ADF Code Corner

048-How-to build XML Menu Model based site menus and how to protect them with ADF Security and JAAS

twitter.com/adfcodecorner

Abstract:

There are different types of menus you can use within an application: breadcrumbs, to navigate a process within unbounded page flows, train stops for the same in bounded sub flows, context menus to operate actions on a specific component, hierarchical structure menus to access tasks and their sub tasks tabs to switch between different views for a current work context and site menus, just to name a few. This blog entry explains how to build site menus to navigate within the page hierarchy of a larger application and how to protect menu items and groups with ADF Security. Navigation menus mirror the user navigation path within an application and should be protected by the same policies that are defined for the pages to implement consistent security. This blog article explains two topics in one: How to build manageable site menus in ADF using the XML MenuModel based on unbounded task flow definitions and how to protect menu items with ADF Security and JAAS based authorization.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
13-MAY-2010

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

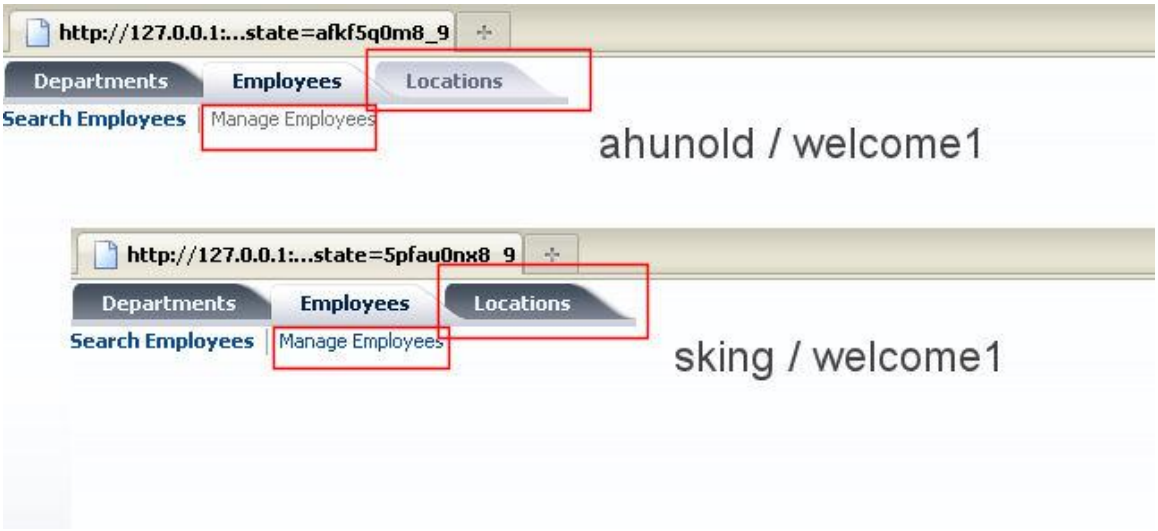
Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

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

Applications are best designed so the troika of usability, performance and security stays in balance. As demonstrated in this article, sometimes, more metadata can mean more security without causing a negative impact on usability and performance.

Topics covered in this article invlude the XML Menu Model, as the data source for the site menu, and application security with ADF Security. The good about using the XML Menu Models is that you can extend the site menu without changing any page sources. The image below shows the runtime UI of the example you can download at the end of this article.

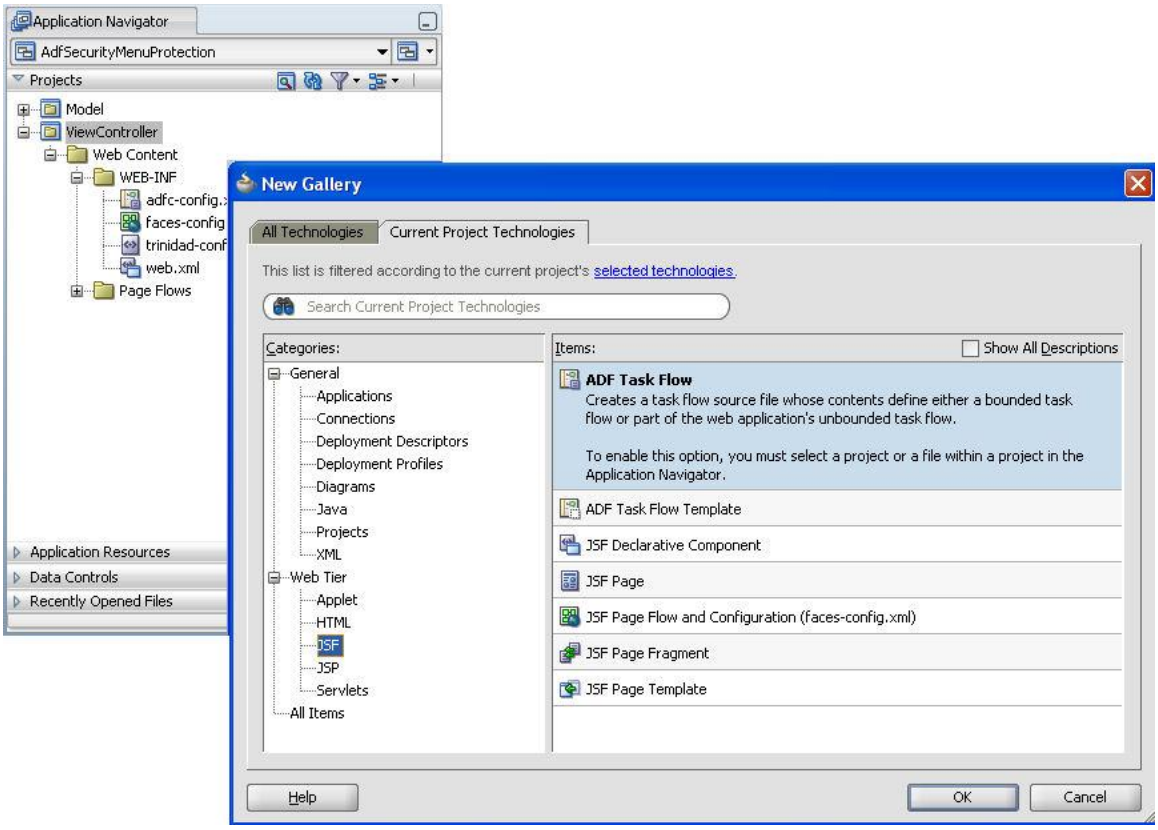


In the image above, the user "ahunold" has no access to the Locations site and the Manage Employees sub view, while "sking" has it all. Of course in a production environment its best to not at all render options that the current user is not authorized for.

Creating the XML Menu Model

The XML Menu Model originates from the MyFaces Trinidad JSF component set and uses a managed bean and a metadata file to describe the site menu structure. Using the ADF Controller, ADF developers can visually build the site hierarchy using one or many unbounded task flow definitions. Each task flow definition uses view activities and wild card navigation cases to represent the menu hierarchy levels. In the example for this article, a simple two level hierarchy is used in which the first level is defined by the "Departments", "Employees" and "Locations" categories. The sub-menu options are defined as "Manage Departments", "Search Departments", "Manage Employees", "Search Employees", "Manage Locations" and "Search Locations". For example, clicking the "Departments" top level navigation item (a tab in the image above) shows the sub level menu option "Manage Departments", "Search Departments" with (optionally) one of them selected by default.

