



**ORACLE®**

## Empowering Multi-tasking with an ADF UI Powerhouse

Steven Davelaar    twitter: [@stevendavelaar](#)  
Technical Director    blog: [blogs.oracle.com/jheadstart](http://blogs.oracle.com/jheadstart)  
Oracle Consulting



# Agenda

- Multi-tasking
  - Options with ADF applications
  - Using UIShell with dynamic tabs
  - Additional usability requirements
- UIShell with dynamic tabs implementation
- Implementing additional requirements



## Multi-tasking options with ADF applications

- Start new browser instance
  - End user easily loses overview
- Open new browser tab within same browser instance
  - ADF tabs mixes with non-ADF tabs
  - ADF session shared accross tabs
- Open new tab within ADF application
  - Options to meet additional requirements



## Multi-tasking and related usability requirements (1)

- Open new tab in various ways
  - Using the menu
  - Using global quick search
  - From within other tabs
  - Conditionally: check whether tab is already open
- Close tab in various ways
  - Use close tab icon on tab label
  - Use close tab button inside tab region
  - Auto-close when task is completed



## Multi-tasking and related usability requirements (2)

- Transaction handling
  - Tabs are independent tasks -> independent transactions
  - Visual indicator that tab contains pending changes
  - Warning when closing tab with pending changes
- Miscellaneous
  - Update browser title based on selected tab
  - Initially displayed tabs
  - Prevent tabs from being closed manually
  - Set maximum number of open tabs
  - Update tab label based on current data inside tab

# Dynamic Tabs UI Shell Functional UI Pattern

ORACLE

Welcome Davelaar ( [Account](#) | [Help](#) | [Sign Out](#) ) United States ▾ Communities ▾ I am a... ▾ I want to... ▾

Products and Services

Downloads

Store

Support

Training

Partners

About

Oracle Technology Network > Developer Tools > Application Development Framework

JDeveloper

Application Testing Suite

SQL Developer

SQL Developer Data Modeler

Application Development Framework

Application Express

APEX Listener

Developer Suite

Developer Tools for Visual Studio

Discoverer

Enterprise Pack for Eclipse

JHeadstart

## Reference: Dynamic Tabs UI Shell Template Functional UI Pattern

This topic contains the following sections:

- [Overview](#)
- [Problem Description](#)
- [Technical Pattern Description](#)
- [The User Experience](#)
- [Pattern Implementation](#)
- [Known Issues](#)

### Overview

The Dynamic Tabs UI Shell Template (UI Shell) is used to address one of the necessary attributes of a software product: usability. Elements addressed are:

ORACLE

# UIShell – Dynamic Tabs with Tree Menu

The screenshot displays the Oracle Dynamic Tabs Demo application. The interface features a blue header with the text "ORACLE Dynamic Tabs Demo" and the Oracle logo. Below the header, there are three tabs: "Home", "Maintain Jobs", and "Search Employees". The "Maintain Jobs" tab is currently active. On the left side, there is a tree menu with a "View" dropdown. The tree menu is expanded to show "Human Resources" with sub-items: "Search Employees", "Maintain Jobs", "View Jobs", and "Departments". The main content area of the "Maintain Jobs" tab is titled "Edit Job" and contains a form with the following fields:

* JobId	<input type="text" value="FI_MGR"/>	MinSalary	<input type="text" value="8200"/>
* JobTitle	<input type="text" value="Finance Manager"/>	MaxSalary	<input type="text" value="16000"/>

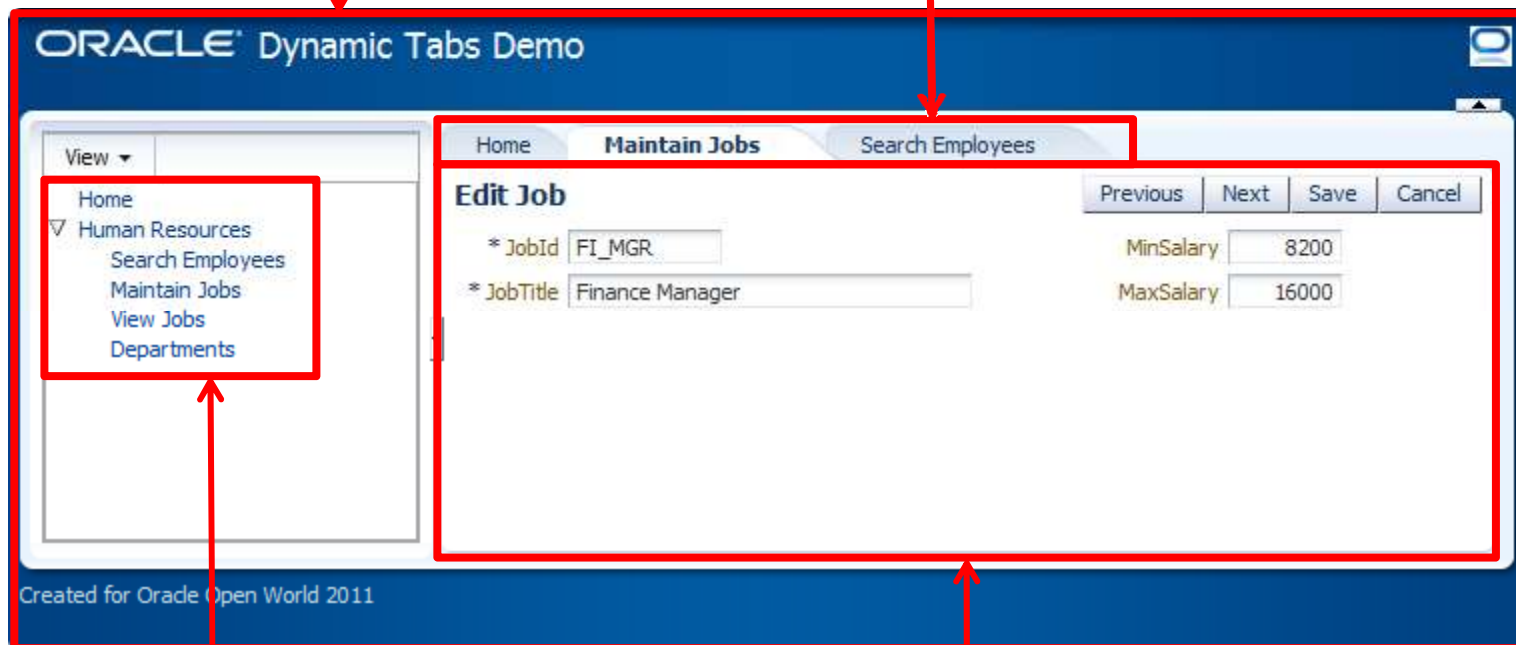
At the top right of the form, there are four buttons: "Previous", "Next", "Save", and "Cancel".

