Quasi-modale Dialoge in Apex

Frank Weyher, ORBIT Gesellschaft für Applikations- und Informationssysteme mbH

Dieser Artikel richtet sich in erster Linie an Apex-Entwickler, die modale Fenster, wie sie in traditionellen GUIs gang und gäbe sind, realisieren möchten.

Im Rahmen einer größeren Apex-Applikationslandschaft bieten sich für beliebige Geschäftsobjekte applikationsübergreifend Funktionen an. Diese Funktionen, wie zum Beispiel eine Notiz- oder Dokumentenanhangs-Funktion, sollten durch einen Klick auf ein Icon erreichbar sein. Seit einiger Zeit unterstützt Apex Plug-ins, um den Entwickler von der Codierung einer eigenen Komponente zur Simulation modaler Dialoge zu befreien. Eine kurze Recherche führte zu dem Plug-in „SkillBuilders Modal Page“ (siehe http://skillbuilders.com/oracle-apex/Apex-Consulting-Training.cfm?category=apex-plug-ins), das diese Anforderungen unterstützt. Der Autor verwendet das Plug-in in der Version 2.0.0 in einer Apex-4.2-Umgebung.

Notwendige Schritte

Die Konfiguration und Verwendung des Plug-ins ist einfach und erfordert einige im Folgenden beschriebene Maßnahmen. Das Plug-in wird über das Apex-Import-Utility des Application Builder importiert. Die Konfigurationseinstellungen in den „Shared Components –> Component Settings“ sind eine Frage des persönlichen Geschmacks beziehungsweise des Corporate Designs. Allerdings brauchte das CSS ein wenig Pflege, siehe Abschnitt „Was noch zu tun bleibt“.

Der nächste Schritt besteht darin, eine Dynamic Action (DA) zu erstellen. Diese soll das Klick-Ereignis auf das Icon verarbeiten. Es darf sich dabei sowohl in einem Bericht als Berichtsspalte als auch in einem Formular befinden. Die „Global Page“ ist der ideale Ort, um diese DA zu definieren, weil sie dann auf jeder Seite zur Verfügung steht. Tabelle 1 zeigt die nötigen Einstellungen (nicht genannte Eigenschaften bleiben bei den Standardwerten). „md“ steht in diesem Fall für „modaler Dialog“. Tabelle 2 zeigt die Eigenschaften der (True-)Action.

Auch bei der Definition von Buttons für Formulare kann man sich doppelte Arbeit ersparen. Dazu legt man sich ein eigenes Template für den Button an, um die nötigen Einstellungen nur einmal definieren zu müssen. Der Autor hat es als Anchor-Element mit einem Bild definiert (siehe Listing 1). Tabelle 3 zeigt die Bedeutung der einzelnen Attribute.

Der Aufruf des quasi-modalen Dialogs kann nun in Berichten anhand eines Column-Links und in Formularen durch einen Button erfolgen. Wichtig ist, dass in den Link-, beziehungsweise Button-Attributen bestimmte Eigenschaften gesetzt sind (siehe Tabelle 4).

Apex besteht darauf, dass die URL einen Link enthält. Er wird jedoch nicht benötigt, deshalb wird dort „javascript:void(0)“ eingetragen. In der Definition der DA wurde ein dynamischer „Event Scope“ mit einem bestimmten Namen festgelegt, der nun hier erforderlich ist. Im Header der Berichtsregion wird „ “ eingefügt.

Ist der „Display Point“ des Buttons „Above …“ oder „ Below Region“, kann das „div“-Tag im Regions-Header des Formulars definiert werden. Andernfalls bleibt nur der Seiten-Header.

```
<a href="#LINK#" style="padding:10px" onclick="return false;" #BUT-
TON_ATTRIBUTES# id="#BUTTON_ID#" class="md-link">
<img border="0" title="Mein Pop-Up" alt=" Mein Pop-Up" src="#IMAGE_
PREFIX#js_libs/famfamfam/icons/comment.png">
</a>
```

Listing 1

Attribut	Wert	Bemerkung
Event	Click	
Selection Type	jQuery Selector	
jQuery Selector	.md-link	
Event Scope	Dynamic	Bitte unbedingt den Punkt vor „md-link“ beachten. Er bedeutet, dass sich der jQuery-Selector auf das class-Attribut bezieht. Dieser Selector sorgt also dafür, dass alle Seitenelemente mit dem class-Attribut „md-link“ empfindlich für die DA sind.
Static Container	#region_with_md_link	Damit in Reports beim Weiterblättern die Linkspalte auch auf den Folgeseiten des Reports funktioniert, muss der Event-Scope den Wert „Dynamic“ haben. Der hier verwendete Name taucht dann wieder als „id“ im Regions-Header des Berichts auf.

