung muss immer auf den Webserver kopiert werden, was die Entwicklung evtl. behindert. Außerdem ist darauf zu achten, dass bei der Übertragung von der Entwicklungsumgebung in die Produktionsumgebung auch alle Javascript Dateien kopiert werden, wenn es sich um unterschiedliche Webserver handelt.

Abb. 1 Einbinden einer Javascript Datei vom Webserver

Hochladen der Javascript Datei als statische Datei

Eine Javascript Datei kann auch als statische Datei bei Shared Components eingebunden werden. Auch hierbei ist der Quellcode für den Benutzer später nicht mehr zu sehen. Die Datei wird aber in der Datenbank gespeichert und muss beim Seitenaufbau erst geladen werden. Dies verlangsamt die Ladezeit. Da die Datei aber in der Apex ist, kann sie durch Importieren oder Exportieren leicht von einer Umgebung zu einer anderen transportiert werden und der Entwickler benötigt keine besonderen Rechte auf dem Webserver. Bei jeder Änderung muss die Datei erneut vom Entwickler hochgeladen werden.

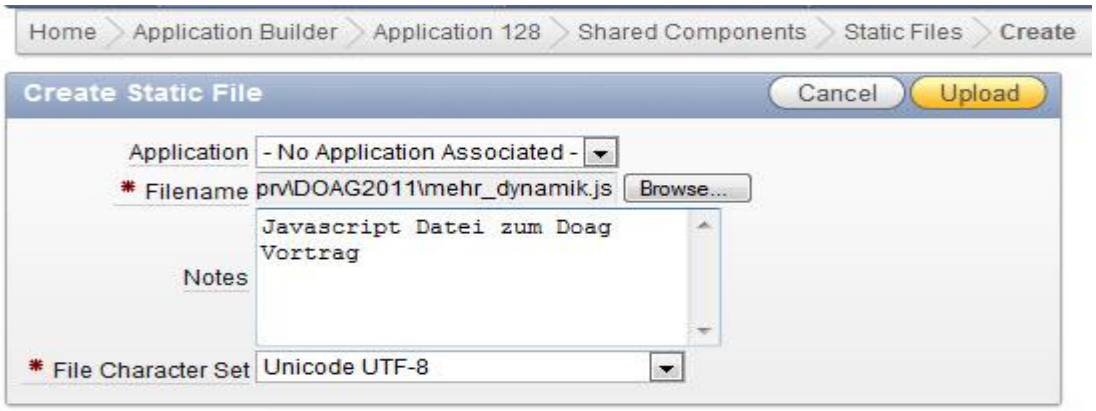

Abb. 2 Hochladen einer Datei als Static File

Je nach dem ob die Datei einer Applikation zugeordnet wird oder nicht, kann sie über die Substituierung #APP_IMAGES# oder #WORKSPACE_IMAGES# referenziert werden.

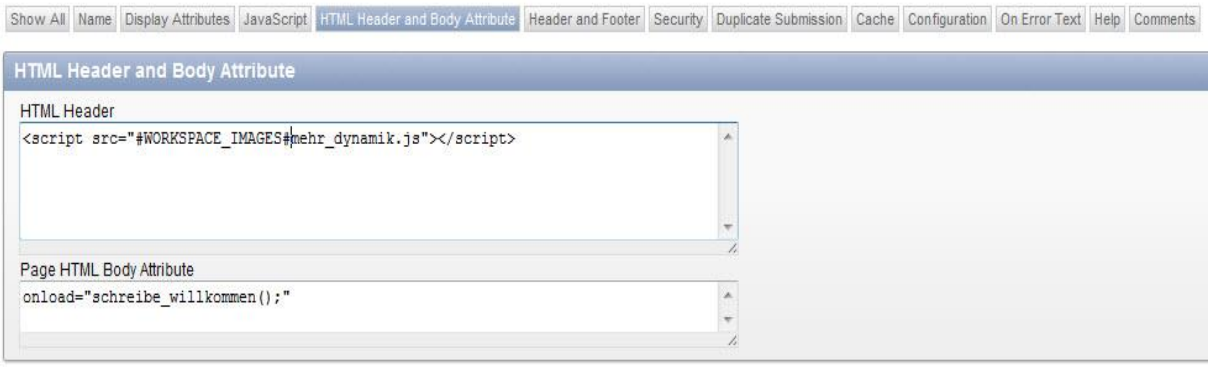

Abb. 3 Referenzierung des Javascripts über #WORKSPACE_IMAGES#

Eintragen des Quellcodes direkt auf einer Apex Seite

Anstatt eine Javascript Datei zu erstellen, kann der Quellcode auch direkt auf der Seite eingetragen werden. Dies erfolgt entweder im Abschnitt HTML Header oder der Entwickler erstellt eine Region vom Typ HTML und fügt darin den Code ein. Der Vorteil ist, dass der Code jederzeit bearbeitet werden kann, ohne dass ein kopieren oder hochladen der Javascript Datei erforderlich ist. Eine Portierung in eine andere Umgebung ist problemlos möglich. Ein Benutzer kann den Code allerdings über den Browser einsehen. Außerdem gibt es Beschränkungen hinsichtlich der Länge des Quellcodes. Beim Eintrag im HTML Header sind nur 4000 Zeichen erlaubt. Beim Eintrag in einer Region sind 32000 Zeichen erlaubt.

Abb. 4 Javascript Quelltext im HTML Header

Anzeigen und Verstecken von Elementen
Dieses Beispiel zeigt, wie eine Region mit Hilfe eines onmouseover Events angezeigt oder versteckt werden kann. Für diese Region muss eine statische ID definiert werden, um sie über Javascript anzusprechen. Die Region wird mittels einer Style Formatierung zunächst versteckt.

