Partial Page Rendering in OAF Page

Method 1:

Partial Page Rendering in OAF Page:

Let us try to implement partial page rendering for a text item.
If value of TextItem1 is null then TextItem2 will not be appeared on UI.
If value of TextItem1 is not null then TextItem2 will be appeared on UI.

1. Create a New OA Workspace and Empty OA Project
File> New > General> Workspace Configured for Oracle Applications
File Name -- PPRProj
Project Name – PPRDemoProj
Default Package -- mahi.oracle.apps.fnd.pprdemo

2. Create Application Module AM
PPRDemoProj right click > New > ADF Business Components > Application Module
Name -- PPRAM
Package -- mahi.oracle.apps.fnd.pprdemo.server

Check Application Module Class: PPRAMImpl Generate JavaFile(s)

3. Create a PPRVO View Object
PPRDemoProj> New > ADF Business Components > View Objects
Name – PPRVO
Package – mahi.oracle.apps.fnd.pprdemo.server

In Attribute Page
Click on **New** button and create transient primary key attribute with the following properties:

Attribute	Property
Name	RowKey
Type	Number
Updateable	Always
Key Attribute	(Checked)

Click **New** button again and create transient attribute with the following properties:

Attribute	Property
Name	TextItem2Render
Type	Boolean
Updateable	Always

Note – No Need to generate any JAVA files for PPRVO

4. Add Your View Object to Root UI Application Module
Right click on PPRAM > Edit PPRAM > Data Model >

Select PPRVO in Available View Objects list and shuttle to Data Model list

5. Create a OA components Page
PPRDemoProj right click > New > OA Components > Page
Name – PPRPG
Package -- mahi.oracle.apps.fnd.pprdemo.webui

6. Modify the Page Layout (Top-level) Region

Attribute	Property
ID	PageLayoutRN
Region Style	pageLayout
Form Property	True
Auto Footer	True
Window Title	PPR Demo Window Title True
Title	PPR Demo Page Header
AM Definition	mahi.oracle.apps.fnd.pprdemo.server.PPRAM

7. Create the Second Region (Main Content Region)
Right click on PageLayoutRN > New > Region

Attribute	Property
ID	MainRN
Region Style	messageComponentLayout

8. Create Two Text Items

Create First messageTextItem --
Right click on MainRN > New > messageTextInput

Attribute	Property
ID	TextItem1
Region Style	messageTextInput
Prompt	Text Item1
Length	20
Disable Server Side Validation	True
Disable Client Side Validation	True
Action Type	firePartialAction
Event	TextItem1Change
Submit	True

Note -- Disable Client Side Validation and Event property appears after you set the **Action Type property** to **firePartialAction**

Create Second messageTextItem --
Select MainRN right click > New > messageTextInput

Attribute	Property
ID	TextItem2
Region Style	messageTextInput
Prompt	Text Item2
Length	20
Rendered	${oa.PPRVO1.TextItem2Render}

9. Add Following code in PPRAMImpl.java

```
import oracle.apps.fnd.framework.OARow;
import oracle.apps.fnd.framework.OAViewObject;
import oracle.apps.fnd.framework.server.OAApplicationModuleImpl;
import oracle.apps.fnd.framework.server.OAViewObjectImpl;
public void handlePPRAction()
{
 Number val = 1;
 OAViewObject vo = (OAViewObject)findViewObject("PPRVO1");
 if (vo != null)
 {
  if (vo.getFetchedRowCount() == 0)
  {
   vo.setMaxFetchSize(0);
   vo.executeQuery();
   vo.insertRow(vo.createRow());
   OARow row = (OARow)vo.first();

   row.setAttribute("RowKey", val);
   row.setAttribute("TextItem2Render", Boolean.FALSE);
  }
 }
}
```

