Using Knowledge Management Functionality in Web Dynpro Applications

SAP NetWeaver 04

Copyright

Icons in Body Text

Icon	Meaning
⚠	Caution
💬	Example
💡	Note
🧭	Recommendation
SYN	Syntax

Typographic Conventions

Type Style	Description
Example text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
`Example text`	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, `F2` or `ENTER`.

Using Knowledge Management Functions in Web Dynpro Applications

Task

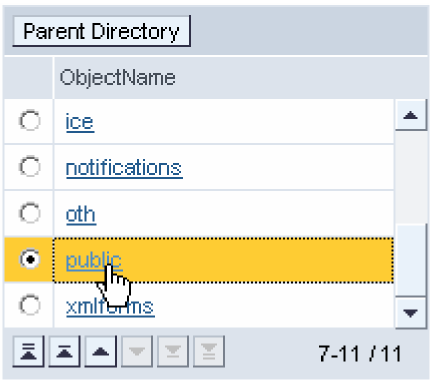
This tutorial shows you how to use SAP Knowledge Management in Web Dynpro applications. In the tutorial you create an application that uses Web Dynpro to access KM APIs in order to browse the KM repository.

The application allows you to walk through the directory structure but does not let you view files.

For more information about KM, see SDN at *Advanced search* → *Knowledge management* → *HowTo Guides.*

Preview

The graphic below shows what you are trying to achieve with your Web Dynpro application: It displays the directory structure of KM, which you can navigate through.



Objectives

By the end of this tutorial, you will be able to:

✓ Create a table

✓ Add UI elements to the table

✓ Do context binding for the table

✓ Add actions to the table

✓ Access the JAR files required for KM

Prerequisites

Systems, installed applications, and authorizations

- ☐ The SAP NetWeaver Developer Studio is installed on your computer.
- ☐ You have access to a SAP J2EE Engine with the SAP Portal including Knowledge Management

Knowledge

- ☐ You have a basic knowledge of Java.

Importing the Project Template

You have to import the project template. It contains a view in which you create the layout.

Prerequisites

- You have access to the SAP Developer Network (http://sdn.sap.com) with a user ID and password.
- The SAP NetWeaver Developer Studio is installed on your computer.

Procedure

Importing the project template into the SAP NetWeaver Developer Studio

1. Call the SAP Developer Network using the URL http://sdn.sap.com and log on with your user ID and the corresponding password. If you do not have a user ID, you must register before you can log on.
2. Navigate to *Web Application Server* area and then to the *Samples and Tutorials* section.
3. Download the ZIP file *TutWD_KMBrowser_Init.zip*, which contains the initial Web Dynpro project *TutWD_KMBrowser_Init* and save it in a local directory or directly in the work area of the SAP NetWeaver Developer Studio.
4. Unzip the contents of the ZIP file *TutWD_KMBrowser_Init.zip* into the work area of the SAP NetWeaver Developer Studio or in local directory.
5. Call the SAP NetWeaver Developer Studio.
6. Import the Web Dynpro project *TutWD_KMBrowser_Init.*
   The Web Dynpro project *TutWD_KMBrowser_Init* then appears in the Web Dynpro Explorer for further processing and editing in the context of this tutorial. You can ignore the warnings triggered by the Web Dynpro project *TutWD_KMBrowser_Init* at this time, since you will extend the Web Dynpro project during the remainder of this tutorial and so the warnings will disappear. You cannot execute the initial Web Dynpro project template *TutWD_KMBrowser_Init* without completing the application.

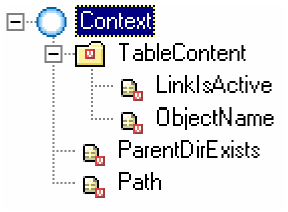
Initial Project Structure

After you have imported the Web Dynpro project template *TutWD_KMBrowser_Init,* the following project structure is displayed in the Web Dynpro Explorer:

Web Dynpro project structure
Web Dynpro project: **`TutWD_KMBrowser_Init`**
Web Dynpro application: **`KMBrowserApp`**
Web Dynpro component: **`KMBrowserAppComp`**
View: **`KMBrowserAppCompView`**

The Context

The context below has already been created for you in the template:



The attributes *LinkIsActive* and *ParentDirExists* have the property *type*, which is set to *boolean*.