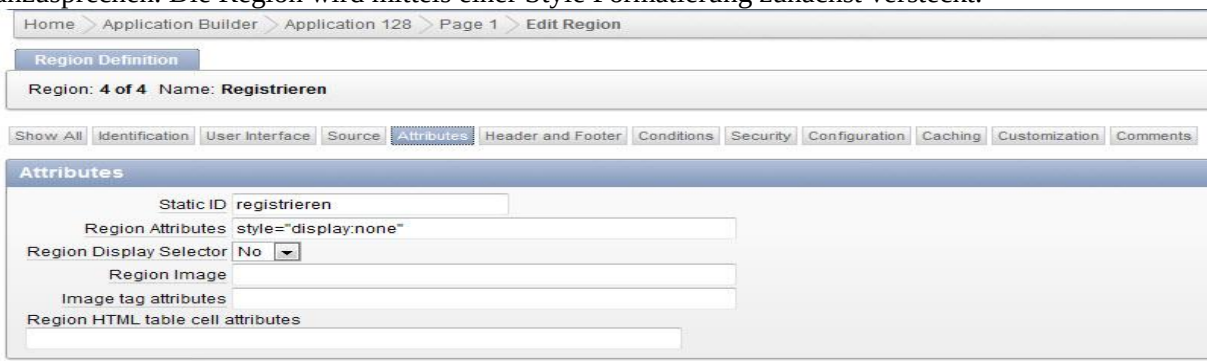

Abb. 5 versteckte Region mit statischer ID

Ein Item in einer anderen Region bekommt ein onmouseover Attribut mit der zugehörigen Javascript Funktion zugewiesen.

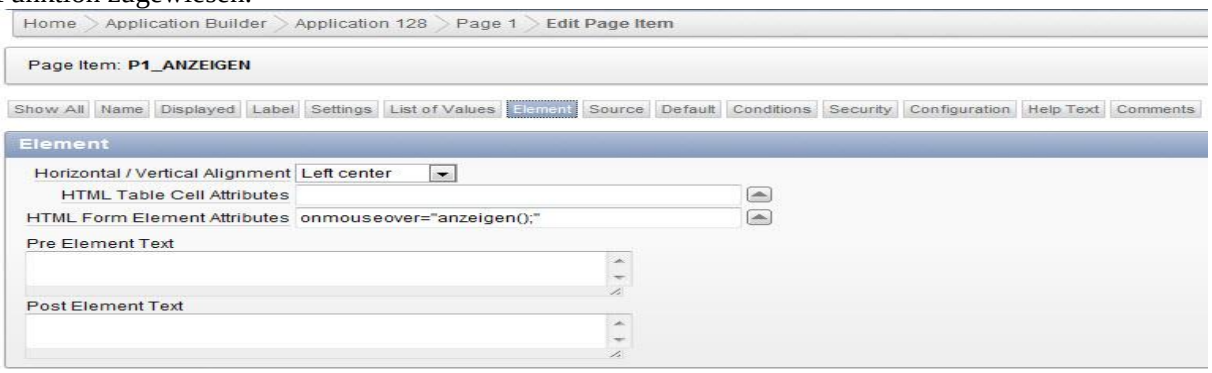

Abb. 6 Item mit onmouseover Attribut

Ein onmouseover Event führt dann folgendes Javascript aus und zeigt damit die versteckte Region an:

```
function anzeigen(){
      document.getElementById("registrieren").style.display="inline";
}
```

Aktualisieren von Select Listen
Manchmal ist der Wert einer Selectliste abhängig von einer anderen Selectliste. Zum Beispiel ist der Ansprechpartner einer Firma abhängig von der ausgewählten Firma.
Für dieses dynamische aktualisieren von Selectlisten in Abhängigkeit eines vorher ausgewählten Wertes musste in Apex 3 ein Javascript Aufruf erfolgen, der mittels Ajax die aktuellen Werte aus der Datenbank ausliest. Diese Werte werden dann als neue Elemente der Selectliste ausgegeben.
Seit Apex 4 ist das nicht mehr erforderlich. Es kann jetzt bei der Definition einer Select Liste angegeben werden, welche abhängigen Elemente zu dieser Auswahl gehören. Bei einer Änderung des Wertes eines abhängigen Elementes werden die Auswahlfelder dynamisch angepasst. Dazu muss lediglich der Item Name des abhängigen Elementes in das Feld *Cascading LOV Parent Item*

eingetragen werden. Die Aktualisierung der Selectliste erfolgt immer dann, wenn sich der Wert dieses Items ändert.