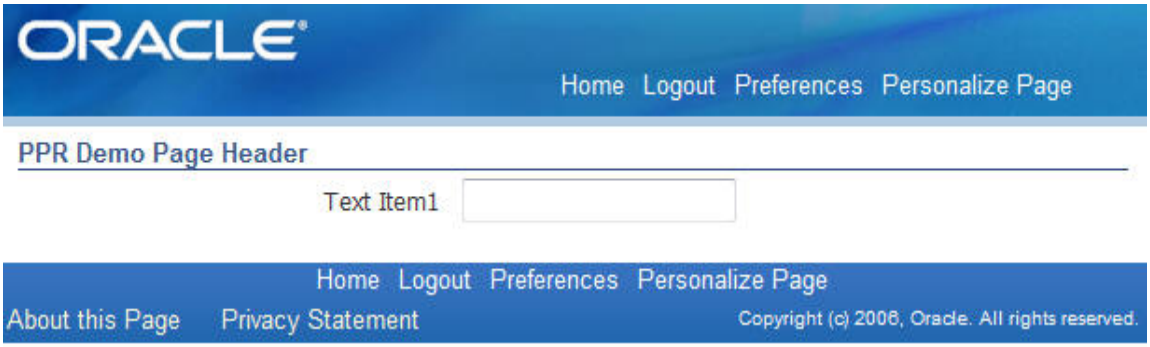
10. Implement Controller for Page

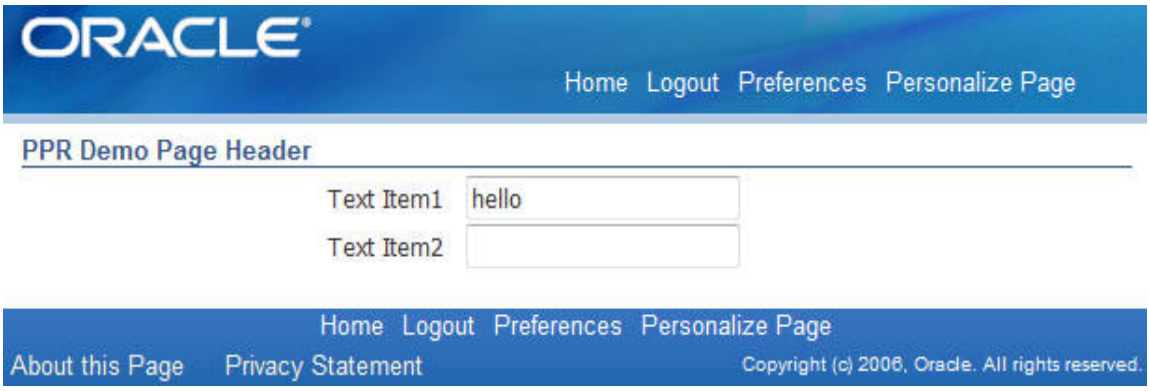
Select PageLayoutRN in Structure pane right click > Set New Controller
Package Name -- mahi.oracle.apps.fnd.pprdemo.webui
Class Name – PPRCO

