PeopleSoft®

EnterpriseOne JDE5 Portal PeopleBook

May 2002

EnterpriseOne JDE5
Portal PeopleBook
SKU JDE5EPR0502

Table of Contents

Overview

Portal Overview

The J.D. Edwards Portal is a configurable gateway to J.D. Edwards software and other applications as well as to the Internet. You can use the Portal to launch J.D. Edwards applications or other programs, perform Internet and intranet searches, and to access other information. The look of the Portal is completely configurable. You can change the Portal and its components so that it looks like the rest of your company's web pages. Access to the Portal and to its resources is governed by J.D. Edwards software security and is based on roles and user IDs.

You can access the J.D. Edwards Portal either through your computer or through a variety of hand-held devices.

Note

You must be licensed to run J.D. Edwards software on a thin client browser before you can receive the J.D. Edwards Portal.

OneWorld Portal Interface

Portal Interface

The J.D. Edwards Portal provides a Web-based framework in which you can manipulate components. A component is any element in the workspace of the Portal and can include a calendar, your e-mail, your work center, national news headlines, and so forth. You can use your PC to view the Portal, or you can use a handheld device that uses Pocket PC or Windows CE. The interface for each of these environments varies; furthermore, the Pocket PC environment does not support full Portal functionality.

Note

If you are using Microsoft Internet Explorer version 6 to view the Portal, you must turn on support for session cookies in Microsoft Internet Explorer before you launch the Portal.

Portal PC Interface

The PC interface for the J.D. Edwards Portal features two toolbars and a workspace area. The system displays workspaces in the workspace area, and it displays components within workspaces.

Toolbars

The top area of the Portal contains two toolbars. The upper toolbar is the Enterprise Navigation Bar. It contains icons (buttons) or hyperlinks that are configured by your system administrator for enterprise-wide navigation. For example, the buttons can provide access to different areas of your company's Web site, other areas within the Portal, or other sites on the World Wide Web. You can choose to display the Enterprise Toolbar Buttons as icons, text, or icons and text. The Enterprise Navigation Bar might also contain an Internet Search Tool. Use User Preferences to hide or show the Internet Search Tool and to select a search engine.

The lower toolbar is the Workspace Navigation Bar. It displays the name of the current workspace. Additionally, the Workspace Navigation Bar might contain a workspace chooser, a role chooser, and a set of Portal- and workspace-related links. A special component type, link components, can be added to the Workspace Navigation Bar to provide hyperlinks in addition to the default hyperlinks provided by the Portal. You can display the workspace chooser either as a drop-down field or as a set of tabs across the bottom part of the Workspace Navigation Bar.

Both toolbars might vary, depending on the current workspace and on how your system administrator configures it and your access to the Portal.

► To configure the Enterprise toolbar buttons

1. Click Personalize on the Secondary Navigation Bar.

If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, click User Preferences.
3. On Display Preferences, for Enterprise Navigation Bar Icon Display, choose how you want to display the buttons and click OK.

► To hide or show the Internet search tool

1. Click Personalize on the Secondary Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, click User Preferences.
3. On Display Preferences, perform one of the following tasks:
   a. To hide the Internet search tool, ensure that Show is not selected, and then click OK.
   b. To show the Internet search tool, click Show and complete the following steps.
      i. Click Personalize Search to configure the tool.
      ii. On Internet Search Preferences, choose one of the search engine options, or specify a different engine by clicking Custom and completing the following fields:
         - Title

           You can give the search engine any title you wish. This title appears next to the Internet Search Tool field on the Enterprise Navigation Bar.
         - Search Home URL

           Enter the URL of the home page for the search engine.
         - Search Query URL

           Enter the URL that the search engine uses for normal searches. To find this URL, verify that the Address Bar is showing in your Web browser, and then go to the search engine home page. Perform a normal search. The URL that appears in the address bar is the search query URL plus the text string you asked to search on (for example, a normal search in yahoo.com for J.D. Edwards renders the following URL: `http://search.yahoo.com/bin/search?p=J.+D.+Edwards`). Delete the text string you searched on and copy the rest of the URL (for example: `http://search.yahoo.com/bin/search?p=`). This is the URL you must place in the Search Query URL field.

      iii. Click OK.
      iv. On Display Preferences, click OK.

► To configure the workspace chooser

1. Click Personalize on the Secondary Navigation Bar.

If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, click User Preferences.
3. On Display Preferences, for Workspace Selection Display Type, choose Dropdown to display the chooser as a drop-down menu or choose Tabs to display the chooser as a set of tabs.
4. Click OK.

Workspace Area

Most of the Portal interface space is devoted to the workspace area: the area in which the Portal displays workspaces. You can create several workspace layouts to logically group components. For example, you might have one workspace for a variety of news and stock information and another workspace for accessing your company's accounting systems. You can configure your private workspaces to show one or more components arranged in one to five columns. You can switch between workspaces by using the workspace chooser on the Workspace Navigation Bar.

Workspaces that you create are private; only you can see them. Your system administrator can create system-wide workspaces. You might not be able to modify these system-wide workspaces. Furthermore, your system administrator can assign one or more roles in the system to you. Your role determines in part which Enterprise Navigation Bar buttons, workspaces, and components you can see.

Components

Most workspaces display several components at once. Components have their own properties and can be viewed in several modes. Most components have a toolbar at the top that displays the component’s title and includes two or more buttons. These buttons allow you to change the component mode. The component modes are:

Restore In restore mode, the component can be viewed in a box in the workspace. Other components (if they exist) can be viewed in the workspace as well. Restore mode usually provides a small amount of content that fits into its portion of the workspace. For example, an e-mail component might display the subjects of your first five pieces of new mail, and a weather component might display a map with temperatures.

Restore mode is the default mode of all components. If a component is in another mode, you can return it to normal mode by clicking the double-box icon on the component toolbar.

Minimize In minimize mode, only the component's toolbar is visible. If you have a large number of components in your workspace, you might want to minimize some to give others more space. To view a component in minimize mode, click the line icon on the toolbar. To return to restore mode, click the double-box icon on the toolbar.

Maximize In maximize mode, the component fills the entire workspace. If the component contains a large amount of content, it might be easier to work with it in this mode. In this mode, some components might display additional features. Not all components have this mode available. To view a component in maximize mode, click the single-box icon on the component toolbar. To return to restore mode, click the double-box icon on the toolbar.

Help In help mode, the component fills the entire workspace, displaying information about the component. Not all components have this mode available. To view a component in help mode, click the question mark icon on the component toolbar. To return to restore mode, click the double-box icon on the toolbar.

Personalize In personalize mode, the component fills the entire workspace. The tasks that you can perform in this mode vary based on the component. You might be able to add or remove links from the component, for example. Not all components have this mode available. To view a component in personalize mode, click the arrow icon on the component toolbar. To return to restore mode, click the double-box icon on the toolbar.

Most components have a restore, minimize, and maximize mode. Help and personalize modes are available only if the designer of the component enabled them.

Portal Windows CE Interface

The Windows CE interface for the J.D. Edwards Portal features two toolbars and a workspace area. The system displays workspaces in the workspace area, and it displays components within workspaces.

Toolbars

The top area of the Portal contains two toolbars. The upper toolbar is the Enterprise Navigation Bar. It contains icons (buttons) or hyperlinks that are configured by your system administrator for enterprise-wide navigation. For example, the buttons can provide access to different areas of your company's Web site, other areas within the Portal, or other sites on the World Wide Web.

The lower toolbar is the Workspace Navigation Bar. It displays the name of the current workspace. Additionally, the Workspace Navigation Bar might contain a workspace chooser,

a role chooser, and a set of Portal- and workspace-related links. A special component type, link components, can be added to the Workspace Navigation Bar to provide hyperlinks in addition to the default hyperlinks provided by the Portal.

Both toolbars might vary, depending on the current workspace and on how your system administrator configures it and your access to the Portal.

► To configure the Enterprise toolbar buttons

1. Click Personalize on the Secondary Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, click User Preferences.
3. On Display Preferences, for Enterprise Navigation Bar Icon Display, choose how you want to display the buttons and click OK.

Workspace Area

Most of the Portal interface space is devoted to the workspace area: the area in which the Portal displays workspaces. You can create several workspace layouts to logically group components. For example, you might have one workspace for a variety of news and stock information and another workspace for accessing your company's accounting systems. You can configure your private workspaces to show one or more components arranged in one to five columns. You can switch between workspaces by using the workspace chooser on the Workspace Navigation Bar.

Workspaces that you create are private; only you can see them. Your system administrator can create system-wide workspaces. You might not be able to modify these system-wide workspaces. Furthermore, your system administrator can assign one or more roles in the system to you. Your role determines in part which Enterprise Navigation Bar buttons, workspaces, and components you can see.

Components

Most workspaces display several components at once. Components have their own properties and can be viewed in several modes. Most components have a toolbar at the top that displays the component’s title and includes two or more buttons. These buttons allow you to change the component mode. The component modes are:

Restore In restore mode, the component can be viewed in a box in the workspace. Other components (if they exist) can be viewed in the workspace as well. Restore mode usually provides a small amount of content that fits into its portion of the workspace. For example, an e-mail component might display the subjects of your first five pieces of new mail, and a weather component might display a map with temperatures.

Restore mode is the default mode of all components. If a component is in another mode, you can return it to normal mode by clicking the double-box icon on the component toolbar.

Minimize In minimize mode, only the component's toolbar is visible. If you have a large number of components in your workspace, you might want to minimize some to give others more space. To view a component in minimize mode, click the line icon on the toolbar. To return to restore mode, click the double-box icon on the toolbar.

Maximize In maximize mode, the component fills the entire workspace. If the component contains a large amount of content, it might be easier to work with it in this mode. In this mode, some components might display additional features. Not all components have this mode available. To view a component in maximize mode, click the single-box icon on the component toolbar. To return to restore mode, click the double-box icon on the toolbar.

Help In help mode, the component fills the entire workspace, displaying information about the component. Not all components have this mode available. To view a component in help mode, click the question mark icon on the component toolbar. To return to restore mode, click the double-box icon on the toolbar.

Personalize In personalize mode, the component fills the entire workspace. The tasks that you can perform in this mode vary based on the component. You might be able to add or remove links from the component, for example. Not all components have this mode available. To view a component in personalize mode, click the arrow icon on the component toolbar. To return to restore mode, click the double-box icon on the toolbar.

Most components have a restore, minimize, and maximize mode. Help and personalize modes are available only if the designer of the component enabled them.

Portal Pocket PC Interface

The Pocket PC interface for the J.D. Edwards Portal features one toolbar and a workspace area. The system displays workspaces or components in the workspace area.

Toolbar

The top area of the Portal contains a toolbar consisting of a drop-down field. Click the drop-down menu to access different workspaces or to log off.

Workspace Area

Most of the Portal interface space is devoted to the workspace area: the area in which the Portal displays workspaces and components. When you first log on, a workspace is displayed

in the workspace area. It appears as one or more hyperlinks. The hyperlinks are components. Click a component to display it.

You cannot create or configure workspaces with the Pocket PC interface.

Components

Most workspaces display several components at once. Components have their own properties and can be viewed in several modes. Most components have a toolbar at the top that displays the component's title and includes two or more buttons. These buttons allow you to change the component mode. The component modes are:

Restore In restore mode, the component can be viewed in a box in the workspace. Other components (if they exist) can be viewed in the workspace as well. Restore mode usually provides a small amount of content that fits into its portion of the workspace. For example, an e-mail component might display the subjects of your first five pieces of new mail, and a weather component might display a map with temperatures.

Restore mode is the default mode of all components. If a component is in another mode, you can return it to normal mode by clicking the double-box icon on the component toolbar.

Minimize In minimize mode, only the component's toolbar is visible. If you have a large number of components in your workspace, you might want to minimize some to give others more space. To view a component in minimize mode, click the line icon on the toolbar. To return to restore mode, click the double-box icon on the toolbar.

Maximize In maximize mode, the component fills the entire workspace. If the component contains a large amount of content, it might be easier to work with it in this mode. In this mode, some components might display additional features. Not all components have this mode available. To view a component in maximize mode, click the single-box icon on the component toolbar. To return to restore mode, click the double-box icon on the toolbar.

Help In help mode, the component fills the entire workspace, displaying information about the component. Not all components have this mode available. To view a component in help mode, click the question mark icon on the component toolbar. To return to restore mode, click the double-box icon on the toolbar.

Personalize In personalize mode, the component fills the entire workspace. The tasks that you can perform in this mode vary based on the component. You might be able to add or remove links from the component, for example. Not all components have this mode available. To view a component in personalize mode, click the arrow icon on the component toolbar. To return to restore mode, click the double-box icon on the toolbar.

Most components have a restore, minimize, and maximize mode. Help and personalize modes are available only if the designer of the component enabled them.

► To log into the Portal

1. Launch a Web browser such as Netscape Navigator or Microsoft Internet Explorer.

Note
Your company may have its own method for users to access the Portal, such as a desktop shortcut.

2. Enter the Portal's URL and press Enter.

   If you do not know what the Portal's URL is, contact your system administrator.

   Alternatively, you can also enter your ID, password, and environment directly into the URL to bypass the login form. For example, if the URL for the Portal was:

   ```
   http://localhost/jde/servlet/com.jdedwards.portal
   ```

   you could instead enter:

   ```
   http://localhost/jde/servlet/com.jdedwards.portal.PortalBui
   lderServlet?User=MYUSERID&Password=MYPASSWORD&Environmen
   t=MYENVIRONMENT
   ```

   where

   MYUSERID = your user ID

   MYPASSWORD = your password

   MYENVIRONMENT = the environment you want to access

   This method of logging in is called parameter-based login.

3. On the J.D. Edwards software login form, enter your user ID, password, and environment, and then click Sign On.

   If you leave the environment field blank, the system provides a list of environments for you to choose from when you click Sign On.

► To log out of the Portal

Click Logout in the Workspace Navigation Bar.

Portal Workspaces

If your system administrator has granted you the ability to personalize workspaces, you can add new workspaces to the Portal.

The workspace that you create is not public; that is, only you can see it. However, you can grant other individuals the following kinds of rights to your workspace:

- View

  Individuals who have View rights can see your workspace and personalize the appearance on their workstations. They cannot affect the workspace in any other way.

- View/Edit

Individuals who have View/Edit rights can view and make changes to the workspace.

- View/Edit/Grant Permissions

  Individuals who have View/Edit/Grant Permissions rights can view and change the workspace, and can also grant rights to other individuals.

You can personalize your workspaces. From the Personalize form, you can change the following characteristics for a workspace:

- Workspace unique ID (Workspace Name)
- Environments in which the workspace can appear (PC, Windows CE, Pocket PC)
- Title
- Components (add, remove, rearrange)
- Columns (add, remove)
- Workspace Navigation Bar links
- Colors
- Background image
- Toolbar logo and background image
- Access privileges

To personalize a workspace, you must have a View/Edit rights to the workspace. You alone are granted View/Edit/Grant Permissions rights automatically with any workspace you create. You might not have access privileges to some public workspaces.

See Also

- See *Setting Workspace Permissions* for instructions on granting access to workspaces. Not all users have this capability.

► To create a new workspace

You can create an original workspace (one that is not based on another workspace), or you can create a workspace that is based on another workspace.

1. Click Personalize on the Secondary Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, perform one of the following:

   a. To create an original workspace, click Add Workspace.

   b. To create a workspace that is based on another workspace, choose a workspace from the Select Workspace field on the Workspace Navigation Bar, and then click Copy Workspace.

Note
When you copy a workspace, the system does not copy any links associated with the workspace. If you want the new workspace to contain the same links as the workspace on which it is based, you must recreate them.

3. Complete the following fields in the Current Workspace section of the form:

   - Workspace Name

     This is the name of the workspace as it appears in the workspace chooser on the Workspace Navigation Bar. You should limit your workspace name to 50 characters.

   - Devices

     Choose in which form factors the new workspace should be available. Choose HTML 4.0 to make the workspace available to PC environments; click one or more of the other choices to make the workspace available to devices that use Windows CE.

     Note
     Making a workspace available to a particular device does not guarantee that all of its components will work properly on that device. A component that includes features that are not supported by a particular device will not function correctly on that device.

   - Page Title

     This text appears on the bottom right side of the Workspace Navigation Bar when the workspace is being viewed.

   - Workspace ID

     This name is the system name for the workspace.

     Caution
     All J.D. Edwards workspaces follow this convention: jde*xxxxxxx*. Do not use the J.D. Edwards convention to name your workspaces. Any Portal element (workspaces, components, and so forth) beginning with the characters jde might be overwritten when the system is upgraded.

4. Format the workspace as desired by adding or removing columns and components, changing colors, and defining different URLs to use for Portal elements.

   See *To change how a workspace looks* and *To change the layout of a workspace* for instructions for making these changes.

5. Click OK.

   If you make changes to several areas of the form, clicking OK in any section saves the changes made to all sections.

To cancel your changes, click the Portal hyperlink on the Workspace Navigation Bar instead of OK.

See Also

- *To change how a workspace looks*

- *To change the layout of a workspace*

► To set workspace permissions

In the following task, if you do not see a particular option, your account probably has not been configured to allow you access to the feature in question. Contact your system administrator for assistance.

1. From the Portal, click Personalize on the Workspace Navigation Bar.
2. On the Workspace Navigation Bar, choose a workspace from Select Workspace.

   Unless you are a system administrator, you can only choose the workspaces that you have created or to which you have been granted access.

3. In the Set Workspace Permissions section, click Permissions.

   Unless you are a system administrator, this hyperlink is available only for those workspaces that you own (that you have created) or to which you have been granted the view/edit/grant permission level of access privileges.

4. In the Add User Relationship for Workspace, choose one of the following options:
   - User

     If you want to define access for a user, choose the user option, and then enter the user's ID.

   - Role

     If you want to define access for a role, choose the role option, and then choose a role from the associated drop-down menu.

   - PUBLIC
5. From the menu below the Public option, choose the type of access to the workspace that you want to grant.
6. Click Update.

   The user, role, or *PUBLIC appears in the Edit/Remove Relationships for Workspace section of the form.

7. To delete a user, role, or *PUBLIC from the access list, click the checkbox in the Remove column next to the line to be deleted, and then click Update.
8. To add other users or roles to the access list, repeat steps 4-6.
9. When you are finished, click OK.

► To change how a workspace looks

1. From the Portal, click Personalize on the Workspace Navigation Bar.

If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. Choose a workspace from the Select Workspace field on the Workspace Navigation Bar.
3. To change the name or title of the workspace or to change on which devices the workspace should be available, change the corresponding fields in the Current Workspace section of the form.

   For more information about these fields, see *To create a new workspace*.

4. To change the colors of different Portal elements, click *Color Scheme Appearance* in the Workspace Details section of the form and perform the following actions:

   a. To choose a color scheme to set all of the Portal elements to different but compatible colors and textures, select a color scheme from the Predefined Schemes drop-down field in the Choose a Scheme for Your Workspace section.

      If you vary the colors or logos of any of the elements in the other sections on this page, you can return to the default values for the scheme by clicking Select (the button next to the Predefined Schemes drop-down field that looks like two right-pointing arrows).

   b. To choose a color for a specific Portal element, perform either of the following tasks in the Modify Workspace Scheme Colors section.

      The colors you choose in this section override your current color scheme, but will be lost if you choose a different color scheme.

      i. For a given element, click a color on its Color Palette.

      ii. For a given element, enter a hexadecimal value in the # field in the New Color column. Click Preview to render the hexadecimal color.

   c. To return an element to the color specified by the current color scheme, click Reset.

      When you choose different options (in any section of the page), the system reflects your choices in the Scheme Preview in the Choose a Scheme for Your Workspace section.

5. To use different images for the workspace's background, logo, and toolbars, enter an appropriate URL in the applicable fields in the Modify Workspace Images and Links section of the page.

   When you choose different options (in any section of the page), the system reflects your choices in the Scheme Preview in the Choose a Scheme for Your Workspace section.

6. When you are finished, click OK.

   If you make changes to several areas of the page, clicking OK in any section saves the changes made to all sections.

7. To cancel your changes, click Cancel in any section of the page.

See Also

- *To create a new workspace*

- *To incorporate your corporate theme into a workspace*

► To change the layout of a workspace

In the following task, if you do not see a particular option, your account probably has not been configured to allow you access to the feature in question. Contact your system administrator for assistance.

1. Click Personalize on the Workspace Navigation Bar.
2. On Personalize, choose a workspace from the Select Workspace field on the Workspace Navigation Bar.
3. In the Workspace Layout Design section of the form, change the number of columns by performing any of the following steps. All buttons are located under the Workspace Layout graphic. When you hover over a button, its title appears in the box between the Workspace Layout graphic and the buttons.
   a. Click *Add a new column* to add a new column to the workspace.

      New columns are added to the right. Workspaces can contain 1-5 columns.

   b. Choose a column and then click *Delete the highlighted column* to remove the column you chose from the workspace.

      If you delete a column with components in it, the components are removed from the workspace only. You cannot use the Personalize form to delete components from the system.

   c. Click *Reset to 3 empty columns* to clear all components and to create three equally-sized columns.
4. To change the width of a column, drag the column dividers to the right and left.

   If a component's width is wider than the column width that you set, the system overrides your column settings to accommodate the component.

5. Add, remove, and rearrange components in the workspace by performing any of the following steps:
   a. To add a component to a column, choose a column, choose the component that you want to add, and click the right-pointing arrow between the Available Components list and the Workspace Layout graphic.
   b. To remove a component from a column, choose the component that you want to remove and click the left-pointing arrow between the Available Components list and the Workspace Layout graphic.
   c. To move a component on the Workspace Layout to a different column, choose the component that you want to move and click the right- or left-pointing arrow below the Workspace Layout graphic, or drag and drop the component to a new location.
   d. To move a component on the Workspace Layout above or below other components in the column, choose the component that you want to move and click the up- or down-pointing arrow below the Workspace Layout graphic, or drag and drop the component to a new location.
6. To list the components by their IDs instead of by their names, click Sort by IDs.
7. When you are finished, click OK.

If you make changes to several areas of the form, clicking OK in any section saves the changes made to all sections.

► To alter hyperlinks on the Workspace Navigation Bar for a workspace

Component developers can make a special kind of component called a link component. When you add link components to the Workspace Navigation Bar, they appear as additional hyperlinks.

In the following task, if you do not see a particular option, your account probably has not been configured to allow you access to the feature in question. Contact your system administrator for assistance.

1. From the Portal, click Personalize on the Workspace Navigation Bar.
2. On the Workspace Navigation Bar, choose a workspace from the Select Workspace field.
3. In the Workspace Details section, click Workspace Navigation Bar Links.
4. Add, remove, and rearrange links by performing any of the following tasks:
   a. To add a link to the Workspace Navigation Bar, choose the link component that you want to add, and click the right-pointing arrow between the Available list and the Selected list.
   b. To remove a link from the Workspace Navigation Bar, choose the link that you want to remove and click the left-pointing arrow between the Available list and the Selected list.
   c. To rearrange the order of the links, choose the link that you want to move and click Move Up or Move Down.
5. To list the components by their IDs instead of by their names, click Sort by IDs.
6. When you are finished, click OK.

► To delete a workspace

You can delete any workspace that you create. However, when you delete a workspace, you do not delete its components from the system.

Caution

You cannot undo a workspace deletion. After you confirm that you want to delete a workspace, the workspace is permanently deleted from the system and cannot be retrieved.

1. Click Personalize on the Workspace Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, choose a workspace from the Select Workspace field on the Workspace Navigation Bar.
3. In the Current Workspace section of the form, click Delete Workspace.

   The system confirms the deletion.