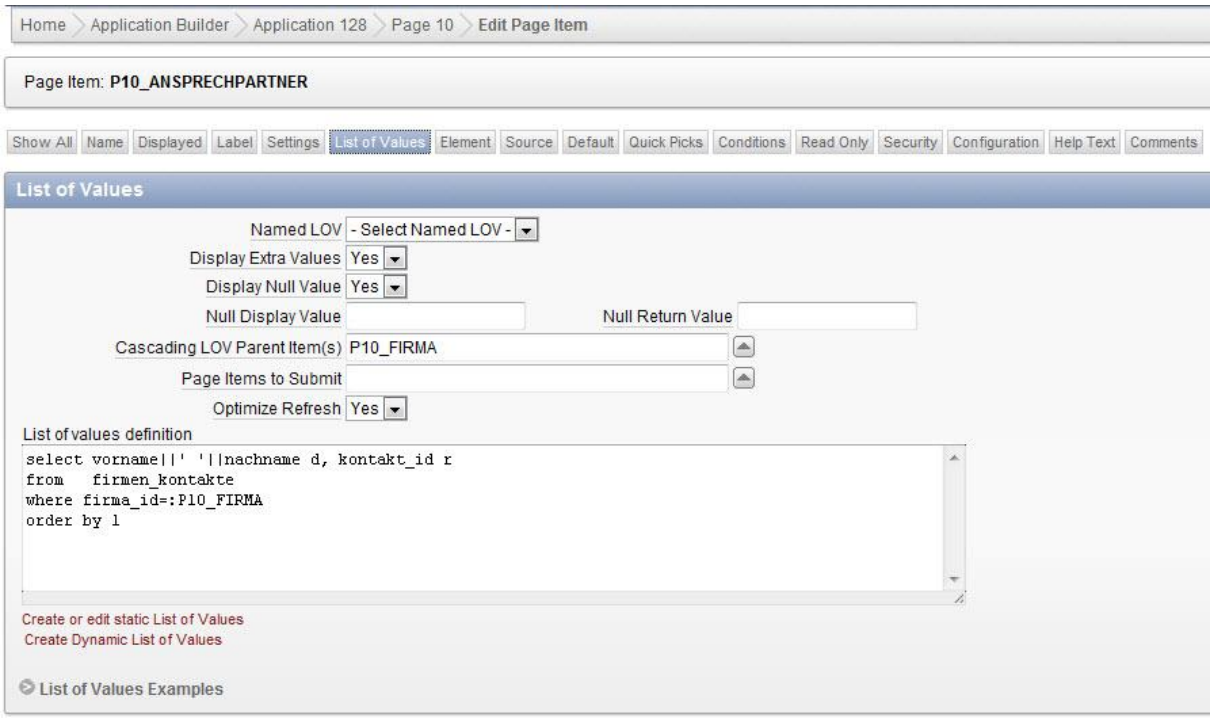

Abb. 7 Select Liste mit Parent Item

Einblenden eines modalen Dialogfensters

Modale Dialogfenster gehören zum Standardumfang von JQuery. Ab Apex 4 ist JQuery in Apex integriert und muss nicht mehr zusätzlich eingebunden werden. Um ein modales Fenster zu erzeugen, erstellen Sie in einer HTML Region ein leeres DIV Tag. Über das ID Attribut können Sie dieses DIV Element später über Javascript ansprechen und es als modalen Dialog definieren.

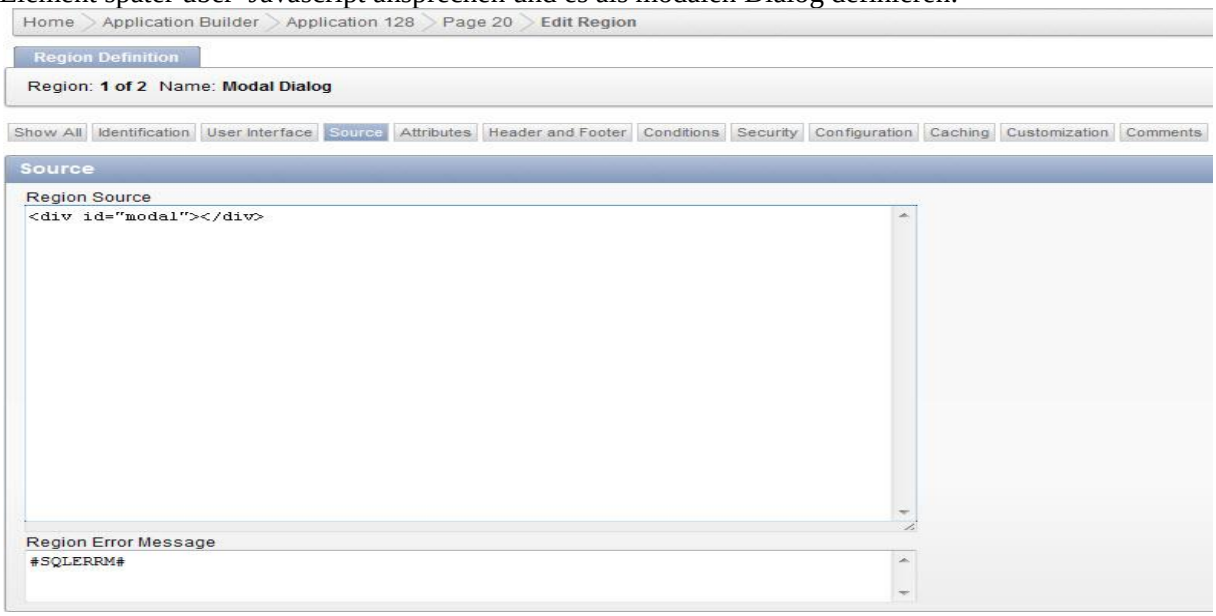

Abb. 8 DIV Element in einer HTML Region

Sobald dem Element die JQuery Eigenschaft *Dialog* mit dem Attribut *modal* zugeordnet wird, öffnet sich ein modales Fenster. Dies erreichen Sie am besten, in dem Sie einen Button erzeugen, der bei *onclick* eine Javascript Funktion aufruft.