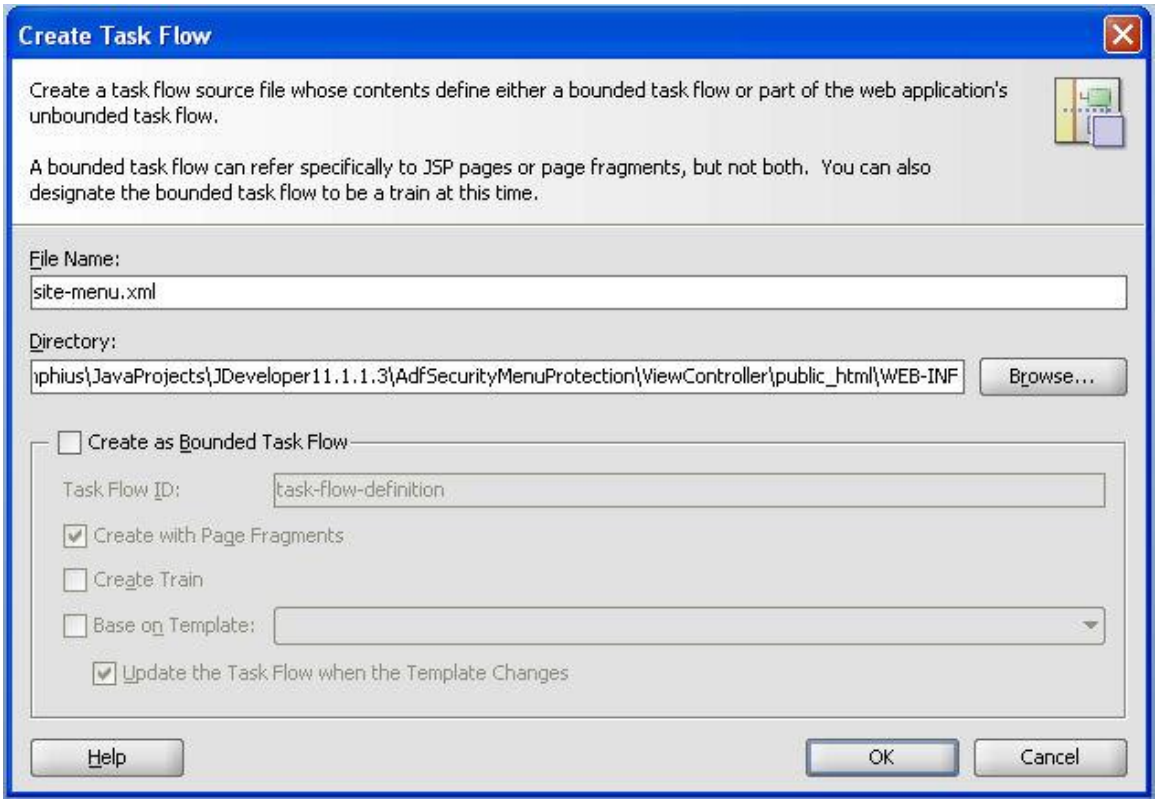
At the start of the development, create a new unbounded task flow from the Oracle JDeveloper New Gallery. The starting point of creating a new task flow definition is the same for unbounded and bounded task flows. So just choose the ADF Task Flow entry in the JSF category as shown in the image below.



By default, the Create as Bounded Task Flow check box option is selected. For the site menu model, we need an unbounded task flow definition (or two, three, four, five ...) and the check box must be explictly de-elected in this dialog. The name of the unbounded task flow is up to you. In the image below its created with the name site-menu.xml. Best practices is to name the XML definition file after the part of the menu hierarchy it covers.

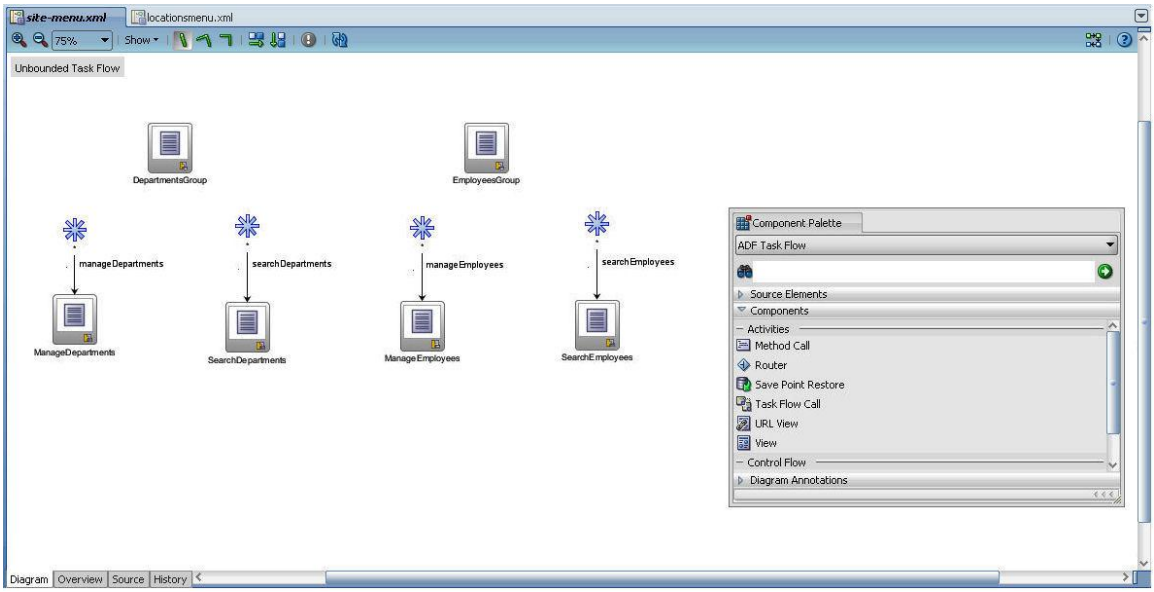
But wait! Didn't you read or hear that at runtime there's only a single instance of unbounded task flow? So how does it work then with the creation of multiple unbounded task flows ? Well, yes, you heard it right! However, while it is true that ADF Controller only uses a single instance of unbounded task flow, it does not mean that this instance must be defined in a single XML metadata configuration.When you create additional unbounded task flow definitions, have a look at adfc-config.xml, the default unbounded task flow, and you'll see that it contains a reference to the additional configuration files.

Note: I don't recommend to use the adfc-config.xml file for the menu creation, though it would work. The reason for my standpoint here is that I prefer things to be separated by duty.



As shown below, in the opened unbounded task flow diagram - site-menu.xml in this example - you create a hierarchy of view activities that may be bound to a physical page later on.

There are two things to point out about the content in the diagram: First there are view activities with navigation cases and some without. Second, navigation cases use wild cards and thus are accessible from all over the application (which is a requirement for activities you want to access from a site menu). The activities without navigation cases - at least in this example - are planned to become menu groups. A bit confusing is that Oracle JDeveloper automatically creates wild card navigations for all view activites that have no navigation case defined when generating the menu model. However, navigation cases are not used by menu groups and therefore I took the liberty to remove them. All task flow items - view activities, wild cards and navigation cases - are accessible from the JDeveloper component palette.