4. Click Yes when the system asks if you really want to delete the workspace.

Incorporating Your Corporate Theme

You can alter a single workspace or the entire J.D. Edwards Portal to match the look and feel of an existing page or site. Changes you make to the jas.ini or portal.css files are global. Changes you make on the Personalize form override the global settings but are applied only to the current workspace.

► To incorporate your corporate theme

1. From the Portal, click Personalize on the Workspace Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, choose a workspace from the Select Workspace field on the Workspace Navigation Bar.
3. To change the background for the workspace area, perform one of the following processes:
   - To change the color, in the Workspace Look & Feel section, click the checkbox next to Workspace Background Color field, and perform one of the following processes:
     - Click a color on the Color Palette, and then click the paintbrush icon under the Color Palette.
     - Enter the hexadecimal value of a color in the Color Code field, and then click the paintbrush icon next to the value you entered.
   - To display an image in the workspace area, enter the URL of the image to display in the Workspace Background Image URL field in the Additional Personalization section of the form.

     Depending on the size of the image, the Portal might tile it. To remove a background image, delete the contents of this field.

4. To change the background for the Enterprise Navigation Bar, perform one of the following processes:
   - To change the color, in the Workspace Look & Feel section, click the checkbox next to Enterprise Navigation Bar Background Color field, and perform one of the following processes:
     - Click a color on the Color Palette, and then click the paintbrush icon under the Color Palette.
     - Enter the hexadecimal value of a color in the Color Code field, and then click the paintbrush icon next to the value you entered.
   - To display an image in the Enterprise Navigation Bar, enter the URL of the image to display in the Enterprise Navigation Background Image URL field in the Additional Personalization section of the form.

     Depending on the size of the image, the Portal might tile it. To remove a background image, delete the contents of this field.

5. To change the logo on the Enterprise Navigation Bar, enter the URL of the image in the Enterprise Navigation Logo Image URL field in the Additional Personalization section of the form.

   For best results, the referenced image should be between 100-120 pixels wide and 25-35 pixels tall.

   To use your logo as the default logo in all Portal workspaces, change the corplogourl setting in the [PORTAL] section of the jas.ini file to point to the image you want to use.

   To make the logo a clickable link, enter a target URL in the Enterprise Navigation Logo Link URL field. When the user clicks on the logo, the target URL appears.

6. To change the colors of different Portal elements, perform one of the following actions in the Workspace Look & Feel section of the form:

   - Select a color scheme from the drop-down menu.

     You can create a selectable color scheme based on your company's colors. These custom schemes are stored in the jas.ini file.

   - Click a color on the Color Palette, click the checkbox next to one or more elements that you want to change, and then click the paintbrush icon under the Color Palette to apply the color to the selected elements.
   - Enter the hexadecimal value of a color in the Color Code field, click the checkbox next to one or more elements that you want to change, and then click the paintbrush icon next to the value you entered to apply the color to the selected element.

7. To change font types, sizes, colors, and related elements, modify the Portal style sheet (portal.css).

   Changes to the Portal's style sheet are reflected in the entire Portal, not just the current workspace.

   Caution
   Changing a web page's style sheet can drastically alter the appearance of the page. For example, increasing a font's point size might displace or obscure some page elements. If you change the style sheet, make a note of your changes and test each one so you can revert to the previous setting if a change has a negative impact.

8. When finished, click OK.

   If you make changes to several areas of the form, clicking OK in any section saves the changes made to all sections.

   To cancel your changes, click the Portal hyperlink on the Workspace Navigation Bar instead of OK.

Portal Design

Portal Design

The flexible nature of the J.D. Edwards Portal is based on the ability of its users to design their own components. A component is an object that appears within the Portal to provide links, information, or other web- or network-related functions. There is no limit to the number of components that you can design or to the number of components that end users can include in their workspaces. You can make components selectable by end users so that they can freely place the components in their workspaces. You can also limit access to components and create system-wide workspaces that end users cannot configure.

Note

To design components and system-wide workspaces, you must have appropriate security access to the Portal. See *Portal Configuration* for more information.

System-wide workspaces are created in exactly the same way as personal workspaces. If your permissions are set properly, you can designate any workspace that you create as a system-wide workspace, making it available to end users. See *Using the Portal* for information about creating workspaces.

To make your components easier to find for inclusion on workspaces, you can group them in folders. You can also indicate that a component is part of a component group. In this context, a component group is a set of components designed for a specific task. Together, the components in a component group might share the same value for attributes like the protocol and the name of the server that hosts the components.

When you design components, the Portal Builder Servlet (Portal Builder) provides a variety of implementation strategies and interface choices. The base component, that is, a component in normal mode, can be designed with HTML code or based on a URI reference and can be designed for use with different form factors (a browser on a PC or on a Windows CE or Pocket PC device). You can also write a Java applet to use as the base component. The Portal Builder imbues all PC and Windows CE components with the ability to be minimized. A minimized component displays its title bar only. Additionally, you can provide any component with functionality for one or more of the following modes: maximize, help, and personalize.

If you provide functionality for a mode, then the Portal Builder includes an appropriate icon on the title bar that the user can click to launch the mode. All three of these alternate modes take up the entire workspace area. Additionally, you can provide functionality for a mode called link. The link mode functions exactly like the maximize mode, but it has no associated icon. If you provide a component with link mode functionality, you will also need to provide users with a way to access the mode from within the component.

In a typical design scenario, the designer renders the component's normal view with HTML, a URI, a Java applet, or a Portal Component Servlet. Then the designer creates either additional web pages for the other modes and then references them with URIs, JavaScripts, or Portal Component Servlets that can be invoked to provide functionality. You can use different solutions for different modes. For example, a component's maximize icon can call a URL while its personalize and help icons can invoke JavaScript functions.

In addition to the type of component designed to appear within a workspace, you can also create link components. Link components are simply hyperlinks pointing to any URL. These components appear as hyperlinks in the Workspace Navigation Bar. End users can add, remove, and rearrange the link components as they wish.

See Also

- *OneWorld Portal Configuration*
- *Using the OneWorld Portal*

Planning Portal Components

Regardless of the kind of components you design for use in the J.D. Edwards Portal, you should be aware of a number of factors that might affect your design decisions. Unless you component is a link component or you are designing it exclusively for Pocket PCs, it will share screen space with several other components when in normal mode. Components in this mode should adapt to changes in width to the greatest extent possible. If your component cannot adapt to changes in width, the Portal framework displays a horizontal scroll bar to allow users to access off-screen content.

A reasonable size for your component is approximately 220 pixels wide by 340 pixels high. The height of the component is not as critical as the width, because the component will push other components located beneath it down. Keeping components within these limits, however, will help to provide a much cleaner interface and easier interoperability with other components. The normal mode of the component should be relatively small. Remember that the portal framework supports a maximized view in which your component can use the majority of the screen. Normal mode should be a reasonably-sized introduction to your main content.

Your component is responsible for storing any information that it needs for personalization or to support its functionality, although cookies are managed by the Portal Builder. You may use any mechanism or technology to store your information. The Portal Component Servlet Java API for the J.D. Edwards Portal also provides a mechanism for storing XML Data in a database, but the Portal Builder Servlet itself is not involved in storing your component's data.

It is your responsibility to determine the browsers on which your components will function properly. Currently, the J.D. Edwards Portal supports the following browsers:

- Netscape 6.01 and higher for PCs
- Microsoft Internet Explorer 5.0 and higher for PCs
- Pocket IE 3.01 and higher for Pocket PCs
- Pocket IE 3.0 and higher for HPC/HPC Pros
- Pocket ID 4.0 and higher for HPC 2000s

If your target audience uses only one browser, your component can take advantage of the features provided by that browser. If you target a generic audience, you will want to use generic functions and ensure that your component works on multiple browsers. Furthermore, especially for pervasive devices, you should test the component for bandwidth. Not all environments can support high volumes of data exchange at acceptable rates.

Finally, pages within a component need to fully qualify its URL's for images and anchors or the Portal Builder Servlet's server information will be assumed. Consider the following example:

```
<html>
<body>
<image src="mypic.jpg">
</body>
</html>
```

In this example, mypic.jpg is a relative link. If this page was located at http://www.mysite.com/mypic.html, then the browser would look for mypic.jpg at http://www.mysite.com/mypic.jpg. Now consider a J.D. Edwards Portal, located at http://www.aap.com/servlet/... tries to call your component. When looking for your image, the browser would try and find mypic.jpg at http://www.aap.com/servlet/mypic.jpg. This problem can be solved by changing the image tag to read:

```
<img src="http://www.mysite.com/mypic.jpg">
```

When building fully qualified URLs for a external (browser) request for a resource, use the externalhost setting in the LOGIN section of the JAS.ini. For example, you would need to set externalhost when the Portal is running behind a load balancer such as a Cisco Local Director. With more advanced load balancing such as port swapping (where the request to the load balancer comes in on one port, but the load balancer forwards the request to the Portal on a different port), you must build external URLs that reference the original machine/port, rather than then machine/port from which the Portal received the response.

For example: A Cisco Local Directory is setup with an Name of CLD such that the command 'ping CLD' pings the Local Director. The local director is set to forward any requests is receives to JASSERVER:PORT. In this way, the Local Directory can load balance to multiple JAS instances running on the same machine, but listening on different ports.

Note

Sticky load balancing with a sticky time greater than the session timeout internal is required to use the Portal with a local director at this time.

Consider the following additional issues when planning new Portal components:

- Will you store your component on a server or in an area accessible only to the J.D. Edwards Portal?

  If the component should be accessible to the J.D. Edwards Portal only, then it must be an HTML component.

  If the Portal can access the component via a server, then you can use straight HTML (including ActiveX components), Java applets, URI components, Isolated URI components, or Portal Component Servlets. If you create an interactive URI, J. D. Edwards recommends using Java or JSP because the Portal itself is built on this technology, so integrating components built with Java or JSP is easier. If you need to create a Portal Component Servlet (that is, a component that requires access to the database information, sign on information, or both), then your component just be a Java component. Note that applet tags reside within the Portal itself and are not accessible outside the Portal, even though the applet itself can be accessed outside of the Portal.

- Will your component need server-side programs to support it?

These programs can be in almost any programming language that is supported by the server that is hosting the component. If you must use server-side programs, you will probably need to create a URI, Isolated URI, or Portal Component Servlet. These three component types can also include applets and ActiveX controls.

- What type of component will you be building?
  - Link components have no function other than to provide end users with a hyperlink pointing to a URL. They are fast and useful for providing users access to URLs that are incompatible with the Portal. You can take advantage of passthrough functions with Link components as well.
  - HTML components require no interaction with an external server, so they provide quick response with very little external resource load. They can contain references to applets, ActiveX controls, and images, but the HTML is taken directly from the J.D. Edwards Portal database, so the content is static. HTML components inherit the Portal's cascading style sheet, so they blend effortlessly with whatever workspace in which they are included. You can take advantage of language and passthrough functions with HTML components as well.
  - URI components always require interaction between the J.D. Edwards Portal and a web server. These components can be written as CGI, Java servlets, or HTML pages, and they can reside on a server other than the Portal server. URI components tend to be slower than the other components, but they allow you to use the inherited trust and passthrough features.
  - Isolated URI components are the most flexible component type because they are not actually integrated into the Portal. They allow you full access to the browser frame and to take existing Web content and display it in the Portal. On pervasive devices, the interface tends to be more clumsy because Isolated URI components do not include i-frame tags. You can take advantage of inherited trust with Isolated URI components.
  - Portal Component Servlets must be written in Java, and they must derive themselves from the PortalComponentServlet class. This class must be accessible from the PortalBuilderServlet through the classpath. The components might need to exist on the Web Server if you want to access them directly. These components are essentially Java servlets, but they implement a different API. Portal Component Servlets are fully integrated into the Portal; they provide the greatest support for pervasive devices and language.
  - Applet components require the browser to access an applet from a web server and to support Java. This applet runs on the client side and requires no resources other than a place to download your applet's classes. Applet components are dynamic because of their direct server connection, and they can be made to inherit cascading style sheet information from the Portal. This type of component requires the most testing out of all of the components because it is dependent on the user's Java Virtual Machine (JVM), browser, and operating system. Currently, no pervasive devices support Java, so they cannot display Applet components.
- Will your component provide access to J.D. Edwards applications, require database access using the ComponentDatabaseInterface, or need to reference the session information?

  To access a J.D. Edwards application, or to interact with the active session (such as when you need to access or pass a user ID), you must create a Portal Component Servlet. Portal Component Servlets are essentially URI components, but do not require a fully-qualified URL; Portal Component Servlets use a class and package name instead. Additionally, Portal Component Servlets must be located on the same server as the Portal because they must be able to access session information.

- Will your component display help or copyright information to users, or need a maximized view or personalization pages?

  To display information in help, maximized, or personalization mode, you must enable the mode for the component and then provide material to be displayed. If this information contains multiple pages, HTML probably will not work because it supports only one page. You can use the Java APIs in the J.D. Edwards Portal to wrap multiple pages into the component, if necessary.

  Additionally, ensure you fully qualify graphics from external locations. Keep in mind that all three of these modes take up the entire workspace of the Portal.

The design approach that you take and the type of component that you create depend on the specifications that you outline here.

Building Link Components

Link components have no function other than to provide end users with a hyperlink pointing to a URL. They are fast and useful for providing users access to URLs that are incompatible with the Portal. You can take advantage of passthrough functions with Link components as well. A link component can appear as a component in a workspace, or you can configure the component to appear as a hyperlink on the Workspace Navigation Bar for a particular workspace.

See Also

- *Planning Portal Components* for more information about determining what kind of component is appropriate for your needs
- *Setting Component Permissions* for instructions on granting access rights to a component

► To create a link component

1. From any J.D. Edwards Portal workspace, click Personalize on the Workspace Navigation toolbar.
2. On Personalize, click the Components hyperlink.
3. In the Create New Component section of the form, choose Link Component in the Component Type field.
4. To put the component in a folder other than the Root, in the Edit Existing Components section of the form, click the folder in which you want the new component to appear.

   Click Add New Folder to create a new folder for the component. For more information about folders and folder creation, see *Organizing Components*.

5. In the Create New Component section of the form, click Create.
6. In the Link Component Entry Form section, complete the following fields:

   - Component ID

     The system name for the component. It is the name that appears when the system provides lists of components for a user to choose from. The maximum field length is 10 characters.

   - Component Name

The name for the component as it appears in the component title bar in the J.D. Edwards Portal. The maximum field length is 30 characters.

- Devices

  Indicates in which devices and environments the component can be viewed.

- Icon Image URL

  If you want the link component to display an image, type the URL for the image in this field. The image appears only when the link component is included on a workspace; it does not appear when the link component is included in the Workspace Navigation Bar.

- Link

  Type the path code for the link target.

- Type

  Choose the target type from the drop-down menu. For J.D. Edwards HTML Application, Portal Component Servlet Class Name, or URL, click Open in New Window if you want the target to appear in a new browser window.

  Additionally, for URL target types, choose Use isolation if you want the URL to appear in a frame independent of the Portal.

7. If the component is part of a set of components for a component group, choose the component group from the drop-down list in the Component Group field.
8. To display the component in additional folders, perform one of the following tasks:
   - To place the component in the Root folder, perform the following steps:
     i. In the Available column, browse and display the contents of the Root folder.
     ii. Ensure none of the folders are highlighted and click the right-pointing arrow.

     To deselect a highlighted folder, hold down the Control button on your keyboard and click the highlighted folder.

   - To place the component in a folder other than the Root folder, perform the following step:
     - Choose a folder in the Available column under Parent Folders and click the right-pointing arrow.

   The component appears in all folders listed in the Selected column.

   Note
   All folders and components appear in the *ALL folder. You cannot select or deselect the *ALL folder.

9. To prevent the component from appearing in a folder, click a folder in the Selected column and click the left-pointing arrow.
10. To see the component as it will appear to an end user, click Preview.

    In Preview Window, click Close Window to return to the component entry form.

11. When you have finished creating the component, click OK.

    The system creates the component.

    Click Cancel to return to the component creation section without saving the component.

► To alter hyperlinks on the Workspace Navigation Bar for a workspace

Component developers can make a special kind of component called a link component. When you add link components to the Workspace Navigation Bar, they appear as additional hyperlinks.

In the following task, if you do not see a particular option, your account probably has not been configured to allow you access to the feature in question. Contact your system administrator for assistance.

1. From the Portal, click Personalize on the Workspace Navigation Bar.
2. On the Workspace Navigation Bar, choose a workspace from the Select Workspace field.
3. In the Workspace Details section, click Workspace Navigation Bar Links.
4. Add, remove, and rearrange links by performing any of the following tasks:
   a. To add a link to the Workspace Navigation Bar, choose the link component that you want to add, and click the right-pointing arrow between the Available list and the Selected list.
   b. To remove a link from the Workspace Navigation Bar, choose the link that you want to remove and click the left-pointing arrow between the Available list and the Selected list.
   c. To rearrange the order of the links, choose the link that you want to move and click Move Up or Move Down.
5. To list the components by their IDs instead of by their names, click Sort by IDs.
6. When you are finished, click OK.

Building HTML Components

HTML components require no interaction with an external server. They can contain references to applets, ActiveX controls, and images, but the HTML is taken directly from the Portal Builder's database. Additionally, HTML components cannot contain multiple pages.

See Also

- *Planning Portal Components* for more information about determining what kind of component is appropriate for your needs
- *Setting Component Permissions* for instructions on granting access rights to a component

► To create an HTML component

1. From any J.D. Edwards Portal workspace, click Personalize on the Secondary Navigation Bar.

If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, click the Components hyperlink.
3. In the Create New Component section of the form, choose HTML Component in the Component Type field.
4. To put the component in a folder other than the Root, in the Edit Existing Components section of the form, click the folder in which you want the new component to appear.

   Click Add New Folder to create a new folder for the component. For more information about folders and folder creation, see *Organizing Components*.

5. In the Create New Component section of the form, click Create.
6. In the HTML Component Entry Form section, complete the following fields:

   - Component ID

     The unique system ID for the component. It is the ID that appears when the system provides lists of components for a user to choose from. The maximum field length is 10 characters.

   - Component Name

     The name for the component as it appears in the component title bar in the J.D. Edwards Portal. The maximum field length is 30 characters.

   - Devices

     Indicates in which devices and environments the component can be viewed.

   - Display

     Indicates whether to display the component title bar in the J.D. Edwards Portal. The title bar does not appear when this option is checked. Because the mode icons reside on the component's title bar, hiding the title bar will prevent users from accessing any associated modes or other Portal-enabled component features such as the minimize feature.

   - Enter/Edit HTML for this component below

     The HTML code (up to 30,000 characters) that defines your component in normal mode. If you use <A> tags, the tags should include a TARGET reference to a new window so that the Portal is not replaced by the content to which the user navigated.

     You can use DHTML code, but be sure to name any controls or JavaScript functions with the following prefix to avoid naming conflicts: com_companyname_applicationname.

   - Personalize

     The URL target to display, Portal Component Servlet to instantiate, or JavaScript to execute when the user clicks the Personalize icon. The maximum field length is 256 characters.

     To display the content of a URL target, choose URL, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Help

  The URL target to display, Portal Component Servlet to instantiate, or JavaScript to execute when the user clicks the Help icon. The maximum field length is 256 characters. To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

  To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

  Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Expand

  The URL target to display, Portal Component Servlet to instantiate, or JavaScript to execute when the user clicks the Expand icon. The maximum field length is 256 characters. To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

  To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

  Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

Note
Avoid tags such as META, HTML, HEAD, and BODY. Do not use the FRAMESET tag.

---

7. If the component is part of a set of components for a component group, choose the component group from the drop-down list in the Component Group field.
8. To display the component in additional folders, perform one of the following tasks:
   - To place the component in the Root folder, perform the following steps:
     i. In the Available column, browse and display the contents of the Root folder.
     ii. Ensure none of the folders are highlighted and click the right-pointing arrow.

     To deselect a highlighted folder, hold down the Control button on your keyboard and click the highlighted folder.
   - To place the component in a folder other than the Root folder, perform the following step:
     - Choose a folder in the Available column under Parent Folders and click the right-pointing arrow.

   The component appears in all folders listed in the Selected column.

Note
All folders and components appear in the *ALL folder. You cannot select or deselect the *ALL folder.

9. To prevent the component from appearing in a folder, click a folder in the Selected column and click the left-pointing arrow.
10. To see the component as it will appear to an end user, click Preview.

    In Preview Window, click Close Window to return to the component entry form.

11. When you have finished creating the component, click OK.

    The system creates the component.

    Click Cancel to return to the component creation section without saving the component.

Example: Designing an HTML Component

Note

The steps in this example describe creating a component using the Portal. Although the process varies slightly, the HTML code itself is identical whether you are using the Portal or the J.D. Edwards application.

► To create the component

In this part of the example, you will create an HTML component called Hello J.D. Edwards.

1. From any J.D. Edwards Portal workspace, click Personalize on the Secondary Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, click the Components hyperlink.
3. In the Create New Component section of the form, choose HTML Component in the Component Type field, and click Create.
4. Fill out the following fields as indicated:
   - Component ID

     Type : OWPHELLOOW

   - Component Name

     Type: Hello J.D. Edwards

   - Devices

     Choose Desktops and Laptops.

- Enter/Edit HTML for this component below

  Type: `<b>Hello J.D. Edwards</b>`

5. Ensure that the remaining fields are blank, and then click OK.

   The Component Manager appears. Your new component, OWPHELLOOW, appears in the component list.

6. Test the component by adding it to a workspace.

   In the Portal, you should see a component, in boldface, that says, "Hello J.D. Edwards." The Personalize, Help, and Expand buttons are not available for your component because you left those sections of the component entry form blank. The next part of the process describes how to enable these modes for the component.

► To add personalization mode JavaScript

In this part of the component creation process, you will add JavaScript to provide a function for the component's personalization icon. Although this example is applied to the personalization mode specifically, the process is identical for adding functionality to the help and expand functions as well.

1. From any J.D. Edwards Portal workspace, click Personalize on the Secondary Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. Click Components.
3. In the Edit Existing Components section, find and select the OWPHELLOOW component that you created at the beginning of this example, and then click Modify Selected.

   The HTML Component Entry form appears, displaying the code that you have entered for the component to this point.

4. Add the following script to the *Enter/Edit HTML for This Component Below* field (add the code above the existing code):

```
<script>
function com_jdewards_hellojde_personalization(){
alert("This is my personalize box");
}
</script>
```

5. In the Personalize section of the form, select the Javascript option, and then enter the following in the text field:

   `com_jdewards_hellojde_personalization();`

6. Click OK, and then return to the workspace view of the Portal to test your changes.

A personalization icon has been added to the component's title bar. When you click it, an Alert box appears, displaying a message that reads "This is my personalize box."

Building URI Components

URI components always require interaction between the J.D. Edwards Portal and a web server. These components can be written as CGI, Java servlets, or HTML pages.

Java URI components should be packaged as:

`com.`*`yourcompany`*`.portal.components:`*`compname`*

where *yourcompany* is the name of your company and *compname* is the name of the component.

See Also

- *Planning Portal Components* for more information about determining what kind of component is appropriate for your needs
- *Setting Component Permissions* for instructions on granting access rights to a component

► To create a URI component

1. From any J.D. Edwards Portal workspace, click Personalize on the Secondary Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, click the Components hyperlink.
3. In the Create New Component section of the form, choose URI Component in the Component Type field.
4. To put the component in a folder other than the Root, in the Edit Existing Components section of the form, click the folder in which you want the new component to appear.

   Click Add New Folder to create a new folder for the component. For more information about folders and folder creation, see *Organizing Components*.

5. In the Create New Component section of the form, click Create.
6. In the URI (Uniform Resource Identifier) Component Entry Form section of the form, complete the following fields, and then click OK:

   - Component ID

     The system ID for the component. It is the name that appears when the system provides lists of components for a user to choose from. The maximum field length is 10 characters.