Created for Oracle Open World 2011

# Dynamic Tabs with Tree Menu - Implementation

UI Shell – Page Template

Dynamic Tabs



XMLMenuModel Tree

ADF Region





## UIShell Page

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http:
<f:view xmlns:f="http://java.sun.com/jsf/core"
        xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
  <af:document title="UIShell" id="doc0">
    <af:form id="dataForm">
      <af:pageTemplate id="pt"
        viewId="/common/pageTemplates/DynamicTabsPageTemplate.jsf">
        <f:attribute name="menuModel" value="#{menuModel}"/>
      </af:pageTemplate>
    </af:form>
  </af:document>
</f:view>
```



## Tree Menu – XMLMenuModel

```
<?xml version="1.0" encoding="UTF-8" ?>
<menu xmlns="http://myfaces.apache.org/trinidad/menu">
  <itemNode id="homeMI" label="Home" action="uishell:Home"
    focusViewId=""/>
  <groupNode id="hr" idref="empMI" label="Human Resources">
    <itemNode id="empMI" label="Search Employees"
      action="uishell:SearchEmployees" focusViewId=""/>
    <itemNode id="jobMI" label="Maintain Jobs"
      action="uishell:Jobs" focusViewId=""/>
    <itemNode id="jobMI" label="View Jobs"
      action="uishell:ViewJobs" focusViewId=""/>
    <itemNode id="depMI" label="Departments"
      action="uishell:Departments" focusViewId=""/>
  </groupNode>
</menu>
```



## Tree Menu – MenuModel Managed Bean

```
<managed-bean id="__3">
  <managed-bean-name>menuModel</managed-bean-name>
  <managed-bean-class>org.apache.myfaces.trinidad.model.XMLMenuModel
</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
  <managed-property id="__4">
    <property-name>source</property-name>
    <value>/WEB-INF/menu.xml</value>
  </managed-property>
</managed-bean>
```

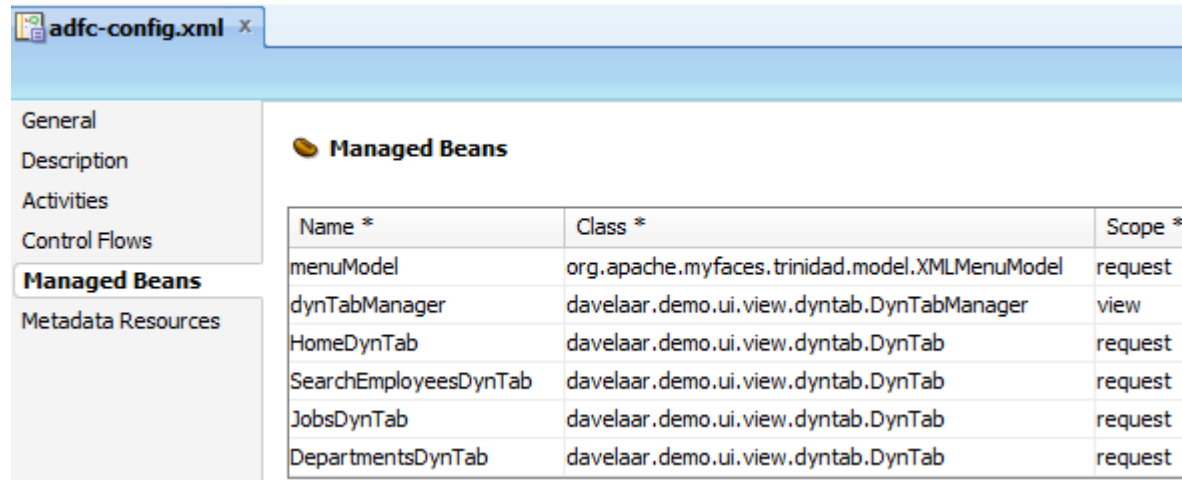
## Dynamic Tabs Page Template – Tree Menu

```
<af:panelCollection id="pctree">
  <af:tree id="Menu1" var="menuItem" contentDelivery="immediate"
    value="#{attrs.menuModel}" summary="Menu Item">
    <f:facet name="nodeStamp">
      <af:group>
        <af:commandLink id="Item1" action="#{menuItem.doAction}"
          textAndAccessKey="#{menuItem.label}"
          partialSubmit="true"
          immediate="true"
          rendered="#{menuItem.focusViewId!=null an
        <af:outputText id="Item2" value="#{menuItem.label}"
          rendered="#{menuItem.focusViewId==null and
      </af:group>
    </f:facet>
  </af:tree>
</af:panelCollection>
```

I

## Adding and Selecting Dynamic Tabs

- Create DynTab class and managed beans
- Create DynTabManager class and managed bean
- Create custom TabsNavigationHandler
  - Hides complexity of adding/selecting tabs



The screenshot shows an IDE window titled 'adfc-config.xml'. On the left, a sidebar contains navigation options: General, Description, Activities, Control Flows, **Managed Beans**, and Metadata Resources. The main area displays the 'Managed Beans' configuration as a table with three columns: Name \*, Class \*, and Scope \*. The table lists several beans, including menuModel, dynTabManager, and several DynTab instances.

Name *	Class *	Scope *
menuModel	org.apache.myfaces.trinidad.model.XMLMenuModel	request
dynTabManager	davelaar.demo.ui.view.dyntab.DynTabManager	view
HomeDynTab	davelaar.demo.ui.view.dyntab.DynTab	request
SearchEmployeesDynTab	davelaar.demo.ui.view.dyntab.DynTab	request
JobsDynTab	davelaar.demo.ui.view.dyntab.DynTab	request
DepartmentsDynTab	davelaar.demo.ui.view.dyntab.DynTab	request



## Create DynTab class and managed beans

- DynTab class holds all info about a dynamic tab
  - Tab title
  - Tab unique identifier
  - Task flow ID
  - Task flow parameters (optional)
- For each menu item, a managed bean using DynTab class is defined



## Maintain Jobs DynTab managed bean