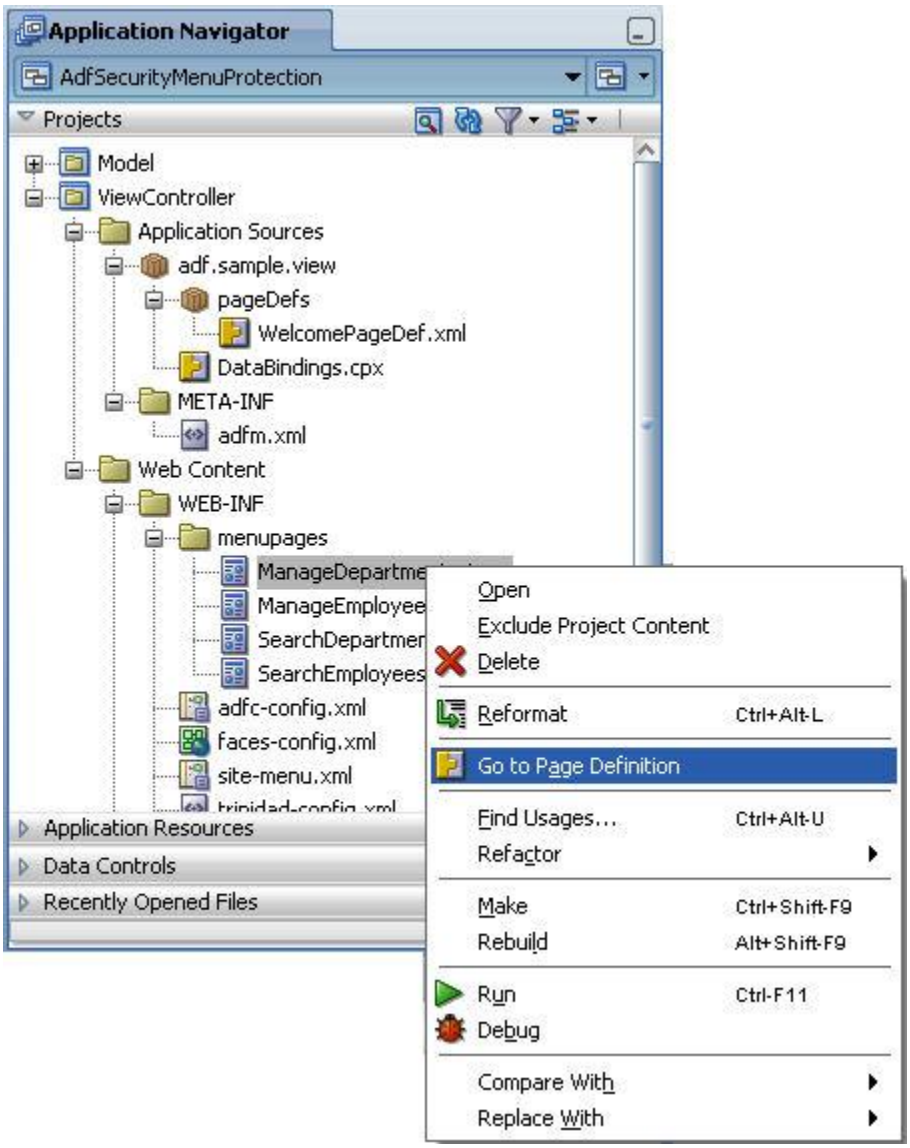
Note: The view activities with an associated wild card navigation case are those that within an application contain UI components for the user to work with. So it is not that the view activities are only used to build the menu. No, no - they are already part of the application to build.

Note: A menu item can also point to a bounded task flow using a task flow call activity in the diagram. However, the task flow must use JSPX documents for this, which excludes direct navigation to ADF Regions. If you want to navigate to an ADF Region, then add the region in a JSPX document and link to this document.

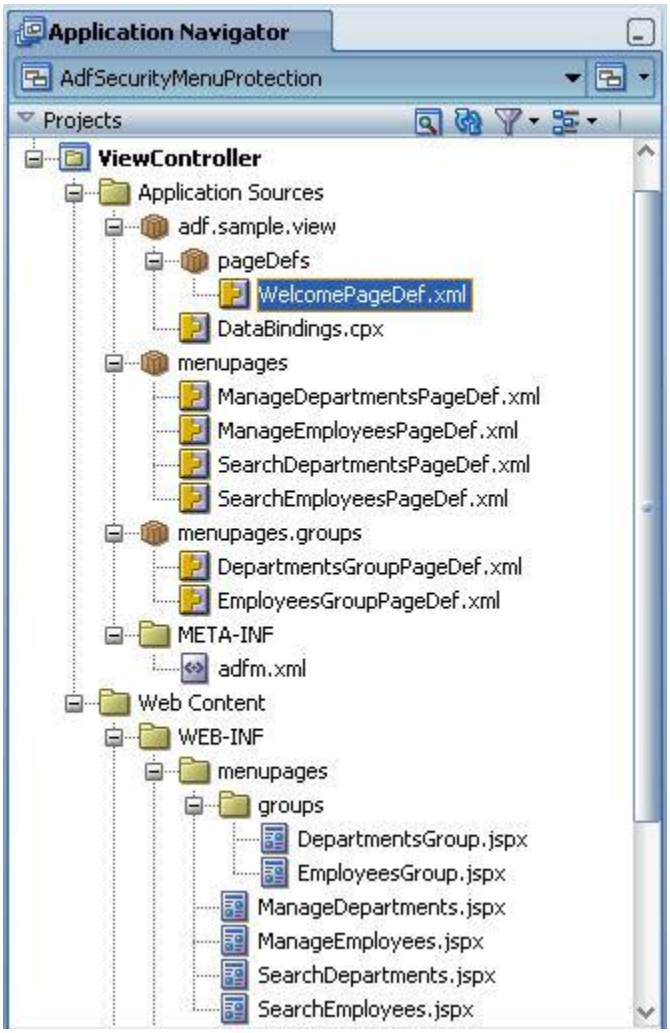
Create the ADF PageBinding for Security and for Use

ADF Security is a JAAS based authorization framework in ADF that protects ADF page bindings and bounded task flows. The easiest option to protect menu items with ADF Security thus is to build JSF pages for each menu item and menu group and to create ADF bindings for each of them. To create a JSF page for a view activity, double click on the view activity entry in the unbounded task flow definition. Shown in the image below, the pages in our example are saved in sub-folders: "menupages" and "menupages | groups". The only JSF page left in the public_html directory is the start page. Note that JSF pages that are stored in the WEB-INF folder are not directly accessible from a browser URL field.

To create the ADF bindings files, select each of the page entries in the Oracle JDeveloper Application Navigator and choose "Go to Page Definition" from the context menu. Note that the page definition files, the ADF binding files are created in a structure similar to those in which the JSF pages are saved. The overhead of creating PageDef files for all pages is minimal as it can be assumed that most of the JSF pages will later contain ADF bound content.



The image below shows the pages and ADF bindings that are created for the example in this article. As mentioned, the workspace can be downloaded at the end to this blog article.

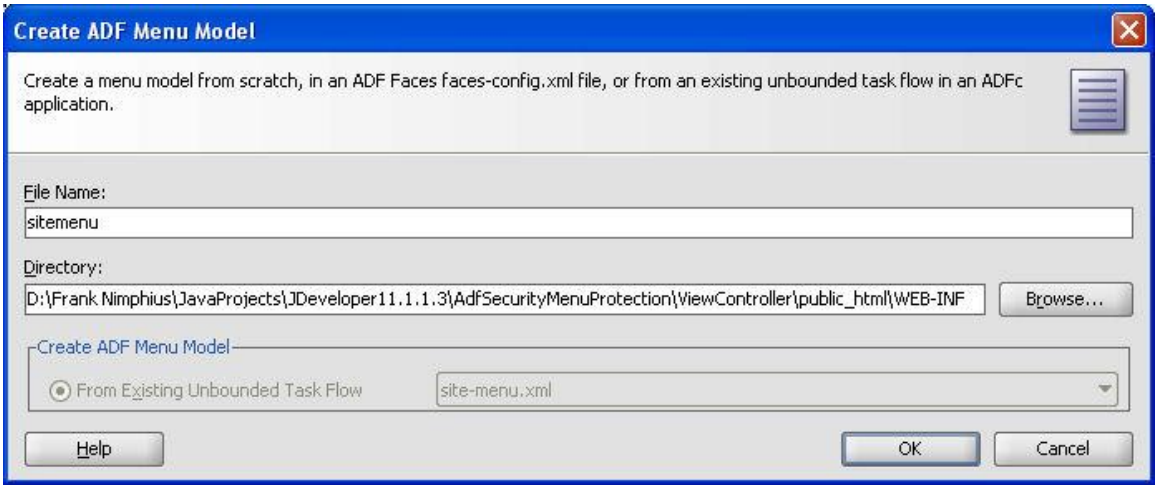


Creating the Site Menu Structure

Until here, we created an unbounded task flow and defined view activities for the menu navigation hierarchy. We also created JSF pages and ADF bindings for each page to support security.

