ADF Code Corner

040. Partial form submit using af:subform and ADF

twitter.com/adfcodecorner

Abstract:

A common application development use case is partial form submit, in which configurable parts of an input form are submitted while others are not. The form submit may be performed on a single page or across views, for example in the course of an entry wizard. ADF Faces provides the af:sub form component to partition input forms for partial submit. By default, the ADF binding layer does not know how to handle partial form submits and instead produce validation errors, unless developers tell it how to. This blog entry is an addition to the af:subform discussion in the Oracle Fusion Developer Guide book by Mc Graw Hill.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
DD-MAR-2010

Oracle ADF Code Corner is a loose blog-style series of how-to documents that provide solutions to real world coding problems.

Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

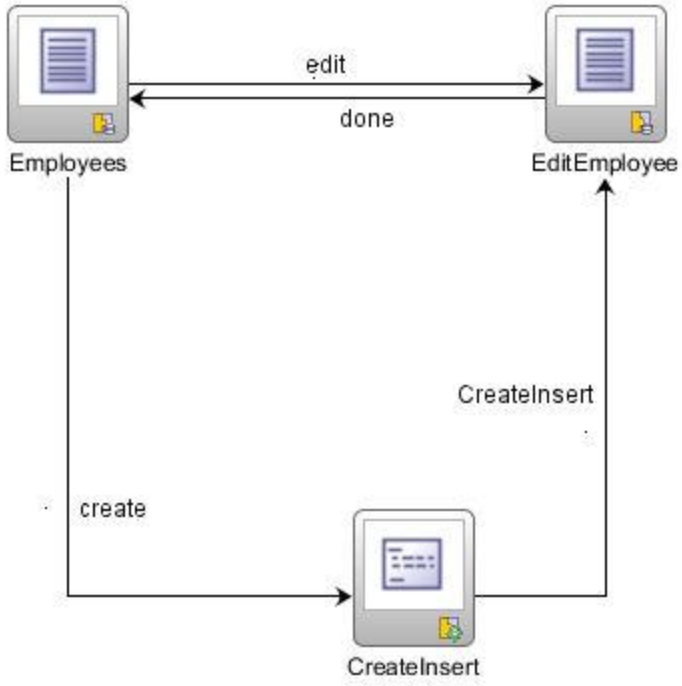
Please post questions or report problems related to the samples in this series on the OTN forum for Oracle JDeveloper: http://forums.oracle.com/forums/forum.jspa?forumID=83

Introduction

The af:subform [docs] is an independent area - or region - within a form that represents a submit boundary, which means that a submit on a sub form does not affect any component outside of it. The contents of a sub form will only be validated if a component inside of the sub form is responsible for submitting the page or if the default attribute is set to true. This allows for comparatively fine-grained control of which components will be validated and pushed into the model without the compromises of using entirely separate form elements. The af:subform addresses the problem of nested af:form elements, which in ADF Faces are not supported (the truth is that HTML does not support this either). In this blog entry you learn

