Key points to remember about actionRegion, actionFunction, actionSupport, actionPoller & actionStatus.

Vimal Tiwari Follow
Jan 21, 2019 · 3 min…

Note: All Action events must be the child of an ***<apex:form>***

<apex:actionRegion>

1. is used to optimize VF performance.
2. tells Force.com Server which components should be processed. So whatever inside an ActionRegion is processed by a server when AJAX request is generated on the event such as “KeyPress” or “onClick” etc.
3. *doesn’t define what areas of the page are re-rendered when the request completes. To achieve that functionality, use other* ***<apex:action****> *tags.*

```
<apex:page standardController="Opportunity">
  <apex:form >
    <apex:pageBlock title="Edit Opportunity" id="thePageBlock" mode="edit">

<apex:pageBlockButtons >
        <apex:commandButton value="Save" action="{!save}"/>
        <apex:commandButton value="Cancel" action="{!cancel}"/>
      </apex:pageBlockButtons>

<apex:pageBlockSection columns="1">
      <apex:inputField value="{!opportunity.name}"/>
      <apex:pageBlockSectionItem>
      <apex:outputLabel value="{!$ObjectType.opportunity.fields.stageName.label}"
                        for="stage"/>

 <!-- Without the actionregion, selecting a stage from the picklist would cause a validation error if you hadn't already entered data in the required name. -->

      <apex:actionRegion>
        <apex:inputField value="{!opportunity.stageName}" id="stage">
          <apex:actionSupport event="onchange" rerender="thePageBlock"
                              status="status"/>
          </apex:inputField>
          </apex:actionRegion>
      </apex:pageBlockSectionItem>
        </apex:pageBlockSection>

</apex:pageBlock>
    </apex:form>
</apex:page>
```

<apex:actionFunction>

<apex:actionFunction name=”myactionfun” action=”{!actionFunctionTest}” reRender=”pgBlock, pbSection” />

1. order of **<apex:param>** is matched by the caller's argument list. **rerender** attr is needed to define <apex:param>.
2. It doesn't have **event** attr. It doesn't add the Ajax Request before calling the Controller method.
3. *can't place **<apex:actionFunction>** inside an iteration component — **<apex:pageBlockTable>**, **<apex:repeat>**, and so on. Put the **<apex:actionFunction>** after the iteration component, and inside the iteration put a normal **JavaScript** function that calls it.*
4. for defination, **name** is mandatory. For example :

```
<apex:actionFunction name='Function1' />
```

<apex:actionSupport>

```
<apex:inputText value="{!dummyString}" >
<apex:actionSupport event="onchange" action="{!actionSupportTest}" reRender="pgBlock, pbSection" />
</apex:inputText>
```

1. It adds the AJAX request to VF page and then Calls the Controller method.
2. It has **event** attr and it is used mainly when we can't get any event like "KeyPress" or "onClick" for an element

```
<-- onClick attr is not appilicable for <apex:outputText> -->
<apex:outputText value="Click Me!: {!count}"/>
            <apex:actionSupport event="onclick"
                                action="{!incrementCounter}"
                                rerender="counter" status="counterStatus"/>
        </apex:outputpanel>
```

3. *can place **<apex:actionSupport>** inside an iteration component — **<apex:pageBlockTable>**, **<apex:repeat>**, and so on.*

4. for definition, nothing is mandatory. For example :

```
<apex:actionSupport />
```

<apex:actionPoller>

```
<apex:actionPoller action="{!incrementCounter}" reRender="counter" interval="15" enabled = "true" />
```

1. A timer that sends an AJAX request to the server according to a time interval that you specify.
2. Enabled attribute is used to make poller as active or inactive by default value is true
3. it won't time out due to inactivity.
4. for definition, nothing is mandatory. For example :

```
<apex:actionPoller/>
```

<apex:actionStatus>

<apex:actionStatus startText=" (incrementing...)" stopText=" (done)" id="counterStatus"/>

1. displays the status of an AJAX update request.

Sample code :

```
<apex:page controller="ActionController">
    <apex:slds />

  <apex:form >
      <script>
          function javaScriptMethod(){
            myactionfun();
          }
      </script>

      <apex:actionFunction name="myactionfun"  action="{!actionFunctionTest}" reRender="pgBlock, pbSection" />
      <apex:pageBlock title="Action Support/Function">
          <apex:pageblockSection >
              <apex:inputText value="{!dummyString}" >
                  <apex:actionSupport event="onchange" action="{!actionSupportTest}" reRender="pgBlock, pbSection" />
              </apex:inputText>
              <apex:inputcheckbox onclick="javaScriptMethod();" />
          </apex:pageblockSection>

      </apex:pageBlock>
      <apex:pageblock title="Output of Action Funtion/Support" id="pgBlock">
          <apex:pageblockSection id="pbSection" >
              {!actionFunTest}
          </apex:pageblockSection>
      </apex:pageblock>

      <apex:pageBlock title="Action Poller">
          <apex:pageBlockSection >
              <apex:outputText value="Watch this counter: {!count}" id="counter"/>
              <apex:actionPoller action="{!incrementCounter}" reRender="counter" interval="10"/>
          </apex:pageBlockSection>
      </apex:pageBlock>

  </apex:form>
</apex:page>


/////////////////////////////////////////////////////////
public class ActionController {

    public static String actionFunTest { get; set; }
    public static String dummyString { get; set; }

    Integer count = 0;

    public void actionFunctionTest(){
      actionFunTest = 'Value from Action Function';
    }

    public void actionSupportTest(){
      actionFunTest = 'Value from Action Support '+dummyString;
    }


    public PageReference incrementCounter() {
        count++;
        return null;
    }

    public Integer getCount() {
        return count;
    }
}
```