The value node *TableContent* contains the data for the table.

The value attribute *LinkIsActive* is used in the table to specify whether a link is active.

The value attribute *ObjectName* contains the name of the object.

The value attribute *ParentDirExists* specifies whether there is a parent directory.

The value attribute *Path* specifies a path.

The Actions

Two actions, which are assigned to view elements, have been created for navigating through the KM repository:

OpenDirectory

This action allows you to navigate to a directory by clicking on the appropriate row in the application.

GoToParentDirectory

The application has a button used to trigger this action. It allows you to navigate up one level.

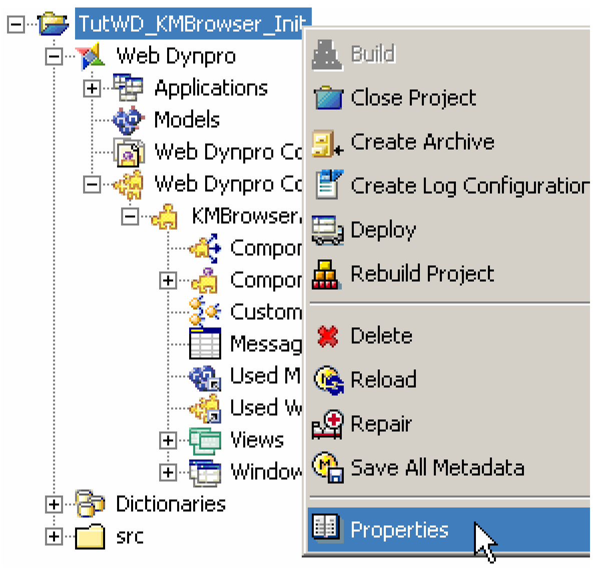
Implementing the View

You have to access external JAR files and implement the view controller.

Procedure

To include the needed .jar files in the classpath proceed as follows:

1. Create two classpath variables.
    a. Choose *Window* → *Preferences*.
    b. Open *Java* in the tree and choose *Classpath Variables* . Then choose *New…*
    c. In the dialog box displayed enter **`WEBAS_HOME`** as the *Name.* Enter the following *Path* for the directory:
    `<harddisk>`\usr\sap\**`<instance name>`**\JC**`<instance number>`**\j2ee\cluster\server**`<server number>`**
    (for example: C:\usr\sap\J2E\JC00\j2ee\cluster\server0)
    d. Create another variable with the *Name* **`PORTAL_HOME`**.Enter the following *Path* for the directory:
    WEBAS_HOME\apps\sap.com\irj\servlet_jsp\irj\root\WEB-INF\portal\
    (for example:
    C:\usr\sap\J2E\JC00\j2ee\cluster\server0\apps\sap.com\irj\servlet_jsp\irj\root\WEB-INF\portal\)
2. Click the project with the secondary mouse button and choose *Properties*:



3. In the dialog box displayed, choose *Java Build Path*, *Libraries* and then *Add Variable…*

4. Mark *WEBAS_HOME* with the primary mouse button and choose *Extend…* Navigate to \bin\ext\com.sap.security.api.sda\com.sap.security.api.jar, mark it with the primary mouse button and choose *OK*.

   This library offers the recent version of SAP's user management.

5. Now proceed in the same way with the variable *PORTAL_HOME* and add the following .jar files:

   - \lib**prtapi.jar**

     The portal runtime APIs
   - \portalapps\com.sap.portal.usermanagement\lib\com.sap.security.api.ep5.jar

     The user management APIs of the Enterprise Portal 5.0 are deprecated, but still in use in SAP NetWeaver 04
   - \portalapps\com.sap.netweaver.bc.rf\lib\bc.rf.framework_api.jar

     KM Repository Framework APIs
   - \portalapps\com.sap.netweaver.bc.rf.service\lib\bc.rf.global.service.urlgenerator_api.jar