   - Component Name

     The name for the component as it appears in the component title bar in the J.D. Edwards Portal. The maximum field length is 30 characters.

   - Devices

     Indicates in which devices and environments the component can be viewed.

   - Display

Indicates whether to display the component title bar in the J.D. Edwards Portal. The title bar does not appear when this option is checked. Because the mode icons reside on the component's title bar, hiding the title bar will prevent users from accessing any associated modes or other Portal-enabled component features such as the minimize feature.

- Inherited Trust

  Indicates whether the component can use the current session user ID and login to log in to the component’s application.

- Support cookies

  Indicates whether the component can support cookies.

- Component URL

  The URL of the component. If you use <A> tags, the tags should include a TARGET reference to a new window so that the Portal is not replaced by the content to which the user navigates.

  You can use DHTML code, but be sure to name any controls or JavaScript functions with the prefix com_companyname_applicationame to avoid naming conflicts.

- Personalize

  The URL target to display, Portal Component Servlet to instantiate, or JavaScript to execute when the user clicks the Personalize icon. The maximum field length is 256 characters.

  To display the content of a URL target, choose URL, and then enter the URL in the space provided.

  To execute JavaScript, choose Javascript, and then enter the complete JavaScript function call (including the parentheses and ending semicolon).

  Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Help

  The URL target to display, Portal Component Servlet to instantiate, or JavaScript to execute when the user clicks the Help icon. The maximum field length is 256 characters. To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

  To display the content of a URL target, choose URL, and then enter the URL in the space provided.

  To execute JavaScript, choose Javascript, and then enter the complete JavaScript function call (including the parentheses and ending semicolon).

  Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Expand

The URL target to display, Portal Component Servlet to instantiate, or JavaScript to execute when the user clicks the Expand icon. The maximum field length is 256 characters. To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To display the content of a URL target, choose URL, and then enter the URL in the space provided.

To execute JavaScript, choose Javascript, and then enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

7. If the component is part of a set of components for a component group, choose the component group from the drop-down list in the Component Group field.
8. To display the component in additional folders, perform one of the following tasks:
   - To place the component in the Root folder, perform the following steps:
     i. In the Available column, browse and display the contents of the Root folder.
     ii. Ensure none of the folders are highlighted and click the right-pointing arrow.

     To deselect a highlighted folder, hold down the Control button on your keyboard and click the highlighted folder.
   - To place the component in a folder other than the Root folder, perform the following step:
     - Choose a folder in the Available column under Parent Folders and click the right-pointing arrow.

   The component appears in all folders listed in the Selected column.

   Note
   All folders and components appear in the *ALL folder. You cannot select or deselect the *ALL folder.

9. To prevent the component from appearing in a folder, click a folder in the Selected column and click the left-pointing arrow.
10. To see the component as it will appear to an end user, click Preview.

    In Preview Window, click Close Window to return to the component entry form.
11. When you have finished creating the component, click OK.

    The system creates the component.

    Click Cancel to return to the component creation section without saving the component.

Calling URLs

If your component must interact with server-side programs, your client-side code should open a new browser frame from which the client-side code interacts with the server without

replacing the Portal's presentation. If the interaction is one-time in nature (such as an HTML form submission) then the server-side program should use the PBSURL request parameter to redirect the resulting response. See *Wrapping URIs* for more information about this parameter.

To support language localization, all URLs entered via the Component Manager can contain the string:

```
*L;
```

This string is replaced with the J.D. Edwards language code string currently in use by the Portal framework. This is known as run-time substitution. The purpose of this is to permit component builders to direct URLs to language-specific directories or files when necessary.

For example, if the user supplied the URL `http://www.myplace.com/lang-*L;/help.html` and the current language code string was E, then the URL would be changed to `http://www.myplace.com/lang-E/help.html` before it was requested.

Similarly, if the user supplied the URL `http://www.myplace.com/help-*L;.html` and the current language code string was F, then the URL would be changed to `http://www.myplace.com/help-F.html` before it was requested.

If the current language is the user's default language, then the code is blank.

The following table lists the J.D. Edwards language codes:

Language Preference	Description	ISO639LanguageCode
	Domestic Language	en
AR	Arabic	ar
C	Czech	cs
CS	Chinese Simplified	zh_CN
CT	Chinese Traditional	zh_TW
DN	Danish	da
DU	Dutch	nl
E	English	en
F	French	fr
FN	Finnish	fi
G	German	de
GR	Greek	el
HU	Hungarian	hu
I	Italian	it
J	Japanese	ja
KO	Korean	ko

Language Preference	Description	ISO639LanguageCode
NO	Norwegian	no
P	Portuguese	pt
PO	Polish	pl
RU	Russion	ru
S	Spanish	es
TH	Thai	th
TR	Turkish	tr
W	Swedish	sv

Note

The J.D. Edwards Language Preference is blank for Domestic Language.

Wrapping URIs

The Portal provides four JavaScript functions that allow you to wrap URIs into the current form. Thus, you can provide multiple pages of material. The four functions are:

- showPersonalizationPage(compname, uri, addextra)

  Use this function to wrap a URI into the personalize mode of a component.

- showHelpPage(compname, uri, addextra)

  Use this function to wrap a URI into the help mode of a component.

- showMaxPage(compname, uri, addextra)

  Use this function to wrap a URI into the maximize mode of a component.

- showLinkPage(compname, uri, addextra)

  Use this function to wrap an external URI into a component without associating a particular mode with the function. The Portal displays the URI in a full-workspace view, identical to the maximize mode, except that the component title bar does not display a mode name.

The arguments for the functions have the following characteristics:

compname The name of the component calling the URI. You can obtain this name from a Java servlet; however, if you are using DHTML, you must hard code the component name into the HTML.

uri The URI to call. Include any parameters (that is, the ?parm=value pairs) that are required to call the page.

addextra An option to send and receive extra parameters. True enables the extra parameters; false disables the feature. These parameters are not user configurable; that is, the component either passes the entire set as described below or not. The parameters are:

- jsessionid - A reference to the session ID. For the purpose of component design, you always use jsessionid to retrieve the session id of the user, as opposed to sessionId or sessionid.
- sessionId - A redundant reference to the session ID. To be removed in future releases.
- sessionid - A redundant reference to the session ID. To be removed in future releases.
- LANGCODE - The language code identifying the current user language preference. This value currently is not used.
- PBSURL - The URI to the Portal Builder Servlet.
- COMPNAME - The name of the component calling the URI as it exists in the J.D. Edwards Portal Database.

Obtaining User Information

To acquire user name and ID from an HTML or Applet component, use a script tag that references the /servlet/com.jdedewards.portal.js.PortalScript servlet. This tag returns a script called JDEUserData which provides the following information:

JDEUserData.name The user's name. If J.D. Edwards Name is unavailable, this variable returns the user's ID instead.

JDEUserData.address.number The user's address book number.

JDEUserData.userId The user's J.D. Edwards ID.

JDEUSERDATA.languagePreference The user's language preference (JDElang). The system can return this data only if provided with a valid address book record.

For example, consider the following code fragment:

```
<html>
<body>
<script language="JavaScript" type="text/javascript"
src="/servlet/com.jdedwards.portal.js.PortalScript">
</script>
<script>
document.write("Hello"+document.JDEUserData.userId);
```

```
</script>
</body>
</html>
```

If a user logged in as SO1234567, the above code would display the following string:

```
Hello SO1234567
```

Inherited Trust

When a component provides access to another application that requires a login, the system prompts the user to log in to that application. With inherited trust, however, you can set URI and Isolated URI components so that the Portal logs in for the user instead. After an initial log in, the log in to that specific application with that specific component is transparent to the user.

To apply inherited trust to a URI or Isolated URI component, select the Inherited Trust option when you create the component.

The Portal's Inherited Trust Manager servlet manages all inherited trust requests. To implement inherited trust, the component accesses this manager by building a URL from the given Portal host name, Portal port, and Portal protocol. For example:

```
protocol://hostname:port/jde/servlet/com.jdedwards.portal.Inhe
ritedTrustManager
```

The Portal Request ID provides a method to determine the age of the request. The Inherited Trust Manager accesses the static list of Request IDs on Portal server, further limiting the possibility of hacking into the system.

The target application must be set to match the password and other information to the user ID. For example, these values can be encrypted in a database using a Secret Key. On each login attempt after the initial log in, the application looks up the password and other information for the user ID, decrypts it, and proceeds with the login as if the user had typed them into the login form.

Each time a manual sign on occurs (such as logging into the application without the Portal), the database entry is checked to see if the password or other information has changed. If it has changed, the database entry is cleared for that user ID and the new data is updated.

Secret Enterprise Key

A secret enterprise key is what makes Inherited Trust logins secure. All applications that wish to use inherited trust must know this key—if it gets changed in one place, it must be changed in all others. If an application is aware of this key, trust is established.

To specify this key on the J.D. Edwards Portal, use the init-parameter called "secretEnterpriseKey" as in the following example:

```
    <servlet>
        <servlet-name>
            com.jdedwards.portal.InheritedTrustManagerServlet
        </servlet-name>
        <servlet-class>
```

```
            com.jdedwards.portal.InheritedTrustManagerServlet
        </servlet-class>
        <init-param>
            <param-name>secretEnterpriseKey</param-name>
            <param-value>this is my secret enterprise key</param-value>
        </init-param>
    </servlet>
```

Specifying the key in this manner prevents it from being accessed via other servlets that might attempt to retrieve this value for malicious use.

ASP Implementation

If your component is written using ASP, you must have the provided COM DLL that interfaces with the Inherited Trust Manager Servlet. Your component does not have to be served by the same computer, only both computers must be able to interact with one another. The DLL file is located in the jdewww/owportal/ directory provided by the install CD.

When you enable the Inherited Trust option for a URI or an Isolated URI component, the following set of parameters is sent to your component:

```
'Get variables required for Inherited Trust
Dim intpbsurl, jdeowpUserID, jdeowpRequestID, jdeowpReferer, jdeowpProtocol,
jdeowpServer, jdeowpPort
jdeowpUserID = Request.QueryString("jdeowpUserID")
jdeowpRequestID = Request.QueryString("jdeowpRequestID")


intpbsurl = Request.QueryString("intpbsurl")
If intpbsurl = "" Then
   'Build server/port from the referer
   jdeowpReferer = Request.ServerVariables("HTTP_REFERER")
   jdeowpProtocol = Split(jdeowpReferer,":")(0)
   jdeowpServer = Split(jdeowpReferer,"/")(2)
   If UBound(Split(jdeowpServer,":")) = 1 Then
          jdeowpPort = Split(jdeowpServer,":")(1)
          jdeowpServer = Split(jdeowpServer,":")(0)
   Else
          jdeowpPort = "80"
   End If
Else
   'Build server/port from intpbsurl (needed when a local director is
involved)
   jdeowpProtocol = Split(intpbsurl,":")(0)
```

```
    jdeowpServer = Split(intpbsurl,"/")(2)
    If UBound(Split(jdeowpServer,":")) >= 1 Then
           jdeowpPort = cStr(Split(jdeowpServer,":")(1))
           jdeowpServer = Split(jdeowpServer,":")(0)
    Else 'Port not given, use default port
           jdeowpPort = "80"
           jdeowpServer = Split(jdeowpServer,":")(0)
    End If
End If
```

After these variables are stored, the page can interface with the COM DLL as follows:

```
Dim iTrust, iTrustResult, iTrustPassword, iTrustProject
If Request.QueryString("PortalUserID") <> "" Then
'Uses a COM DLL:
    'To Register Run (once):  regsvr32.exe pathToDll/InheritedTrust.dll
    'To Unregister Run:  regsvr32.exe /u pathToDll/InheritedTrust.dll

'Execute the Inherited Trust Model
    Dim secretEnterpriseKey
    secretEnterpriseKey = "this is my secret enterprise key"
    Set iTrust =
Server.CreateObject("InheritedTrust.InheritedTrustForPortal")
    iTrustResult =
iTrust.VerifyInheritedTrust(CStr(Request.QueryString("PortalProtocol")), _
    Cstr(Request.QueryString("PortalServer")), _
    Cstr(Request.QueryString("PortalPort")), _
    Cstr(Request.QueryString("PortalRequestID")), _
    Cstr(Request.QueryString("PortalUserID")), _
    Cstr(secretEnterpriseKey) )
    If iTrustResult = "true" Then
           'It is safe to assume that the PortalUserID is really the user
trying to get in
           'Get the User ID/Password/Project from the Database and log the
user in
    End If
End If
```

Java Implementation

If your component is written in Java, you must have the inheritedtrust.jar file in your classpath to interface with the Inherited Trust Manager Servlet. Your component does not have to be served by the same computer, only both computers must be able to interact with one another. The jar file is located in the jdewww/owportal/ directory provided by the install CD.

When you enable the Inherited Trust option for a URI or an Isolated URI component, the following set of parameters is sent to your component:

- `intpbsurl`
- `jdeowpUserID`
- `jdeowpRequestID`

To interface with the Inherited Trust library, the server protocol, server IP address, and server port must be parsed out of the intpbsurl parameter. The other two values are passed, and the secretKey must be the secret enterprise key as specified in the init-parameter of the Inherited Trust Manager Servlet. The method to use from the JAR file is:

```
com.jdedwards.portal.inheritedtrust.InheritedTrustForPortal.VerifyInheritedT
rust(

String protocol, String server, String port,

String requestID,  portalUserID, String secretKey);
```

This method returns true if Inherited Trust is verified and false if it failed. Failure may be caused by mismatching enterprise keys, too much time passed from initial request until verification, or a network error.

Example: Designing a URI Component

Create a text file on a web server that contains the following text:

```
<html>

<body>

<script>

/*

* This function parses &-separated name=value pairs.

/

function com_jdedwards_HelloJDE_getArgs(){

var args = new Object();

var query = location.search.substring(1);

var pairs = query.split("&");

for (var i = 0; i < pairs.length; i++){

var pos = pairs[i].indexOf(`=');

if (pos == -1) continue;

var argname = pairs[i].substring(0,pos);
```

```
var value = pairs[i].substring(pos+1);
args[argname]=unescape(value);
}
var args = com_jdedwards_HelloJDE_getArgs();
if (args.mode == null)
document.writeln("<b>Hello J.D. Edwards</b>");
else if (args.mode=="personalize")
document.writeln("This is my personalization screen for Hello J.D.
Edwards");
</script>
</body>
</html>
```

When the page is loaded into the browser, the JavaScript is executed. First it retrieves the arguments from the URL. If the mode argument does not exist, then "Hello J.D. Edwards" is displayed in bold letters. If mode = personalize, then "This is my personalization screen for J.D. Edwards" is displayed instead.

To test the functionality of the component, save your work and point your browser to the URL of the component. The screen should display "Hello J.D. Edwards" in bold letters. To test the personalize function, add "?mode=personalize" to the URL. Now the page should display "This is my personalization screen for J.D. Edwards."

► To create the component

The first step in the process of designing a URI component is to create the actual URI component.

1. From any J.D. Edwards Portal workspace, click Personalize on the Secondary Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. In the Create New Component section of the form, choose URI Component in the Component Type field, and click Create.
3. In the URI (Uniform Resource Identifier) Component Entry Form section of the form, complete the following fields as indicated:
   - Component ID

     Type: OWPHELLOO2

   - Component Name

     Type: Hello J.D. Edwards Number 2

   - Devices

     Choose Desktops and Laptops

   - Component URL

Enter the URL of the text file that you created before you started this process. Even though you did not create the file using Portal tools, it is technically a component.

4. Choose the URL option in the Personalize section, and then enter the same URL that you entered in the Component URL field with ?mode=personalize appended to the end of it.
5. Leave the remaining fields blank, and then click OK.

   The Component Manager appears. Your new component, OWPHELLOOW2, appears in the component list.

6. Test the component by adding it to a workspace.

   In the Portal, you should see a component, in boldface, that says "Hello J.D. Edwards Number 2." Click the Personalize icon to see the message "This is my personalization screen for J.D. Edwards."

7. Modify the OWPHELLOO2 file so that it reads as follows:

```
<html>
<body>
<script>
/*
* This function parses &-seperated name=value pairs.
*/
function com_jdedwards_HelloJDE_getArgs(){
var args = new Object();
var query = location.search.substring(1);
var pairs = query.split("&");
for (var i = 0; i < pairs.length; i++){
var pos = pairs[i].indexOf(`=');
if (pos == -1) continue;
var argname = pairs[i].substring(0,pos);
var value = pairs[i].substring(pos+1);
args[argname]=unescape(value);
}
}
function com_jdedwards_HelloJDE_show(){
if (document.JDEPortal != null)
showPersonalizationPage(args.COMPNAME,"http://localhost/jde/owportal
/components/HelloJDE.html?mode=personalize2", true);
else
window.location="http://localhost/jde/owportal/components/HelloJDE.h
tml?mode=personalize2";
```

```
}
var args = com_jdedwards_HelloJDE_getArgs();

if (args.mode == null)

document.writeln("<b>Hello J.D. Edwards</b>");

else if (args.mode=="personalize")

{

document.writeln("<p>This is my personalization screen for Hello
J.D. Edwards.</p>");

document.writeln(`<p><a href="JavaScript:
com_jdedwards_HelloJDE_show()">');

document.writeln("Goto Personalize 2</a></p>");

}

else if (args.mode=="personalize2")
{

document.writeln("<p>This is the second personalization page</p>");

document.writeln("<p> jsessionid="+args.jsessionid+"<br>");

document.writeln("LANGCODE="+args.LANGCODE+"<br>");

document.writeln("PBSURL="+args.PBSURL+"<br>");

document.writeln("COMPNAME="+args.COMPNAME+"<br></p>");

}

</script>

</body>

</html>
```

Note that the com_jdedwards_HelloJDE_show() function checks to see if the calling component is currently displayed in the J.D. Edwards Portal. If it is, then document.JDEPortal will never be null. If it is not, then document.JDEPortal will be null. This check is necessary to support the J.D. Edwards standard that Portal components be able to function as well alone as within the Portal. Because the JavaScripts used in this example are specific to the Portal, you should always check to ensure that the J.D. Edwards Portal is displaying an object before using one of these scripts. In this example, if the J.D. Edwards Portal is unavailable, the code calls the URL directly.

After the com_jdedwards_HelloJDE_show() function checks for the J.D. Edwards Portal and finds that it is available, it then executes the showPersonalizationPage function. Notice that the component name can be obtained from the arguments because the COMPNAME is provided to the personalization page by default. The second argument references a text file called HelloJDE.html, and the last argument tells the J.D. Edwards Portal to send the extra arguments to the component.

This example also displays the second personalization page when mode=personalize2. This function lists a page that displays all of the parameters provided by the Portal Builder Servlet.

Typically, you will want to gather the extra component information as in this example; however, depending on the source of the URI, you might not always find it advantageous to do so. For example, you might have a component that provides data and exists on a foreign host such as Yahoo that returns server-generated errors if it receives invalid parameters (as

a protection against hacking). In this scenario, passing the extra parameters to a foreign host might prevent this page from loading. Of course, if the information is not provided, then the component will know nothing about itself and might not be able to display additional information in the J.D. Edwards Portal.

The showMaxPage, showHelpPage, and showLinkPage functions work the same way as the showPersonalizationPage function in this example. Additionally, you set up showFramedLinkPage as you do showPersonalizationPage. However, showFramedLinkPage displays the component and the Portal's two navigation bars each in their own frame.

Building Isolated URI Components

Like URI components, Isolated URI components always require interaction between the J.D. Edwards Portal and a web server. These components can be written as CGI, Java servlets, or HTML pages. Isolated URI components differ from regular URI components in that they function completely outside of the Portal in their own frame. Consequently, Isolated URI components are the most flexible of all Portal components. On the other hand, Isolated URI components cannot take advantage of Portal functionality the way the other component types can.

Java URI components should be packaged as:

`com.`*`yourcompany`*`.portal.components:`*`compname`*

where *yourcompany* is the name of your company and *compname* is the name of the component.

See Also

- *Planning Portal Components* for more information about determining what kind of component is appropriate for your needs
- *Setting Component Permissions* for instructions on granting access rights to a component

► To create an Isolated URI component

1. From any J.D. Edwards Portal workspace, click Personalize on the Secondary Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, click the Components hyperlink.
3. In the Create New Component section of the form, choose Isolated URI Component in the Component Type field.
4. To put the component in a folder other than the Root, in the Edit Existing Components section of the form, click the folder in which you want the new component to appear.

   Click Add New Folder to create a new folder for the component. For more information about folders and folder creation, see *Organizing Components*.

5. In the Create New Component section of the form, click Create.

6. In the URI (Uniform Resource Identifier) Component Entry Form section of the form, complete the following fields, and then click OK:

   - Component ID

     The system ID for the component. It is the name that appears when the system provides lists of components for a user to choose from. The maximum field length is 10 characters.

   - Component Name

     The name for the component as it appears in the component title bar in the J.D. Edwards Portal. The maximum field length is 30 characters.

   - Devices

     Indicates in which devices and environments the component can be viewed.

   - Display

     Indicates whether to display the component title bar in the J.D. Edwards Portal. The title bar does not appear when this option is checked. Because the mode icons reside on the component's title bar, hiding the title bar will prevent users from accessing any associated modes or other Portal-enabled component features such as the minimize feature.

   - Inherited Trust

     Indicates whether the component can use the current session user ID and login to log in to the component’s application.

   - Width

     Indicates the width in pixels of the component IFRAME.

   - Height

     Indicates the height in pixels of the component IFRAME.

   - Scrolling

     Choose Yes to always display scrollbars in the IFRAME. Choose No to never display scrollbars. Choose Auto to allow the IFRAME to display scrollbars if they are needed.

   - Component URL

     The URL of the component. If you use <A> tags, the tags should include a TARGET reference to a new window so that the Portal is not replaced by the content to which the user navigates.

     You can use DHTML code, but be sure to name any controls or JavaScript functions with the prefix com_companyname_applicationame to avoid naming conflicts.

   - Personalize

     The URL target to display, Portal Component Servlet to instantiate, or JavaScript to execute when the user clicks the Personalize icon. The maximum field length is 256 characters.

To display the content of a URL target, choose URL, and then enter the URL in the space provided.

To execute JavaScript, choose Javascript, and then enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Help

  The URL target to display, Portal Component Servlet to instantiate, or JavaScript to execute when the user clicks the Help icon. The maximum field length is 256 characters. To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

  To display the content of a URL target, choose URL, and then enter the URL in the space provided.

  To execute JavaScript, choose Javascript, and then enter the complete JavaScript function call (including the parentheses and ending semicolon).

  Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Expand

  The URL target to display, Portal Component Servlet to instantiate, or JavaScript to execute when the user clicks the Expand icon. The maximum field length is 256 characters. To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

  To display the content of a URL target, choose URL, and then enter the URL in the space provided.

  To execute JavaScript, choose Javascript, and then enter the complete JavaScript function call (including the parentheses and ending semicolon).

  Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

7. If the component is part of a set of components for a component group, choose the component group from the drop-down list in the Component Group field.
8. To display the component in additional folders, perform one of the following tasks:
   - To place the component in the Root folder, perform the following steps:
     i. In the Available column, browse and display the contents of the Root folder.
     ii. Ensure none of the folders are highlighted and click the right-pointing arrow.

     To deselect a highlighted folder, hold down the Control button on your keyboard and click the highlighted folder.

   - To place the component in a folder other than the Root folder, perform the following step:
     - Choose a folder in the Available column under Parent Folders and click the right-pointing arrow.

The component appears in all folders listed in the Selected column.

Note
All folders and components appear in the *ALL folder. You cannot select or deselect the *ALL folder.