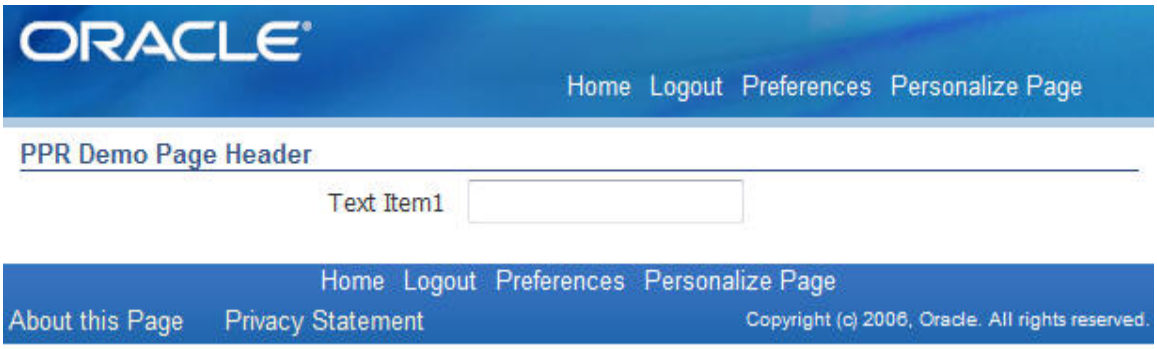
Write following code in processFormRequest function of PPRCO Controller

```
import oracle.apps.fnd.framework.OARow;
import oracle.apps.fnd.framework.OAViewObject;

public void processRequest(OAPageContext pageContext, OAWebBean webBean)
{
 super.processRequest(pageContext, webBean);
 PPRAMImpl am = (PPRAMImpl)pageContext.getApplicationModule(webBean);

 am.invokeMethod("handlePPRAction");
}
public void processFormRequest(OAPageContext pageContext, OAWebBean webBean)
{
 super.processFormRequest(pageContext, webBean);

 PPRAMImpl am = (PPRAMImpl)pageContext.getApplicationModule(webBean);
 OAViewObject vo = (OAViewObject)am.findViewObject("PPRVO1");
 OARow row = (OARow)vo.getCurrentRow();

 if ("TextItem1Change".equals(pageContext.getParameter(EVENT_PARAM)))
 {
  if (pageContext.getParameter("TextItem1").equals(""))
  {
   row.setAttribute("TextItem2Render", Boolean.FALSE);
  }
  else
  {
   row.setAttribute("TextItem2Render", Boolean.TRUE);
  }
 }
}
```

11. Congratulation you have successfully finished. Run Your PPRPG page and Test Your Work







Method 2:

PPR (Partial Page Rendering) in OAF :

While developing pages, we may face some scenarios where the requirement is to make modifications on the current page itself, instead of navigating to different page. Such as selecting an item from a choice list might result in modifications to other fields or clicking a button to add a new row in table region and many more like these.
All these tasks can be performed by simply refreshing the page however this could result in performance issues. Another alternative is to use javaScript which will give faster results but implementing complex functions can be a tough job using javaScript.

UIX Framework provides a solution for this: **Partial Page Rendering** i.e. to re-render only a limited portion of a page.

Working of PPR :-
==========
PPR is a three step process:

1) Partial page event

Partial page events are quite similar to full page events. However, there are two important differences between partial and full page events. First, partial page events specify partial page rendering-specific event parameters which are not present on the full page event equivalents. For example, partial page events may include an event parameter which identifies the set of partial targets, or nodes that should be re-rendered.
The second difference between partial page events an full page events is how the events are sent. Unlike full page events, partial page events must be sent in a way which does not force the browser to reload the current page. To implement this capability, UIX PPR uses a hidden iframe as a communication channel between the browser and the web application running on the middle-tier. Partial page events are sent by forcing a navigation in the hidden iframe, which in turns causes a request to be sent to the application on the middle-tier. Since the iframe is hidden, the process of sending a partial page event and rendering partial page contents can take place behind the scenes, without discarding the contents of the current page.

2) Partial Page Rendering Pass

When the partial page event is received by the application, the application responds by determining the set of partial targets to render and performing the partial page rendering pass. The partial page rendering pass is similar to a full page rendering pass. In both cases, the UINode tree is traversed by calling render() on each node in the tree. However, in the PPR case, only the contents generated by the partial targets are actually sent back to the browser. All other contents are dropped. So, the partial page response is generally much smaller, since only the modified contents are sent back to the browser.

3) Partial Page Replacement

When the browser receives the partial page response, the new contents for each partial target node are copied from the hidden iframe into the main browser window, replacing the existing contents for each target node. For example, in the table navigation case, rather than replacing the entire page, just the contents of the table itself are replaced.

Now lets create a simple page to implement PPR. In this page, there are 3 items (employee number, employee name & job). I have attach a PPR event on employee number field. Hence, user will enter the emp number and as soon as he tabs out, name and job will populate on their respective fields without refreshing the page.