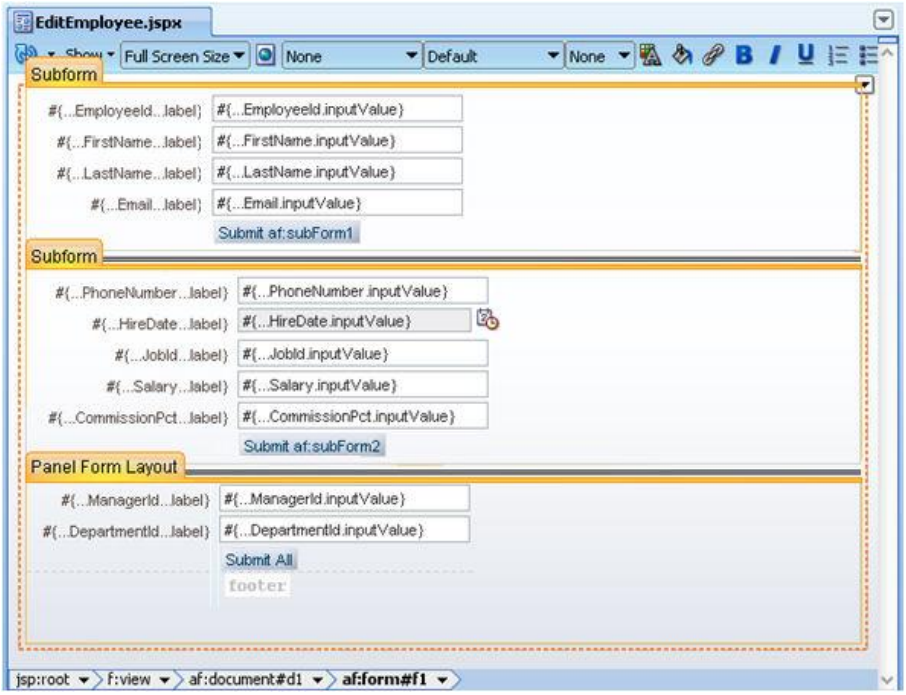
- How to partition a form into sub form using af:subform
- How to configure the ADF binding layer to suppress validation on partial submit
- About the ADF skipValidation binding property

Note that an advantage of the af:subform compared to other options of handling partial submit, like using the partialSubmit="true" and immediate="true" properties on the component, is ease of use as there is less code to write. In the sample I use in this blog post, the input form is based on the Employees table of the Oracle HR demo schema. Users start on the "Employees" browse page to create or edit employee records.

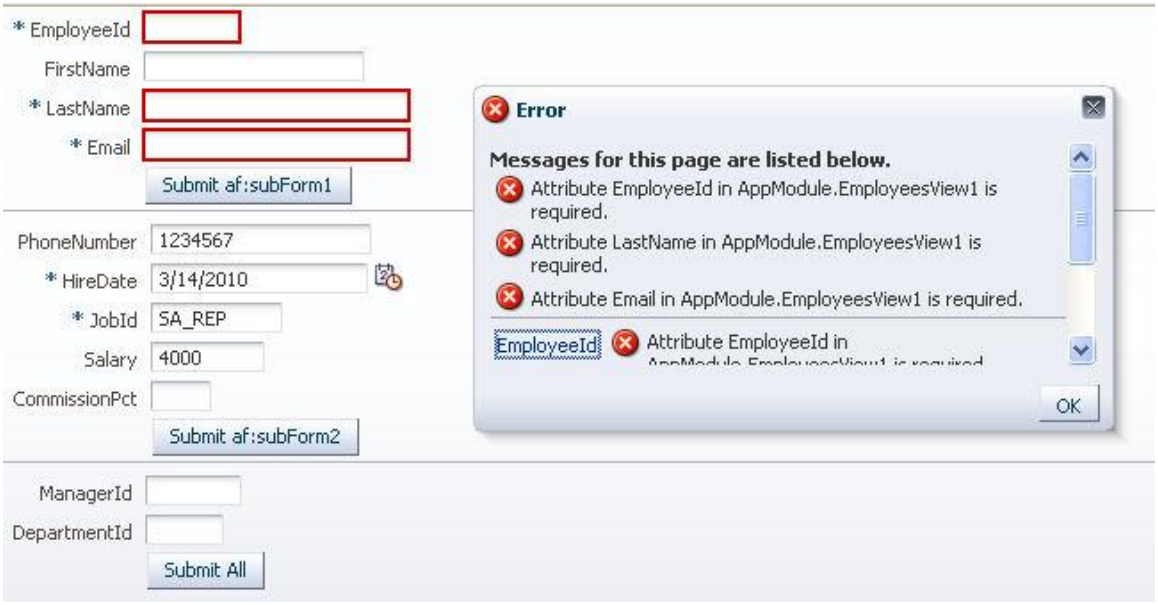


Building the Form

To build the form, I used an af:form tag that contains two af:subform elements and a af:panelFormLayout component. The two af:subform components contain the input fields I like to be included in a partial page submit. The input text component in the af:panelFormLayout component are submitted when the af:form submits.