Tabelle 1: Parameter der Dynamic Action

Attribut	Wert	Bemerkung
Fire On Page Load	No	
Dialog Title	Mein Pop-Up	
URL Location	Attribute of Triggering Element	Da kontextabhängige Informationen an die Seite im modalen Dialog transportiert werden sollen, kann es nicht „Statically Defined", sondern muss „Attribute of Triggering Element" sein.
Attribute Name	href_md	Dass hier nicht „href" verwendet wird, sondern das selbst definierte Attribut „href_md", liegt an der Art und Weise, wie Apex-Buttons die URLs-Aufrufe behandelt. Sie werden durch einen JavaScript-Aufruf maskiert, was dazu führt, dass der Aufruf des Plug-ins nicht funktioniert. Dadurch kann der URL nicht direkt über die Apex-Eigenschaften definiert werden, sondern muss direkt in den Link-Attributen erfolgen.
Autoclose On Element Selector	div#success-message	Standard – Die Dialoge werden nicht automatisch geschlossen, sondern müssen aktiv durch den Benutzer beendet werden.
Dialog Height/Width Mode	Auto	

Tabelle 2: Parameter der True Action der Dynamic Action

Attribut	Wert	Bemerkung
onclick	"return false;"	Damit der Klick auf den Button/Icon durch die DA behandelt wird.
class	md-link	
href	"#LINK#"	Wird nicht verwendet.
Style	"padding:10px"	Beliebig. In diesem Fall sorgt es für einen 10-Pixel-Abstand zu den anderen Buttons.

Tabelle 3: Button-Template-Attribute

Attribut	Wert	Bemerkung
class	md-link	Der Name der Klasse „md-link" muss mit dem jQuery-Selektor aus dem Abschnitt „Definition einer Dynamic Action zum Aufruf des Plug-ins" übereinstimmen.
onclick	"return false"	Dieser „onclick"-Handler sorgt dafür, dass die DA den Click behandelt.
title	"Mein Pop-Up"	Der Titel produziert einen Tooltip, der natürlich beliebig ist.
href_md	"f?p=&APP_ID. :10001 :&SESSION. :DEFAULT :&DEBUG. :<Parameter>"	Das Attribut „href_md" muss mit dem Attributnamen aus der Definition der DA übereinstimmen und ist die URL, die beim Klick auf den Link aufgerufen wird, in diesem Fall die Applikationsseite „10001". Sie übergibt als „REQUEST" den Kontext „DEFAULT". Die weiteren Parameter sind natürlich applikationsspezifisch und enthalten typischerweise einen Objekttyp und die Id des Objekts.

Tabelle 4: Link-Eigenschaften

Die Popup-Fenster zeigen herkömmliche Apex-Seiten auf der Grundlage des Popup-Templates. Dieses bietet die typische reduzierte Optik. Es wird auch durch einige Standard-Themes bereitgestellt; gegebenenfalls muss man es noch an seine Bedürfnisse anpassen.

Für die Parameter-Übergabe an Popup-Seiten hat man alle auch sonst in Apex verfügbaren Möglichkeiten. Deshalb sind sie eine elegante Methode, um eigene Auswahldialoge zu implementieren oder auch zentrale Funktionen bereitzustellen.

Zu beachten ist, dass diese Seiten nicht zu unterschiedliche Größen besitzen sollten, besonders dann, wenn man mit horizontalen Listen in den Dialogen arbeitet, die die Registeroptik nachempfinden. Dies führt zu einem ruhigeren Bild beim Wechsel der Seiten.

Was noch zu tun bleibt

Zurzeit muss jede nutzende Applikation innerhalb der Applikationslandschaft die DA selber definieren. Ein weiteres Ziel ist – um den Aufwand in den nutzenden Applikationen möglichst gering zu halten –, dass sie nur einen Link aufrufen und die zentrale Apex-Applikation sich um den Rest kümmert. Leider verhält sich die zentrale DA, die etwa auf „Page Load" als auslösendes Ereignis reagiert, nicht wie gewünscht.

Das Plug-in bietet ein sogenanntes „Custom Theme" an, das in den Component-Settings des Plug-ins eingestellt ist. Die dafür nötige „css"-Datei darf dazu nicht bei den Plug-in-Dateien liegen. Sie muss, damit sie vom Plug-in gefunden wird, im Bereich der Static-Files abgelegt werden. Zurzeit gibt es allerdings noch Schwierigkeiten beim Laden der zusätzlichen Ressourcen (wie zum Beispiel Bilder). Der Ersetzungsmechanismus für „#WORKSPACE_IMAGES#" funktioniert nicht wie erwartet. Der Autor hat sich damit beholfen, ein existierendes Theme gemäß seinen Wünschen anzupassen.

Frank Weyher
frank.weyher@orbit.de