Force.com Visualforce Pages

Lesson 06 : Javascript in Visualforce

Lesson Objectives

By the end of this lesson, you will be able to :

- Describe the use of AJAX with in Visualforce
- Create access to standard actions via the URLFOR expression
- Create reusable functions within <script> tags
- Create partial page refreshers on JavaScript events
- Describe the usage of the AJAX action components

Action Binding and JavaScript

At this point, only actions that are shared across all objects are exposed through standard controllers.
But further standard Salesforce action are available by using JavaScript and the expression syntax with the !URLFOR and $Action keywords.

- Additional standard actions will be added into standard controllers in future releases
- See the online help for more information regarding global $variables

Example

```
<apex:PageBlockButtons>
   <apex:commandButton value="Edit" action={!edit}"/>
   <apex:commandButton onclick="parent.window.location.href='{!URLFOR($Action.Position__c.Clone,position.id)}'" value="Clone"/>
</apex:PageBlockButtons>
```

JavaScript Functions

It is possible to create reusable JavaScript functions with HTML <script> tag.

- **This creates functions that can be called from within the Visualforce page**

Creating functions can be helpful if the same JavaScript code is needed multiple times on a page.

These functions can then be used on the embedded Javascript event attributes in many Visualforce tags.

- **Example: onclick=“changeFont();”**

It can also be useful to isolate the Javascript at the top of the page for maintenance purposes.

JavaScript Functions

```
<apex:page id="thePage">
<!- - the following script element contains a simple function for changing the font. - - >
<script language="JavaScript" type="text/javascript">
function changeFont(input, textid) {
       if (input.checked) document.getElementById(textid).style.fontWeight = "bold";
        else document.getElementById(textid).style.fontWeight = " normal";
}
</script>
<!- - the first output panel calls the function, passing in the existing occurrence of the
checkbox, as well as the DOM ID of the output component- - >
   <apex:outputPanel layout="block">
       <label for="checkbox">Click this box to change text font:</label>
   <apex:input id="checkbox" type="checkbox"
onclick="changeFont(this,'{!$Component.thePanel}');"/>
    </apex:outputPanel>
<!- - the second output panel contains the text that will be changed - - >
  <apex:outputPanel id="thePanel"layout="block">Change me!
   </apex:outputPanel>
</apex:page>
```

Partial Page Update

Due to the data binding capabilities of Visualforce, AJAX is not required for API access.

AJAX is a way to create better performing websites through partial page refreshes.

- By just refreshing a small section of the page, performance is improved by not reloading the entire page.

Partial Page Update (cont.)

The simplest way to implement a partial page update is to use the reRender attribute on a button or link.

First, isolate the portion of the page that should be rerendered when the button or link is clicked by surrounding it with the <apex:outputPanel> tags.

- Be sure to give the **outputPanel** an **id**

Then, create the command button or link that will trigger the partial refresh.

- Add the **reRender** attribute to the button or link tag and assign it the value of the **id** of the **outputPanel** created earlier

Partial Page Update (cont.)

```
<apex:page standardController=“Account”>
   <apex:pageBlock title=“Hello {!$User.FirstName}!”>
         You are displaying contacts from the {!account.name} account.
         Click a contact’s name to view his or her details.
   </apex:pageBlock>
 <apex:pageBlock title=“Contacts”>
     <apex:dataTable value=“{!account.Contacts}” var=“contact” cellPadding=“4” border=“1”>
       <apex:column>
          <apex:commandLink rerender=“detail”>
               {!contact.Name}
              <apex:param name=“cid” value=“{!contact.id}”/>
          </apex:commandLink>
      </apex:column>
  </apex:pageBlock>
  <apex:outputPanel id=“detail”>
       <apex:detail subject={!$CurrentPage.parameters.cid}” relatedList=“false” title=“false”/>
  </apex:outputPanel>
</apex:page>
```

Asynchronous Operation Status

AJAX behaviors (like partial page updates) are asynchronous events that occur in the background while a user continues to work on the page.

In the event that these updates take longer than near-instantaneous. Visualforce supports displaying status updates using the actionStatus tag.

- **You can display text at the beginning or end of the asynchronous event using the startText and stopText tags**
- **You can specify what is to be displayed after the refresh by using the** <apex:facet name="stop"> **tags**

Asynchronous Operation Status

```
<apex:page standardController=“Account”>
<! - - Same as previous example - - >
          <apex:commandLink rerender=“detail” status=“detailStatus”>
               {!contact.Name}
              <apex:param name=“cid” value=“{!contact.id}”/>
          </apex:commandLink>
<apex:pageBlock>
  <apex:outputPanel id=“detail”>
       <apex:actionStatus startText=“Requesting…” id=“detailStatus”>
       <apex:facet name=“stop”>
        <apex:detail subject=“{!$CurrentPageReference.parameters.cid}” relatedList=“false”
title=“false”/>
   </apex:facet>
 </apex:actionStatus>
  </apex:outputPanel>
</apex:page>
```

AJAX Behavior on Events

Visualforce Components & Javascript Events

Many Visualforce tags relate to different UI components.

These often have attributes equivalent to JavaScript events to handle user actions to those components.

AJAX Behavior on Events
Javascript Events

Most Visualforce tags that produce concrete HTML elements support appropriate Javascript event attributes.

Atribute	The event occurs when….
onabort	Loading of an image is interrupted
onblur	An element loses foucs
onchange	User changes the contents of the field
onclick	Mouse clicks an object
ondblclick	Mouse double clicks an object
onerror	An error occurs while loading a document or image
onfocus	An element gets focus
onkeydown	A keyboard key is pressed
onkeypress	A keyboard key is pressed or held down
onkeyup	A keyboard key is released
onload	A page or an image is finished loading
onmouseover	The mouse is moved over an element
onselect	Text is selected
onsubmit	The submit button is clicked

AJAX Behavior on Events

This method of having the button or link click trigger the refresh is simple to implement, but often there are better ways to improve the user experience.

- **Example: instead of clicking a link, the detail section should refresh when the mouse hovers over a name**

If the event is happening to the same component that should take the action, use the built-in Javascript event attributes.

If the event is happening to a different component than will take the action, use the actionSupport tag to handle the event.

AJAX Behavior on Events

```
<apex:page standardController=“Account”>
<! - - Same as previous  example - - >
          <apex:outputPanel>
               <apex:actionSupport event=“onmouseover” rerender=“detail”>
                   <apex:param name=“cid” value=“{!contact.id}”/>
               </apex:actionSupport>
              {!contact.Name}
          </apex:outputPanel>
      </apex:column>
  </apex:dataTable>
</apex:form>
<apex:pageBlock>
  <apex:outputPanel id=“detail”>
       <apex:actionStatus startText=“Requesting…”>
       <apex:facet name=“stop”>
        <apex:detail subject=“{!$CurrentPageReference.parameters.cid}” relatedList=“false” title=“false”/>
   </apex:facet>
 </apex:actionStatus>
  </apex:outputPanel>
</apex:page>
```

AJAX Behavior on Events

There are a few additional AJAX components.
actionPoller: similar to actionSupport, but the event is based on a timer instead of a user action
actionFunction: provides support for invoking a controller action from Javascript code using an AJAX request by defining a new JavaScript function
actionRegion: used to demarcate which parts of the page the server should reprocess

- It does not, however, say which areas should be rerendered, which should be specified with a panel

Demo

Creating Partial Page refreshers for Conditional Fields

Summary

<apex:component> tag can be used to create your own custom, reusable components

Template Tags

Visualforce Performance Considerations