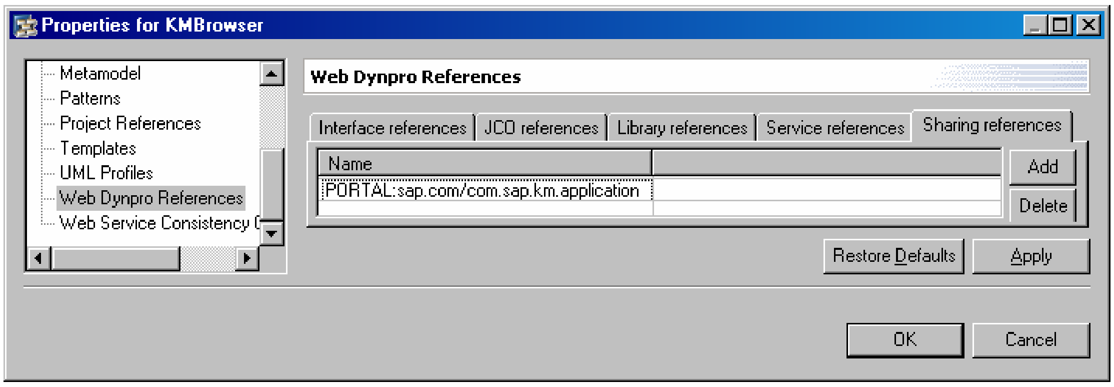
     Repository Framework Utility: URL Generator
   - \portalapps\com.sap.netweaver.bc.sf\lib\bc.sf.framework_api.jar

     Repository Framework: Repository Services
   - \portalapps\com.sap.netweaver.bc.util\lib\bc.util.public_api.jar

     Repository Framework Utilities

6. Navigate to *Web Dynpro References* in the left part of the dialog box and then choose *Sharing References*. Check if a reference is entered there. If it is not the case, choose the *Add* button add the following reference:

   ```
   PORTAL:sap.com/com.sap.km.application
   ```



Web Dynpro References

Name	
PORTAL:sap.com/com.sap.km.application	

Now you have to add some coding.

Add the following coding to the method `wdDoInit()`:

wdDoInit()

```
    //Set initial path to root directory "/"
    wdContext.currentContextElement().setPath("/");
    //refresh the table, here it means filling it initially
    refreshTable();
```

Add the following coding to the method `wdDoModifyView()`:

wdDoModifyView()

```
    //establish a ResourceID for the current path
     RID pathRID = RID.getRID(wdContext.currentContextElement().getPath());
     //if this ResourceID represents the repository's root…
     if (pathRID.isRoot()){
        //…disable the "Parent Directory" button
        wdContext.currentContextElement().setParentDirExists(false);
     } else {
        //…otherwise enable it
        wdContext.currentContextElement().setParentDirExists(true);
     }
```

Add the following coding to the method `wdActionOpenDirectory()`:

wdActionOpenDirectory()

```
    //get index of selected table row
    int i = wdContext.nodeTableContent().getLeadSelection();
    //get the name of the according object (resource)
    String name =
wdContext.nodeTableContent().getTableContentElementAt(i).getObjectName();
    //create a RID for the current path
    RID pathRID = RID.getRID(wdContext.currentContextElement().getPath());
    //create another RID for the directory that shall be opened
    RID directoryToOpenRID;
    //if the current directory is the root…
    if (pathRID.isRoot()){
       //…attach the selected directory's name without a leading slash /
       directoryToOpenRID = RID.getRID(pathRID.toString() + name);
    } else {
       //…attach the selected directory's name with a leading slash /
       directoryToOpenRID = RID.getRID(pathRID.toString() + "/" + name);
    }
    //set the selected directory as the new path
    wdContext.currentContextElement().setPath(directoryToOpenRID.toString());
    //refresh the table, i.e. show the content of the selected directory
    refreshTable();
```

Add the following coding to the method `onActionGoToParentDirectory()`:

onActionGoToParentDirectory()