9. To prevent the component from appearing in a folder, click a folder in the Selected column and click the left-pointing arrow.
10. To see the component as it will appear to an end user, click Preview.

    In Preview Window, click Close Window to return to the component entry form.

11. When you have finished creating the component, click OK.

    The system creates the component.

    Click Cancel to return to the component creation section without saving the component.

Building Portal Component Servlets

Portal Component Servlets are specialized URI components that you must use if you want:

- To use a OneWorld application.
- To reference or update the session object.
- To send and receive data from the database using the ComponentDatabaseInterface or any other Database access through the OneWorld API.

Like URI components, Portal Component Servlets always require interaction between the OneWorld Portal and a web server. These components must be written in Java as Portal Component Servlet servlets.

Java Portal Component Servlets should be packaged as:

```
com.mycompany.oneworld.owportal.components.compname
```

where *mycompany* is the name of your company.

Building Portal Component Servlets is similar to building URI components. Instead of providing a fully-qualified URI, however, you provide a class name that is accessible to the PortalBuilderServlet through the server's classpath. Use the same method to pass parameters to the Portal Component Servlet that you use to pass parameters from a browser. For example, consider the following URL:

```
http://myserver.com/servlets/com.mycompany.oneworld.owportal.
components.componentName.MyServlet?mode=personalize
```

To declare a Portal Component Servlet from with the Portal, you might define this URL in this way:

```
com.mycomponent.oneworld.owportal.components.componentName.MySer
vlet?
mode=personalize
```

See Also

- *Planning Portal Components* for more information about determining what kind of component is appropriate for your needs
- *Setting Component Permissions* for instructions on granting access rights to a component
- *Calling URLs*

- *Wrapping URIs*

► To create a Portal Component Servlet

1. From any OneWorld Portal workspace, click Personalize on the Secondary Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, click the Components hyperlink.
3. In the Create New Component section of the form, choose Portal Component Servlet in the Component Type field.
4. To put the component in a folder other than the Root, in the Edit Existing Components section of the form, click the folder in which you want the new component to appear.

   Click Add New Folder to create a new folder for the component. For more information about folders and folder creation, see *Organizing Components*.

5. In the Create New Component section of the form, click Create.
6. On Portal Component Servlet Entry Form, complete the following fields:

   - Component name

     The system name for the component. It is the name that appears when the system provides lists of components for a user to choose from. The maximum field length is 10 characters.

   - Description

     The name for the component as it appears in the component title bar in the OneWorld Portal. The maximum field length is 30 characters.

   - Devices

     Indicates in which devices and environments the component can be viewed.

   - Display

     Because the mode icons reside on the component's title bar, hiding the title bar prevents users from accessing any associated modes or other Portal-enabled component features such as the minimize function.

   - Class Name

     The class declaration for the component.

   - Personalize

To display a Portal Component Servlet, choose OneWorld, and then enter its class name.

To display the content of a URL target, choose URL, and then enter the URL in the space provided.

To execute JavaScript, choose Javascript, and then enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Help

  To display a Portal Component Servlet, choose OneWorld, and then enter its class name.

  To display the content of a URL target, choose URL, and then enter the URL in the space provided.

  To execute JavaScript, choose Javascript, and then enter the complete JavaScript function call (including the parentheses and ending semicolon).

  Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Exapnd

  To display a Portal Component Servlet, choose OneWorld, and then enter its class name.

  To display the content of a URL target, choose URL, and then enter the URL in the space provided.

  To execute JavaScript, choose Javascript, and then enter the complete JavaScript function call (including the parentheses and ending semicolon).

  Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

7. To display the component in additional folders, perform one of the following tasks:

   - To place the component in the Root folder, perform the following steps:

     i. In the Available column, browse and display the contents of the Root folder.

     ii. Ensure none of the folders are highlighted and click the right-pointing arrow.

     To deselect a highlighted folder, hold down the Control button on your keyboard and click the highlighted folder.

   - To place the component in a folder other than the Root folder, perform the following step:

     - Choose a folder in the Available column under Parent Folders and click the right-pointing arrow.

   The component appears in all folders listed in the Selected column.

Note
All folders and components appear in the *ALL folder. You cannot select or deselect the *ALL folder.

8. To prevent the component from appearing in a folder, click a folder in the Selected column and click the left-pointing arrow.
9. To see the component as it will appear to an end user, click Preview.

   In Preview Window, click Close Window to return to the component entry form.
10. When you have finished creating the component, click OK.

    The system creates the component.

    Click Cancel to return to the component creation section without saving the component.

Building Java Applet Components

When you create an applet component, you enter all of the parameters necessary for the APPLET tag and any PARAM tag information into the Applet Component Entry form. If you want to use the toolbar icons to invoke expand, personalize, or help functionality, you must make these functions part of the applet's public interface. The J.D. Edwards Portal framework provides the JavaScript, which invokes your applet's functions whenever you click one of the component's toolbar icons.

Your applet may change its interface to reflect the new state (help or personalize) but you will probably want to generate a new frame window in response to the expand function invocation. Applets can communicate with the server, from which they were served via HTTP or RMI. Applets can communicate with servlets or other server-side programs so that the applet can display information in the database or other dynamic information.

See Also

- *Setting Component Permissions* for instructions on granting access rights to a component

► To build a Java applet component

1. From any J.D. Edwards Portal workspace, click Personalize on the Secondary Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.
2. On Personalize, click the Components hyperlink.
3. In the Create New Component section of the form, choose Applet Component in the Component Type field.
4. To put the component in a folder other than the Root, in the Edit Existing Components section of the form, click the folder in which you want the new component to appear.

Click Add New Folder to create a new folder for the component. For more information about folders and folder creation, see *Organizing Components*.

5. In the Create New Component section of the form, click Create.
6. Complete the following fields:
   - Component name

     The system name for the component. It is the name that appears when the system provides lists of components for a user to choose from. The maximum field length is 10 characters.

   - Description

     The name for the component as it appears in the component title bar in the J.D. Edwards Portal. The maximum field length is 30 characters.

   - Devices

     Indicates in which devices and environments the component can be viewed.

   - Display

     Because the mode icons reside on the component's title bar, hiding the title bar prevents users from accessing any associated modes or other Portal-enabled component features such as the minimize feature.

   - Scheme Flag

     Indicates whether to pass color parameters to the applet.

   - Codebase

     The codebase argument of the applet. The maximum field length is 256 characters.

   - Code

     The code argument of the applet. The maximum field length is 256 characters.

   - Width

     The width of argument of the applet. The maximum field length is 4 digits.

   - Height

     The height argument of the applet. The maximum field length is 4 digits.

   - Align

     The align argument of the applet. The maximum field length is 30 digits.

   - Vspace

     The vspace argument of the applet. The maximum field length is 4 digits.

   - Hspace

     The hspace argument of the applet. The maximum field length is 4 digits.

   - Archive

The archive argument of the applet. The argument is valid only for HTML version 4.0 and later. The maximum field length is 256 characters.

- Parameters; name=val|name=val|val|name=val,...(No spaces, NO CRs)

  The param value or values that follow the applet. Each value must be expressed in the form of name argument = value argument. To enter more than one param value, separate each value with a pipe (|). Do not use spaces or carriage returns in this field. This field accepts up to 30,000 characters.

- Alt (Alternate HTML if applets are not enabled (255 chars max))

  The alt argument of the applet. The maximum field length is 256 characters. Leave this field blank unless you are using HTML version 4.0 or later.

- Personalize

  To display the content of a URL target, choose URL, and then enter the URL in the space provided.

  To invoke an applet, choose Public Function, and then enter its public function name. The response to the function call depends on the applet and might include opening additional Java frame windows as the user interacts with the applet.

  To execute JavaScript, choose Javascript, and then enter the complete JavaScript function call (including the parentheses and ending semicolon).

  Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Help

  To display the content of a URL target, choose URL, and then enter the URL in the space provided.

  To invoke an applet, choose Public Function, and then enter its public function name. The response to the function call depends on the applet and might include opening additional Java frame windows as the user interacts with the applet.

  To execute JavaScript, choose Javascript, and then enter the complete JavaScript function call (including the parentheses and ending semicolon).

  Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Maximize

  To display the content of a URL target, choose URL, and then enter the URL in the space provided.

  To invoke an applet, choose Public Function, and then enter its public function name. The response to the function call depends on the applet and might include opening additional Java frame windows as the user interacts with the applet.

  To execute JavaScript, choose Javascript, and then enter the complete JavaScript function call (including the parentheses and ending semicolon).

  Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

7. If the component is part of a set of components for a component group, choose the component group from the drop-down list in the Component Group field.
8. To display the component in additional folders, perform one of the following tasks:
   - To place the component in the Root folder, perform the following steps:
     i. In the Available column, browse and display the contents of the Root folder.
     ii. Ensure none of the folders are highlighted and click the right-pointing arrow.

     To deselect a highlighted folder, hold down the Control button on your keyboard and click the highlighted folder.
   - To place the component in a folder other than the Root folder, perform the following step:
     - Choose a folder in the Available column under Parent Folders and click the right-pointing arrow.

   The component appears in all folders listed in the Selected column.

   Note
   All folders and components appear in the *ALL folder. You cannot select or deselect the *ALL folder.

9. To prevent the component from appearing in a folder, click a folder in the Selected column and click the left-pointing arrow.
10. To see the component as it will appear to an end user, click Preview.

    In Preview Window, click Close Window to return to the component entry form.
11. When you have finished creating the component, click OK.

    The system creates the component.

    Click Cancel to return to the component creation section without saving the component.

Scheme Flag Parameters

When you create a Java applet component, you can choose to have the component return color scheme data for the elements in the current workspace. The following table lists the parameters that are returned:

Flag	Parameter Name	Typical Value
1	jdeowpBodyFont	Airal,Helvetica,Verdana,Sans-serif
2	jdeowBodyTextColor	#000000
3	jdeowpToolbarBackground	#585969
4	jdeowpToolbarTextColor	#000000
5	jdeowpToolbarNavBackground	#ABABB7
6	jdeowpMainNavBackground	#C5C5CD

Flag	Parameter Name	Typical Value
7	jdeowpMainNavImage	/jde/owportal/images/enbdefault.jpg
8	jdeowpMainNavTextColor	#FFFFFF
9	jdeowpSecNavBackground	#C5C5CD
10	jdeowpSecNavTextColor	#000000
11	jdeowpBorderColor	#ABABB7
12	jdeowpTableBackground	#FFFFFF
13	jdeowpTableTextColor	#000000
14	jdeowpAltTableBackground	#EDEDEF
15	jdeowpAltTableTextColor	#000000

Building Java Servlet Components

In the J.D. Edwards Portal, JavaScript code can use the open function to create new browser windows, invoke the functionality of applets or objects contained in the user entered HTML, and so forth. Script-generated frame windows can contain dynamically generated HTML for the user to interact with. Alternately, the SRC parameter in the open call can reference a URL whose output makes up the content of the new window. Whichever path you choose, make sure that your functionality does not impose itself on the Portal presentation. If your client-side code must interact with server side programs, it should open a new browser frame from which it interacts with the server without replacing the Portal's presentation.

The J.D. Edwards Portal provides the Portal Component Servlet API. To gain access to the API, you must have the jar file for the J.D. Edwards Portal in your class path, and then you must import `com.jdedwards.portal.components.*` into your class. Currently, this package includes the following classes:

ComponentDatabaseInterface Provides a method for storing data to and retrieving data from a JAS database. The package also provides a convenient way to import and export data to and from Sun's XML parser so that you can easily use data with the DOM.

ComponentDatabaseException An exception model for database errors generated by the ComponentDatabaseInterface.

All JavaScripts within your component should be named:

`com_`*yourcompany*`_`*yourcomponent*`_`*scriptname*

where *yourcompany* is the name of your company and *yourcomponent* is the name of your component.

All other files (non-JavaScript files) should reside in directories named:

yourcompany`/owportal/components/`*yourcomponent*

where *yourcompany* is the name of your company and *yourcomponent* is the name of your component.

Note

For custom servlet development, you should place your java objects (.class files) in their own directory or jar file. You should not add them to the jdedwards jas.jar file. When you create it, you must add your custom directory or jar file location to the Websphere class path.

Critical State Notification Scripts

Use these scripts to instruct the J.D. Edwards Portal and the user when the Portal is about to execute a function that will jeopardize the persistence of a component's data. A component in this condition is said to be in a critical state. For example, if a user was entering new employees into J.D. Edwards software via a Portal component and attempted to switch to a different workspace before completing a transaction, the Portal could send the component instructions to terminate the user's J.D. Edwards software session.

The Portal notifies each component in a critical state on a workspace that includes either of the Portal-to-component notification scripts—one for URLs and one for Portal Component Servlets. However, if a workspace contains multiple critical state components that contain the user notification script, the user receives only one message.

These scripts respond only to changes in the Portal's condition brought on by one of the Portal's command functions, not to changes in the Portal's condition brought on by a component's function. For example, notification occurs when a user tries to execute a function on either of the Portal's navigation bars such as accessing a different URL in the current session or logging out. Notification does not occur if the user accesses a different URL in the current session though a link on one of the components in the workspace.

You can include these scripts in HTML, URI, and Portal Component Servlets. The functions provided by these scripts are available only on JavaScript-enabled platforms.

jdeowpAddCSNotificationURL Script

Use this script to notify a URL when a component is in a critical state.

This script has the following format:

```
jdeowpAddCSNotificationURL(compname, value)
```

where

compname is the component name

value is the URL that you want to access

Follow value with a question mark and a command, if necessary.

jdeowpAddCSNotificationClass Script

Use this script to notify a Portal Component Servlet when a component is in a critical state.

This script has the following format:

```
jdeowpAddCSNotificationClass(compname, value)
```

where

compname is the component name

value is the Portal Component Servlet that you want to access

Follow value with a question mark and a command, if necessary.

jdeowpCSWarnUser() Script

Use this script to notify a user when a component is in a critical state. You cannot define jdeowpCSWarnUser for a component unless you have also defined either jdeowpAddCSNotificationURL or jdeowpAddCSNotificationClass for the component first.

Call this script to enable it for a component.

Component Look and Feel

You can set different elements of a component such as it borders, body, or toolbar to match the scheme of whichever workspace it resides. You can also override or specify component toolbar elements such as the component description and the button icons. You can also specify buttons beyond the basic set of modes (minimize, maximize, personalize, and help) you set up when you created the component.

Portal Design Standards

Components in normal mode should adapt to changes in width to the greatest extent possible. A reasonable size for your component is approximately 250-300 pixels wide by 250-300 pixels high.

All components should be able to function independently of the J.D. Edwards Portal. Your component is responsible for storing any information that it needs for personalization or to support its functionality.

Always fully qualify all graphics.

When designing HTML components, avoid tags such as META, HTML, and BODY. Do not use the FRAMESET tag.

Java URI and Portal Component Servlets should be packaged as:

```
com.yourcompany.portal.components:compname
```

where *yourcompany* is the name of your company and *compname* is the name of the component.

All JavaScripts within your component should be named:

```
com_yourcompany_yourcomponent_scriptname
```

where *yourcompany* is the name of your company and *yourcomponent* is the name of your component.

All other files (non-JavaScript files) should reside in directories named:

yourcompany/owportal/components/*yourcomponent*

where *yourcompany* is the name of your company and *yourcomponent* is the name of your component.

Note

For custom servlet development, you should place your java objects (.class files) in their own directory or jar file. You should not add them to the jdedwards jas.jar file. When you create it, you must add your custom directory or jar file location to the Websphere class path.

Scheme Information

Each workspace has its own style sheet which the Portal loads automatically when the user views the workspace. Your component can acquire scheme information from this style sheet and cause the component to conform to the scheme, if desired. Depending on the type of component you are creating, you can get and set scheme specifications in one of three ways.

Use the Stylesheet servlet to acquire and set scheme information for Link, HTML, URI, Independent URI, and Portal Component Servlets for display on PCs or on pervasive devices based on CE for the HPC2000.

Use the Worksapce Java Class to acquire and set scheme information for Portal Component Servlets for display in any form factor.

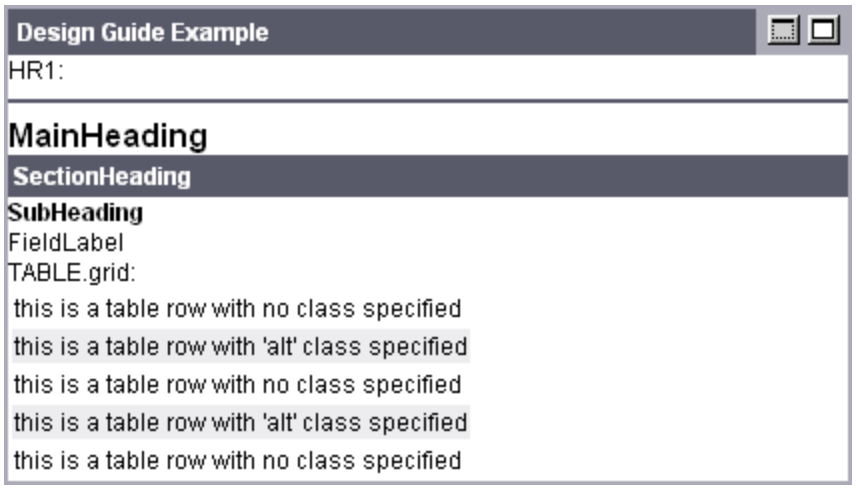
You can also set Java Applet components to acquire scheme information as a set of Param tags.

Stylesheet Servlet

The Stylesheet Servlet allows you to set the following classes based on the current workspace style sheet:

HR1	Supplies a visual separator.
MainHeading	Supplies the font, color, and size of titles and first level headers.
SectionHeading	Supplies the font, color, and size of second level headers.
SubHeading	Supplies the font, color, and size of third level headers.
FieldLabel	Supplies the font, color, and size of field labels in your application.
TABLE.grid	Supplies the parameters for a table. Use the alt class on rows that you want to be colored alternatively from the default grid colors.

Consider the following illustration:



The following code fragment produced the component pictured above:

```
HR1:<HR CLASS="HR1">
<SPAN CLASS="MainHeading">MainHeading</SPAN><BR>
<SPAN CLASS="SectionHeading">SectionHeading</SPAN><BR>
<SPAN CLASS="SubHeading">SubHeading</SPAN><BR>
<SPAN CLASS="FieldLabel">FieldLabel</SPAN><BR>
TABLE.grid:
<TABLE CLASS="grid">
<TR><TD>this is a table row with no class specified</TD></TR>
<TR CLASS="alt"><td>this is a table row with 'alt' class
specified</TD></TR>
<TR><TD>this is a table row with no class specified</TD></TR>
<TR CLASS="alt"><td>this is a table row with 'alt' class
specified</TD></TR>
<TR><TD>this is a table row with no class specified</TD></TR>
</TABLE>
```

You can also use the Servlet to get and set specific classes in the Portal's underlying spreadsheet.

StyleInfo

In addition to using the Web GUI package to generate various HTML controls in harmony with the look and feel of the Portal, you can retrieve individual colors that match those used by the current workspace as well. Use the StyleInfo class inside any PortalComponentServlet to retrieve color and image information.

The following code shows an example of how to retrieve the text color of the Portal workspace:

```
//env is the Login Environment

StyleInfo info = new StyleInfo(env);

String textColor = getColor(StyleInfo.TEXT_COLOR);
```

The following table shows the list of valid constants for color.

StyleInfo.ALT_TABLE_COLOR	Returns the alternate table background color. The alternate table background color is the background color of an alternate table cell.
StyleInfo.ALT_TABLE_TEXT_COLOR	Returns the alternate table text color. The alternate table text color is the text color of an alternate table cell.
StyleInfo.BACKGROUND_COLOR	Returns the background color. The background color is the color of the background of the workspace.
StyleInfo.BORDER_COLOR	Returns the border color. The border color is the color of the border around components.
StyleInfo.ENTERPRISE_BAR_COLOR	Returns the fixed background color. The fixed background color is the background color of the Enterprise Navigation bar.
StyleInfo.NAVIGATION_BAR_COLOR	Returns the menu color. The menu color is the color of the Workspace Navigation bar.
StyleInfo.NAVIGATION_BAR_TEXT_COLOR	Returns the greeting color. The greeting color is the color of the text in the Workspace Navigation bar.
StyleInfo.TABLE_COLOR	Returns the table background color. The table background color is the background color of a standard table cell.
StyleInfo.TABLE_TEXT_COLOR	Returns the table text color. The table text color is the text color of a standard table cell.
StyleInfo.TEXT_COLOR	Returns the text color. The text color is the color of the text of the workspace.
StyleInfo.COMPONENT_TOOLBAR_COLOR	Returns the toolbar color. The toolbar color is the color of the component's toolbar.
StyleInfo.COMPONENT_TOOLBAR_TOOLS_COLOR	Returns the toolbar tools color. The toolbar tools color is the color behind the component's buttons (including the maximize and minimize buttons).
StyleInfo.ENTERPRISE_BAR_TEXT_COLOR	Returns the top text color. The top text color is the text color of the Enterprise Navigation bar.

The following code sample shows how to retrieve an image URL used on the workspace for the corporate logo:

```
//env is the Login Environment

StyleInfo info = new StyleInfo(env);

String url = getUrl(StyleInfo.CORPORATE_LOGO_URL);
```

The following table shows the list of valid constants for image URLs:

StyleInfo.ALT_TABLE_IMAGE_URL	Returns the alternate table image URL. The alternate table image is the background image of an alternate table cell.
StyleInfo.BACKGROUND_URL	Returns the background URL. The background color is the color of the background of the workspace.
StyleInfo.CORPORATE_LOGO_URL	Returns the corporate logo URL. The corporate logo URL is the URL of the logo in the upper left corner of the Enterprise Navigation bar.
StyleInfo.NAVIGATION_BAR_IMAGE_URL	Returns the menu image URL. The menu image is the background image of the Workspace Navigation bar.
StyleInfo.NAVIGATION_BAR_SEPARATOR_IMAGE_URL	Returns the menu separator image URL. The menu separator image is the image used to separate menu items on the Workspace Navigation bar.
StyleInfo.TABLE_IMAGE_URL	Returns the table image URL. The table image is the background image of a standard table cell.
StyleInfo.COMPONENT_TOOLBAR_IMAGE_URL	Returns the toolbar image URL as a string. The toolbar image is the background image used for the toolbar.
StyleInfo.COMPONENT_TOOLBAR_TOOLS_IMAGE_URL	Returns the toolbar tools image URL. The toolbar tools image is the image used as aye background for the navigation section of the component toolbar.
StyleInfo.ENTERPRISE_BAR_IMAGE_URL	Returns the top image URL. The top image URL is the image in the background of the Enterprise Navigation bar.

The following code shows how to retrieve a font used on the workspace:

```
//env is the Login Environment

StyleInfo info = new StyleInfo(env);

String font = getFont(StyleInfo.BODY_FONT);
```

The following table shows the list of valid constants for fonts:

StyleInfo.BODY_FONT	Returns the font used by the workspace.

Implementing the Web GUI Package

The J.D. Edwards Portal was designed to bring disparate content together on one workspace. Each workspace can have its own color scheme which can be used to tie the content together. If your component does not integrate well into whatever workspace it is placed, the component might confuse your users and set an expectation for inconsistent formatting.

> "Consistency gives polish to a site and encourages visitors to stay by creating an expectation about the structure."
>
> – Yale Web Style Guide

J.D. Edwards provides a set of Java APIs that help ensure consistency, giving your content a professional appearance. This API set is designed to work with Portal Component Servlets. It has been designed to abstract from you, the developer, the details of making certain controls look the way they do. You do not need to set colors or even specifying proper StyleSheet classes. Furthermore, this abstraction allows automatic translation to more primitive devices that do not support the latest in HTML standards.