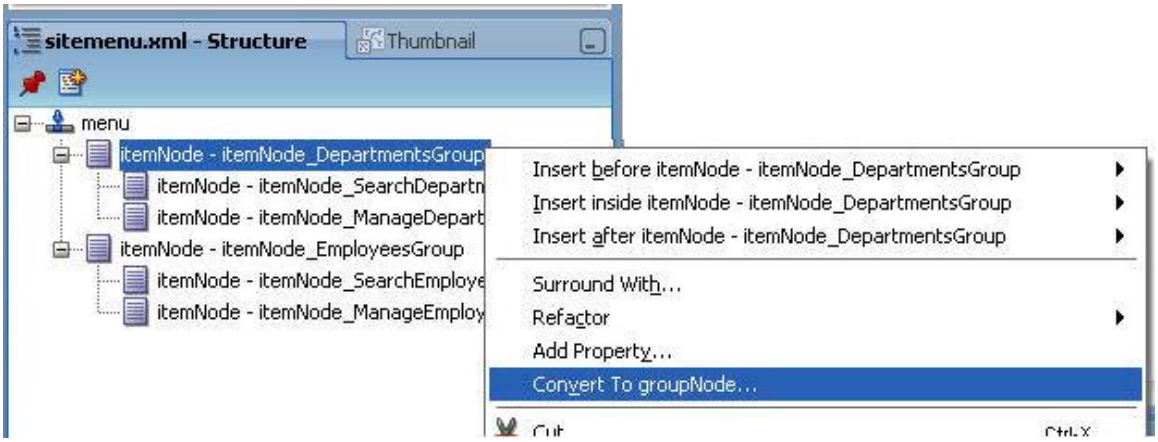
If you want to use your menu structure unprotected then you create PageDef files only when ADF bound components are used on a page. To create the menu definition - which is just another metadata artifact - you select the unbounded task flow and choose **"Create ADF Menu Model"** from the context menu. The menu yo create in this step can be updated later on without overriding any customizations you do in the following. So no worries if you don't yet know how many menu entries are to follow.

For simplicity, just give the menu the same name as the unbounded task flow so you have a visible relationship, which will prove beneficial when working in teams.



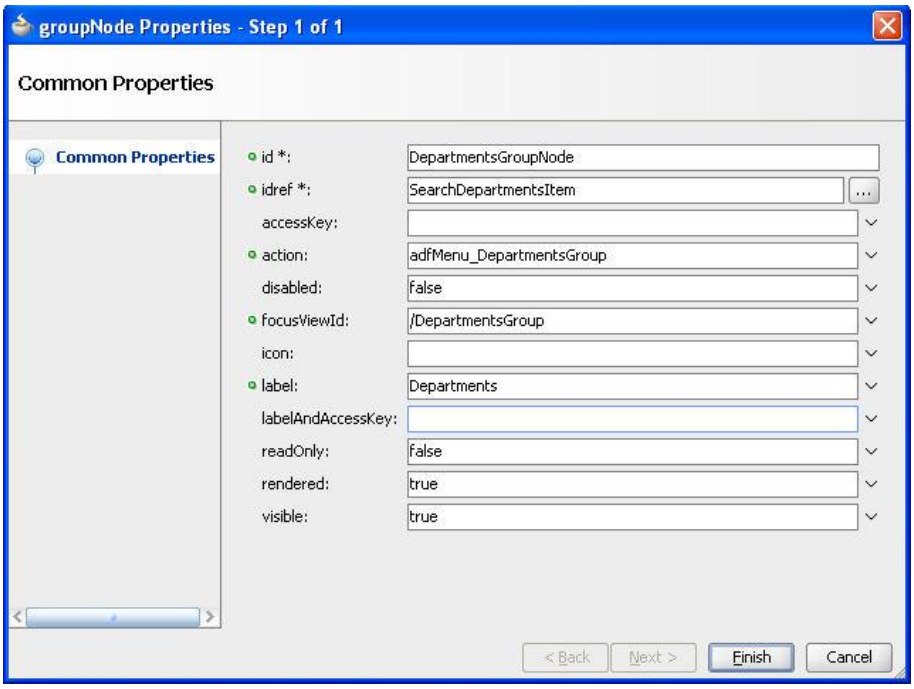
The ADF menu dialog creates a menu item for each of the view activities in the unbounded task flow definition. Because it does so even for the view activities we wanted to become menu groups, there is a bit of customization that needs to follow.

In a first customization, you move all related menu items under the group node. So in our example, we select the **SearchDepartmentsItem** and the **ManageDepartmentItem** and drag them under the **DepartmentsGroupItem**. This way you create the hierarchical structure for your site menu. Once this is done, you select the nodes that should become group nodes and choose **"Convert to GroupNode"** from the context menu.



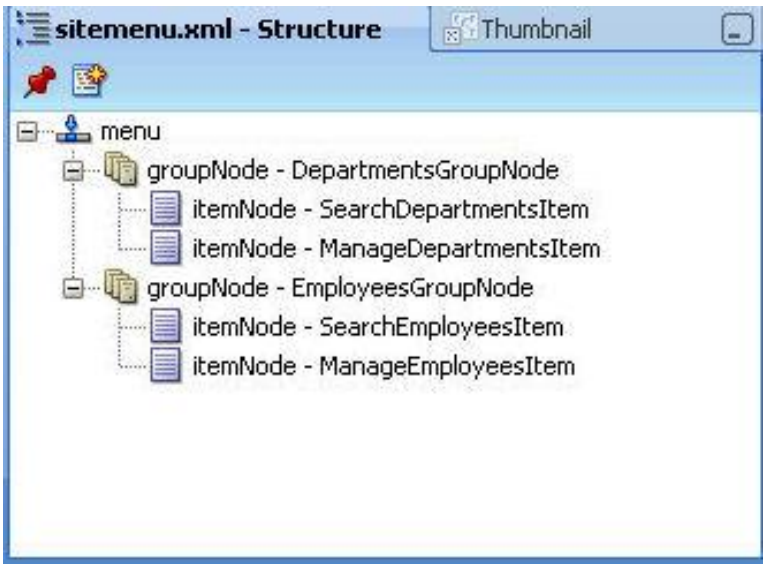
Group nodes don't perform navigation and therefore the action property of the menu item and some other properties are not available. A dialog that you press OK on informs you about the removal of these properties.

For each menu item and group node, you can open the Oracle JDeveloper Property Inspector to further shape the menu entries. For example you want to change the Id to a shorter but still unique and meaningful name and define a user friendly label, which can also be referenced from a resource bundle (labels are seldom hard coded in this global economy)



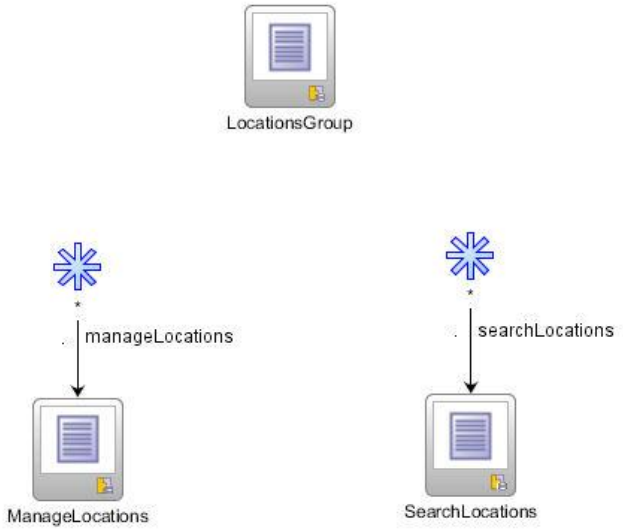
The image below shows the end result of the menu created in this example.

But wait! Where is the Locations tab from the image on top? Well we could have added this to the "site-menu.xml" unbounded task flow created above, but then we would have missed the opportunity to explain how to use multiple unbounded task flows to define the site menu structure. So the next - optional - section continues with building the menu.