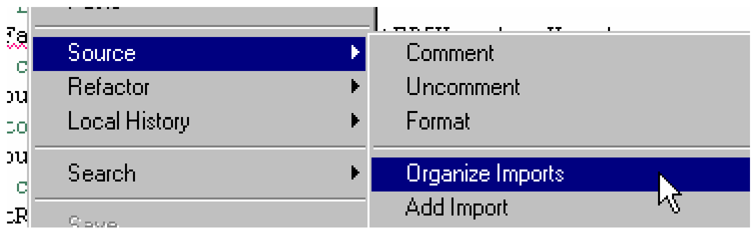
```
    //get a RID for the current path
    RID pathRID = RID.getRID(wdContext.currentContextElement().getPath());
    //set the path to its parent's value and save it
    wdContext.currentContextElement().setPath(pathRID.parent().toString());
    //show the parent's content
    refreshTable();
```

In the section *others* (between *//@@begin others* and *//@@end others*) enter the following code:

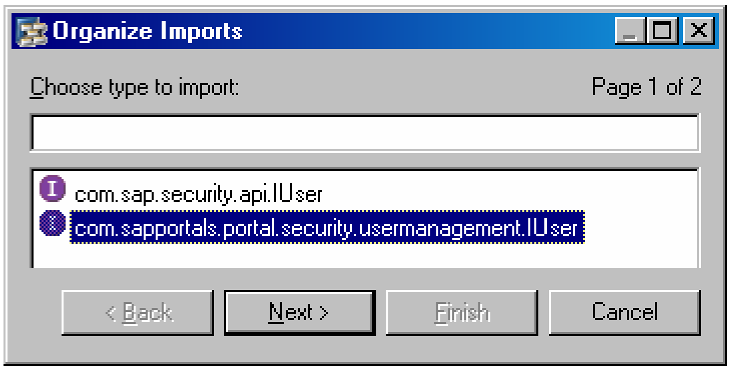
//@@begin others* and *//@@end others

```
    private void refreshTable(){
        //clear the table
        wdContext.nodeTableContent().invalidate();
        IPrivateKMBrowserAppCompView.ITableContentElement contentElement;
        try {
            //create an user object from the current user
            IWDClientUser wdClientUser = WDClientUser.getCurrentUser();
            com.sap.security.api.IUser sapUser = wdClientUser.getSAPUser();
            //create an ep5 user from the retrieved user
            IUser ep5User = WPUMFactory.getUserFactory().getEP5User(sapUser);
            //establish resource context
            IResourceContext resourceContext = new ResourceContext(ep5User);
            //get a resource factory
            IResourceFactory resourceFactory = ResourceFactory.getInstance();
            //get a RID from the current path to display the according content
            RID pathRID =
RID.getRID(wdContext.currentContextElement().getPath());
            //get a Iresource object to work on
            IResource resource = resourceFactory.getResource(pathRID,
resourceContext);
            //cast the object to a Collection
            ICollection collection = (ICollection) resource;
            //get the Collection's children
            IResourceList resourceList = collection.getChildren();
            //and finally get an iterator to walk through the set of children
            IResourceListIterator resourceListIterator =
resourceList.listIterator();
            //now read all elements…
            while (resourceListIterator.hasNext()) {
                //… and create a new context element for each of them
                contentElement = wdContext.createTableContentElement();
                IResource tempResource = resourceListIterator.next();
                if (!tempResource.isCollection()) {
                    contentElement.setLinkIsActive(false);
                } else {
                    contentElement.setLinkIsActive(true);
                }
                contentElement.setObjectName(tempResource.getName());
                wdContext.nodeTableContent().addElement(contentElement);
                }
            }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
```

To add the import statements required click the source code with the secondary mouse button and choose *Organize Imports.*

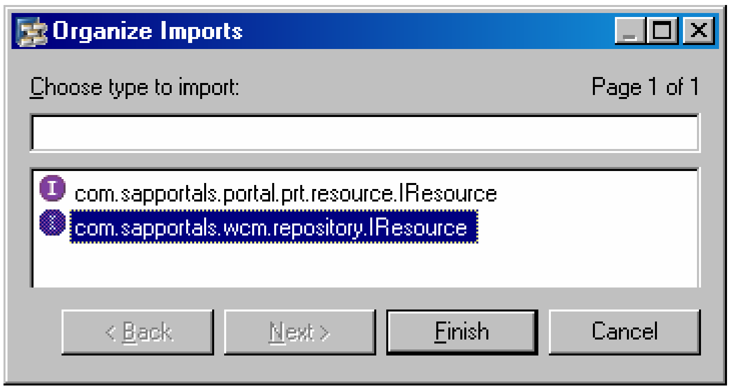


When asked to choose the type to import, select the type displayed below:


Organize Imports

Choose type to import: Page 1 of 2

com.sap.security.api.IUser
com.sapportals.portal.security.usermanagement.IUser


Organize Imports

Choose type to import: Page 1 of 1

com.sapportals.portal.prt.resource.IResource
com.sapportals.wcm.repository.IResource

Save the metadata by clicking your project with the secondary mouse button and choosing *Save All Metadata*.