The Web GUI APIs are located in the com.jdedwards.base.webgui Java package. Please refer to the JavaDocs for code examples and detailed explanations.

The Web GUI package consists of the following classes:

- WebTable
- WebTabBody
- WebTab
- WebMenu
- WebMenuBar
- WebMenuItem
- WebLabel
- WebGroupBox
- WebGraphicButton
- WebCell

WebTable and WebCell

The following sample code displays a typical grid where the top row of controls is a Query by Example (QBE) line, and the regular gird cells are framed by header cells:

```
Vector row1 = new Vector();
for (int i=0; i<5; i++)
    row1.addElement(new WebCell(WebCell.CONTROL_CELL, "control"));

Vector row2 = new Vector();
for (int i=0; i<5; i++)
    row2.addElement(new WebCell(WebCell.HEAD_CELL, "header"));

Vector row3 = new Vector();
row3.addElement(new WebCell(WebCell.HEAD_CELL, "header"));
for (int i=0; i<4; i++)
    row3.addElement(new WebCell(WebCell.GRID_CELL, "regular grid cell"));

Vector row4 = new Vector();
row4.addElement(new WebCell(WebCell.HEAD_CELL, "header"));
for (int i=0; i<4; i++)
    row4.addElement(new WebCell(WebCell.GRID_CELL, "regular grid cell"));

WebTable theTable = new WebTable();
theTable.addRow(row1);
theTable.addRow(row2);
theTable.addRow(row3);
theTable.addRow(row4);

out.println(theTable.toStringBuffer(env));
```

The output for this code appears as follows:

control	control	control	control	control
header	header	header	header	header
header	regular grid cell	regular grid cell	regular grid cell	regular grid cell
header	regular grid cell	regular grid cell	regular grid cell	regular grid cell

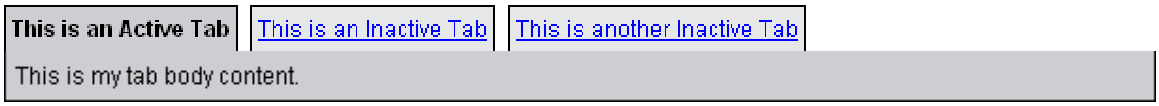
WebTabBody and WebTab

The following sample code produces a table with one header row and two rows of regular cells:

```
WebTabBody webtabbody = new WebTabBody("This is my tab body content.");
 webtabbody.setWidth(600);
 webtabbody.addTab(new WebTab("This is an Active Tab", WebTab.TYPE_ACTIVE));
```

```
 webtabbody.addTab(new WebTab("<a href=\"\">This is an Inactive Tab</a>",
WebTab.TYPE_INACTIVE));
 webtabbody.addTab(new WebTab("<a href=\"\">This is another Inactive
Tab</a>", WebTab.TYPE_INACTIVE));

  out.println(webtabbody.toStringBuffer(env));
```

The output for this code one looks like this:



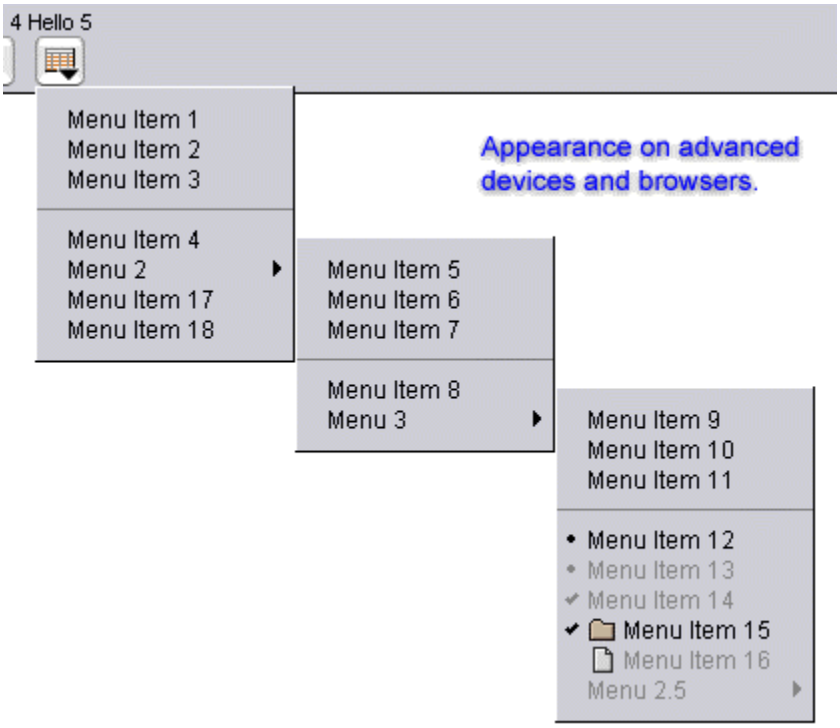
WebMenu, WebMenuItem and WebMenuBar

The following sample code creates a menu with two submenus. This example illustrates many of the capabilities that menu items can have:

```
WebMenu webmenu1 = new WebMenu("Menu 1");

 webmenu1.addMenuChild(new WebMenuItem("Menu Item 1",
"javascript:alert('Hello World');"));

 webmenu1.addMenuChild(new WebMenuItem("Menu Item 2",
"javascript:alert('Hello World');"));

 webmenu1.addMenuChild(new WebMenuItem("Menu Item 3",
"javascript:alert('Hello World');"));

 webmenu1.addMenuChild(new WebMenuItem("-"));

 webmenu1.addMenuChild(new WebMenuItem("Menu Item 4",
"javascript:alert('Hello World');"));


 WebMenu webmenu2 = new WebMenu("Menu 2");

 webmenu2.addMenuChild(new WebMenuItem("Menu Item 5",
"javascript:alert('Hello World');"));

 webmenu2.addMenuChild(new WebMenuItem("Menu Item 6",
"javascript:alert('Hello World');"));

 webmenu2.addMenuChild(new WebMenuItem("Menu Item 7",
"javascript:alert('Hello World');"));

 webmenu2.addMenuChild(new WebMenuItem("-"));

 webmenu2.addMenuChild(new WebMenuItem("Menu Item 8",
"javascript:alert('Hello World');"));


 webmenu1.addMenuChild(webmenu2);

 webmenu1.addMenuChild(new WebMenuItem("Menu Item 17",
"javascript:alert('Hello World');"));

 webmenu1.addMenuChild(new WebMenuItem("Menu Item 18",
"javascript:alert('Hello World');"));


 WebMenu webmenu2Point5 = new WebMenu("Menu 2.5");
```

```
 webmenu2Point5.setEnabled(false);


 WebMenu webmenu3 = new WebMenu("Menu 3");

 webmenu3.addMenuChild(new WebMenuItem("Menu Item 9",
"javascript:alert('Hello World');"));

 webmenu3.addMenuChild(new WebMenuItem("Menu Item 10",
"javascript:alert('Hello World');"));

 webmenu3.addMenuChild(new WebMenuItem("Menu Item 11",
"javascript:alert('Hello World');"));

 webmenu3.addMenuChild(new WebMenuItem("-"));

 webmenu3.addMenuChild(new WebMenuItem("Menu Item 12",
"javascript:alert('Hello World');", null, WebMenuItem.SELECTION_TYPE_RADIO,
true, true));

 webmenu3.addMenuChild(new WebMenuItem("Menu Item 13",
"javascript:alert('Hello World');", null, WebMenuItem.SELECTION_TYPE_RADIO,
false, true));

 webmenu3.addMenuChild(new WebMenuItem("Menu Item 14",
"javascript:alert('Hello World');", null, WebMenuItem.SELECTION_TYPE_CHECK,
false, true));

 webmenu3.addMenuChild(new WebMenuItem("Menu Item 15",
"javascript:alert('Hello World');",
"../owportal/images/jdefolderclosed.gif", WebMenuItem.SELECTION_TYPE_CHECK,
true, true));

 webmenu3.addMenuChild(new WebMenuItem("Menu Item 16",
"javascript:alert('Hello World');", "../owportal/images/jdefileicon.gif",
WebMenuItem.SELECTION_TYPE_NONE, false, false));

 webmenu3.addMenuChild(webmenu2Point5);

  webmenu2.addMenuChild(webmenu3);


 WebMenuBar menubar = new WebMenuBar();


 WebGraphicButton webMenuButton = new
WebGraphicButton("../img/row_exit.gif","../img/row_exitmo.gif",null,null,"th
is is my hover text","MenuBarTest5", "Hello 5",
WebGraphicButton.TYPE_STANDARD_LARGE_ICON);

 webmenu1.setWebGraphicButton(webMenuButton);

  menubar.addItem(webmenu1.toStringBuffer(env));

  out.println(menubar.toStringBuffer(env));
```

The output for the web menu looks like this:



The output for the web menu bar looks like this:

WebLabel

The following sample code shows three types of a field style label:

```
WebLabel label = new WebLabel("This is my example label.",
WebLabel.STYLE_FIELD_LABEL, WebLabel.TYPE_NOWRAP);

 out.println(label.toStringBuffer(env));

 label = new WebLabel("This is my example label.",
WebLabel.STYLE_FIELD_LABEL, WebLabel.TYPE_WRAP, "100");

 out.println(label.toStringBuffer(env));

 label = new WebLabel("This is my example label.",
WebLabel.STYLE_FIELD_LABEL, WebLabel.TYPE_NOWRAP_CROPPED, "100");

 out.println(label.toStringBuffer(env));
```

The output for this code looks like this:

This is my example label.

This is my
example label.

This is my exampl

This is my example description label.

This is my
example
description label.

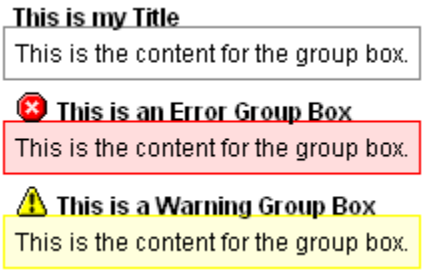
This is my exampl

WebGroupBox

The following sample code creates three types of group boxes:

```
WebGroupBox gb = new WebGroupBox(WebGroupBox.TYPE_DEFAULT, "This is my
Title", "This is the content for the group box.");

 out.println(gb.toStringBuffer(env));

 gb = new WebGroupBox(WebGroupBox.TYPE_ERROR, "This is an Error Group Box",
"This is the content for the group box.");

 out.println(gb.toStringBuffer(env));

 gb = new WebGroupBox(WebGroupBox.TYPE_WARNING, "This is a Warning Group
Box", "This is the content for the group box.");

 out.println(gb.toStringBuffer(env));
```

The output for this code looks like this:



WebGraphicButton

The following sample code creates five types of web graphic buttons:

```
out.println("<table cellspacing=0 cellspadding=10><tr><td>");

 out.println( (new
WebGraphicButton("../img/hc1.gif","../img/hc1mo.gif","javascript:alert('hell
o world');",null,"this is my hover
text","WebGraphicButtonTest1")).toStringBuffer(env) );

 out.println("</td><td>");

 out.println( (new
WebGraphicButton("../img/hc2.gif","../img/hc2mo.gif","javascript:alert('hell
o world');",null,"this is my hover text","WebGraphicButtonTest2", "A
WebGraphicButton", WebGraphicButton.TYPE_TEXT_ON_TOP)).toStringBuffer(env)
);

 out.println("</td><td>");

 out.println( (new
WebGraphicButton("../img/hc950.gif","../img/hc950mo.gif","javascript:alert('
hello world');",null,"this is my hover text","WebGraphicButtonTest3", "A
WebGraphicButton", WebGraphicButton.TYPE_TEXT_ON_RIGHT)).toStringBuffer(env)
);

 out.println("</td><td>");

 out.println( (new
WebGraphicButton("../img/hc951.gif","../img/hc951mo.gif","javascript:alert('
hello world');",null,"this is my hover text","WebGraphicButtonTest4", "A
WebGraphicButton",
WebGraphicButton.TYPE_TEXT_ON_BOTTOM)).toStringBuffer(env) );

 out.println("</td><td>");

 out.println( (new
WebGraphicButton("../img/gridtab.gif","../img/gridtabmo.gif","javascript:ale
rt('hello world');",null,"this is my hover text","WebGraphicButtonTest5", "A
WebGraphicButton", WebGraphicButton.TYPE_TEXT_ON_LEFT)).toStringBuffer(env)
);

 out.println("</td></tr></table>");
```

The output for this code looks like this:

ComponentDatabaseInterface Class

The ComponentDatabaseInterface class provides access to the set of database tables where your components store data. The class allows this data to be read from and written to an XML Parser (using streams). It also allows strings to be imported and exported. You can convert such strings using the loadTemplate method of the ComponentServlet class. See *Understanding the loadTemplate() Method* for more information.

This class requires a LoginEnvironment; therefore, you must derive your class from the PortalComponentServlet class.

PortalComponentServlet Class

The PortalComponentServlet allows your component to run on the JAS server as well as to function with the J.D. Edwards Portal. If you extend this class when you make your own components, you can take advantage of J.D. Edwards features that help you integrate into other components or allow your component to function in a stand-alone environment. These features include parameter passthrough, style information, Web GUI components, user information, and access to other parameters associated with running inside of the Portal.

The PortalComponentServlet class is an abstract class that allows you to integrate your own Java code with the Portal. The class also allows a component to acquire J.D. Edwards environment and session information from either the Portal or direct execution. By deriving itself from the J.D. Edwards component class, your component loads into the Portal faster then standard servlet components and uses a minimum of bandwidth. Although the PortalComponentServlet class implements the standard service method, you must provide some additional information not normally required by servlets.

Note

The Portal Builder expects that all servlet components will be derived from the PortalComponentServlet class.

Extending the PortalComponentServlet Class

The following servlet is the minimum servlet required to extend the PortalComponentServlet class correctly:

```
package com.jdedwards.portal.components.helloworld;

import com.jdedwards.portal.components.*;

import com.jdedwards.services.login.*;

class HelloWorldServlet extends PortalComponentServlet

{

protected void service(LoginEnvironment env, PrintWriter out,
HttpServletRequest req,

HttpServletResponse res)throws ServletException, IOException

{

out. println(“Hello J.D. Edwards”);

}

public Boolean requiresJ.D.EdwardsPortal()

{

return false;

}

}
```

The service method must be implemented by every class extending PortalComponentServlet. Because the standard service method is implemented in the parent class, this service method does not follow the Java servlet specification. For servlets that require a Login environment, the PortalBuilderServlet must bypass the web server and instantiate the class directly. It provides a PrintWriter and a LoginEnvironment to your class. You must not close the PrintWriter or instantiate a new one in your derived class; instead, use the one that is provided.

The servlet receives the ability to write data to the output stream so that it will appear in the user's browser from the HTTPServletResponse.getWriter. The servlet also receives the Login environment via LoginEnvironment. If you need to obtain information about the user and his or her login, you must use the Login environment. When executed, the servlet checks to see if it received information from the login. If not, then the user is not logged in. If the user is not logged in, the servlet closes the output stream and terminates.

The requiresJ.D.EdwardsPortal method tells the PortalComponentServlet class or the PortalBuilderServlet whether a login and a Login environment is needed. If this method returns false, you can expect the Login Environment to be null. Use this method when you want to write a Portal Component Servlet that does not use the database or session information but still makes use of the more robust interface and quicker instantiation of the Portal Component Servlet mechanism.

getPbsUrl() Method and Parameter Passthrough

You can pass parameter values to your PortalComponentServlet from a web page you generate by calling the Portal with all the parameters you have created. The Portal passes these values to your component.

First, you must create a link to the Portal. Use the following code to pass a parameter to the Portal.

```
 String value = request.getParameter("COM_MYCOMPANYNAME_PARAM");

 StringBuffer sb = new StringBuffer();

 sb.append("<SCRIPT>\n");

 sb.append(" function myTestComponentFunction(){\n");

 sb.append("  alert('Test Component is up and running.');\n");

 if (value != null){

   sb.append("  alert('The value passed through the Portal was
"+value+".');\n");

 }

 sb.append(" }\n");

 sb.append("</SCRIPT>\n");

  topbar.setOnLoadScript("myTestComponentFunction();");

 String link = PortalJSPLib.printFullURL( request, getPbsUrl(req)+ "?"+
PortalConstants.PROP_COMPNAME+"="+getComponentName(req)+ "&"+
PortalConstants.PROP_COMPSTATE+"="+ComponentManager.NOR
+"&COM_MYCOMPANYNAME_PARAM=25");

  out.println("<a href=\""+link+"\">Link back to portal</a>");

  out.println(sb);
```

Use the method, PortalJSPLib.printFullURL, to create a fully-qualified link. Use PortalJSPLib.printFullURL instead of a relative URL because the method encapsulates load balancing.

The method, getPbsUrl, returns a link to the Portal and includes the webapp setup.

The parameter, PortalConstants.PROP_COMPNAME, ensures that you link to the correct component. You must set the value of this parameter to the value returned by the getComponentName method. The component name is the Component ID that the creator of the component established for the component in the Component manager interface in the Portal.

The parameter, PortalConstants.PROP_COMPSTATE, is an optional parameter used to control the state in which you want the component to appear. If the component is in restore (multi component) mode and it does not return this parameter, the component automatically switches to Maximize mode (single component view) if you link to the Portal. However, in any other mode, if the parameter is not returned, the component remains in its current mode.

The table below shows the valid values for PortalConstants.PROP_COMPSTATE:

Mode Constant	Mode	AddExtra text value
ComponentManager.NOR	Restore mode	norm
ComponentManager.MAX	Maximize mode	max
ComponentManager.MIN	Minimize mode	min
ComponentManager.HLP	Help mode	help
ComponentManager.PER	Personalize Mode	per
ComponentManager.LNK	Link Mode	link
ComponentManager.FRMLNK	Frame Link mode	framelink

Any application-specific parameters that you pass must use the convention:

COM_*MYCOMPANYNAME_PARAMETERNAME*

To retrieve a parameter from the request object returned by the browser, use the standard servlet call, request.getParameter("COM_*MYCOMPANYNAME_PARAMETERNAME*").

The following table lists the Portal-specific parameters that can be sent on a passthrough request:

Parameter	Description	Required?
PortalConstants.PROP_COMPNAME	The current component id.	Yes
PortalConstants.PROP_COMPSTATE	The mode in which you want the component to appear.	No
PortalConstants.PROP_ADDEXTRA	A request for additional parameters. If this parameter is added and set to true, extra parameters are returned. These parameters are listed in the	No

Parameter	Description	Required?
	next table.	
PortalConstants.PROP_COMPCLASS	A different PortalComponentServlet to which you want to pass parameters. The value should be the full package name of the class you want to call.	No
PortalConstants.PROP_COMPURL	The URL to which you want to pass parameters.	No

The following table lists the parameters returned when ADDEXTRA is set to true:

Parameter	Description
ComponentManager.PBSURL	The PbsUrl, equivalent to getPbsUrl().
ComponentManager.COMPNAME	The current component id, equivalent to getComponentName().
ComponentManager.SESSIONID	The current user session id.
ComponentManager.LANG	The ISO language code. Equivalent to LoginEnvironment.getLanguageCode().
ComponentManager.INTPBSURL	The internal PbsUrl (PortalBuilderServlet URL). Use this URL to connect directly to the PortalBuilderServlet instead of through an HTML page. The value for the host is set in the localhost setting of the JAS.INI file in the PORTALCONFIGURATION section.
ComponentManager.STATE	An integer representing the current mode that can be tested against the constants listed above in the ComponentManager table.
ComponentManager.STATENUM	A text representation of the current mode. The values are listed above in the ComponentManager table.
ComponentManager.FORMFACT	A text representation of the current form factor. Equivalent to LoginEnvironment.getFormFactor.toString()
PortalConstants.PROP_ANON	An flag indicating whether the current user is an anonymous user. PortalConstants.PROP_ANON_TRUE is returned as the value if the current user is an anonymous user. Equivalent to LoginEnvironment.isAnonymousUser().

To access data such as an image on the host, use the following code to build the link:

```
String imgUrl = PortalJSPLib.printFullURL( req,
com.jdedwards.util.WebApp.WEB_APP_PATH +
"/owportal/mycomponent/images/maximize2.gif") ;
```

Use the com.jdedwards.util.WebApp.WEB_APP_PATH constant rather than hard-coding a value such as /jde, because the value might change depending on the installation. Fully qualify the path with the PortalJSPLib.printFullURL static method call.

getTopBar() and setTopBar() methods

You have the ability to determine the exact contents of a component's toolbar. The toolbar consists of a title (the component description), a title icon, a set of buttons, and the onLoad event Javascript. To access the toolbar, use the getTopBar method. After acquiring the toolbar, you can access each element. You may also create an entirely new toolbar by using the following code:

```
ComponentTopBar topBar = new ComponentTopBar();

SetTopBar(topBar);
```

Component Toolbar Buttons

The default buttons that appear in the toolbar depend on the component's mode. You can alter, remove, or add new buttons to the toolbar in any component mode. Following is a description of the default buttons associated with each mode:

- Restore mode: This mode displays the Minimize and Maximize buttons. The Help and Personalize buttons appear also if the component was configured with those modes when it was created. If the user has editable rights to the component, the Edit button appears as well.
- Maximize and Personalize modes: This mode displays the Restore button. The Help and Personalize buttons appear as appropriate.
- Minimize mode: This mode displays the Restore and Maximize buttons.

Note

If you remove the Restore button, you might want to provide the user an alternative link to return to the Restore mode. For example, might include a Close button on your component that redirects to the Portal with the Restore mode parameter set.

The following example shows how the icons may be manipulated.

```
ComponentTopBar topbar = getTopBar();


//Change the URL for the personalize button

Vector buttons = topbar.getButtons();

for(int i =0; i < topbar.getNumberOfButtons(); i++){

 PortalButton pb = (PortalButton)buttons.get(i);

 if(pb instanceof PersonalizePortalButton){

  pb.setUrl("http://www.jdedwards.com");
```

```
   }
  }


  //Create a new button
    PortalButton pb = new PortalButton();
    pb.setId("BUT1");
    pb.setText("Mail");
    pb.setUrl("http://www.jdedwards.com");
    pb.setTarget("new_window");
       pb.setIcon( PortalJSPLib.printFullURL( req,
com.jdedwards.util.WebApp.WEB_APP_PATH + "/owportal/images/maximize2.gif")
);
    topbar.addButton(pb);


out.println(“Hello J.D. Edwards”);
```

The PortalButton might be an instance of one of the following button classes:

- PersonalizePortalButton
- NormalPortalButton (The Restore button)
- MaximizePortalButton (The Maximize button)
- MinimizePortalButton (The Minimize button)
- HelpPortalButton

All other buttons are PortalButtons.

Component Title and Title Icon

The title defaults to the value entered in the Component Name field when the component was created. You can remove or change this value by using the following code:

```
ComponentTopBar topbar = getTopBar();
 topbar.setTitle("My new title");
```

By default, the title does not have an associated icon. If you specify an icon, it appears to the left of the title. To add a title icon, you must specify a URL for the icon. For example:

```
ComponentTopBar topbar = getTopBar();
 topbar.setTitleIconUrl( PortalJSPLib.printFullURL(request,
WebApp.WEB_APP_PATH + "/owportal/images/maximize2.gif") );
```

Onload Scripts

If the HTML code generated by the component requires the browser's onLoad event, you must provide the component toolbar with the javascript method name that you want to be called. For example:

```
 StringBuffer sb = new StringBuffer();
```

```
sb.append("<SCRIPT>\n");
sb.append(" function myTestComponentFunction(){\n");
sb.append("  alert('Test Component is up and running.');\n");
sb.append(" }\n");
sb.append("</SCRIPT>\n");


ComponentTopBar topbar = getTopBar();
topbar.setOnLoadScript("myTestComponentFunction();");
out.println(sb);
```

PrintWriter

To write content to the browser, use any of the standard PrintWriter methods. The fastest way to generate output is to create a StringBuffer and then append the output data to the StringBuffer. After generating the output, use the following code to send it to the browser:

```
out.println( myStringBuffer );
```

Do not close the PrintWriter because it will be used by other components.