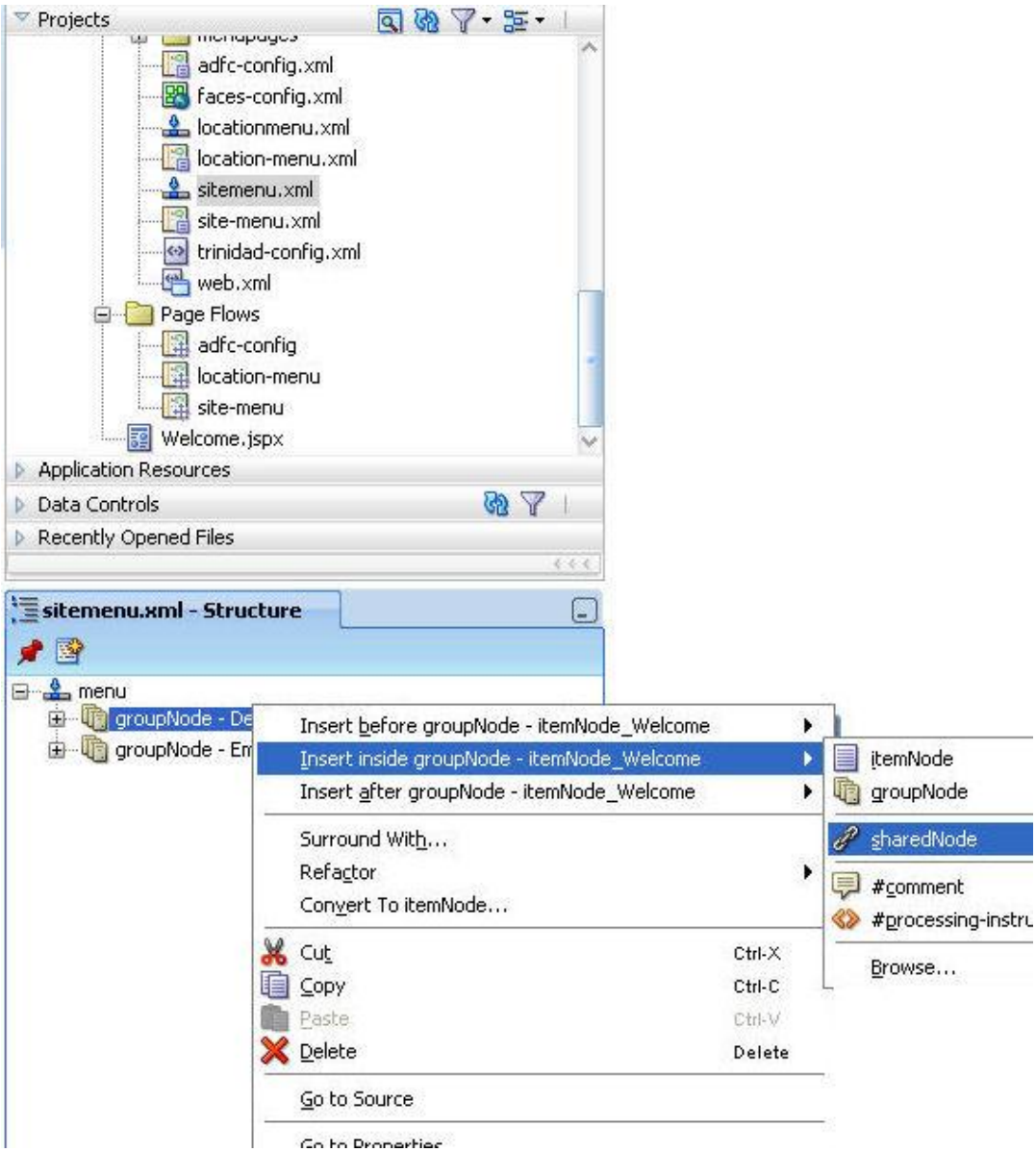


Optional Exercise: Include Hierarchies of a Second Unbounded Task Flow to the Menu

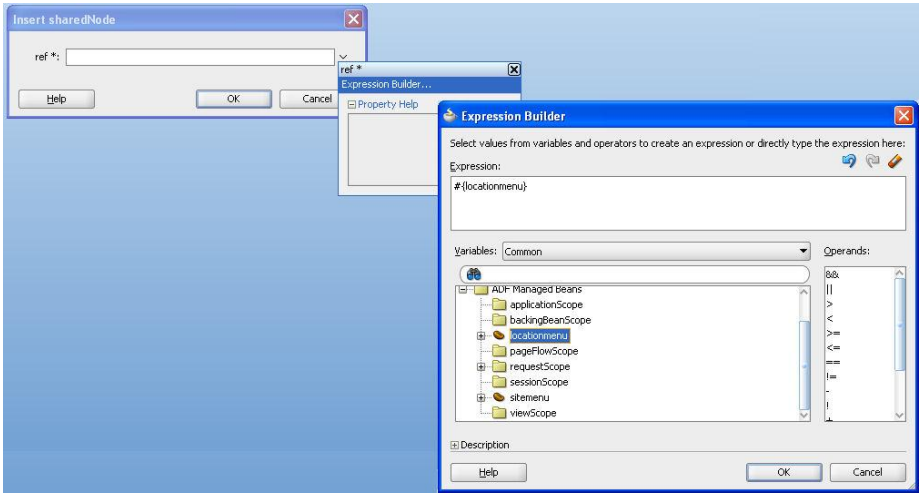
When building site menus, it may happen that you don't see the wood for the trees. In such cases it is good practice to split the unbounded task flow definition for the site menu hierarchy into two or more unbounded task flows. You can use this approach to modularize the site structure, which also makes it easier to work in teams. In the example we built for this article, the Locations tab and its sub menu entries are created in another unbounded taskflow location-menu.xml. You create this unbounded task flow as explained before for the site-menu. Then you also add view activities and create wild card navigation cases, JSF pages and ADF page bindings. Nothing different here.



You then create a menu definition from the location-menu.xml unbounded task flow, which results in a new menu entry "locationmenu.xml" (shown in the image below) getting created. In a next step, we need to link the two menus "sitemenu" and "location-menu" together, and this really is the only thing new to learn in this section. To link two menus, you add a shared node to the parent menu. In our case the parent menu is the "sitemenu". Note that whenever you create a new menu configuration, a managed bean is created that represents the menu model for this menu hierarchy. In the next step, we will make use of this managed bean.