Result

You can now build, deploy and run your application.

Building, Deploying, Configuring, and Running the Application

Some preparation is still essential before you can deploy and run the application successfully on the J2EE engine. Go through each of the following prerequisites carefully.

Prerequisites

You have made sure that the relevant SAP basis system, which you will be accessing remotely to retrieve the flight data, is currently available and contains flight data.

You have made sure that the SAP J2EE Engine has been launched. To do this, refer to Starting and Stopping the SAP J2EE Engine [Extern].

You have checked that the configuration settings for the J2EE server and for the SDM server are entered correctly in the SAP Netweaver Developer Studio.

To check the server settings, choose *Window* → *Preferences* → *SAP J2EE Engine*.The connection parameters for the SLD are defined in the J2EE Visual Administrator.

Procedure

Building the project

1. If you have not already done so, save the metadata for your project in its current state.

2. In the Web Dynpro Explorer, click the project node with the secondary mouse button and choose *Rebuild Project*. Make sure that the *Tasks* view does not display any errors for your project.

Deploying the project

1. In the Web Dynpro Explorer, click the project node with the secondary mouse button and choose *Create Archive*.

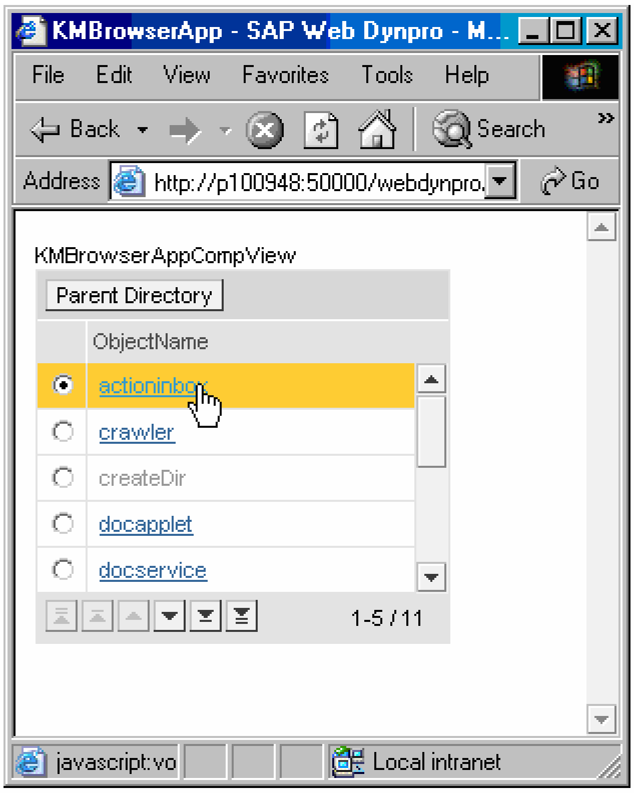
2. Click the project node with the secondary mouse button and choose *Deploy*.

Launching the application

In the Web Dynpro Explorer, click the application node with the secondary mouse button and choose *Run*.
Note you need a portal to access KM because KM runs on the portal.

Result

The SAP Netweaver Developer Studio performs the deployment process and then automatically launches your application in the Web browser. Your application allows you to browse the KM repository.


KMBrowserApp - SAP Web Dynpro - M...
File Edit View Favorites Tools Help
Back
Search
Address http://p100948:50000/webdynpro.
Go
javascript:vo
Local intranet

KMBrowserAppCompView

Parent Directory

	ObjectName
●	actioninbox
○	crawler
○	createDir
○	docapplet
○	docservice

1-5 / 11