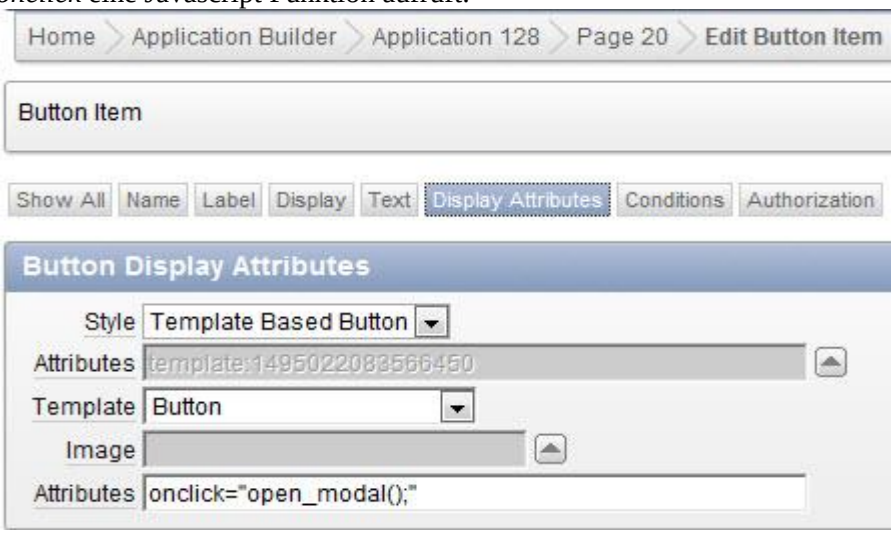

Abb. 9 Aufruf einer Javasrcipt Funktion bei onclick eines Buttons

Die meisten Apex Templates lassen es in Apex 4 nicht zu, die Button Attribute zu setzen. Damit das *onclick* Attribut aufgerufen werden kann, muss daher noch das Button Template um #BUTTON_ATTRIBUTES# ergänzt werden.

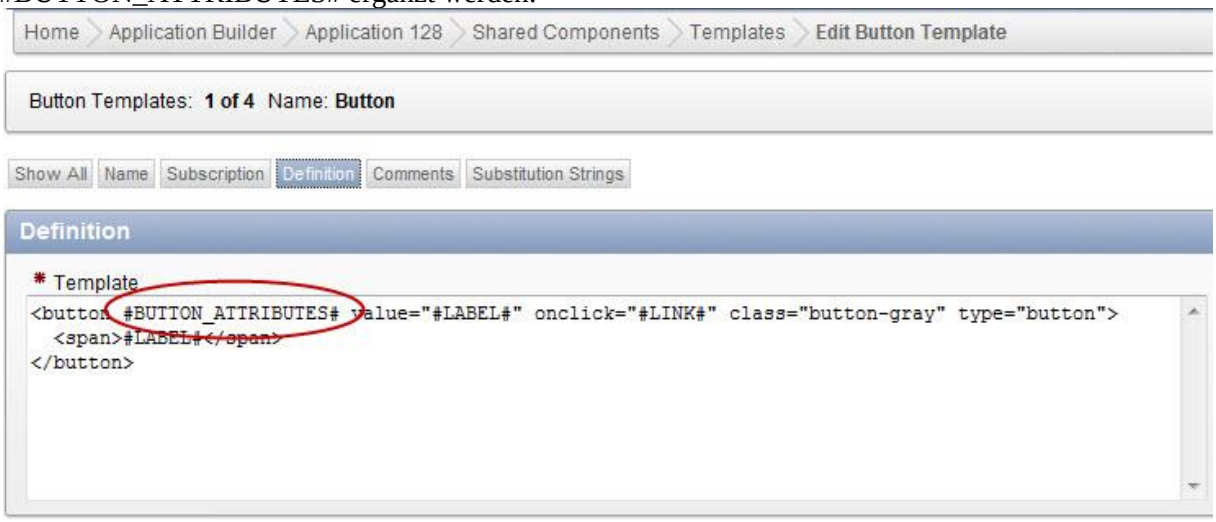

Abb. 10 Button Template um #BUTTON_ATTRIBUTES# ergänzen

Das mit onclick aufgerufene Javascript sieht folgendermaßen aus:

```
function open_modal(){
$("#modal").html('<div id="in_modal">Dies ist ein modales
Dialogfenster</div>');
$("#modal").dialog({modal:true});
}
```

Mit `$("#modal")` wird auf das oben beschriebene DIV Element zugegriffen. Über den Aufruf .html wird der Inhalt des modalen Fensters definiert. Der Aufruf .dialog erzeugt das modale Fenster.

Darstellung eines Radio Elementes als Rating Element
Mit Hilfe eines Plugins für JQuery lässt sich ein Radio- oder Selectitem als Rating Star anzeigen. Dazu müssen Sie zunächst das Plugin einbinden. Am einfachsten geht es, wenn das Plugin auf den Webserver kopiert werden kann. Ist das nicht möglich, muss der Style des Plugins für Apex angepasst werden. Nachfolgend wird beschrieben, wie das Plugin in Apex eingebunden werden kann, ohne die Dateien auf den Webserver zu kopieren.
Laden Sie zunächst die notwendigen Bild und Javascript Dateien unter Shared Components hoch. Dann erstellen Sie eine HTML Region und kopierem die Style Definitionen in den Regioninhalt. Jetzt müssen Sie den Verweis auf die Bilddatei in der Styledefinition auf #APP_IMAGES# ändern und nach der Styledefinition das Javascript des Plugins einbinden. Damit ist das Plugin in Apex eingebunden und kann verwendet werden.