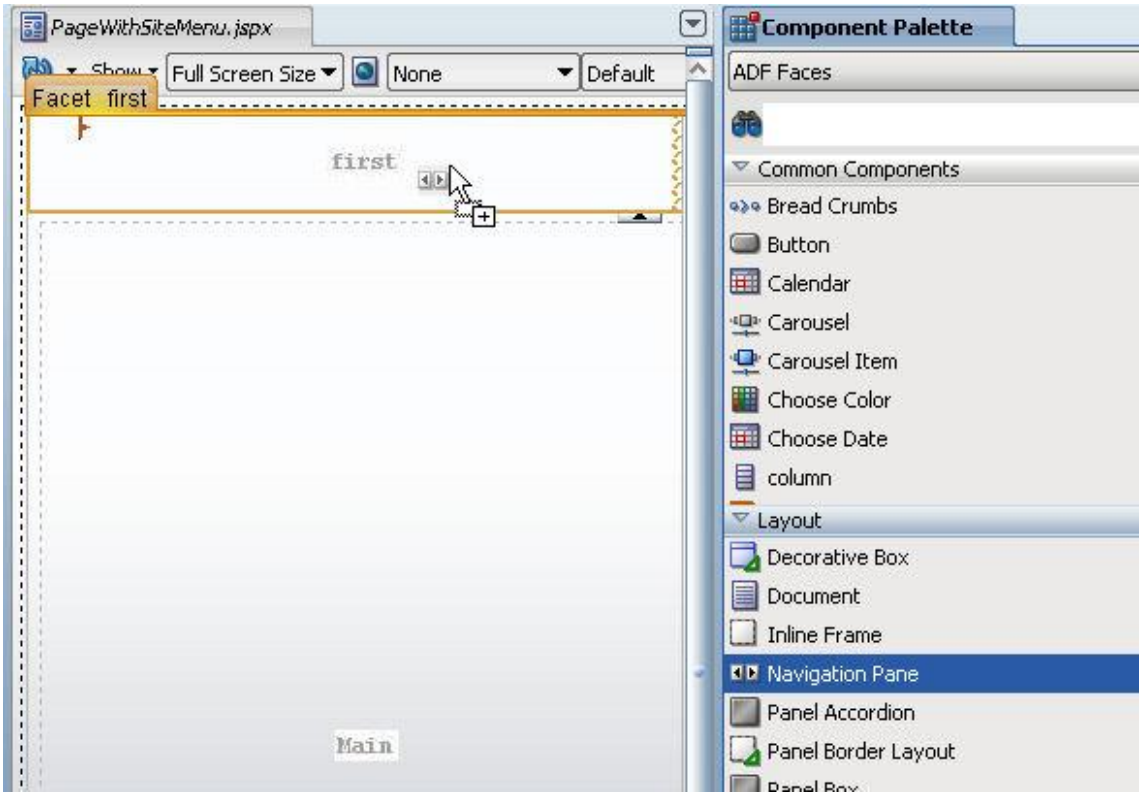


When you add a shared node to the parent menu, then the dialog shown below opens for you to use the ExpressionBuilder entry from the context menu to locate the managed bean that automatically got created when building the "locationmenu" menu. So the reference to this managed bean, which is a menu model implementation, is all you need to do to bring the menu structures together. Easy - he ?



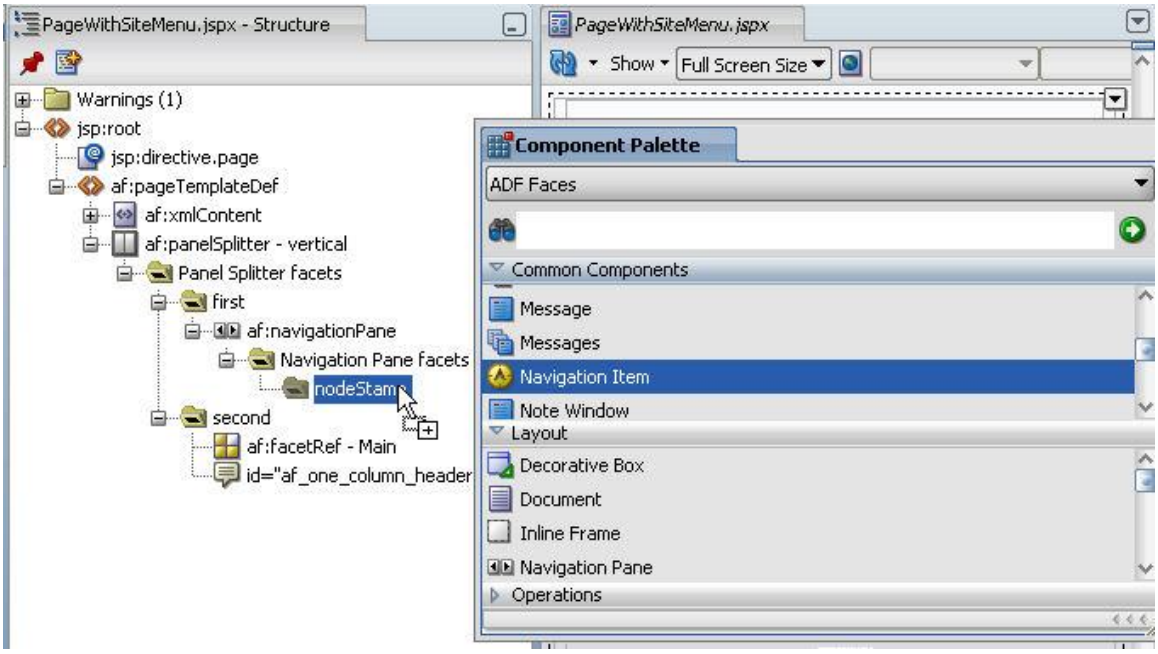
Creating the Site Menu in a Page Template

There is a lot of magic that happened and yet it doesn't really feel we developed a menu model for site navigation. However, we just did - yes, we developed a menu model without using Java. Sorry, but this is what makes ADF so great ;-) So all we need to do, to create a site menu that dynamically renders itself, including all future changes to the menu structure, is to us the ADF Faces Navigation Pane component and bind it to the menu model. For this, create an ADF Faces Page Template and add the Navigation Pane from the Component Palette. You could use the Navigation Pane without a page template, but a template saves you work.



Into the Navigation Pane component, add a Navigation Item component to the nodeStamp facet. The content of the nodeStamp facet is printed for all elements that are on a specific menu level.

The Navigation Pane has two properties to set that decide about which menu hierarchy level is rendered in it and how it is rendered. Into the Navigation Pane component, add a Navigation Item component to the nodeStamp facet. The content of the nodeStamp facet is printed for all elements that are on a specific menu level.



The Navigation Pane has two properties to set that decide about which menu hierarchy level is rendered in it and how it is rendered. The first image in the next section shows two Navigation Pane components surrounded by an af panelGroupLayout. The reason for having two Navigation Pane components is that the menu we built in this article is two level deep.

The reason for surrounding the Navigation Pane components in a Panel Group Layout is because facets can only have a single direct child component. So looking at the image below, you see the first menu hierarchy level - Departments, Employees and Locations - being rendered as a tab panel. The second level menu items are rendered "bar" in the second Navigation Pane. If you have a look at the image on top of this article then this is exactly what you see.

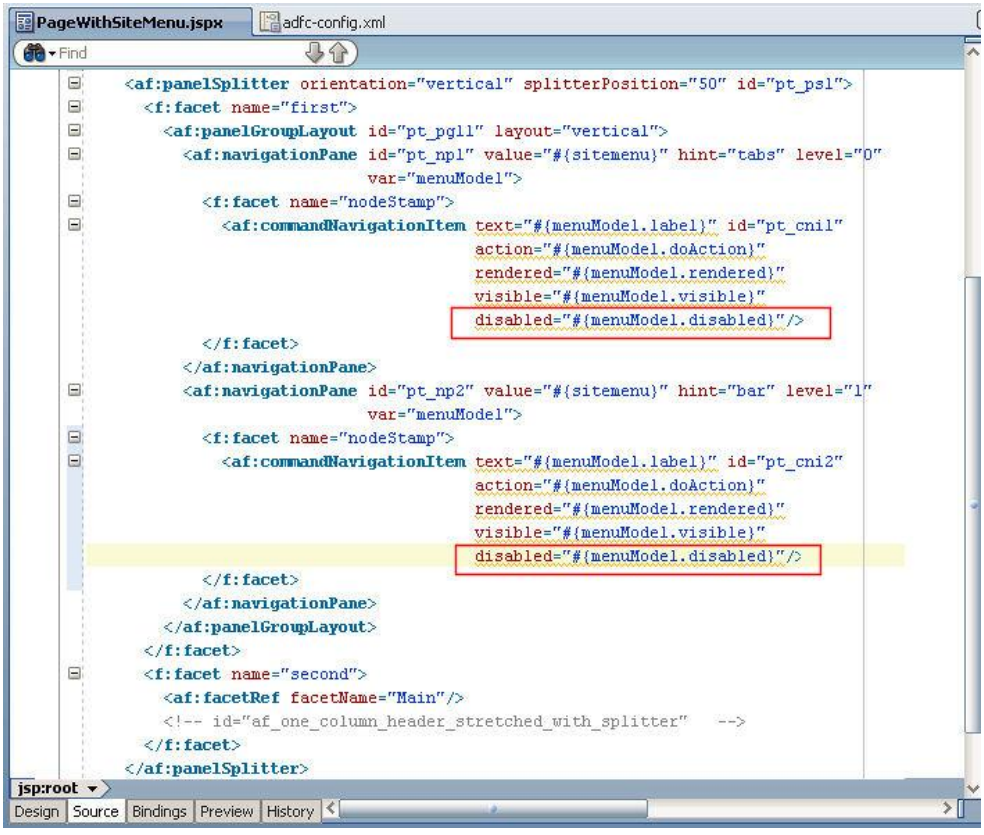
Applying Security

As mentioned, the basic requirement for using ADF Security to protect menu items is that the pages referenced by view activities have an ADF binding created (a PageDef file). Of course, you could manually create an OPSS resource permission and authorize page menu items using the ADF security context, but this is more - non declarative - work without getting the benefits of a multi line of defense strategy that is implemented using ADF security.