Note that the command button "Submit af:subForm1" and "Submit af:subForm2" are added for demo reasons to submit the sub forms. This submit could also be through setting autosubmit="true" on the input fields, in which case explicit buttons are not required. Lets see what happens when I run the page, provide input values for the 2nd subform and press the "Submit af:subForm2" button.



Ouch! Did af:subform work at all, and if where is this error coming from. Well, the af:subform worked and the error we see is one coming from the server side.

If af:subform did not work, we would see client side validation errors using a different popup dialog. So af:subform indeed submitted the content of subForm2 only. However, the partial submit updated the model, which using ADF is the ADF binding layer.

By default, the ADF binding layer does not distinguish between partial form submit and full form submit, which means that all attributes of the DataControl row are validated, leading to a failure of attributes in subform 1.

The page source for the form is shown below

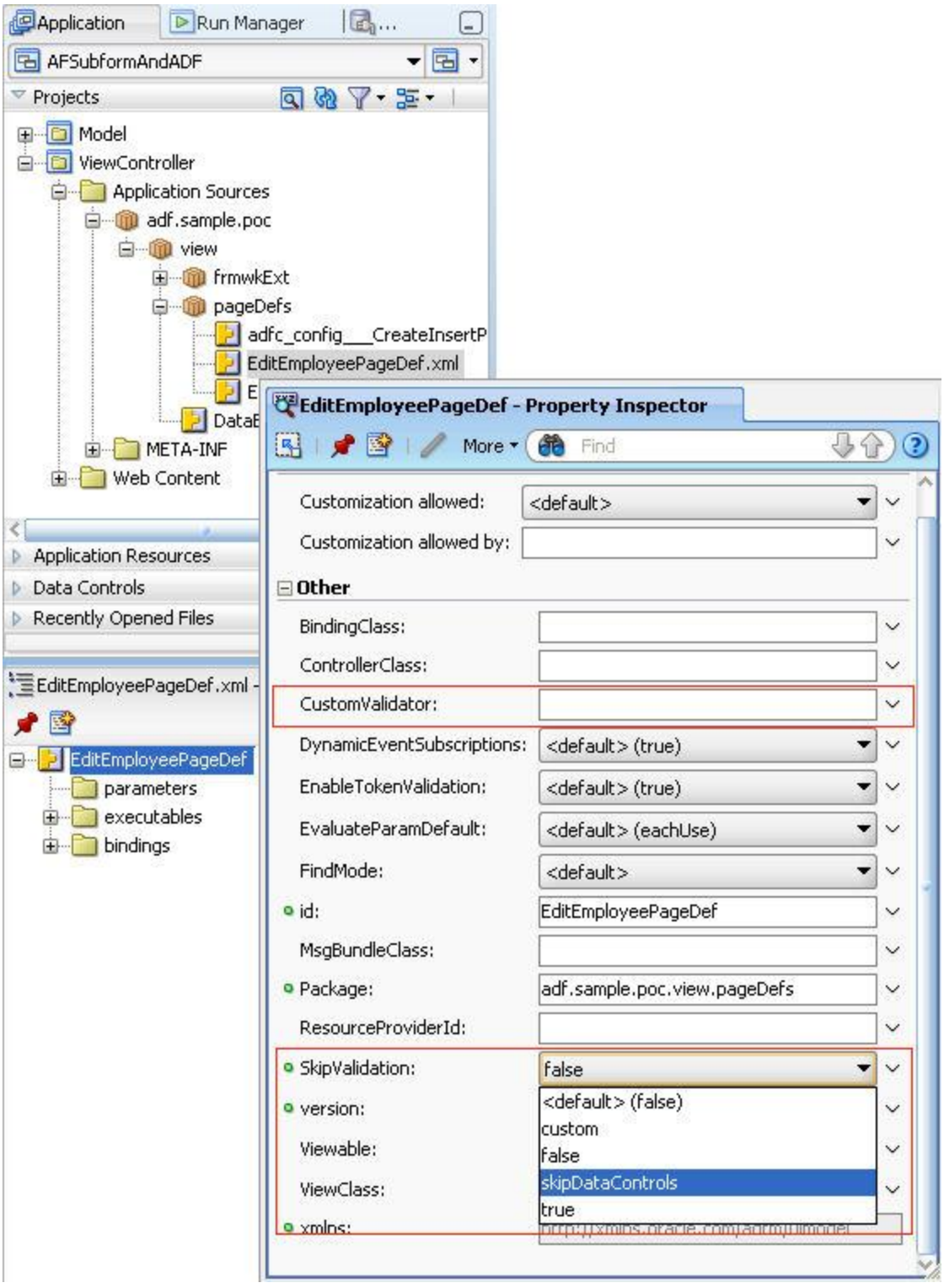
```
<af:form id="f1">
  <af:subform id="s1" default="true">
    <af:panelFormLayout id="pfl2">
      <af:inputText value="#{bindings.EmployeeId.inputValue}"
             label="#{bindings.EmployeeId.hints.label}"
             required="#{bindings.EmployeeId.hints.mandatory}"
             columns="#{bindings.EmployeeId.hints.displayWidth}"
             maximumLength="#{bindings.EmployeeId.hints.precision}"
             shortDesc="#{bindings.EmployeeId.hints.tooltip}"
            id="it1">
       <f:validator binding="#{bindings.EmployeeId.validator}"/>
       <af:convertNumber groupingUsed="false"
                         pattern="#{bindings.EmployeeId.format}"/>
      </af:inputText>
      ...
      <af:commandButton text="Submit af:subForm1" id="cb2"/>