```
<managed-bean id="__13">
  <managed-bean-name>JobsDynTab</managed-bean-name>
  <managed-bean-class>davelaar.demo.ui.view.dyntab.DynTab</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
  <managed-property>
    <property-name>title</property-name>
    <value>Maintain Jobs</value>
  </managed-property>
  <managed-property>
    <property-name>uniqueIdentifier</property-name>
    <value>Jobs</value>
  </managed-property>
  <managed-property>
    <property-name>taskFlowIdString</property-name>
    <value>/WEB-INF/jobs-tf.xml#jobs</value>
  </managed-property>
</managed-bean>
```



## DynTabManager class and managed bean

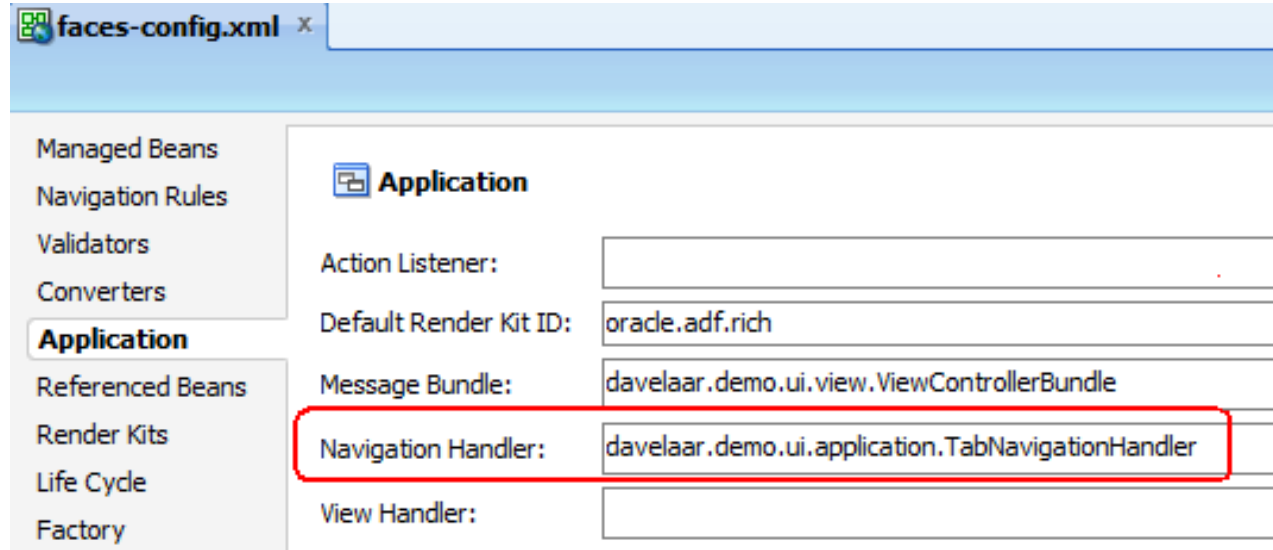
- DynTabManager class controls tab behavior
  - Housekeeping of open tabs and current tab
  - Launch tab: add new or select existing tab
  - Remove tab
  - Mark current tab dirty

```
<managed-bean id="__49">
  <managed-bean-name>dynTabManager</managed-bean-name>
  <managed-bean-class>davelaar.demo.ui.view.dyntab.DynTabManager
</managed-bean-class>
  <managed-bean-scope>view</managed-bean-scope>
  <managed-property>
    <property-name>maxNumberOfTabs</property-name>
    <property-class>java.lang.Integer</property-class>
    <value>10</value>
  </managed-property>
</managed-bean>
```



# Navigating Using Dynamic Tabs

- Create custom TabNavigationHandler
  - configure in faces-config.xml
  - Provides standard JSF navigation through superclass
  - Allows setting adding/selecting dynamic tab: tab name specified after navigation outcome, separated by colon



The screenshot shows the configuration interface for a JSF application. The left sidebar lists various configuration categories, with 'Application' selected. The main area displays the configuration for the 'Application' component, including fields for Action Listener, Default Render Kit ID, Message Bundle, Navigation Handler, and View Handler. The 'Navigation Handler' field is highlighted with a red box and contains the value 'davelaar.demo.ui.application.TabNavigationHandler'.

Managed Beans	
Navigation Rules	
Validators	
Converters	
<b>Application</b>	<b>Application</b>
Referenced Beans	
Render Kits	
Life Cycle	
Factory	

Action Listener:	
Default Render Kit ID:	oracle.adf.rich
Message Bundle:	davelaar.demo.ui.view.ViewControllerBundle
Navigation Handler:	davelaar.demo.ui.application.TabNavigationHandler
View Handler:	



## TabNavigationHandler

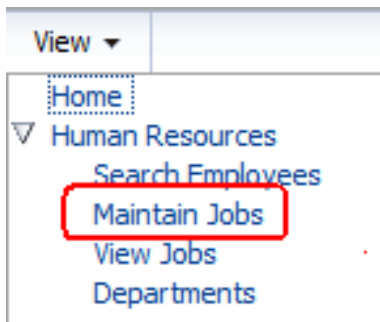
```
public void handleNavigation(FacesContext facesContext, String action, String outcome)
{
    sLog.info("handleNavigation action=" + action + ", outcome=" + outcome);
    if (outcome != null && outcome.indexOf(":")>-1)
    {
        int pos = outcome.indexOf(":");
        String shellAction = outcome.substring(0, pos);
        String tabName = outcome.substring(outcome.indexOf(":") + 1);
        sLog.info("Navigating to " + shellAction + ", opening dynamic tab " + tabName);