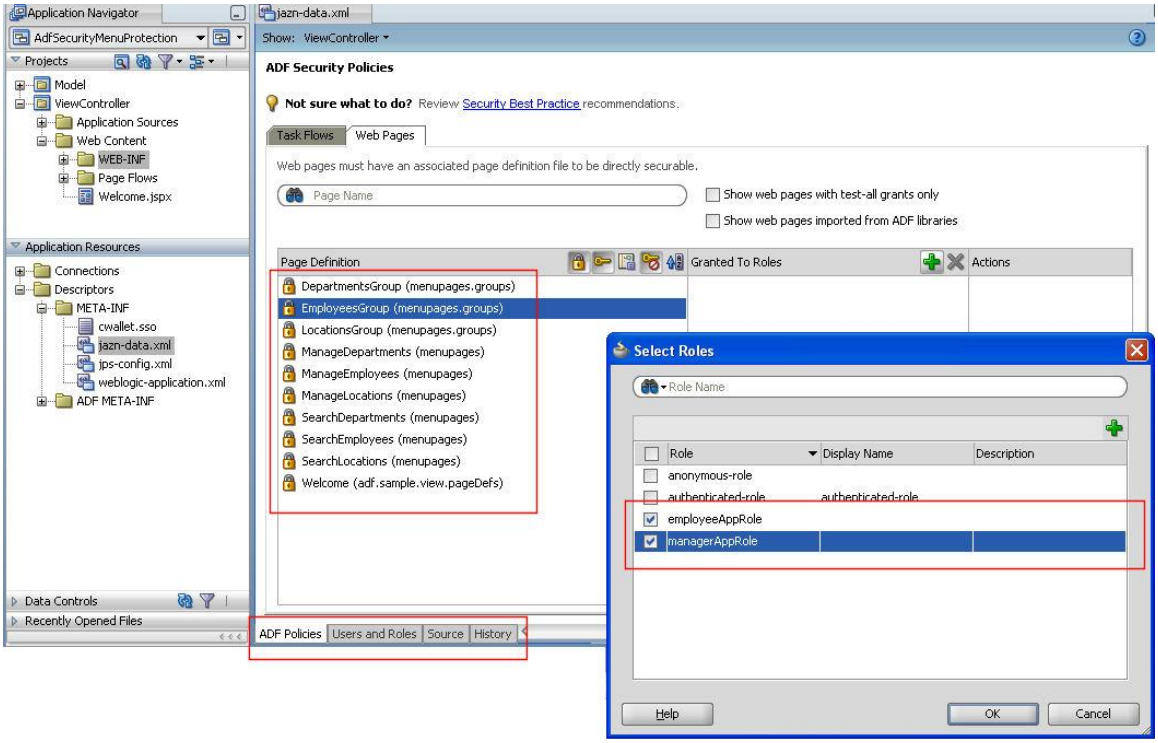
To implement ADF Security:

- Choose Application | Secure | Configure ADF Security from the JDeveloper menu
- Run the wizard and accept the default settings. Make sure you use the authentication and authorization option and don't use automatic grants (its the default, but I want to make sure you keep it that way)
- Double click the jazn-data.xml file in Application Resources | Descriptors | META-INF panel of Oracle JDeveloper's Application Navigator
- Use the User and Roles tab to create test users: sking/welcome1 and ahunold/welcome1 in this example. In the same tab, create enterprise roles (user groups) and application roles that have the enterprise groups associated. The application roles is what you use to grant page permissions to
- Shown in the image below, the af:commandNavigationItem has disabled, visible and rendered properties, which point to the equivalent menu model properties. All of the three properties can be used to implement security.

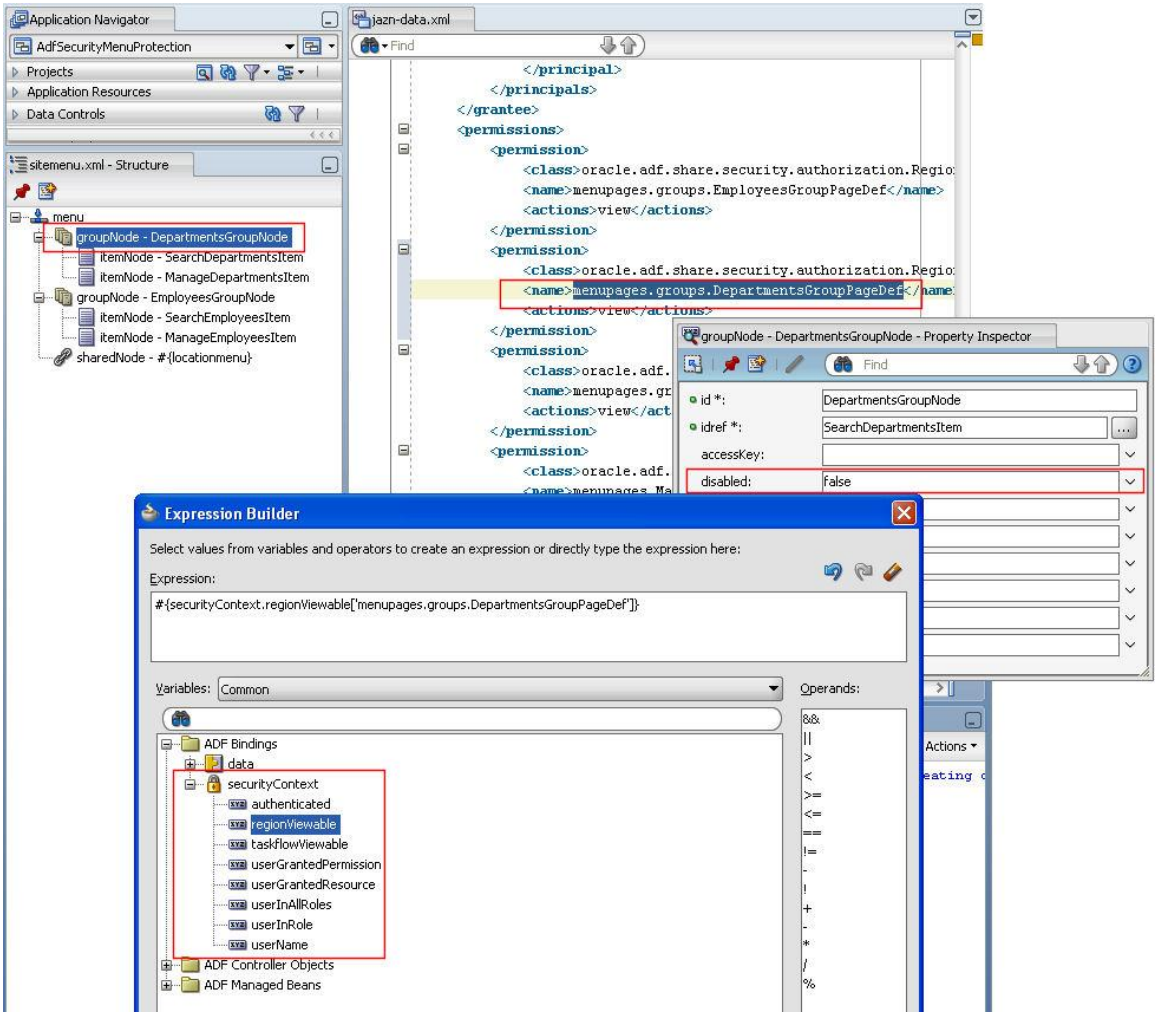
Note: In this example we use the disabled property for a better visual effect. The disabled property grays out menu options that users are not authorized to use an option, I really urge you to not use this option in production but to usevisible or rendered instead . The simple reason for my recommendation is that if a user is not allowed to use a menu option, why do you want to tease him/her by showing how less privileged he/she is and what would be possible otherwise? If you show me what I am not supposed to use, I swear I spend time looking for a way around this - just because I am a curious person. So you better hide what users are not allowed to use (my recommendation, but your choice). Thedisabled property is good to use if a menu item requires a specific condition to be true before users can access it. For example, a customer must be selected before orders can be worked on.