    </af:panelFormLayout>
```

```
  </af:subform>
  <af:separator id="s3"/>
  <af:subform id="s2" default="true">
    <af:panelFormLayout id="pfl3">
      <af:inputText
         value="#{bindings.PhoneNumber.inputValue}"
         label="#{bindings.PhoneNumber.hints.label}"
         required="#{bindings.PhoneNumber.hints.mandatory}"
         columns="#{bindings.PhoneNumber.hints.displayWidth}"
         maximumLength="#{bindings.PhoneNumber.hints.precision}"
         shortDesc="#{bindings.PhoneNumber.hints.tooltip}"
         id="it10">
        <f:validator binding="#{bindings.PhoneNumber.validator}"/>
      ...
      <af:commandButton text="Submit af:subForm2" id="commandButton1"/>
    </af:panelFormLayout>
  </af:subform>
  <af:separator id="s4"/>
  <af:panelFormLayout id="pfl1">
    <af:inputText value="#{bindings.ManagerId.inputValue}"
                  label="#{bindings.ManagerId.hints.label}"
                  required="#{bindings.ManagerId.hints.mandatory}"
                  columns="#{bindings.ManagerId.hints.displayWidth}"
                  maximumLength="#{bindings.ManagerId.hints.precision}"
                  shortDesc="#{bindings.ManagerId.hints.tooltip}"
                  id="it8">
      <f:validator binding="#{bindings.ManagerId.validator}"/>
      <af:convertNumber groupingUsed="false"
                        pattern="#{bindings.ManagerId.format}"/>
    </af:inputText>
    ...
    <af:commandButton text="Submit All" id="cb1" action="done"/>
  </af:panelFormLayout>
</af:form>
```

Note that both af:subform have their "default" attribute set to true, which means that a submit of the af:form tag also submits the subform. This ensures that a form submit still behaves as without the use of af:subform. The difference though is that the submit in one of the subforms does not submit the form as a whole (which exactly is what we want).