When you use PrintWriter, the content type has already been set on the request, so do not set it in the code.

The HTML you generate should avoid the use of the HTML, HEAD, and BODY tags since you are already within a body tag.

LoginEnvironment

The LoginEnvironment encapsulates the data pertinent to the current session and has the following access methods:

TagMathNumeric getAddressNumber()	Returns the current user's address number.
Object getAttribute(String key)	Retrieves an object stored on the session with the given key.
Hashtable getAttributes()	Returns the complete cache, Hashtable.
ResourceBundle getBundle()	Returns a resource bundle for the user's language.
String getEnvironment()	Returns the string representing the current logged in environment.
FormFactor getFormFactor()	Returns the FormFactor bean containing information about the user's current device and browser. See the description of the FormFactor bean below.
String getLanguagePreference()	Returns the user’s JDEdwards language code.
String getLanguageCode()	Returns the user’s ISO language code.
NetManager getNetManager()	Returns the JAS NetManager object.
String getPassword()	Returns the string representation of the current login password.
RoleEntry getRole()	Returns the RoleEntry object of the user's current role.
Vector getRoles()	Returns a vector of all the current user's roles.
String getUserId()	Returns the current user's userid.
UserName getUserName()	Returns a Username bean that represents the user’s text name.
boolean isAnonymousUser()	Returns true if the user is logged in as the anonymous user.
boolean isRoleChooserOn()	Returns true if roles are being used.
setAttribute(String key, Object object)	Sets an attribute on the session cache and stores it with the associated key. If the object added is an instance of LoginEnvironmentBindingListener, the bound method is called on the object with the following parameters: LoginEnvironment env, String name where the name is the key that will be used to store the object on the Hashtable.
getHtmlOwEnv()	Gets the current HTMLOWEnv. This method is deprecated and should not be used.

The FormFactor bean has the following public interface:

float getBrowserLevel()	Returns the browser level as a float.
int getBrowserType()	Returns the browser type as an int. Valid return values are: • FormFactor.NS – Netscape browser • FormFactor.IE – Internet Explorer browser • FormFactor.MOZILLA – Mozilla browser • FormFactor.OTHER – Another browser
int getPlatform( )	Returns the browser type as an integer. Valid return values are: • FormFactor.WIN – Windows • FormFactor.MAC – Macintosh • FormFactor.PPC – Pocket PC • FormFactor.WINCE – Windows CE • FormFactor.OTHER – Another platform
int getFormat()	Returns the browser type as an int. Valid return values are: • FormFactor.HTML – HTML format • FormFactor.OTHER – Another format

The RoleEntry bean has the following public interface:

String getRoleDescription()	Returns the current role description.
String getRoleName()	Returns the current role name.
boolean getInclude()	Returns true if the role is included in *ALL.

The UserName bean has the following public interface:

String getFirstName()	Returns the user's first name or initial.
String getLastName()	Returns the user's last name or initial.
String getMiddleName()	Returns the user's middle name or initial.
String getTitle()	Returns the user's title.
String getFullName()	Returns the user's name as stored in the database.

The UserName bean supports the following formats:

- LAST
- FIRST LAST
- FIRST MIDDLE LAST
- LAST, FIRST
- LAST, FIRST MIDDLE

- FIRST LAST, TITLE
- FIRST MIDDLE LAST, TITLE
- LAST, FIRST, TITLE
- LAST, FIRST MIDDLE, TITLE

Constructing a Database Interface

You can instantiate the ComponentDatabaseInterface in the following manners:

```
new ComponentDatabaseInterface(owEnv, componentName);

new ComponentDatabaseInterface(owEnv, componentName, UserId);
```

In the first constructor, LoginEnvironment is passed in as well as the component description (the COMPNAME field sent by the Portal Builder Servlet). This constructor assumes that the default user ID is *public. Consequently, any method that uses the default user ID instead of specifying one explicitly will retrieve only those entries that are available to the public.

Conversely, with the second constructor, you can specify a specific J.D. Edwards user ID.

Using the Public Fields

Several static fields are available in the ComponentDatabaseInterface class. One of these fields is public and the rest are protected so that they may be overridden. All are static so they may be referenced without instantiating the class.

The PUBLIC_USER field contains the string representation of the public component user ID, that is, *public. If this field is overridden with another value, then any instantiation of the ComponentDatabaseInterface that does not provide a user ID will use this instead.

The second static field is protected and stores the name of the database table in the J.D. Edwards environment where data for the class is stored. This field is called DB_NAME. Overloading this field will change the database table that is accessed whenever the class is instantiated. This makes it easy to extend the ComponentDatabaseInterface if you need to use different database tables.

The last static field is also protected and contains the user override type. It is named UOTY. The user override type is an entry type identifier. Most components will have a user override type of PC"(Portal Component). You can, however, override this value by extending this class. All of the functions within the ComponentDatabaseInterface would then use this user override type.

getComponentName Method

The getComponentName method returns the name of the component that instantiated the method. Because this name is provided in the constructor, it can be retrieved via this method at any time. Consider the following example:

```
package com.jdedwards.portal.components.helloWorld;

import com.jdedwards.services.login.*;

import com.jdedwards.portal.components.*;

class HelloWorldServlet extends PortalComponentServlet
```

```
{
protected void service(LoginEnvironment owEnv, PrintWriter out,
HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException

{

ComponentDatabaseInterface cda = new
ComponentDatabaseInterface(owEnv, "This is my Component");

System.out.println(cda.getComponentName());

}

public String getTitle()

{

return "Hello J.D. Edwards",

}

public Boolean requiresJDE()

{

return true;

}

}
```

This code returns the following text string:

```
This is my Component
```

setComponentName Method

The setComponentName method allows you to change the component name after the class is instantiated. Consider the following example:

```
protected void service(LoginEnvironment owEnv, PrintWriter out,
HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException

{

ComponentDatabaseInterface cda = new
ComponentDatabaseInterface(owEnv, "This is my Component");

cda.setComponentName("myComponent");

System.out.println(cda.getComponentName());

}
```

This code returns the following text string:

```
myComponent
```

getData Method

This method has two versions:

```
getData();
```

```
getData(String);
```

Both versions of this method return a string of data for a component in the database. Usually the data returned would be an XML document, but the returned value is not restricted in its format. Note, however, that the method allows only one data field per user, per component.

The first method uses the default user ID that is passed to the class during its instantiation, while the second method requires that you pass it the user ID. If no user ID is specified, then the first method uses the value of the PUBLIC_USER field.

Consider the following example:

```
protected void service(LoginEnvironment owEnv, PrintWriter out,
HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException

{

ComponentDatabaseInterface cda = new
ComponentDatabaseInterface(owEnv, "myComponent", "ME000001");

System.out.println(cda.getData());

System.out.println(cda.getData("ME000000"));

}
```

The above code retrieves the data field for the user ME000001, and then retrieves data for the user ME000000.

setData Method

The setData method has two versions:

```
setData(data);

setData(data, userId);
```

The first version sets the data to a string using the default user ID. The second one saves data using a specified user ID. After this data is set, it is also committed to the database.

getDataInputSource Method

This method is the same as the getData method. It has two versions, just like the getData method does, and the differences between the two versions are the same. What makes the getDataInputStream method different from the getData method, however, is that getDataInputStream returns an input stream instead of a string. This input stream implementation aids in the integration of some XML DOM parsers. Many DOM parsers expect a file or some other input stream to get the Data into the DOM.

See *getData Method* for more information about the getData method.

getDataOutputStream Method

This method has two versions:

```
getDataOutputStream();

getDataOutputStream(UserId);
```

This method provides an output stream which is required by many DOMs to save their data. This stream caches all of the data sent to it, and when the stream is closed, it saves the data to the database.

The first method uses the default user ID that was passed to the class during its instantiation, while the second method requires that you pass it the user ID. If no user ID is specified, then the first method uses the value of the PUBLIC_USER field.

getDefaultUserId Method

The getDefaultUserId method returns a string containing the default user ID that was used when the class was instantiated, set with the last setDefaultUserId method, or the value of the PUBLIC_USER field if no user ID was specified. This ID is used in every method that uses the default user ID.

setDefaultUserId Method

The setDefaultUserId method changes the value of the default user ID.

Organize Components

All Portal components reside in a directory structure. The directory structure facilitates finding a particular component. You can construct the directory however you want, and you can place a component in multiple folders.

When creating or personalizing workspaces, you see the directory structure when you select components for inclusion on a workspace. You can sort the list by component name (ID) or by description. Folders always appear at the top, and you double-click a folder to see its contents.

When creating or modifying a component, you see the same directory structure. Use it in the same way to identify one or more folders in which the component should appear.

Modify the directory structure on the same form you use to create a component.

► To create a component folder

1. From any J.D. Edwards Portal workspace, click Personalize on the Secondary Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, click the Components hyperlink.
3. In the Edit Existing Components section, perform one of the following actions:
   - Click Add New Folder to create a new folder that is not based on another folder.
   - Choose a folder in the Components list and click Copy Selected to create a new folder based on the folder you chose.
4. In the Folder Entry Form section, complete the following fields:
   - Component ID

     The system name for the folder. This name appears in the component list when the user clicks Sort by IDs. The maximum field length is 10 characters.

- Name

  The long name for the folder. This name appears in the component list when the user chooses not to sort by IDs (by not clicking Sort by IDs). The maximum field length is 30 characters.

5. To nest the folder under one or more other folders, choose a folder in the Available column under Parent Folders and click the right-pointing arrow.

   The folder will appear as a child folder in all folders listed in the Selected column.

   To keep the folder from appearing as a child of a folder, click a folder in the Selected column and click the left-pointing arrow.

6. Click OK.

   The system creates the folder.

   Click Cancel to return to the component creation section without saving the folder.

► To modify a component folder

1. From any J.D. Edwards Portal workspace, click Personalize on the Secondary Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, click the Components hyperlink.
3. In the Edit Existing Components section, choose the folder you want to modify from the components list and click Modify Selected.
4. In the Folder Entry Form section, change the description of the folder, if desired.

   The Name is the long name for the folder. This name appears in the component list when the user chooses not to sort by IDs (by not clicking Sort by IDs). The maximum field length is 30 characters.

5. To nest the folder under one or more other folders, choose a folder in the Available column under Parent Folders and click the right-pointing arrow.

   The folder will appear as a child folder in all folders listed in the Selected column.

   To keep the folder from appearing as a child of a folder, click a folder in the Selected column and click the left-pointing arrow.

6. Click OK.

   The system modifies the folder.

   Click Cancel to return to the component creation section without saving changes to the folder.

► To delete a component folder

Note

You cannot delete a folder that contains components or other folders. Depending upon your permissions, you might not be able to see all of the components or folders in a folder, so you might receive an error message when you are trying to delete a folder even though the folder appears to be empty.

1. From any J.D. Edwards Portal workspace, click Personalize on the Secondary Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. On Personalize, click the Components hyperlink.
3. In the Edit Existing Components section, choose the folder you want to delete from the components list and click Delete Selected.

   The system confirms the deletion.

4. Click OK.

   The system deletes the folder if it is empty.

Exporting and Importing Portal Components

The J.D. Edwards Component Exporter and the J.D. Edwards Component Importer allow you to archive all elements of a component, component group, or workspace and then install that component, component group, or workspace in any instance of the J.D. Edwards Portal. You can use these applications to save your Portal objects and to package and distribute them. You can also use the Importer to import Portal objects from the J.D. Edwards Update Center on the J.D. Edwards Knowledge Garden. When you import, the Importer automatically grants the person using the Import application Viewable, Selectable, Editable rights to the object. All J.D. Edwards Portal components consist of a database entry in the Component Definition table (F9060) and (optionally) a set of files residing on the Web server.

You access both the Import and Export applications for all Portal objects through the Portal Component Importer/Exporter component.

The Portal Component Importer/Exporter component requires Microsoft Internet Explorer 5 or higher or Netscape 6 or higher. The component is not supported on pervasive devices.

Note

IE5.5 sp 1 has a bug which prevents downloads. This bug is confirmed by Microsoft (MS knowledge base article Q279667).

Before You Begin

- Add the following section to your JAS.ini file:

```
[PORTALCONFIGURATION]

jde=jdelocation

servlet=servletlocation

backup=backuplocation
```

where

- *jdelocation* is the fully qualified location of the Web directory (WebSphere 3.5) or webclient.war directory (WebSphere 4.0) corresponding to the JDE virtual directory
- *servletlocation* is the fully qualified location of a directory in the application servers classpath where servlet classes are stored
- *backuplocation* is the fully qualified location of a directory where you want to backup overwritten files

Note

You can override the backup setting by using the personalize feature of the Portal Component Importer/Exporter component.

- If you are importing components, you must first download the archive from the J.D. Edwards update center and save it on a local machine. Then you can use the Importer to restore files from your local copy of the archive.

► To export a Portal workspace

From your Portal, choose a workspace that contains the Portal Component Importer component.

1. From the Portal Component Importer component, click Export Workspaces.
2. On Select Workspaces, in the Available Workspaces list, choose the workspace you want to export, and then click the right-pointing arrow to move your selection to the Selected Workspaces list.

   You can move as many workspaces as you want into the Selected Workspaces list.
3. Click Next.
4. On Preview Selected Workspaces, verify that the objects that will be exported are correct and click Next.
5. On Select Servlets to include with the component, choose the servlets you want to include from the Available Servlets list, and then click the right-pointing arrow to move your selection to the Selected Servlets list.

   You can move as many servlets as you want into the Selected Servlets list.
6. Click Next.
7. On Select Resources to include with the component, choose the resources you want to include from the Available Resources list, and then click the right-pointing arrow to move your selection to the Selected Resources list.

   You can move as many resources as you want into the Selected Resources list.

8. Click Next.

   The system prompts you to repeat steps 5 through 8 for each component you are exporting.

9. On Add special instructions to this archive, enter any additional information you want to provide to anyone who will be importing the archive, and click Next.

   The Save the Portal Component Archive form appears, and the system automatically begins the archive process. If the process does not appear to begin automatically, click the Manually download hyperlink.

   To abort the process, click Return to Workspace.

10. On File Download, click portal.car.
11. On Save As, enter the location where you want to save the file and click OK.

    The system completes the archive process.

► To export a Portal component

From your Portal, choose a workspace that contains the Portal Component Importer component.

1. From the Portal Component Importer component, click Export Portal Components.
2. On Select the Portal Component to export, navigate to the folder containing the component you want to export.
3. In the Available Components list, choose the component you want to export, and then click the right-pointing arrow to move your selection to the Selected Components list.

   You can move as many components as you want into the Selected Components list.

4. Click Next.
5. On Select Servlets to include with the component, choose the servlets you want to include from the Available Servlets list, and then click the right-pointing arrow to move your selection to the Selected Servlets list.

   You can move as many servlets as you want into the Selected Servlets list.

6. Click Next.
7. On Select Resources to include with the component, choose the resources you want to include from the Available Resources list, and then click the right-pointing arrow to move your selection to the Selected Resources list.

   You can move as many resources as you want into the Selected Resources list.

8. Click Next.

   The system prompts you to repeat steps 5 through 8 for each component you are exporting.

9. On Add special instructions to this archive, enter any additional information you want to provide to anyone who will be importing the archive, and click Next.

The Save the Portal Component Archive form appears, and the system automatically begins the archive process. If the process does not appear to begin automatically, click the Manually download hyperlink.

To abort the process, click Return to Workspace.

10. On File Download, click Download portal.car.
11. On Save As, enter the location where you want to save the file and click OK.

    The system completes the archive process.

► To export a Portal component group

From your Portal, choose a workspace that contains the Portal Component Importer component.

1. From the Portal Component Importer component, click Export a Component Group and related Components.
2. On Select the Portal Component Group to Export, navigate to the folder containing the component group you want to export.
3. In the Available Applications list, choose the component group you want to export, and then click the right-pointing arrow to move your selection to the Selected Applications list.

   You can move as many component groups as you want into the Selected Applications list.

4. Click Next.
5. On Select Servlets to include with the component, choose the servlets you want to include from the Available Servlets list, and then click the right-pointing arrow to move your selection to the Selected Servlets list.

   You can move as many servlets as you want into the Selected Servlets list.

6. Click Next.
7. On Select Resources to include with the component, choose the resources you want to include from the Available Resources list, and then click the right-pointing arrow to move your selection to the Selected Resources list.

   You can move as many resources as you want into the Selected Resources list.

8. Click Next.

   The system prompts you to repeat steps 5 through 8 for each component in all of the component groups that you are exporting.

9. On Add special instructions to this archive, enter any additional information you want to provide to anyone who will be importing the archive, and click Next.

   The Save the Portal Component Archive form appears, and the system automatically begins the archive process. If the process does not appear to begin automatically, click the Manually download hyperlink.

   To abort the process, click Return to Workspace.

10. On File Download, click Download portal.car.

11. On Save As, enter the location where you want to save the file and click OK.

    The system completes the archive process.

► To import a Portal component from a file

Use this process to import individual components as well as component groups.

From your Portal, choose a workspace that contains the Portal Component Importer component.

1. From the Portal Component Importer component, click Import Portal Components.
2. On Select the Portal Component to Import, click Browse and locate the archive you want to import.
3. Click Next.
4. On Validate Imported Portal Component Archives, click Next.

   The system notifies you if the archive contains files that already reside on your system. Additionally, if the archive includes special instructions, you see them now.

5. On Select the target folders for imported components, select one or more folders in the Available Folders column and click the right-pointing arrow to move your selections to the Selected Folders column.

   Folders in this instance exist merely for organizational purposes and are unrelated to a file structure. Each component will appear in each folder you select when a user searches for that component in the Portal interface.

6. Click Next.

   The system launches the import processes. It notifies you when it has completed the process.

7. On Confirmation, click Return to Workspace.

Portal Configuration

Portal Configuration

Through the J.D. Security Workbench application, you can determine, for anonymous and system users, groups, and *PUBLIC, who can access the J.D. Edwards Portal and its configuration forms such as personalization and component management. Through the Portal, you can exercise an even greater level of control by granting access privileges to different workspaces and components. You can also customize the two Portal toolbars.

Configuring J.D. Edwards Portal Security

The J.D. Edwards Portal Security application allows you to control, for individuals or for groups, access to J.D. Edwards Portal and to the following J.D. Edwards Portal interface forms:

- Portal Personalization - The Portal Personalization form allows you to specify components, colors, and so forth on a user's Portal workspace.
- Component Management - The Component Management form allows you to add, change, and delete Portal components from the system.
- Relationships - The Relationships form allows you to define view, select, and edit permissions for individual Portal components or workspaces based on user IDs or roles. A user who is an administrator in the Portal but who does not have this option enabled, will not be able to assign permissions to workspaces and components for roles or for *PUBLIC.
- Toolbars - The Toolbars form allows you to configure the Enterprise Navigation and Secondary Navigation toolbars.

Note

Do not confuse the Personalization form for the Portal with the personalization mode for Portal components. You cannot prevent users from accessing the personalization mode for components with the J.D. Edwards Portal Security application.

Security for specific workspaces and components is set individually. However, you can also give certain users administrative rights to the Portal. An individual with administrative rights is granted automatic access to all workspaces (viewable/editable) and components (viewable/selectable/editable). To grant a user administrative rights, set or create the admin parameter of the PORTALCONFIGURATION section of the JAS.ini to that user's ID. For example:

```
PORTALCONFIGURATION
admin=AB123456
```

► To configure Portal Security

From the Security Maintenance menu (GH9052), choose Security Workbench.

1. On Work with User/Group Security, from the Form menu, choose Setup Security and then choose Portal Security.

2. On Portal Security, in the User/Role field, enter the ID of the user or the role for whom you want to establish security.
3. Perform one of the following steps:
   - To deny access to the J.D. Edwards Portal, click Hide.
   - To allow access to the J.D. Edwards Portal, click View.
4. To allow access to the various J.D. Edwards Portal interface forms, click the appropriate checkboxes.
5. Click OK when finished.

Setting Workspace Permissions

After creating a workspace for the J.D. Edwards Portal, you can designate access of varying degrees for users, roles, or *PUBLIC.

Note

You must have the Relationships Option enabled for your account in the J.D. Edwards Portal Security application to be able assign permissions to roles and to *PUBLIC.

Workspace permissions work in conjunction with component permissions. If you grant users access to a workspace but do not grant them access to the components on that workspace, then users will see a blank workspace. The Portal resolves permission conflicts by providing the highest level of access allowed.

See Also

- *Portal Workspaces*

▶ To set workspace permissions

In the following task, if you do not see a particular option, your account probably has not been configured to allow you access to the feature in question. Contact your system administrator for assistance.

1. From the Portal, click Personalize on the Workspace Navigation Bar.
2. On the Workspace Navigation Bar, choose a workspace from Select Workspace.

   Unless you are a system administrator, you can only choose the workspaces that you have created or to which you have been granted access.
3. In the Set Workspace Permissions section, click Permissions.

   Unless you are a system administrator, this hyperlink is available only for those workspaces that you own (that you have created) or to which you have been granted the view/edit/grant permission level of access privileges.
4. In the Add User Relationship for Workspace, choose one of the following options:
   - User

     If you want to define access for a user, choose the user option, and then enter the user's ID.

- Role

  If you want to define access for a role, choose the role option, and then choose a role from the associated drop-down menu.

- PUBLIC

5. From the menu below the Public option, choose the type of access to the workspace that you want to grant.
6. Click Update.

   The user, role, or *PUBLIC appears in the Edit/Remove Relationships for Workspace section of the form.

7. To delete a user, role, or *PUBLIC from the access list, click the checkbox in the Remove column next to the line to be deleted, and then click Update.
8. To add other users or roles to the access list, repeat steps 4-6.
9. When you are finished, click OK.

Setting Component Rights

After creating a component for the J.D. Edwards Portal, you can designate different rights for users, roles, or *PUBLIC:

Note

You must have the Relationships Option enabled for your account in the J.D. Edwards Portal Security application to be able assign rights to roles and to *PUBLIC.

- View

  Individuals who have View rights to your component can view it if it resides on a workspace to which the user has at least View rights.

- View/Select

  Individuals who have View/Select rights to your component can not only view the component, but can choose to include it on workspaces as well.

- View/Select/Edit

  Individuals who have View/Select/Edit rights to your component can not only view it and include it on workspaces, but can make changes to the component as well.

- View/Select/Edit/Grant Permissions

  Individuals who have View/Select/Edit/Grant Permissions rights to your component can not only view, select, and edit the component, but can grant other individuals access rights to the component as well.

Component permissions work in conjunction with workspace permissions. If you grant users access to a workspace but do not grant them access to the components on that workspace, then users will see a blank workspace. The Portal resolves permission conflicts by providing the highest level of access allowed.

► To set component permissions

In the following task, if you do not see a particular option, your account probably has not been configured to allow you access to the feature in question. Contact your system administrator for assistance.

1. From the Portal, click Personalize on the Workspace Navigation Bar.
2. Click the Components hyperlink.
3. Choose a component, and then click Modify Selected.
4. In the Set Component Permissions section, click Permissions.

   Unless you are a system administrator, this hyperlink is available only for those components that you have created or to which you have been granted view/select/edit/grant permission rights.