        // launch dyn tab
        DynTabManager.getCurrentInstance().launchTab(tabName);

        // navigate to uishell page if needed (usually not needed because only
        // page is UIShell page)
        super.handleNavigation(facesContext, action, shellAction);
    }
    else
    {
        super.handleNavigation(facesContext, action, outcome);
    }
}
```

## Navigating Using Dynamic Tabs

- Action "uishell:Jobs"
  - navigates to UShell.jsf page (if needed)
  - Call launchTab with tabName "Jobs" on DynTabManager
  - TaskFlowId and params picked up from JobsDynTab bean



```
<itemNode id="jobMI" label="Maintain Jobs"
  action="uishell:Jobs" focusViewId="" />
```

```
<af:tree id="Menu1" var="menuItem" contentDelivery="immediate" v
  summary="Menu Item">
  <f:facet name="nodeStamp">
    <af:group>
      <af:commandLink id="Item1" action="#{menuItem.doAction}"
        textAndAccessKey="#{menuItem.label}" parti
        immediate="true"
        rendered="#{menuItem.focusViewId!=null anc
```



## DynTabManager – Launch tab

```
public void launchTab(String tabName)
{
    DynTab tab = DynTab.getInstance(tabName);
    addOrSelectTab(tab);
}

public static DynTab getInstance(String tabName)
{
    String beanName = tabName + "DynTab";
    String expr = "#{ " + beanName + " }";
    DynTab tab = (DynTab) JsfUtils.getExpressionValue(expr);
    return tab;
}

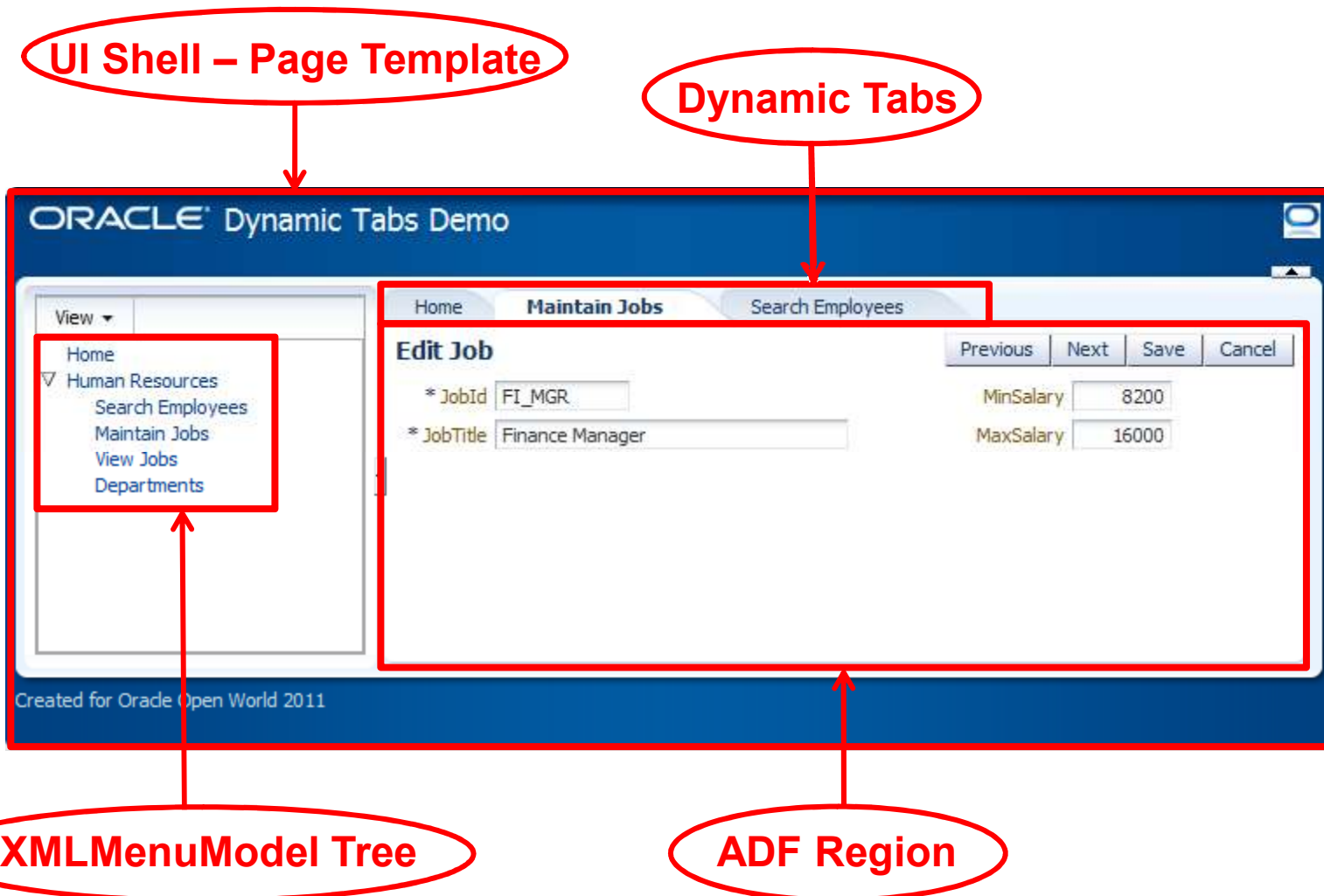
public void addOrSelectTab(DynTab tab)
{
    DynTab existingTab = getMatchingTab(tab);
    if (existingTab != null)
        setSelectedTab(existingTab);
    else
        addTab(tab);
}
```



## DynTabManager – Add tab