Fixing the ADF binding layer

The ADF binding layer - the pageDef file - has two new properties in JDeveloper 11g that allow you to suppress validation either completely or conditionally.



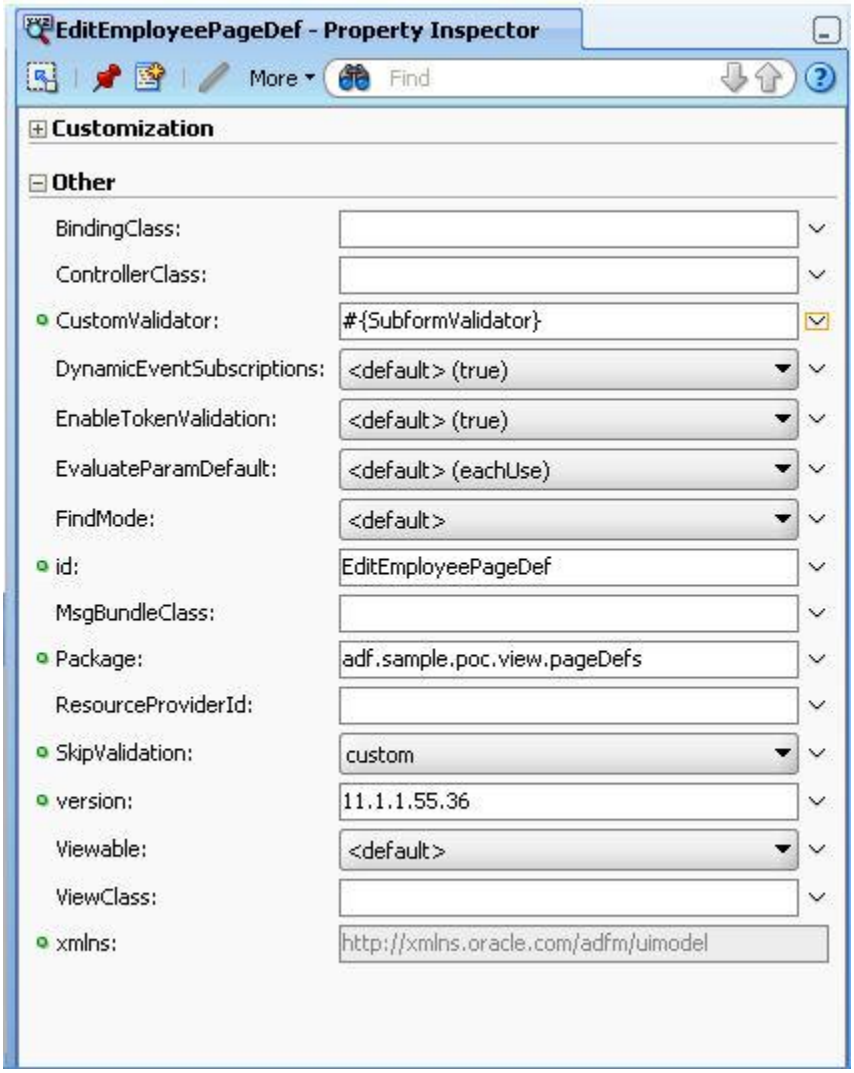
SkipValidation

The skipValidation setting defines when and if validation is omitted on the ADF binding layer. The default value is "false", which means that validation is always performed, leading to the validation error

upon partial submit shown earlier. Setting this value to "true" always skips validation for this page. Always skipping validation might be good if the page is part of a set of wizard pages where you want to validate the data control upon wizard completion but not earlier. Usually however you want to define conditional validation for which the "custom" entry can be used. Setting the skipValidation property to "custom" requires you to configure the "CustomValidator" property with an EL pointing to a managed bean that has the `BindingContainerValidator` interface implemented.