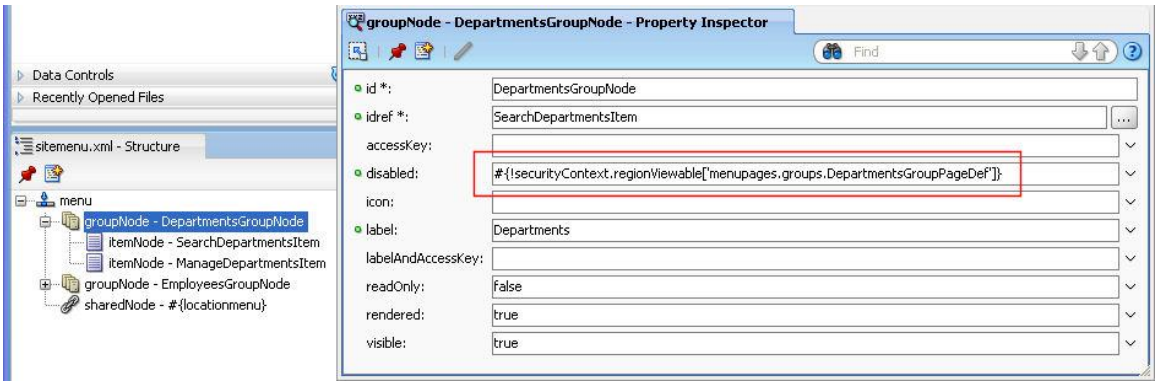
Before we can enforce authorization on a page, we need to configure page security in ADF Security. So with the jazn-data.xml file opened, select the ADF Policies tab and choose the Web Pages tab therein. You see a list of pages, including those we want to display as menu groups (as I said, we accept a little overhead in metada creation and ADF lifecycle handling for more security). For each page, you use the green plus icon to grant page access to one of the application roles you created. In this example I created the following application roles: managerAppRole, employeeAppRole. When granting page access to an application role, implicitly a JAAS permission definition is generated by Oracle JDeveloper and saved in the jazn-data.xml file. Granting access to a page protects the page from unauthorized access, which is one layer in a multi layer defense strategy. The other layer is to check the page permission in the menu model, which is what we do next.



The image below shows on its right hand side the page permission that gets created when granting pages to application roles. The permission type is **RegionPermission** (not to confuse with ADF Regions using the af:regions tag). The name of a permission is the absolute package and PageDef name of the protected ADF bound JSF page. For each of the protected ADF pages, you now select the corresponding menu item or group node and open the Property Inspector. Here you click the **arrow icon** next to either the disabled, rendered or visible property field to bring up the ExpressionLanguage builder. As shown in the image below, the ADF Security Context entry helps you to build the EL expression used to protect the menu option. Note that you will need to manually add the [' .... '] part of the permission check. This is why the jazn-data.xml file is opened in source view in the image below: to copy and paste the permission to avoid typos.

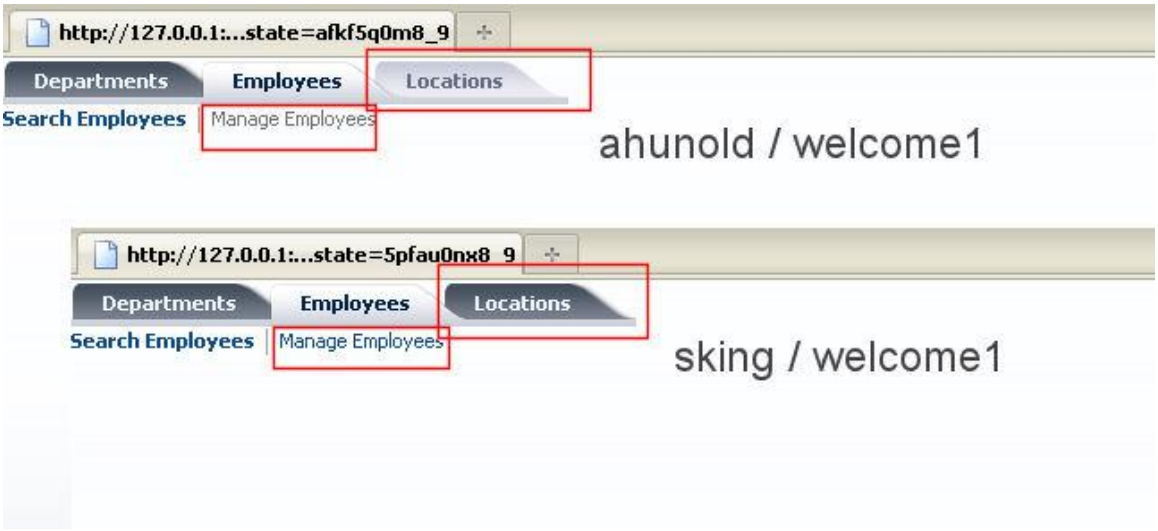


For the security expression to return true when an authenticated user does not have the required permission granted to access a menu option, you need to invert the expression as shown below. Otherwise you implement the opposite, which is that menu options would become active if the user is not authorized (still however ADF Security would prevent page access in this case - which is a benefit of using multi layer security)



Runtime View

Below is the runtime view of this example. Note that the sample you can download at the end of this blog article does not have anything on the pages other than the menus and applied authorization. There is so much more to say and explore about using XML Menu Models in ADF, which however may become subject of follow up articles. Note that each of the pages you access from the menu may also contain ADF Regions.



Note: If the page comes up with no menu rendered and no stack trace is shown in the JDeveloper message log window, then this is because the unbounded task flows that you created to build the menu structure in are not referenced from the adfc-config.xml file. The reference should be added automatically for you, but you know how it is with "should be" and "should not happen" in software development ;-)

Note: How to code against the XMLMenuModel in Java goes beyond the scope of this article. The JavaDoc reference is listed in the "Related Documentation" sections and - the source code is available on the MyFaces Trinidad website (http://myfaces.apache.org/trinidad).

Sample Download

The workspace to download from ADF Code Corner

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

is of JDeveloper 11g R1 PS2. However, this blog article holds true for all releases of Oracle JDeveloper 11g. To try it, just configure the database connection to use a HR schema in your reach and run Welcome.jspx. Connect as sking/welcome1 or ahunold/welome1, where ahunold is the less privileged account.

RELATED DOCOMENTATION

☒	XML Menu Model – Product Documentation http://download.oracle.com/docs/cd/E15523_01/web.1111/b31974/taskflows_complex.htm#BABHIFFE
☒	Page Templae tag - http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e12419/tagdoc/af_pageTemplate.html
☒	Navigation Pane tag documentation - http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e12419/tagdoc/af_navigationPane.html
☒	Command Navigation Item tag - http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e12419/tagdoc/af_commandNavigationItem.html
☒	ADF Security Documentation - http://download.oracle.com/docs/cd/E15523_01/web.1111/b31974/adding_security.htm#BGBGJEAH
☒	XML Menu Model JavaDoc http://www.docjar.com/docs/api/org/apache/myfaces/trinidad/model/XMLMenuModel.html
☒	XML Menu Model Documentation (Trinidad) - http://myfaces.apache.org/trinidad/devguide/xmlMenuModel.html
☒	Oracle Fusion Developer Guide – McGraw Hill Oracle Press, Frank Nimphius, Lynn Munsinger http://www.mhprofessional.com/product.php?cat=112&isbn=0071622543