```
public void addTab(DynTab tab)
{
    // Assign unique id that we use to link the DynTab with the
    // corresponding Taskflow
    tab.setId(createId());
    // add the new tab to the tab list and map
    tabList.add(tab);
    tabMap.put(tab.getId(), tab);
    setSelectedTabId(tab.getId());
    // Create taskflow binding attrs required by ADFm multiTaskFlow binding
    TaskFlowBindingAttributes tfAttr = new TaskFlowBindingAttributes();
    tfAttr.setId(tab.getId());
    tfAttr.setTaskFlowId(tab.getTaskFlowId());
    tfAttr.setRefreshCondition("ifNeeded");
    tfAttr.setParametersMap(tab.getParameterMapExpression());
    taskFlowBindingAttrsList.add(tfAttr);
    refreshTabsAndContentArea();
}
```

# Dynamic Tabs with Tree Menu - Implementation



## Dynamic Tabs Page Template – Dynamic Tabs

```
<af:panelSplitter orientation="horizontal" splitterPosition="200" id="pt_ps2">
  <f:facet name="first">
    <af:decorativeBox theme="default" styleClass="TabletPageContent" id="pt_db2">
      <f:facet name="center">
        <af:panelCollection id="pctree">
          <af:tree id="Menu1" ... /f:facet> </af:tree>
        </af:panelCollection>
      </f:facet>
    </af:decorativeBox>
  </f:facet>
  <f:facet name="second">
    <af:declarativeComponent viewId="/common/declarativeComponents/DynamicTabs.jsff"
      id="dyntdc"/>
  </f:facet>
</af:panelSplitter>
```



# Dynamic Tabs Declarative Component

```
<af:panelTabbed id="ptb0" tabRemoval="all"
    binding="#{viewScope.dynTabManager.tabsNavigationPane}">
  <af:forEach var="tab" varStatus="vs"
    items="#{viewScope.dynTabManager.tabList}">
    <af:showDetailItem stretchChildren="first" id="sdi0"
      textAndAccessKey="#{tab.title}"
      disclosedTransient="true"
      remove="#{tab.closeable ? 'inherit' : 'no'}"
      disclosed="#{tab.id==viewScope.dynTabManager.selectedTabId}"
      disclosureListener="#{viewScope.dynTabManager.selectTab}"
      itemListener="#{viewScope.dynTabManager.removeTab}">
      <af:region value="#{bindings.multiRegion1.taskFlowBindings[tab.id].regionModel}"
        id="r1"/>
      <f:attribute name="tabId" value="#{tab.id}"/>
    </af:showDetailItem>
  </af:forEach>
</af:panelTabbed>
```



# UIShell Page Definition – MultiTaskFlow binding new in Jdev 11.1.2

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
                id="UIShellDTPageDef"
                Package="davelaar.demo.ui.view.pageDefs">
  <parameters/>
  <executables>
    <variableIterator id="variables"/>
    <multiTaskFlow id="multiRegion1"
                  taskFlowList="{viewScope.dynTabManager.taskFlowList}"
                  activation="deferred" Refresh="ifNeeded"
                  xmlns="http://xmlns.oracle.com/adf/controller/binding"/>
  </executables>
</pageDefinition>
```



## JDev 11.1.1.x Implementation

- No MultiRegion executable
- Cannot use af:region inside af:forEach tag
- Page Template Page Definition with 15 pre-defined taskflow bindings
- DynamicTabs declarative component defines 15 regions, only one rendered at a time

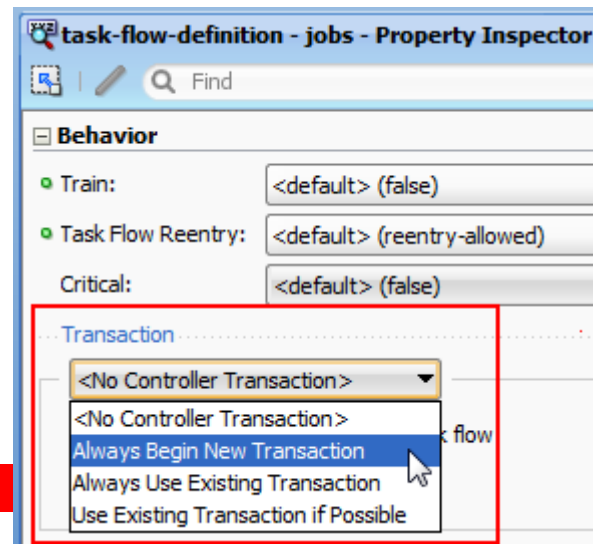
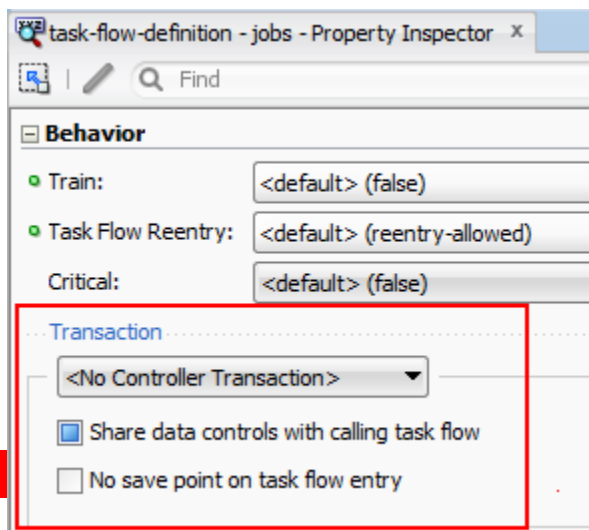