1) Create a new OA page:

==========

Right click on project (xxcus) --> New --> Web Tier --> OA Components --> select 'Page' item. Click OK. (This will open a popup window)

We are creating a page for implementing PPR, so specify the details of page as below:

Name: XxPprDemoPG

Package: xxcus.oracle.apps.fnd.pprdemo.webui

2) Create a new view objects (VO):

==========

Right click --> New View Object (This will open a wizard having 7 steps).

Step 1

Package: xxcus.oracle.apps.fnd.pprdemo.server

Name: XxPprDemoVO

Choose the radio button 'Read-only Access' (as no DML operation).

Click Next.

Step 2

Enter the below query in 'Query Statement':

```
select EMPNO, ENAME, JOB from employee
```

Keep defaults for step3, 4, 5, 6 & 7 and click Finish. Save All.

3) Create a new Application Module (AM):

==========

Step 1

Package: xxcus.oracle.apps.fnd.pprdemo.server

Name: XxPprDemoAM. Click Next.

Step 2

Here we will add an instance of the VO created in (2). Select XxPprDemoVO from 'Available View Objects' and shuttle it to 'Data Model' using ">" button.

Keep defaults for all other steps (3, 4). Click Finish.

4) Create a new Controller (CO):

==========

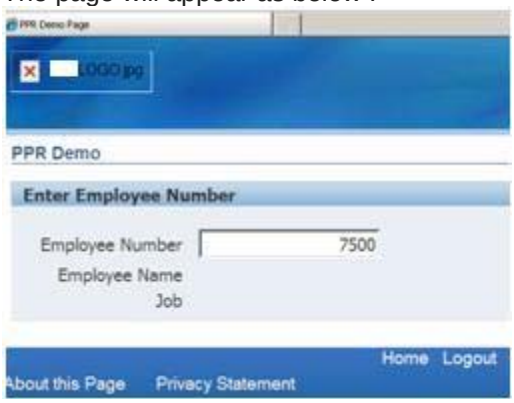
Right click on pageLayout region and create a new controller:

Name: XxPprDemoCO

package: xxcus.oracle.apps.fnd.pprdemo.webui

5) Creating the Page Layout & Setting its Properties :-

The page will appear as below :



1) Create a new region under pageLayout of type defaultSingleColumn.
2) Create 3 items under defaultSingleColumn region (messageTextInput, messageStyledText, messageStyledText).

Now change the properties for each field from property inspector as shown below:

pageLayout:
- ID: pageLayoutRN
- AM Definition: xxcus.oracle.apps.fnd.pprdemo.server.XxPprDemoAM
- Window Title: PPR Demo Page
- Title: PPR Demo

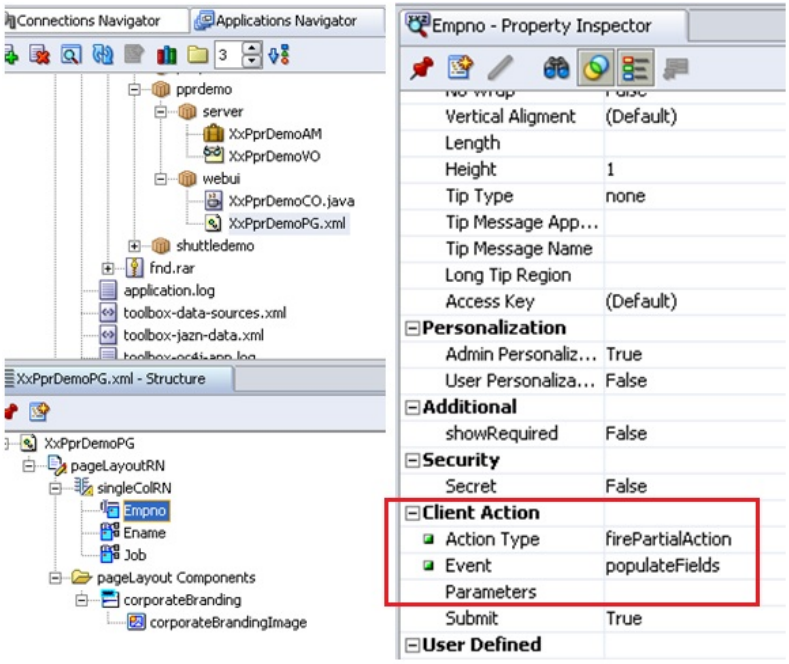
defaultSingleColumn:
- ID: singleColRN
- Text: Enter Employee Number