5. In the Add User Relationship for Component section, choose the User, Role, or Public option.
   a. To define rights for a user, choose the user option, and then enter the user's ID.
   b. To define rights for a role, choose the role option, and then select a role from the associated menu. If you do not see this feature, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.
6. From the menu directly beneath the Public option, choose the type of rights to the component that you want to grant.
7. Click Update.

   The user, role, or *PUBLIC appears in the Edit/Remove Relationships for Component section of the form.

8. To change rights for a user, role, or *PUBLIC, choose a different option from the Permissions drop-down menu, and then click Update.
9. To delete a user, role, or *PUBLIC from the access list, click the checkbox in the Remove column next to the line to be deleted, and then click Update.
10. Repeat steps 5-7 to add other users or roles to the access list.
11. When you are finished, click OK.

Configuring the Component Group List

When you develop a J.D. Edwards Portal component, you can indicate that it is part of a component group. In this context, a component group is a set of components designed for a specific task. Together, the components in a component group might share the same value for attributes such as the protocol and the name of the server that hosts the components. When you indicate that a component is part of a component group, the Portal assumes that all of the components have the same base URL.

To make a component part of a component group, you select the component group from a predefined list at design time. This topic describes how to create and maintain the application list itself.

► To configure the component group list

You can configure the component group list by adding, editing, or deleting component group entries.

1. Click Personalize on the Workspace Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. Click the Enterprise Settings hyperlink.

   If you do not see the Enterprise Settings hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

3. In the Component Groups section of the form, perform any of the following tasks:
   - To add a component group entry to the list, complete the following steps:
     i. Click Create.
     ii. Complete the following fields and click OK:
       - Name

         Enter the system name for the component group entry in this 10-character field.

       - Description

         Enter the descriptive name for the component group entry. When you create a component and then associate it with a component group, this is the name you see in the drop-down list.

       - Document Base URL

         Enter the URL to use to fully qualify a component in the component group with a relative URL. The Portal uses standard browser protocol to resolve this URL. Consequently, if the first character of the URL is a slash, the Portal checks for the URL at the root of the path. On the other hand, if the first character of the URL is not a slash, then the Portal checks for the URL in the terminal directory indicated by the path.

         For example, if the path is http://servername/xyz, then a document base URL of /foldername/x.htm causes the Portal to check for x.htm in the following location: http://servername/foldername. A document base URL of foldername/x.htm causes the Portal to check for x.htm in the following location: http://servername/xyz/foldername.
   - To edit a component group entry, click the Edit hyperlink next to the component group entry you want to change.
   - To delete a component group entry from the list, click the Delete hyperlink next to the component group entry you want to delete. The system confirms the deletion.

Controlling the User Experience

You have several options available as to how the user logs in to the Portal. You also have control over the buttons and hyperlinks on the two Portal navigation bars.

When you install the Portal, you can use the Setup Page utility to designate a default workspace for your users. To run the utility, log in to the Portal as an administrator and execute the servlet http://hostname /jde/servlet/com.jdedwards.portal.SetupPageServlet. Choose the default workspace you want users to see when they launch the Portal. If you want to prevent users from accessing any other workspaces, click *Limit all users to view ONLY this workspace*. Click OK to save your settings. You must reset your server for the changes to take effect.

In addition, you can use the JAS.ini file to control the following aspects of the Portal interface:

[PORTALCONFIGURATION]

ShowCurrentEnvironmentRole= TRUE	Environment display. When set to TRUE, the system displays the current environment in the Workspace Navigation bar.
DefaultWorkspaceOnly=TRUE	Allows access to the default workspace only.
DefaultWorkspace=[WORKSPACE ID]	The workspace to display when no other workspace is specified or when DefaultWorkspaceOnly is set to TRUE.

[OWWEB]

DefaultEnvironment=[environment]	Default environment for login.

[LOGIN]

DisplayEnvironment=[Show|Hidden|ReadOnly|UseDefault]	Environment display rule if a default environment has been set with DefaultEnvironment. Show: User can override the default environment at login. Hidden: The user cannot see the environment box at login. ReadOnly: The user cannot tamper with the environment setting at login. UseDefault: The user can select different environments at login, but the system overrides any other user choice and logs in to the default environment anyway.
DisablePasswordAboutToExpire= [False|True]	Disables notification of impending password expiration when set to TRUE.

Configuring Portal Login

After establishing access to the Portal and its features with Security Workbench, users with access rights to the Portal and login using their J.D. Edwards ID and password. You can define the language to use for the login forms (assuming translated versions of the forms exist). In the JAS.ini file, in the OWWEB section, set InitialLanguageCode to the ISO code representing the initial language you want to use. This setting overrides any local language preferences until the user is logged in to the Portal.

You can make use of cookies to allow a local workstation to record the details of a user's login (including password). Doing so streamlines the login process for the user. You can also create an anonymous user account to allow users to login anonymously to the Portal.

If desired, you can configure the Portal so that instead of using the standard Portal login feature, you can use a simplified basic authentication procedure instead.

In addition to indicating when a user's password has expired, the system can notify users of an impending expiration, depending on the user's login method. A user can elect to change his or her password at the time of notification or can ignore the warning. If the user changes his or her password at the time of notification, then the system logs the user out and back in again with the new password.

The following login methods do not support password condition notification:

- Anonymous
- Parameter-based
- Direct (via cookie)

Notification of impending password expiration occurs automatically by default. If you want to suppress this notification, set the JAS.ini parameter, DisablePasswordAboutToExpire (in the [LOGIN] section), to TRUE.

Anonymous Users

J.D. Edwards Portal accepts an anonymous user login if the JAS server running the Portal instance has been configured to accept it and if the local JAS.ini file has been configured properly. You establish access rights to workspaces and components for the anonymous login as a separate user. The anonymous user is not a part of the *PUBLIC group. If you choose to establish an anonymous user, you must explicitly grant it access to workspaces and components.

A Portal instance that has been configured to accept anonymous users behaves differently from a regular instance. When a user browses to the Portal, the Portal displays the initial Welcome form instead of requiring a login. The user can view workspaces and components to which the anonymous user account has been granted access. The user cannot perform any command actions such as changing user options and personalizing the Portal, although if the user has an account, the user can choose to login normally by clicking the Login hyperlink in the Workspace Navigation Bar. After logging in, the user has access to all workspaces, components, and command functions to which his or her account is entitled. Upon logging out, the user sees the Welcome form instead of the login form.

The anonymous user account is a valid account, and a user can use it to log in to the Portal in the same way that a user can log in using a regular account. You can grant the anonymous user account access to command functions and other Portal features just as you can a normal account. A user who logs in using the anonymous account can then access the command actions and features you granted the account.

► To configure an instance of the Portal on the JAS server to accept an anonymous user login

1. Create a specific, unique user account in J.D. Edwards software to be used as the anonymous user account.
2. Add the settings to the command line parameters appropriate to your environment:
   - For the JVM for Tomcat/WebSphere 3.5, add these settings:

- Danon.user.ouid=*USR12345*

  where *USR12345* is a specific, unique user ID that you set up expressly to be the anonymous user account.

- Danon.user.pwd=*PASS12345*

  where *PASS12345* is the password that you want to use for the anonymous user account.

- Danon.user.env=*ENV12345*

  where *ENV12345* is the instance of the Portal for which you want to allow anonymous logins.

If you are using Tomcat, add these parameters to Project|Project Properties|Run|VM Parameters.

- For WebSphere 4.0, add the following runtime variables for the Web server:

Name	Value
anon.user.ouid	A specific, unique user ID that you set up expressly to be the anonymous user account.
anon.user.pwd	The password that you want to use for the anonymous user account.
anon.user.env	The instance of the Portal for which you want to allow anonymous logins.

3. Update the command line arguments as appropriate to your environment:

   a. If you are using WebSphere 3.5, use the WebSphere Administrative Console and go to the Application Server menu level to update the Command Line Arguments under the General tab as follows:

      ```
      –Danon.user.ouid=[USR] –Danon.user.pwd=[PWD] –
      Danon.user.env=[ENV]
      ```

   b. If you are using WebSphere 4, complete these steps:

      i. Navigate to the JVM Settings tab of the Application Server.

      ii. Add the `anon.user.ouid`, `anon.user.pwd` and `anon.user.env` entries with their respective values in the System Properties section.

      iii. Click Add, and then click Apply.

      The system updates the command line arguments.

4. Add the following line to the [OWWEB] section of the JAS.ini file:

   ```
   AnonAccess=true
   ```

5. To create a hyperlink that allows the user to launch a regular Portal login process, add the following line to the [PORTALCONFIGURATION] section of the JAS.ini file:

   ```
   ShowSignin=TRUE
   ```

   When you enable this feature, the login hyperlink appears on the Workspace Navigation Bar. A user logging in anonymously can use it to launch the regular Portal login process. In this way, a user with a Portal account who logs in anonymously can then log in as a regular user.

6. Restart the application server instance on the server for the changes to take effect.

Direct Login

You can use cookies to record a user's user name, environment, role, language, and even password to streamline the login process the next time the user logs in.

To configure the Portal to record user name, environment, role, and language (but not password), add or edit the following line in the JAS.ini file in the SECURITY section:

```
UseLogonCookie=TRUE
```

If the user logs on before the cookie expires, he or she is prompted only for a password.

To configure the Portal to record all user-related information, including password, add or edit the following line in the JAS.ini file in the SECURITY section:

```
UseLogonCookie=DIRECT
```

If the user logs on before the cookie expires, the login process is transparent to the user, and the Portal appears to launch directly. With this setting, a standard encryption key is used when the system records the password. However, you can use your own encryption key when writing passwords. To set an encryption key, add or edit the following line in the JAS.ini file in the LOGIN section:

```
PassKey=[alphanumeric key]
```

You can also set cookie expiration time. When the cookie expires, its information is deleted. If a user logs in after his or cookie has expired, the user must provide login information again. To set cookie expiration time, add or edit the following line in the JAS.ini file in the SECUIRTY section:

```
CookieLifeTime=[time in days]
```

Basic Authentication

You can configure the Portal to use a generic web login process instead of the default Portal login process. Although basic authentication streamlines the login process, the user cannot specify role or environment. Consequently, when enabled, all users log in to the default environment with the default role.

Typically, basic authentication is used by third-party products to log into the Portal. For a third-party product to use basic authentication, the product must submit a basic authentication header to the Portal.

To enable basic authentication, add or edit the following lines to the specified sections of the server's JAS.ini file:

Section	Setting
SECURITY	SSOEnabled=TRUE
OWWEB	DefaultEnvironment=[an environment]
OWWEB	DefaultRole=[a role]

Configuring the Portal Toolbars

You can create and modify buttons for the Enterprise Navigation Bar and links for the Workspace Navigation Bar. All buttons and links execute URLs when clicked. Besides specifying their functions, you can also specify the appearance of buttons on the Enterprise Navigation Bar. URLs which are not fully qualified are assumed to reside on the JAS server.

The Portal comes with 34 standard icons that can be used as Enterprise Navigation Bar buttons. These icons comprise the palette from which you choose when you create buttons. You can add new images to this palette.

► To add icons to the button palette

J.D. Edwards recommends that button icon images measure 27x27 pixels.

1. Add or edit the JAS.INI entry, NumberOfIcons, in the PORTALCONFIGURATION section to reflect the number of icons you want loaded into the palette.
2. Copy the icon image files onto the server in the /owportal/images directory.

   Use this convention to name images:

   icon*X*.gif

   where *X* is a number from 1 to the number of images you want shown in the palette. The icons appear on the palette in the order in which they are numbered.

► To create an Enterprise Navigation Bar button

1. Click Personalize on the Workspace Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. Click the Enterprise Settings hyperlink.

   If you do not see the Enterprise Settings hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

3. In the Enterprise Navigation Bar Icon Editor section of the form, click Add.
4. Complete the following fields:
   - Icon ID

     Enter the system name for the button in this 10-character field.

   - Description

     The button's description will be used as its hover help. That is, the description appears when the user momentarily holds the mouse pointer over the button.

   - Image URL

     Click a button image or enter the URL of a different image that you want to use.

   - Link URL

To link to a specific workspace, append the following to the end of the URL: ?FORMACTION=LOADING&NEWLAYOUT=*myworkspace* where *myworkspace* is the system name of the workspace that you want to display.

5. Click OK.

   The new button is added to the list of buttons available on the Enterprise Navigation toolbar.

Note
You must assign the button a sequence number or it will not appear on the toolbar.

► To configure the Enterprise Navigation Bar

This process describes how to determine the appearance and the order of buttons on the Enterprise Navigation Bar. For instructions on choosing a different URL for the toolbar's background image, see *Changing the Appearance of the Portal*.

1. Click Personalize on the Workspace Navigation Bar.

   If you do not see the Personalize hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

2. Click the Enterprise Settings hyperlink.

   If you do not see the Enterprise Settings hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

3. In the Enterprise Navigation Bar Icon Editor section of the form, perform any of the following tasks:

   - To reorder the available buttons, use the drop-down fields in the Sequence column.

     Buttons with a dash in the Sequence column instead of a number will not appear on the toolbar.

   - To change the icon, description, or other characteristic of a button, click the associated Edit hyperlink for the button.
   - To delete the button from the system, click the associated Delete hyperlink for the button.

4. When finished, click OK.

► To alter hyperlinks on the Workspace Navigation Bar for a workspace

Component developers can make a special kind of component called a link component. When you add link components to the Workspace Navigation Bar, they appear as additional hyperlinks.

In the following task, if you do not see a particular option, your account probably has not been configured to allow you access to the feature in question. Contact your system administrator for assistance.

1. From the Portal, click Personalize on the Workspace Navigation Bar.
2. On the Workspace Navigation Bar, choose a workspace from the Select Workspace field.
3. In the Workspace Details section, click Workspace Navigation Bar Links.
4. Add, remove, and rearrange links by performing any of the following tasks:
    a. To add a link to the Workspace Navigation Bar, choose the link component that you want to add, and click the right-pointing arrow between the Available list and the Selected list.
    b. To remove a link from the Workspace Navigation Bar, choose the link that you want to remove and click the left-pointing arrow between the Available list and the Selected list.
    c. To rearrange the order of the links, choose the link that you want to move and click Move Up or Move Down.
5. To list the components by their IDs instead of by their names, click Sort by IDs.
6. When you are finished, click OK.

Default User Preferences

You can configure default settings for all user-configurable parameters in the OneWorld Portal. The system applies these settings until a user overrides them by changing preferences.

► To configure default user preferences

From the Portal, click Enterprise Settings on the Workspace Navigation Bar.

If you do not see the Enterprise Settings hyperlink, your account probably has not been configured to allow you access to this feature. Contact your system administrator for assistance.

1. In the Default User Preferences section, click Modify Default User Preferences.
2. On Personalize, in the Default User Preferences section, choose the settings you want to use for the default value for each option.
3. Click OK.

Configuring Inherited Trust

Some components can effectively log a user in to an application using inherited trust. To enable this function, you must first register the InheritedTrustManagerServlet servlet in WebSphere.

After registering the servlet, if you have problems connecting to the server, try increasing the value of cach_itrust_purge in the PORTALCONFIGURATION section of the JAS.ini file.

Inherited Trust

When a component provides access to another application that requires a login, the system prompts the user to log in to that application. With inherited trust, however, you can set URI and Isolated URI components so that the Portal logs in for the user instead. After an initial log

in, the log in to that specific application with that specific component is transparent to the user.

To apply inherited trust to a URI or Isolated URI component, select the Inherited Trust option when you create the component.

The Portal's Inherited Trust Manager servlet manages all inherited trust requests. To implement inherited trust, the component accesses this manager by building a URL from the given Portal host name, Portal port, and Portal protocol. For example:

```
protocol://hostname:port/jde/servlet/com.jdedwards.portal.InheritedTrustManager
```

The Portal Request ID provides a method to determine the age of the request. The Inherited Trust Manager accesses the static list of Request IDs on Portal server, further limiting the possibility of hacking into the system.

The target application must be set to match the password and other information to the user ID. For example, these values can be encrypted in a database using a Secret Key. On each login attempt after the initial log in, the application looks up the password and other information for the user ID, decrypts it, and proceeds with the login as if the user had typed them into the login form.

Each time a manual sign on occurs (such as logging into the application without the Portal), the database entry is checked to see if the password or other information has changed. If it has changed, the database entry is cleared for that user ID and the new data is updated.

Secret Enterprise Key

A secret enterprise key is what makes Inherited Trust logins secure. All applications that wish to use inherited trust must know this key—if it gets changed in one place, it must be changed in all others. If an application is aware of this key, trust is established.

To specify this key on the J.D. Edwards Portal, use the init-parameter called "secretEnterpriseKey" as in the following example:

```
<servlet>
    <servlet-name>
        com.jdedwards.portal.InheritedTrustManagerServlet
    </servlet-name>
    <servlet-class>
        com.jdedwards.portal.InheritedTrustManagerServlet
    </servlet-class>
    <init-param>
        <param-name>secretEnterpriseKey</param-name>
        <param-value>this is my secret enterprise key</param-value>
    </init-param>
</servlet>
```

Specifying the key in this manner prevents it from being accessed via other servlets that might attempt to retrieve this value for malicious use.

ASP Implementation

If your component is written using ASP, you must have the provided COM DLL that interfaces with the Inherited Trust Manager Servlet. Your component does not have to be served by the same computer, only both computers must be able to interact with one another. The DLL file is located in the jdewww/owportal/ directory provided by the install CD.

When you enable the Inherited Trust option for a URI or an Isolated URI component, the following set of parameters is sent to your component:

```
'Get variables required for Inherited Trust

Dim intpbsurl, jdeowpUserID, jdeowpRequestID, jdeowpReferer, jdeowpProtocol,
jdeowpServer, jdeowpPort

jdeowpUserID = Request.QueryString("jdeowpUserID")

jdeowpRequestID = Request.QueryString("jdeowpRequestID")


intpbsurl = Request.QueryString("intpbsurl")

If intpbsurl = "" Then

   'Build server/port from the referer

   jdeowpReferer = Request.ServerVariables("HTTP_REFERER")

   jdeowpProtocol = Split(jdeowpReferer,":")(0)

   jdeowpServer = Split(jdeowpReferer,"/")(2)

   If UBound(Split(jdeowpServer,":")) = 1 Then

          jdeowpPort = Split(jdeowpServer,":")(1)

          jdeowpServer = Split(jdeowpServer,":")(0)

   Else

          jdeowpPort = "80"

   End If

Else

   'Build server/port from intpbsurl (needed when a local director is
involved)

   jdeowpProtocol = Split(intpbsurl,":")(0)

   jdeowpServer = Split(intpbsurl,"/")(2)

   If UBound(Split(jdeowpServer,":")) >= 1 Then

          jdeowpPort = cStr(Split(jdeowpServer,":")(1))

          jdeowpServer = Split(jdeowpServer,":")(0)

   Else 'Port not given, use default port

          jdeowpPort = "80"

          jdeowpServer = Split(jdeowpServer,":")(0)

   End If

End If
```

After these variables are stored, the page can interface with the COM DLL as follows:

```
Dim iTrust, iTrustResult, iTrustPassword, iTrustProject

If Request.QueryString("PortalUserID") <> "" Then

'Uses a COM DLL:

   'To Register Run (once):  regsvr32.exe pathToDll/InheritedTrust.dll

   'To Unregister Run:  regsvr32.exe /u pathToDll/InheritedTrust.dll


'Execute the Inherited Trust Model

   Dim secretEnterpriseKey

   secretEnterpriseKey = "this is my secret enterprise key"

   Set iTrust =
Server.CreateObject("InheritedTrust.InheritedTrustForPortal")

   iTrustResult =
iTrust.VerifyInheritedTrust(CStr(Request.QueryString("PortalProtocol")), _

   Cstr(Request.QueryString("PortalServer")), _

   Cstr(Request.QueryString("PortalPort")), _

   Cstr(Request.QueryString("PortalRequestID")), _

   Cstr(Request.QueryString("PortalUserID")), _

   Cstr(secretEnterpriseKey) )

   If iTrustResult = "true" Then

          'It is safe to assume that the PortalUserID is really the user
trying to get in

          'Get the User ID/Password/Project from the Database and log the
user in

   End If

End If
```

Java Implementation

If your component is written in Java, you must have the inheritedtrust.jar file in your classpath to interface with the Inherited Trust Manager Servlet. Your component does not have to be served by the same computer, only both computers must be able to interact with one another. The jar file is located in the jdewww/owportal/ directory provided by the install CD.

When you enable the Inherited Trust option for a URI or an Isolated URI component, the following set of parameters is sent to your component:

- `intpbsurl`
- `jdeowpUserID`
- `jdeowpRequestID`

To interface with the Inherited Trust library, the server protocol, server IP address, and server port must be parsed out of the intpbsurl parameter. The other two values are passed, and the

secretKey must be the secret enterprise key as specified in the init-parameter of the Inherited Trust Manager Servlet. The method to use from the JAR file is:

```
com.jdedwards.portal.inheritedtrust.InheritedTrustForPortal.VerifyInheritedT
rust(

String protocol, String server, String port,

String requestID,  portalUserID, String secretKey);
```

This method returns true if Inherited Trust is verified and false if it failed. Failure may be caused by mismatching enterprise keys, too much time passed from initial request until verification, or a network error.

▶ To configure inherited trust

From your desktop, launch the WebSphere Administrative Console.

1. On WebSphere Administrative Console, expand the tree to reveal the Portal application ([WebApp Name]).

   This is a typical tree expansion:

   WebSphere Administration Domain
   -[Server Node Name]
   -[AppServer Port]
   -[ServletEngine Port]
   -[WebApp Name]

2. Right-click [WebApp Name] and choose Create and then Servlet.
3. On Create Servlet, complete the following fields:
   - Servlet Name

     Enter `com.jdedwards.oneworld.owportal.InheritedTrustManagerServlet`

   - Servlet ClassName

     Enter `com.jdedwards.oneworld.owportal.InheritedTrustManagerServlet`

4. Click Add under the Servlet Web Path List box and enter the following value:

   `servlet/com.jdedwards.oneworld.owportal.InheritedTrustManagerServlet`

   This action appends VH_OneWorld_X/jde/ so that it reads as follows: "VH_OneWorld_X/jde/servlet/com.jdedwards.oneworld.owportal.InheritedTrustManagerServlet".

5. Click the Advanced Tab and complete the following fields:
   - Init Parm Name

     Enter `secretEnterpriseKey`

   - Init Parm Value

Enter the value you want to use for the secret enterprise key.

6. Change the value of Load at Startup from true to false, and click OK.
7. On WebSphere Administrative Console, right-click [WebApp Name] and choose Create and then Servlet.
8. On Create Servlet, complete the following fields:
   - Servlet Name

     Enter `com.jdedwards.portal.InheritedTrustManagerServlet`
   - Servlet ClassName

     Enter `com.jdedwards.portal.InheritedTrustManagerServlet`
9. Click Add under the Servlet Web Path List box and enter the following value:

   ```
   servlet/com.jdedwards.oneworld.owportal.InheritedTrustManag
   erServlet
   ```

   This action appends VH_OneWorld_X/jde/ so that it reads as follows: "VH_OneWorld_X/jde/servlet/com.jdedwards.oneworld.owportal.InheritedTrustManag erServlet".
10. Click the Advanced Tab and complete the following fields:
    - Init Parm Name

      Enter `secretEnterpriseKey`
    - Init Parm Value

      Enter the value you want to use for the secret enterprise key.
11. Change the value of Load at Startup from true to false, and click OK.
12. Close the WebSphere Administrative Console and restart the server.

    You must restart the server for the change to take effect.