## Implementing some Additional Requirements

- Tabs should have independent transactions
- Pending changes warning when closing tab
- Opening new tab from within other tab
- Opening new tab from global search
- Auto-closing a tab
- Initially displayed tabs
- Update browser window title

# Data Control Scope and Transactions

- Data Control Scope
  - Shared: all TF's share same app module instance
  - Isolated: Each TF instance has own app module instance
- Transaction Scope
  - With isolated data control scope each TF has own transaction
  - With shared data control scope, transaction is shared by default, but can be changed using Transaction setting





## Marking a Tab Dirty – Pending Changes Alert

- Pass DynTabManager as task flow parameter
- Create custom page controller class and implement refreshRegion method
- get the data control through binding container, check for changes and mark current tab dirty using DynTabManager
- Configure custom page controller class in page definition
- Show dialog when closing dirty tab
- Show dirty tab label in italics

## Pass DynTabManager as Task Flow Parameter

```
<input-parameter-definition id="__6">
  <name>dynTabManager</name>
  <value>#{pageFlowScope.dynTabManager}</value>
  <class>davelaar.demo.ui.view.dyntab.DynTabManager</class>
</input-parameter-definition>
```

```
public void addTab(DynTab tab)
{
    // Assign unique id that we use to link the DynTab with the
    // corresponding Taskflow
    tab.setId(createId());
    // add the DynTabManager instance as taskflow parameter so the
    // taskflow itself can open new tabs, and close tabs.
    tab.getParameters().put("dynTabManager", this);
}
```

# Custom Page Controller Class

```
public class TabRegionController
    implements RegionController
{
    @Override
    public boolean refreshRegion(RegionContext rc)
    {
        int refreshFlag = rc.getRefreshFlag();
        rc.getRegionBinding().refresh(refreshFlag);
        setTabDirtyState(rc);
        return false;
    }
}
```

---

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
    version="11.1.2.60.17" id="EditJobPageDef"
    Package="davelaar.demo.ui.view.pageDefs"
    ControllerClass="davelaar.demo.ui.controller.TabRegionController">
<parameters/>
```

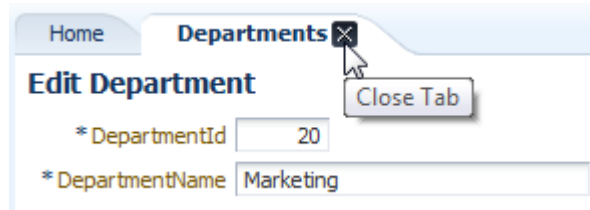


## Custom Page Controller Class – setTabDirtyState

```
protected void setTabDirtyState(RegionContext rc)
{
    DynTabManager tabManager = DynTabManager.getCurrentInstance();
    if (tabManager!=null)
    {
        DCBindingContainer cont = (DCBindingContainer) rc.getRegionBinding();
        if (cont.getIterBindingList().size()>0)
        {
            DCIteratorBinding ib = null;
            for (Object iterBinding : cont.getIterBindingList())
            {
                DCIteratorBinding currentIb = (DCIteratorBinding)iterBinding;
                if (currentIb.getDataControl() != null)
                {
                    ib = currentIb;
                    break;
                }
            }
            if (ib != null && ib.isRefreshed())
            {
                boolean changes = ib.getDataControl().isTransactionDirty()
                    || ib.getDataControl().isTransactionModified();
                tabManager.markCurrentTabDirty(changes);
            }
        }
    }
}
```



# Alert for Pending Changes

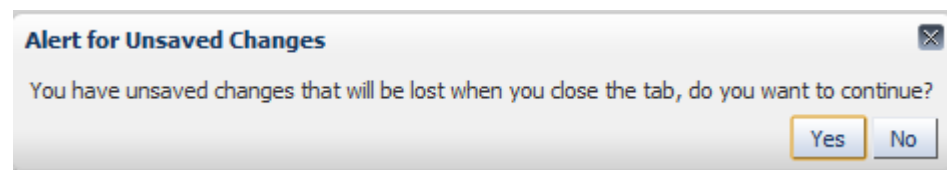


Home Departments Close Tab

**Edit Department**

\* DepartmentId

\* DepartmentName



**Alert for Unsaved Changes**

You have unsaved changes that will be lost when you close the tab, do you want to continue?

```
<af:showDetailItem stretchChildren="first" id="sdi0"
  textAndAccessKey="#{tab.title}"
  disclosedTransient="true"
  remove="#{tab.closeable ? 'inherit' : 'no'}"
  disclosed="#{tab.id==viewScope.dynTabManager.selectedTabId}"
  disclosureListener="#{viewScope.dynTabManager.selectTab}"
  itemListener="#{viewScope.dynTabManager.removeTab}">
  <af:region value="#{bindings.multiRegion1.taskFlowBindings[tab.id]}
    id="r1"/>
  <f:attribute name="tabId" value="#{tab.id}"/>
</af:showDetailItem>
```

## DynTabManager Remove Tab

```
public void removeTab(ItemEvent itemEvent)
{
    UIComponent component = itemEvent.getComponent();
    String tabId = String.valueOf(component.getAttributes().get("tabId"));
    DynTab tab = getTab(tabId);
    if (tab.isDirty())
    {
        RichClientUtils.getInstance().showPopup(getTabDirtyPopup(), null);
        return;
    }
}
```