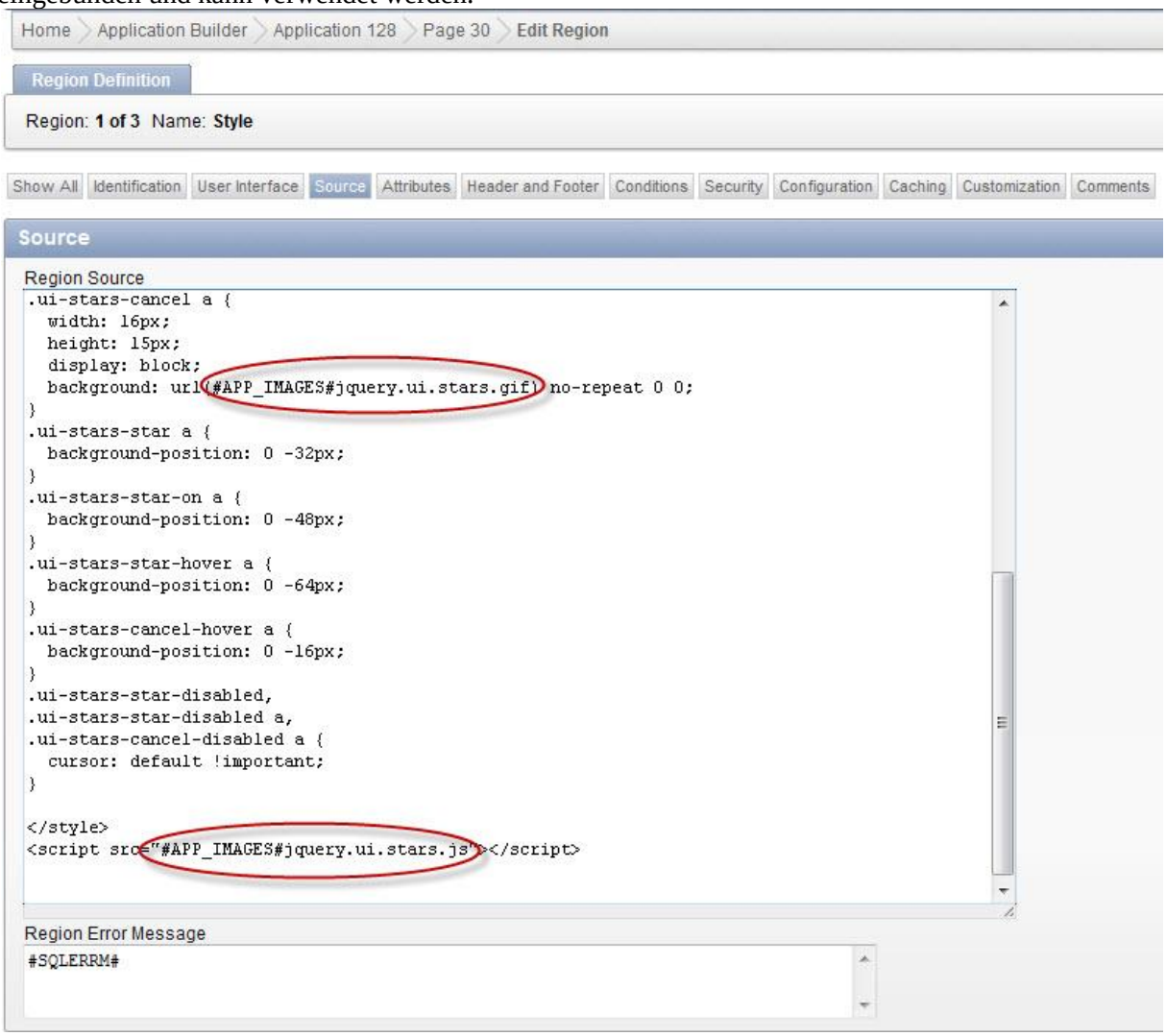

Abb. 11 Verweise auf Bilddatei durch #APP_IMAGES# erstellen und Javascript einbinden

Erstellen Sie ein Radioitem und betten sie es durch *Pre* und *Post Element Text* in ein DIV Element ein. Setzen Sie das ID Attribut auf stars_wrapper, damit dieses DIV vom Plugin verwendet werden kann.

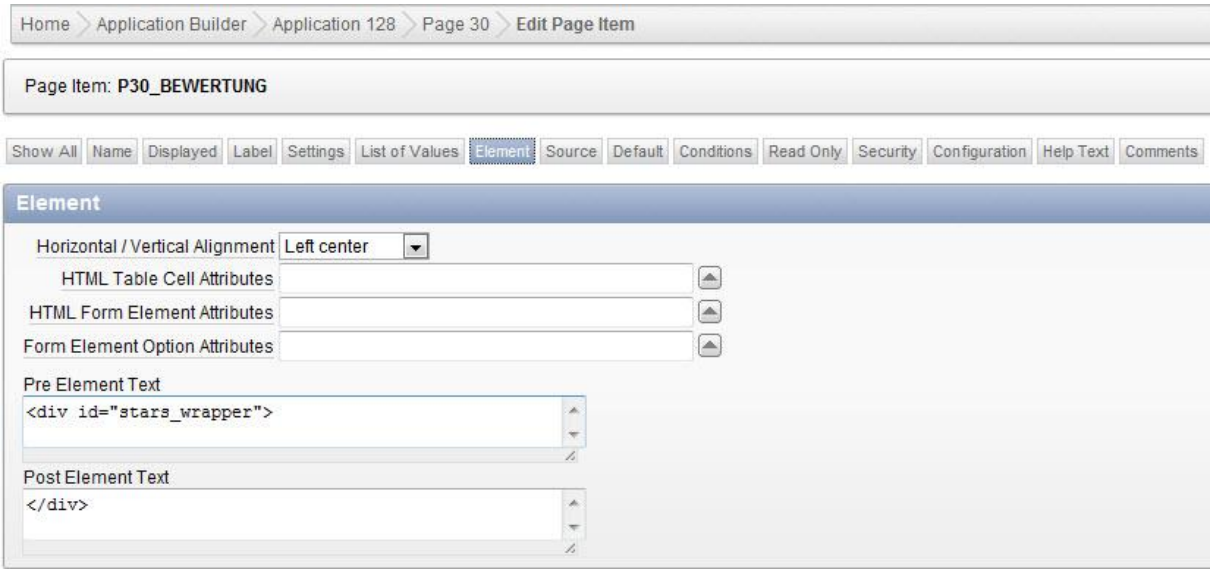

Abb. 12 Radioitem mit Pre und Post Element Text

Damit das Radioitem als Starrating angezeigt werden kann, muss im letzten Schritt dem DIV Element die Eigenschaft *.stars()* beim Laden der Seite zugewiesen werden. Das erfolgt über folgenden Javascript Aufruf:

```
<script>
$(document).ready(function(){
    $("#stars_wrapper").stars();
})
</script>
```

Javascript und Dynamic Actions

Für die dynamische Aktualisierung eines Reports basierend auf einer Änderung einer Selectliste werden Dynamic Actions benötigt.