messageTextInput (applied applied PPR event on this item):
- ID: Empno
- Prompt: Employee Number
- **Action Type: firePartialAction**
- **Event: populateFields**

messageStyledText (for Employee Name):
- ID: Ename
- Prompt: Employee Name
- View Instance: XxPprDemoVO1
- View Attribute: Ename

messageStyledText (for Job):
- ID: Job
- Prompt: Job
- View Instance: XxPprDemoVO1
- View Attribute: Job

The declarative page structure in jDev will be similar to as shown below:



We will capture populateFields event (PPR event) in PFR method of controller and call the appropriate method from AMImpl class as shown in below code snippet:

```
1
2   public void processFormRequest(OAPageContext pageContext,
3                                  OAWebBean webBean) {
4    super.processFormRequest(pageContext, webBean);
5
6    OAApplicationModule am = pageContext.getRootApplicationModule();
7
8    if
    ("populateFields".equals(pageContext.getParameter(EVENT_PARAM))) {
9       String empNum = pageContext.getParameter("Empno");
10      if (!("".equals(empNum.trim()))) {
11         Serializable[] param = { empNum };
12         String result =
13           am.invokeMethod("firePprEvent", param).toString();
14           if ("N".equals(result)) {
15              throw new OAException("Please enter valid employee
    number.",
16                                    OAException.ERROR);
17           }
18      } else {
19          String empnum = null;
20          Serializable[] param = { empnum };
21          am.invokeMethod("firePprEvent", param);
22          throw new OAException("Please enter employee number",
23                                OAException.ERROR);
24      }
25   }
    }
```

Code for firePprEvent method in XxPprDemoAMImpl.java file:

```
1
2    public String firePprEvent(String empNum) {
3      try {
4          XxPprDemoVOImpl vo = getXxPprDemoVO1();
5          vo.setWhereClause(null);
6          vo.setWhereClauseParams(null);
7          vo.setWhereClause("EMPNO = :1");
8          vo.setWhereClauseParam(0, empNum);
9          vo.setMaxFetchSize(-1);
10         vo.executeQuery();
11         XxPprDemoVORowImpl row = (XxPprDemoVORowImpl)vo.first();
12         if (row == null) {
13             return "N";
14         }
15     } catch (Exception e) {
16         e.printStackTrace();
17     }
18     return "Y";
   }
```

User will enter employee number and tabs out. This will auto-populate the other two fields without full page refresh as shown below:

Method 3:

Programmatic PPR in OAF:

I was recently helping a friend in a customization in OAF, where through personalization, he wanted to put a ppr action in a seeded page in a seeded item. He was trying to do that customization using fire action, which was not acceptable by customer due to obvious reasons. Since, I think this section is missing in developers' guide, and is quite simple to approach, here is code you can write in process request of CO to attach programmatic PPR. lets take a simple example of attaching PPR to a message choice bean :

```
//In Process Request()
{
//Please attach PPR only to those UIX beans which support it
//otherwise it may not work
OAMessageChoiceBean mcb=(OAMessageChoiceBean)webBean.findChildRecursive("");
FireAction firePartialAction = new FirePartialAction("empPositionChange");
mcb.setAttributeValue(PRIMARY_CLIENT_ACTION_ATTR,firePartialAction);
}

//In process form request
if
("empPositionChange".equals(pageContext.getParameter(OAWebBeanConstants.EVENT_PA
RAM)))
{
//ur logic on PPR

//if PPR is attached in a table/hgrid child then we can find the row
//reference by
String rowReference
=pageContext.getParameter(OAWebBeanConstants.EVENT_SOURCE_ROW_REFERENCE);
}
```