```
<af:popup id="pt_tabdirty" clientComponent="true" contentDelivery="lazy"
          binding="#{viewScope.dynTabManager.tabDirtyPopup}">
  <af:dialog title="Alert for Unsaved Changes" type="yesNo" id="pt_d2"
            dialogListener="#{viewScope.dynTabManager.handleDirtyTabDialog}">
    <af:outputText value="You have unsaved changes that will be lost when
                      you close the tab, do you want to continue?"
                  id="pt_ot7"/>
  </af:dialog>
</af:popup>
```



## Showing dirty tab label in italics

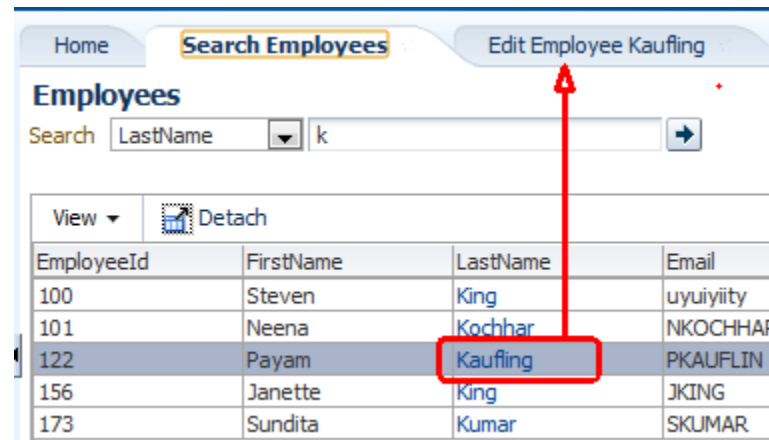
- Currently not possible to change styling of an individual tab within panelTabbed component (ER logged)
- Work around
  - Use navigation pane to render tabs
  - Use separate af:forEach loop to render the regions

## Showing dirty tab label in italics

```
<af:navigationPane id="dynTabsPane"
  binding="#{viewScope.dynTabManager.tabsNavigationPane}"
  rendered="#{not empty viewScope.dynTabManager.selectedTabId}"
  value="#{viewScope.dynTabManager.tabMenuModel}" var="tab" itemRemoval="all">
  <f:facet name="nodeStamp">
    <af:commandNavigationItem id="tabIndex"
      actionListener="#{viewScope.dynTabManager.tabActivatedEvent}"
      inlineStyle="#{tab.dirty ? 'font-style: italic' : ''}"
      partialSubmit="true"
      textAndAccessKey="#{tab.title}" immediate="true"
      remove="#{tab.closeable ? 'inherit' : 'no'}"
      itemListener="#{viewScope.dynTabManager.removeTab}">
      <f:attribute name="tabId" value="#{tab.id}"/>
    </af:commandNavigationItem>
  </f:facet>
</af:navigationPane>

<af:forEach var="tf" varStatus="vs" items="#{bindings.multiRegion1.taskFlowBindingList}">
  <af:region value="#{tf.regionModel}" id="reg"
    rendered="#{tf.name==viewScope.dynTabManager.selectedTabId}"/>
</af:forEach>
```

## Opening a Tab From Within Other Tab



The screenshot shows an Oracle APEX application interface. At the top, there are three tabs: 'Home', 'Search Employees', and 'Edit Employee Kaufing'. The 'Search Employees' tab is active, and a search query for 'k' is entered. Below the search bar, there is a table of employees. The table has columns for EmployeeId, FirstName, LastName, and Email. The row for EmployeeId 122, Payam Kaufing, is highlighted. A red box is drawn around the 'Kaufing' text in the LastName column, and a red arrow points from this box to the 'Edit Employee Kaufing' tab header.

EmployeeId	FirstName	LastName	Email
100	Steven	King	uyuiyiity
101	Neena	Kochhar	NKOCHHAR
122	Payam	Kaufing	PKAUFLIN
156	Janette	King	JKING
173	Sundita	Kumar	SKUMAR

- Set up EditEmployee task flow for deeplinking
- Pass TabManager instance as task flow parameter
- Define Edit EmployeeDynTab managed bean inside “Search Employees” task flow
- Set commandLink action to “uishell:EditEmployee”
- Use setActionListener on commandLink to pass parameters

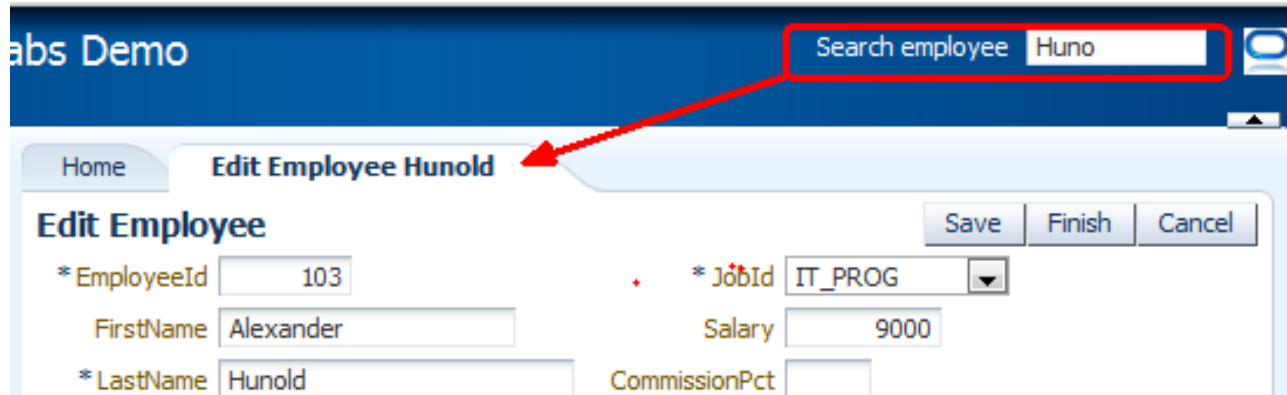


## Opening a Tab From Within Other Tab

```
<af:commandLink id="ot3" text="#{row.LastName}" action="uishell:EditEmployee">
  <af:setActionListener to="#{EditEmployeeDynTab.title}"
    from="#{af:formatString('Edit Employee {0}',row.LastName)}" />
  <af:setActionListener to="#{EditEmployeeDynTab.uniqueIdentifier}"
    from="#{row.EmployeeId}" />
  <af:setActionListener to="#{EditEmployeeDynTab.parameters['rowKeyValue']}"
    from="#{row.EmployeeId}" />
</af:commandLink>
```