Erstellen Sie zunächst die Selectliste mit den Auswahlwerten und definieren sie ein onchange Attribut, welches eine Javascript Funktion aufruft.

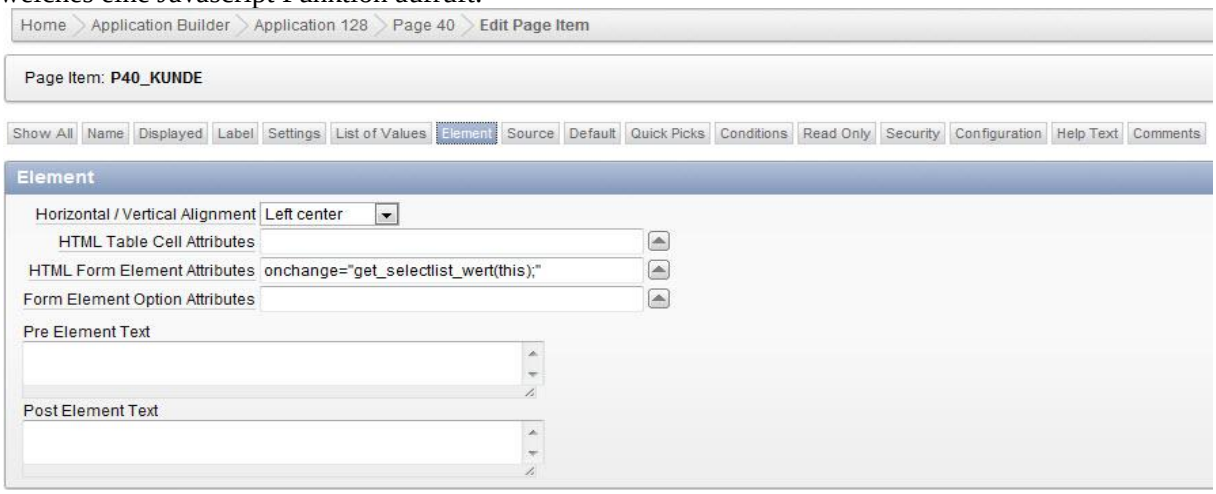

Abb. 13 Selectliste mit onchange Attribut

Um die Aktualisierung des Reportes auszuführen, benötigen sie ein verstecktes Item, in das Mittels Javascript der aktuell ausgewählte Wert der Selectliste eingetragen wird und dann ein change Event für dieses Item aufgerufen wird. Dieses Javascript hat folgenden Aufbau:

```
function get_selectlist_wert(elem){
var company_id=elem.options[elem.selectedIndex].value;
alert("ausgewählte Firmennummer: "+company_id);
$("#P40_FIRMEN_NR").val(company_id);
$("#P40_FIRMEN_NR").change();
}
```

Erstellen sie nun die Dynamic Action, die auf das Change Event des versteckten Items reagiert.