CustomValidator

The custom validator is a Java class that implements the `BindingContainerValidator` interface and that is configured as a managed bean to be EL accessible. The managed bean is referenced from the *CustomValidator* field in the ADF binding layer.



The custom validator class in this blog example is shown below

```
/**
 * Custom ADF binding container validator class to skip validation
```

```
based on session context
 * information
 */
public class SubformValidator implements BindingContainerValidator {
    //sessionn attribute that contains true/false and that is set by a
    //setPropertyListener on the af:subform

   public static final String ENABLE_DC_VALIDATION =
                          "adf.sample.poc.frmwExt.dc_val_enabled_attr";

   public SubformValidator() {
        super();
   }

   public void validateBindingContainer(
                     BindingContainer bindingContainer) {

       FacesContext fctx = FacesContext.getCurrentInstance();
       ExternalContext ectx = fctx.getExternalContext();
       Map sessionMap = ectx.getSessionMap();
       Object dataControlValidation =
             sessionMap.get(ENABLE_DC_VALIDATION);
        //skip validation if flag is set to true
        if ((dataControlValidation != null) &&
          ((String) dataControlValidation).equalsIgnoreCase("TRUE")){
           //avoid validation
          ((DCBindingContainer)bindingContainer).
                                          setSkipValidation(true);
         }
        //validate
        else{
            return;
        }
    }
}
```

The validator class looks in the session for an attribute "adf.sample.poc.frmwExt.dc_val_enabled_attr" which is set by the submit buttons of the input form (after a slight modification shown next). If the attribute exist and the value is set to "TRUE", the validation is suppressed and the binding layer does no longer produce errors. If the value is missing or set to "FALSE", validation is enforced.

Changes to the af:subForm page submit

To set the session attribute value that is looked for by the custom validation class, the page source is changed as follows

```
<af:commandButton text="Submit af:subForm1" id="cb2">
```

```
  <af:setPropertyListener from="#{'true'}"
     to="#{sessionScope['adf.sample.poc.frmwExt.dc_val_enabled_attr']}"
     type="action"/>
</af:commandButton>
```

An af:setPropertyListener tag is added to all of the command buttons in this sample to set the `"adf.sample.poc.frmwExt.dc_val_enabled_attr"` to true or false. Note the use of square brackets to address attribute that have dots in their namings.

Rerun the Sample

As you expect, running the sample again only updates the model in the binding layer but not the data control until you submit the full form.

af:subform and Form Layouts

Using the af:subform in an af:form requires changes to the standard way JDeveloper creates input forms. Usually a single panelFormLayout component is added as the child of the af:form tag.

This panelFormLayout component then contains the form fields, which are nicely layed out in columns and rows, with their labels right aligned. When you use af:subform as a child of af:panelFormLayout, then the definition of rows and columns applies to the af:subform, no longer to the input text components. This basically destroys the previously well organized form layout. To fix this, you make af:subform a child of the af:form component and add an af:panelFormLayout as a child. This then produces a smilar layout

then the default. I say similar because the vertical alignment of input components of different af:subform may not be 100%. In this case you would need to surround the input components with an af:panelLabelAndMessage component so you can use spacer or panelGroupLayout components to adjust the offsets. There is no straight rule of "just do this and it will oook beautiful" and instead you will need to play with it to get the best result.

Download JDeveloper Workspace

The Oracle JDeveloper workspace you can download from ADF Code Corner

http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html

is configured to handle partial submit properly. To see the default behavior, edit the PageDef file of the EditEmployee page and set skipValidation to "false". The model project connects to the HR demo schema and needs to be configured to access a local database of yours.

RELATED DOCOMENTATION

⊠	af:subform tag doc - http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e12419/tagdoc/af_subform.html
⊠	Oracle Fusion Developer Guide (McGraw Hill) - http://www.mhprofessional.com/product.php?isbn=0071622543&cat=7
⊠	Oracle JDeveloper Handbook - http://www.mhprofessional.com/product.php?isbn=0071602380&cat=7