```
<managed-bean id="__8">
  <managed-bean-name>EditEmployeeDynTab</managed-bean-name>
  <managed-bean-class>davelaar.demo.ui.view.dynTab.DynTab</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
  <managed-property>
    <property-name>taskIdString</property-name>
    <value>/WEB-INF/edit-employees-tf.xml#edit-employees</value>
  </managed-property>
</managed-bean>
```

## Opening a Tab From Global Search



The screenshot shows a web application interface with a blue header bar. On the right side of the header, there is a search bar labeled "Search employee" containing the text "Huno". A red box highlights this search bar, and a red arrow points from it to a tab labeled "Edit Employee Hunold". The main content area has a "Home" tab and the "Edit Employee Hunold" tab. Below the tabs, there is a form titled "Edit Employee" with fields for EmployeeId (103), JobId (IT\_PROG), FirstName (Alexander), Salary (9000), and LastName (Hunold). There are also buttons for Save, Finish, and Cancel.

- Set up EditEmployee task flow for quick search query
- Set commandLink action to “uishell:EditEmployee”
- Use setActionListener on commandLink to pass parameters
- Use subform and default command to auto submit when tabbing out search field

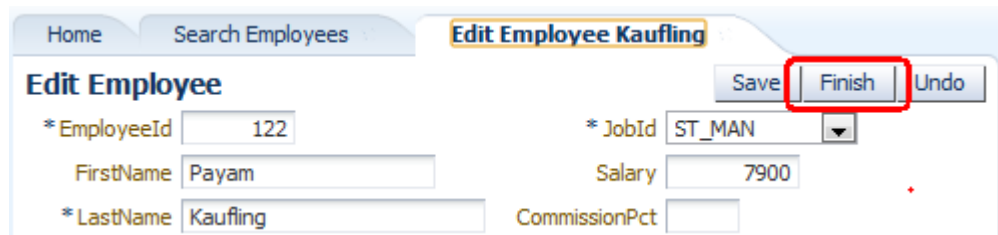
## Opening a Tab From Global Search

```
<af:subform id="pt_s1" defaultCommand="cb3">
  <af:panelGroupLayout id="pt_statWrp" layout="horizontal"
    halign="end" styleClass="AFBrandingBarItem">
    <f:facet name="separator">
      <af:spacer width="5" id="pt_s2"/>
    </f:facet>
    <af:inputText value="#{pageFlowScope.globalSearchString}"
      columns="10" label="Search employee" id="pt_it1"/>
    <af:commandButton id="cb3" text="Go" action="uishell:EditEmployee"
      visible="false">
      <af:setActionListener to="#{EditEmployeeDynTab.parameters['searchString']}"
        from="#{pageFlowScope.globalSearchString}"/>
      <af:setActionListener to="#{EditEmployeeDynTab.uniqueIdentifier}"
        from="#{pageFlowScope.globalSearchString}"/>
    </af:commandButton>
    <af:statusIndicator id="pt_statInd"/>
  </af:panelGroupLayout>
</af:subform>
```

I



## Auto-closing a tab



The screenshot shows a web application interface with three tabs: 'Home', 'Search Employees', and 'Edit Employee Kaufling'. The 'Edit Employee' form contains the following fields and buttons:

- \* EmployeeId: 122
- \* JobId: ST\_MAN (dropdown menu)
- FirstName: Payam
- Salary: 7900
- \* LastName: Kaufling
- CommissionPct: (empty field)
- Buttons: Save, Finish (highlighted with a red box), Undo

```
<af:commandButton actionListener="#{bindings.Commit.execute}"  
  text="Finish" id="cb5"  
  action="#{pageFlowScope.dynTabManager.removeCurrentTabForce}"/>
```

## Initially Displayed Tabs

```
<managed-bean id="__49">
  <managed-bean-name id="__50">dynTabManager</managed-bean-name>
  <managed-bean-class id="__51">davelaar.demo.ui.view.dyntab.DynTabManager</managed-bean-class>
  <managed-bean-scope id="__52">view</managed-bean-scope>
  <managed-property id="__66">
    <property-name id="__67">initialTabs</property-name>
    <property-class>java.util.List</property-class>
    <list-entries>
      <value>#{viewScope.HomeDynTab}</value>
    </list-entries>
  </managed-property>
</managed-bean>
```

```
@PostConstruct
public void init()
{
    // add tabs that should be initially displayed
    for (DynTab tab: initialTabs)
    {
        addTab(tab);
    }
}
```



## Update browser window/tab title

- Add method to DynTabManager, called when selecting another tab.

```
public void updateDocumentTitle()
{
    if (isDoUpdateDocumentTitle())
    {
        String js = "AdfPage.PAGE.findComponent('" + getDocumentId()
            + "').setTitle('" + getDocumentTitle() + "')";
        RichClientUtils.getInstance().writeJavaScriptToClient(js);
    }
}
.....
public String getDocumentTitle()
{
    String tabTitle = getSelectedTab() != null ?
        getSelectedTab().getTitle() : "";
    return getDocumentTitlePrefix() + tabTitle;
}
```



## Summary

- Use dynamic tabs with UIShell to support multi-tasking
- Set up infrastructure classes and beans to use ADF regions as dynamic tabs in UIShell
- Create custom tabs navigation handler to hide complexity for developers
- Think carefully about data control scope and transaction settings
- Create custom page controller for marking tab dirty
- Pass TabManager instance as task flow parameter for “inside-out” tab handling



## Next steps

- Download sample applications for JDev 11.1.2 and 11.1.1.4 from JHeadstart blog
- Follow-on session
  - Building Highly Reuseable ADF Task Flows  
Wednesday 10.15-11.15  
Mariott Marquis Golden Gate A

**SOFTWARE. HARDWARE. COMPLETE.**