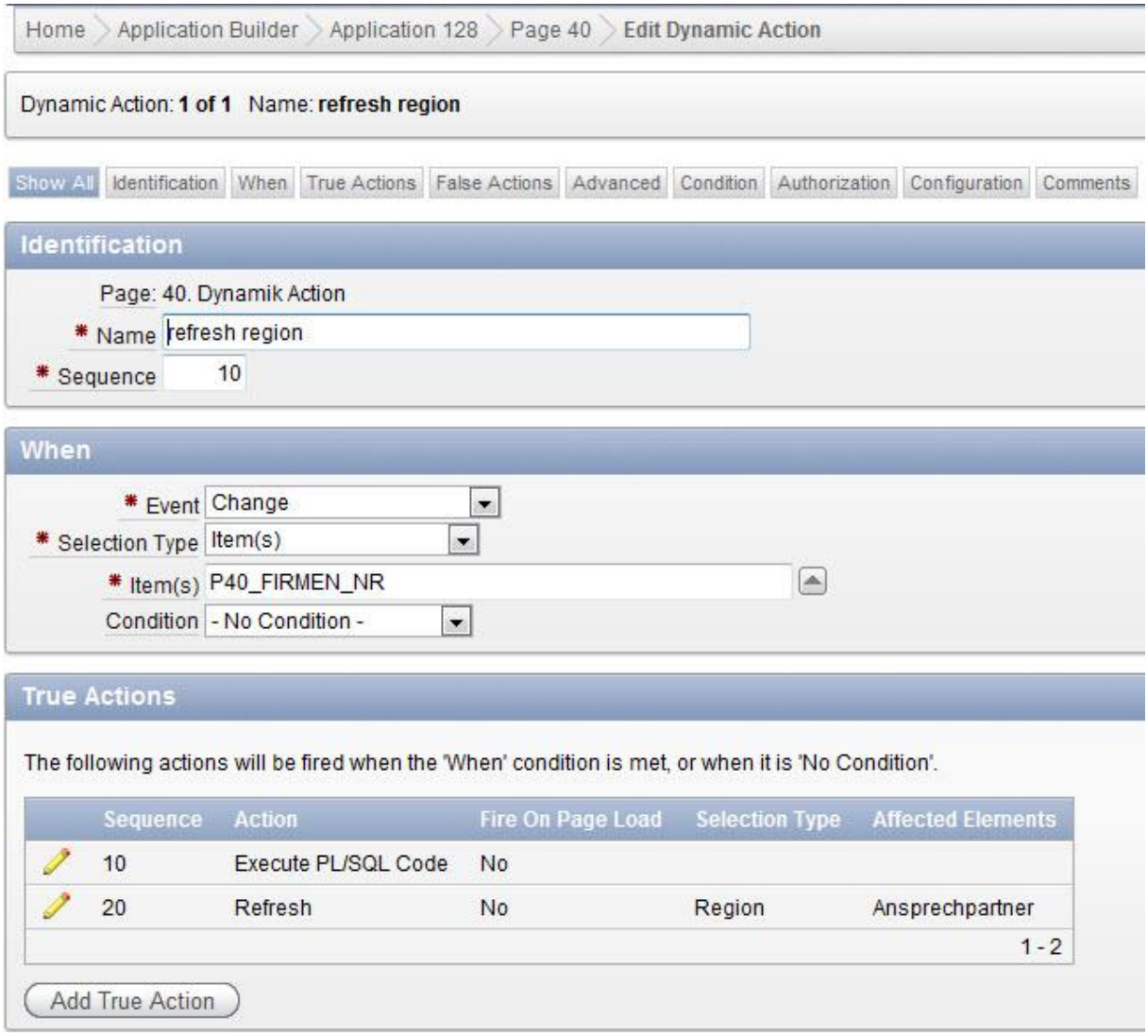

Abb. 14 Dynamik Action mit PL/SQL und Refresh

Die Dynamik Action besteht aus zwei Teilen. Zuerst muss der ausgewählte Wert der Session verfügbar gemacht werden, danach erfolgt ein Refresh des Reports. Der ausgewählte Wert wird mittels eines PL/SQL Blockes an die Session übergeben.

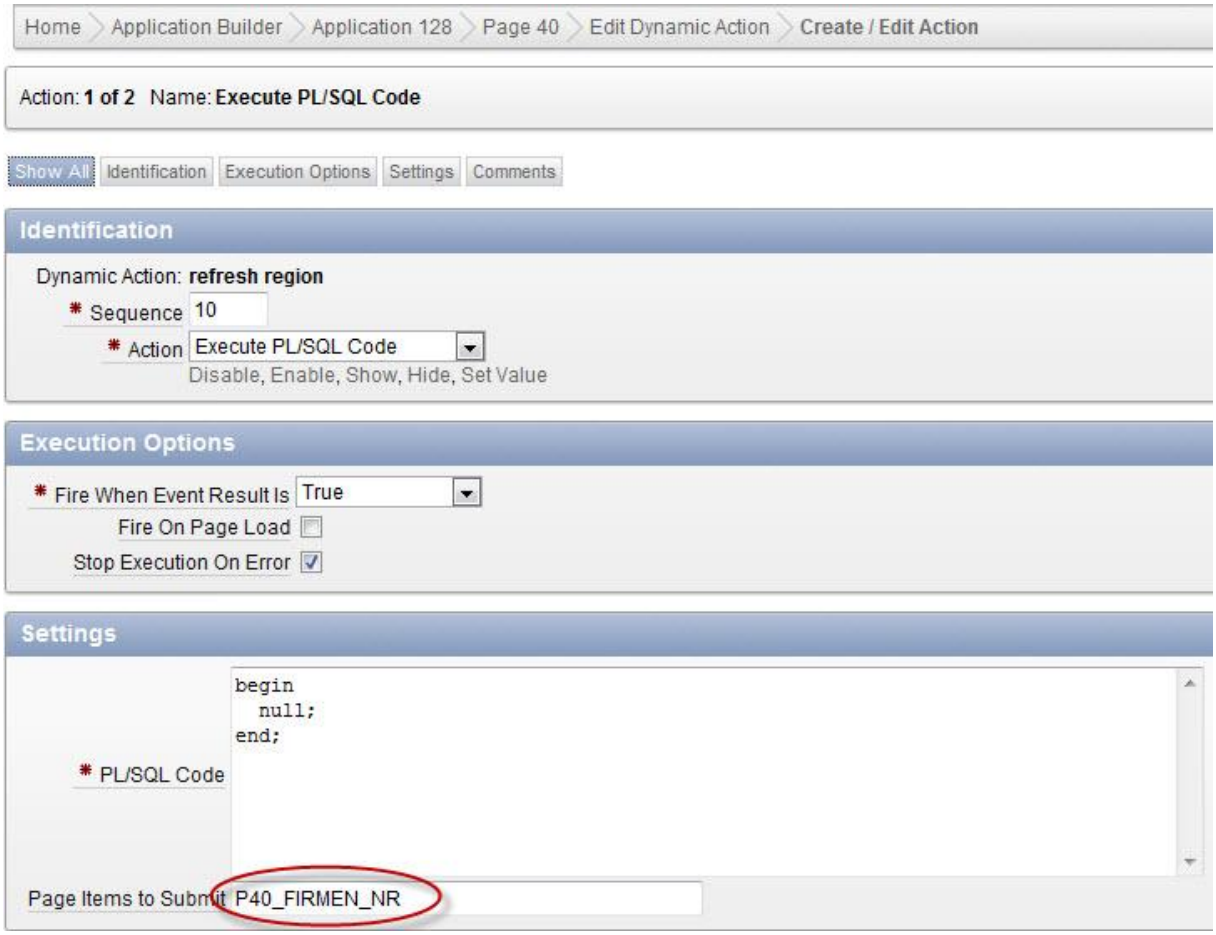

Abb. 15 PL/SQL Block mit Page Item to Submit

Der PL/SQL Block führt gar nichts aus. Er ist aber für die korrekte Syntax erforderlich. Lediglich der Itemname des versteckten Items wird in das Feld Page Items to Submit eingetragen. Dies bewirkt, dass der Wert des Items nun auch in der Session verfügbar ist und der Report diesen Wert verwenden kann.

Javascript in interaktiven Reports

Um zu demonstrieren wie Javascript in einen Interaktiven Report eingebunden werden kann, soll eine Spalte an einen Report hinzugefügt werden, die auf ein onmouseover Event reagiert und weitere Details zu einem Datensatz anzeigt.
Die SQL Abfrage für den Report sieht folgendermaßen aus:

```
SELECT
vorname,
nachname,
'<div id="f01_'||to_char(ansprechpartner_id)||'"
onmouseover="zeige_details(this)">mehr</div>' Details
FROM firmen_kontakte
```

Die Abfrage wird um eine Spalte ergänzt, die als String ein HTML DIV Tag erzeugt. Diesem DIV Tag wird als ID der Primärschlüsselwert zugeordnet. Außerdem wird ein onmouseover Attribut eingefügt, dass die Javascript Funktion *zeige_details* aufruft. Damit diese Spalte nicht als String sondern als

HTML angezeigt werden kann, muss bei den Spalten Attributen unter Display Type für diese Spalte der Wert *Standard Report Column* eingestellt werden. Außerdem sollte es dem Benutzer nicht erlaubt werden auf dieser Spalte Aktionen wie zum Beispiel Filtern oder Sortieren durchzuführen.
Da sich die ID des DIV Tags für jeden Datensatz unterscheidet, liefert eine Filterung im interaktiven Report das falsche Ergebnis, da nicht der angezeigte Wert sondern der zu Grunde liegende String gefiltert wird.
Die einzustellenden Spaltenattribute sehen folgendermaßen aus:

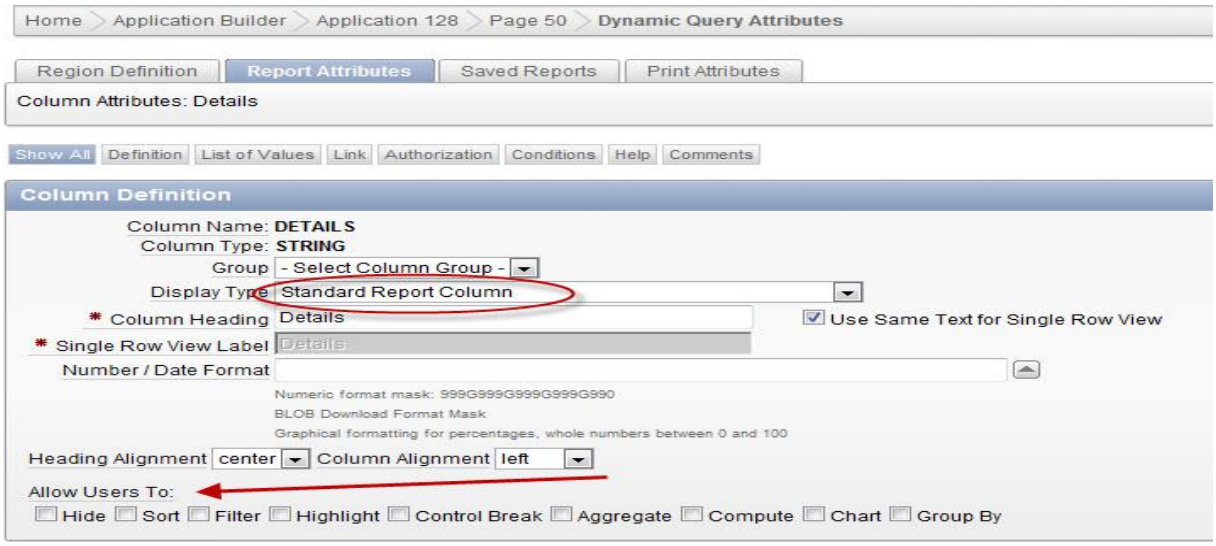

Abb. 16 Die Spaltenattribute "Display Type" und "Allows Users to" müssen angepasst werden

Das Javascript, das bei onmouseover aufgerufen wird, erzeugt für dieses Beispiel lediglich einen Alert, das den Primärschlüsselwert für den ausgewählten Datensatz anzeigt. Diese Funktion sieht folgendermaßen aus:

```
function zeige_details(elem){
 id=elem.getAttribute("id",0);
 startpos=id.indexOf("_")+1;
 alert("User ID:"+id.substr(startpos));
}
```

Schlussbemerkung
In den oben beschriebenen Beispielen wurde gezeigt wie Javascript mit Standard APEX Elementen kombiniert werden kann, um Anwendungen dynamischer zu gestalten. Die Beispiele zeigen dabei die grundsätzliche Vorgehensweise und verzichten auf komplexe Anwendungslogik. Je nach konkreter Aufgabenstellung können in den Javascript Funktionen Anwendungslogik oder auch Ajax Aufrufe für Datenbankzugriffe implementiert werden. Grundsätzlich kann alles, was mit Javascript gemacht werden kann, auch in Apex integriert werden.

Kontaktadresse:

Alexander Scholz
its-people Hochtaunus GmbH
Lyoner Straße 44-48
D-60528 Frankfurt am Main

Telefon: +49 (0) 69-2475210-0
Fax: +49 (0) 69-2475210-21
E-Mail alexander.scholz@its-people.de
Internet: www.its-people.de