Portal-Related JAS.ini Settings

The following table lists the JAS.INI settings that apply to how the Portal handles login/logout:

INI Section	INI Setting	Description
OWWEB	AnonAccess=TRUE	Enables access to the Portal an anonymous user account.
SECURITY	SSOEnabled=TRUE	Enables basic authentication in place of the default Portal login.
OWWEB	LogoutProcessTimeout=[number of seconds]	For an HTML client, the time allowed after logout or a timeout for a business function to complete processing. The system halts the business function after the allotted time has passed.
SECURITY	UseLogonCookie=TRUE	Enables the use of cookies. When the user logs out, the cookie records the user

INI Section	INI Setting	Description
		user logs out, the cookie records the user name, role, and environment (but not password). The user is also prompted to either log out or log in as another user.
SECURITY	UseLogonCookie=DIRECT	Enables the use of cookies. When the user logs out, the cookie records the user name, role, environment, and password in encrypted format. The user is also prompted to either log out or log in as another user.
OWWEB	InitialLanguageCode=[ISO language code]	Initial language code to use for login and web page display. If this line is not included in the .ini file, the initial language defaults to English.
SECURITY	CookieLifeTime=[time in days]	The amount of time the login cookie can reside before being deleted. Each time the user logs in, the cookie's life is renewed.
PORTALCONFIGURATION	ShowCurrentEnvironmentRole=TRUE	Environment display. When set to TRUE, the system displays the current environment in the Workspace Navigation bar.
OWWEB	DefaultEnvironment=[environment]	Default environment for login.
LOGIN	PassKey=[alphanumeric key]	Override for the key used to encrypt cookies.
LOGIN	externalhost=[machine:port]	The machine:port of the local director. This setting causes links form the Portal go back through the local director in cases of port swapping. **Note** Sticky load balancing with a sticky time greater than the session timeout internal is required to use the Portal with a local director at this time.
LOGIN	DisplayEnvironment=[Show|Hidden|ReadOnly|UseDefault]	Environment display rule if a default environment has been set with DefaultEnvironment. Show: User can override the default environment at login. Hidden: The user cannot see the environment box at login. ReadOnly: The user cannot tamper with the environment setting at login. UseDefault: The user can select different

INI Section	INI Setting	Description
		environments at login, but the system overrides any other user choice and logs in to the default environment anyway.
LOGIN	DisablePasswordAboutToExpire=TRUE	Override that suppresses password about to expire notification.
PORTALCONFIGURATION	admin=[user ID|user ID|user ID...]	The list of users with administrative rights to the Portal. This list is pipe (|) delimited.
PORTALCONFIGURATION	servlet=[directory where servlets reside]	The directory where Portal servlets reside. The Component Importer/ Exporter uses the directory to find and write servlets. This is a directory that the JAS administrator must create and configure to be included in the JDE Web Application's classpath.
PORTALCONFIGURATION	jde=[jdewww directory location]	The jdewww directory created at JAS install time. The Component Importer/Exporter uses the directory to find and write HTML resources such as .html, .gif, and jpg files.
PORTALCONFIGURATION	backup=[backup directory]	A location where files that are about to be overwritten by the Component Importer are saved. This provides a backup of the overwritten files.
PORTALCONFIGURATION	cache_workspace_purge=[time in ms]	The time in milliseconds to retain a workspace in cache without being accessed before being deleted.
PORTALCONFIGURATION	cache_workspace_expire=[time in ms]	For expirable components, the time in milliseconds in addition to the last loaded timestamp before a workspace is deleted. To make a component expirable, implement `public long getLastLoadedTimestamp();`. A workspace does not expire unless a user accesses it.
PORTALCONFIGURATION	cache_workspace_timeout=[time in ms]	The interval in milliseconds the system should wait before checking cache for items to purge or expire. The lower the number, the better the memory conservation, but the slower the cache. If set to zero, the system never expires or purges workspaces from the cache.
PORTALCONFIGURATION	cache_workspace_refresh=[time in ms]	The time in milliseconds before the system deletes all workspaces from the cache.
PORTALCONFIGURATION	cache_component_purge=[time in ms]	The time in milliseconds to retain a component in cache without being

INI Section	INI Setting	Description
		accessed before being deleted.
PORTALCONFIGURATION	cache_component_expire=[time in ms]	For expirable components, the time in milliseconds in addition to the last loaded timestamp before a component is deleted. To make a component expirable, implement `public long getLastLoadedTimestamp();`. A component does not expire unless a user accesses it.
PORTALCONFIGURATION	cache_component_timeout=[time in ms]	The interval in milliseconds the system should wait before checking cache for items to purge or expire. The lower the number, the better the memory conservation, but the slower the cache. If set to zero, the system never expires or purges components from the cache.
PORTALCONFIGURATION	cache_component_refresh=[time in ms]	The time in milliseconds before the system deletes all components from the cache.
PORTALCONFIGURATION	cache_itrust_purge=[time in ms]	The time in milliseconds to retain an inherited trust session in cache without being accessed before being deleted.
PORTALCONFIGURATION	cache_itrust_expire=[time in ms]	For expirable components, the time in milliseconds in addition to the last loaded timestamp before an inherited trust session is deleted. To make a component expirable, implement `public long getLastLoadedTimestamp();`. An inherited trust session does not expire unless a user accesses it.
PORTALCONFIGURATION	cache_itrust_timeout=[time in ms]	The interval in milliseconds the system should wait before checking cache for items to purge or expire. The lower the number, the better the memory conservation, but the slower the cache. If set to zero, the system never expires or purges inherited trust sessions from the cache.
PORTALCONFIGURATION	cache_itrust_refresh=[time in ms]	The time in milliseconds before the system deletes all inherited trust sessions from the cache.
PORTALCONFIGURATION	cache_entbutton_purge=[time in ms]	The time in milliseconds to retain an Enterprise Navigation Bar button in cache without being accessed before being deleted.
PORTALCONFIGURATION	cache_entbutton_expire=[time in ms]	For expirable components, the time in milliseconds in addition to the last loaded

INI Section	INI Setting	Description
		timestamp before an Enterprise Navigation Bar button is deleted. To make a component expirable, implement `public long getLastLoadedTimestamp();`. An Enterprise Navigation Bar button does not expire unless a user accesses it.
PORTALCONFIGURATION	cache_entbutton_timeout=[time in ms]	The interval in milliseconds the system should wait before checking cache for items to purge or expire. The lower the number, the better the memory conservation, but the slower the cache. If set to zero, the system never expires or purges Enterprise Navigation Bar buttons from the cache.
PORTALCONFIGURATION	cache_entbutton_refresh=[time in ms]	The time in milliseconds before the system deletes all Enterprise Navigation Bar buttons from the cache.
PORTALCONFIGURATION	DefaultWorkspaceOnly=TRUE	Allows access to the default workspace only.
PORTALCONFIGURATION	DefaultWorkspace=[WORKSPACE ID]	The workspace to display when no other workspace is specified or when DefaultWorkspaceOnly is set to TRUE. Use upper case letters to set this parameter.
PORTALCONFIGURATION	ShowSignin=TRUE	Shows the regular login hyperlink on the Workspace Navigation Bar when the user logs in anonymously.
PORTALCONFIGURATION	startworkspace=WELCOME	workspace (uppercase)
PORTALCONFIGURATION	hlpimg=[help icon]	Default help icon file path and name such as: /jde/images/pic.gif.
PORTALCONFIGURATION	perimg=[personalize icon]	Default personalize icon file path and name such as: /jde/images/pic.gif.
PORTALCONFIGURATION	maximg=[expand icon]	Default expand icon file path and name such as: /jde/images/pic.gif.
PORTALCONFIGURATION	minimg=[minimize icon]	Default minimize icon file path and name such as: /jde/images/pic.gif.
PORTALCONFIGURATION	resimg=[minimize icon]	Default restore icon file path and name such as: /jde/images/pic.gif.
PORTALCONFIGURATION	Retimg=[return icon]	Default return icon file path and name such as: /jde/images/pic.gif.
PORTALCONFIGURATION	searchprefsurl=[search personalization servlet]	The URL the Portal calls to perform search personalization.

INI Section	INI Setting	Description
PORTALCONFIGURATION	archivespec=[Java applet archive]	The file and path name for the Java applet archive for personalization.
PORTALCONFIGURATION	pbsurl=[PortalBuilder servlet URL]	The URL of the PortalBuilder servlet.
PORTALCONFIGURATION	persurl=[Personalization servlet URL]	The URL of the Personalization servlet.
PORTALCONFIGURATION	prefurl=[Preferences servlet URL]	The URL of the Preferences servlet.
PORTALCONFIGURATION	clisturl=[Component List servlet URL]	The URL of the Component List servlet.
PORTALCONFIGURATION	cmsurl=[Component Manager servlet URL]	The URL of the Component Manager servlet.
PORTALCONFIGURATION	rulimg=[Rule image URL]	The URL of the default rule image URL. The rule image is used as the space between objects on the Portal navigation bars.
PORTALCONFIGURATION	edtimg=[Edit image URL]	The URL of the default edit icon URL. The edit icon is used for buttons that allow the user to alter an object.
PORTALCONFIGURATION	corplogourl=[Corporate logo image]	The URL of the default corporate logo. This image is used when the current workspace does not specify an image.
PORTALCONFIGURATION	corplogolinkurl=[Corporate logo link]	The URL of the default corporate logo hyperlink. This link is used when the current workspace does not specify a link.
PORTALCONFIGURATION	pagegreeting=[default page greeting text]	The text that appears on the default greeting page.
PORTALCONFIGURATION	CustomSchemesOnly=YES	Prevents the color schemes that were shipped with the Portal from appearing in the color scheme list when users personalize their Portal.
PORTALCONFIGURATION	CustomScheme[X]=[scheme-name|theme-color|page-color|enb-color|page-url|enb-image-url|logo-image-url|logo-link-url]	Pipe delimited color schemes where X (the key) is a number of a scheme, allowing you to enter a number of unique schemes. Custom schemes must be numbered consecutively starting with CustomScheme1, or they will not appear in the color scheme list. (enb stands for Enterprise Navigation Bar.) • scheme-name: The name of the scheme as it appears in the list. You may use spaces. • theme-color, page-color, enb-color: Hexadecimal color codes in the standard web format of #RRGGBB. • page-url, enb-image-url, logo-image-url: Either partially or fully-

INI Section	INI Setting	Description
		qualified URLs. • logo-link-url: A fully qualified URL.
PORTALCONFIGURATION	NumberOfIcons=[integer]	Number of Enterprise Navigation bar icons to display.
PORTALCONFIGURATION	DataMigrationHasOccured=TRUE	True if current pristine component and workspace data has been updated. This flag is set by the system when a Portal update has been successfully completed.
PORTALCONFIGURATION	useThreadsInSingleMode=TRUE	Indicates that the system should use threads to generate a component in maximize mode. Setting this parameter to TRUE might result in a performance enhancement, but the end user will see a blank screen whenever he or she clicks a link until the system is ready to display the object in full.

Upgrading the OneWorld Xe Portal

Upgrading the OneWorld Xe Portal

Version Xe SP17 of the Portal requires a minimum of Websphere 3.5.3 and the Servlet 2.2 Specification. This is an upgrade from the requirements of previous Xe releases.

If you are upgrading from a version of the Portal previous to SP17 to version SP18, then you must also be aware of this upgrade.

Two tables which were added to support form-factor for pervasive devices and component organization must be provided for the new system to function properly through the deployment of an ESU. The F9065 and F9066 tables must be serialized.

Version Xe SP17 of the Portal can use data created by previous releases of the Portal. However, the data it creates for components and workspaces is not compatible with these previous versions. Therefore, you should not run the two versions on the same set of database tables. Running the Xe and the Xe SP17 versions of the Portal with the same tables might cause the following conditions:

- All components created or modified using the Xe SP17 Portal will appear in minimize mode when viewed in the Xe Portal; the component toolbars are all hidden.
- In the Xe Portal, whenever a you attempts to edit a component that was created or modified using the Xe SP17 Portal, you will get Null Pointer Exceptions.

Furthermore, you might experience compatibility issues with older components that you created. You should test all non-J.D. Edwards components developed in the Xe environment before deploying them to the Xe SP17 Portal environment. Also note that when you update the Portal, the definitions of the J.D. Edwards pristine components are updated as well. As a result, any customizations you made to these components will be lost. The pristine components affected are:

- ACTIVNAV (to be updated)
- ENVCHECK (to be updated)
- WEBCHECK (to be updated)
- WELCOME (to be added)
- JDEIMPORT (to be added)

When you upgrade the Portal, old data from the pristine database is checked (via Component and Workspace beans) and updated as follows:

PortalBuilderServlet checks for the existence of a JAS.ini flag saying that J.D. Edwards data has been updated. If the flag is set then the servlet sets a static Boolean and the check ends. If the flag is not present, the Portal checks the ActivEraNav component to see what version the definition is. If the ActivEraNav component is version 2.0 then the servlet updates the JAS.ini, sets a static Boolean, and ends the check.

If the ActivEraNav component is less then version 2.0 and the user is not an admin user, then a message box appears informing the user that the Portal might not work correctly because some of its data needs to be upgraded. It will tell them to contact their system administrator. At this point, the Portal will continue running on the old data.

If the ActivEraNav component is less then version 2.0 and the user is an admin user, the user will be informed that some changes to the database tables need to be made to ensure compatibility and warn them that by doing so, the data in the current tables might become unusable by previous versions of the Portal system. If you cancel at this point, the system runs the check the next time you log in. If you continue, then you are directed to a servlet which will perform the database data upgrade.

This message will only appear once per sign on.

Deprecated Classes and Methods

Deprecated Classes and Methods

Several Portal classes and methods will be deprecated in future versions of the Portal. If you have components that use these classes and methods, you should modify them to make use of updated Portal classes and methods.

This topic details all those classes and methods supported by the version Xe SP17 of the Portal and lower which will be deprecated in future versions.

Understanding the ComponentServlet Class

The ComponentServlet allows your component to run under the JAS as well as to function with the OneWorld Portal. If you extend this class when you make your own components, you can take advantage of J. D. Edwards enhancements that will help you integrate into other components or allow your component to function in a stand-alone environment.

Extending the Component Servlet

The following servlet is the minimum servlet required to extend the component servlet correctly:

```
package com.jdedwards.oneworld.owportal.components.helloWorld;

import com.jdedwards.oneworld.owportal.components.*;

class HelloWorldServlet extends ComponentServlet

{

  protected void service(HttpServletRequest req, HttpServletResponse
res)throws servletException, IOException

  {

    super.service(req, res);
    PrintWriter out = res.getWriter();
    out.println(getTitle());
    out.close();

  }

  public String getTitle()

  {
     return "Hello OneWorld";
  }

}
```

First, the service class must be implemented by every servlet. It is part of the standard Java Servlet API. The service method calls the same method in the super class. By including this call in your code, you ensure that your component has access to a OneWorld Java Server. You will also automatically set up any required interfaces to other components.

Second, the servlet gets a writer from the HTTPServletResponse.getWriter. This allows you to write data to the output stream so that it will appear in the user's browser.

Third, the servlet gets the OneWorld environment that was referenced by the jsessionid. It is important to understand that the OneWorld Portal retrieves information from a URI via a web server, and therefore requires two session IDs. One session ID is the one that the servlet was called with; the other is the session ID that the OneWorld Portal Builder Servlet was called with. If you need to obtain information about the user and his or her login, you must get the OneWorld environment for the OneWorld Portal Builder with the PortalBase.jdeServletComponentLogin method. If the user has not logged in, the jdeServletComponentLogin will return null to the servlet and will then proceed to log the user into the system in its own thread.

Fourth, the servlet checks to see if it received information from the login. If not, that indicates that the user was not logged in. The servlet closes the output stream and terminates.

Finally, the servlet uses the getTitle method to return the name of the current component to an output string. Every child of the ComponentServlet class must override this method. It will help you access other portals, the ComponentServlet, or any object that the component servlet assumes exists.

See Also

- *Wrapping URIs* for more information about the extra parameter, jsessionid

Understanding the loadTemplate() Method

The loadTemplate method in the ComponentServlet class loads a template from an input stream or a URI. A template is a document encoded with a special version of HTML that allows data to be embedded in the document. For example, a template file might look like the following (assuming the component name is mycomp):

```
<html>
<head>
<title><PageDescription/></title>
<Common.Scripts.All>
function com_jdewards_mycomp_getOneWorldPortal()
{
return (document.ActivEraPortal != null)?true:false;
}
<Common.Scripts.All/>
</head>
<body>
<a href="http://{#jas}/jde/owportal">Click Here</a>
</body>
</html>
```

This example shows that, with a few exceptions, templates look much like HTML documents. For instance, some tags such as <Common.Scripts.All> are not standard to HTML.

Additionally, the file includes XML entities such as <PageDescription/> and a key, {#jas}, within one of the tags. Because the Portal cannot interpret templates, you must use the loadTemplate method to convert the template into an HTML format that it can understand. When loadTemplate reads this file, you can instruct it to modify it, as follows:

```
<html>
<head>
<title>Hello OneWorld</title>
<script language="JavaScript">
function mycompOwpWrapForm(url, myForm, addextra)
{
...
}
function mycompOwpWrapPage(url, addextra)
{
...
}
function mycompOwpTargetedPage (mylink)
{
...
}
function com_jdewards_mycomp_getOneWorldPortal()
{
return (document.ActivEraPortal != null)?true:false;
}
</script>
</head>
<body>
<a href="http://www.mysite.com/jde/owportal">Click Here</a>
</body>
</html>
```

To accomplish this conversion, the loadTemplate method uses the ComponentTemplateParser class to return a standard HTML template with a few additions that Portal components can use. See *Understanding the ComponentTemplateParser Class* for more information about how the ComponentTemplateParser class works.

You can call loadTemplate in any of the following ways:

- Stream loadTemplate functions
    - loadTemplate (InputStream, HTTPServletRequest);
    - loadTemplate (InputStream, HTTPServletRequest, boolean);

- URI loadTemplate functions
  - loadTemplate (String, HTTPServletRequest);
  - loadTemplate (String, HTTPServletRequest, boolean);

The stream functions can accept an InputStream of any source, including a file, an HTML stream, or even an input stream from the ComponentDatabaseInterface class. See *Understanding the ComponentDatabaseInterface Class* for more information about using this class. The URI functions, on the other hand, accept a URI of a template, create their own http stream, and then call their corresponding stream functions.

Two of the calls to loadTemplate contain Boolean operands. If set to true, loadTemplate converts the template to an HTML file as demonstrated at the beginning of this topic. If set to false, then the loadTemplate function returns the template without any replacements. The two calls without Boolean operands assume that you want to convert the template; that is, they function in the same way as their Boolean-enabled counterparts with the boolean argument set to true.

The conversion includes replacing two keys and four tags. Keys are used for inter-tag replacement. For example, {#jas} in the template example above is a key. Keys are case-sensitive, cannot include any white spaces, and are always in the form of {#key}. Tags, on the other hand, cannot be embedded inside of other tags. Tags are not case-sensitive, can include white spaces, and can either be in the form of <tag></tag> or <tag/>. Tags from the template example above are <PageDescription/> and <Common.Scripts.All> </Common.Scripts.All>. Also notice that each tag must either have a <tag></tag> equivalent or have a <tag/> equivalent. You cannot perform <tag> replacements.

The following tags and keys constitute the standard replacements:

- Keys
  - {#jas}

    This key is replaced with the location of the jde server.

  - {#portalurl}

    This key is replaced with the URL of the Portal.

  - {#componentname}

    This key is replaced with the name of the component as specified by the name tag in the Portal Builder.

- Tags
  - <Common.Scripts.All/>

    This tag is replaced with all of the scripts from <Common.Scripts.owpWrapForm/>, <Common.Scripts.owpWrapPage/>, and <Common.Scripts.owpTargetedPage/>. These scripts support Netscape 4 and Microsoft Internet Explorer 4 and allow greater integration between the Portal and the component without making the component dependent on the Portal.

  - <Common.Scripts.owpWrapForm/>

    This tag is replaced with a script that accepts a URL and an HTML form as data. If the component is running within the ActivEra Portal, the script builds the URL from the fields in the HTML form using the get method. It then submits the URL to the Portal Builder's standard JavaScript routine, which wraps the page

referenced in the URL in a Portal Builder interface. The data contained in the HTML form is submitted via a get method to the URL that you specify. You can also hard code some get method parameters in the URL because this script determines whether a ? symbol already exists in the URL.

If the servlet is being run outside of the ActivEra Portal, the script submits the HTML form using the method that is specified in the HTML form header. The data for this tag replacement is taken from the protected string getOwpWrapForm() method in the ComponentServlet class. See *Understanding the OwpWrapForm Script* for more information.

- <Common.Scripts.owpWrapPage/>

  This tag behaves the same as the OwpWrapForm JavaScript function except that it does not append HTML form data. If the component is being displayed in the Portal, the provided URL is sent to the Portal's standard JavaScript function. Otherwise, the URL is called directly. See *Understanding the OwpWrapPage Script* for more information.

- <Common.Scripts.OwpTargetedPage/>

  This tag behaves the same as the OwpWrapPage JavaScript function except that it handles a target. The specified target can be any valid target; however, a target of _portal will be wrapped in the Portal Builder if the servlet is running in the Portal. If the servlet is not running in the Portal, then the tag will act the same as _top. See *Understanding the OwpTargetedPage Script* for more information.

Unless specifically noted above, all tag replacements also include the unmodified tag contents. To illustrate, the tag

<Common.Scripts.OwpTargetedPage>
document.write("me");</Common.Scripts.OwpTargetedPage>

would be replaced with this code (assuming that the component name is mycomp):

```
<SCRIPT language="JavaScript">
function mycompOwpTargetedPage (url, target){
...
}
document.write("me");
</SCRIPT>
```

The tags themselves (<Common.Scripts.owTargetedPage> and </Common.Scripts.OwpTargetedPage>) would be replaced as shown, but the body (document.write("me")) would be carried over intact.

Understanding the OwpWrapForm Script

The OwpWrapForm script is used to submit the contents of an HTML form so that the HTML form can be wrapped into the Portal. The function has the following format:

```
componentnameOwpWrapForm(url, Form, addExtra, type)
```

- *componentname* is the system name of the component (not its description).
- `url` is the URL of the object to which you would like to submit the form. You can include ?property-value properties in the URL.
- `Form` is the name of the HTML form.
- `addExtra` is a Boolean value specifying whether the extra fields should be added to the URL. The following is a list of the extra fields:
  - jsessionid (contains the Portal Builder Servlet's session ID string)

    This parameter allows components to dynamically generate relevant state-based content.

  - STATE (contains one of the following strings: MIN, MAX, PER, HLP, RES, FRM, LNKFRM)

    This parameter aids in component language localization. MIN stands for minimized mode, MAX stands for maximized or link mode, PER stands for personalize mode, HLP stands for help mode, and RES stands for normal mode. These codes are subject to change in future releases; consequently, you might want to create your own vehicle for returning component mode.

  - LANGCODE (contains the language code string currently in use in the portal environment)

    This parameter allows a component to redirect to the PortalBuilderServlet when necessary.

  - PBSURL (contains the URL of the PortalBuilderServlet)
- `type` specifies the mode in which the URL target should be displayed. Valid types are Personalize, Maximize, Help, or Link. It also checks to make sure that the component is being run from the ActivEra Portal.

The script is already aware of the component name, so component name is not a required argument.

You can also wrap URIs with the four JavaScript functions, showPersonalizationPage, showHelpPage, showMaxPage, and showLinkPage. These functions mirror the OwpWrapForm JavaScript with the advantage of the ability to run in environments that do not support Java servlets. The advantage of using the Java servlet, however, is that the servlet does not require the component's ID. See *Wrapping URIs* for more information about these four JavaScript functions.

The following is an example of the OwpWrapForm script. If the component is inside the ActivEra Portal, the script executes the showPersonalizationPage method with the following URL:

```
http://www.mysite.com/test.html?id=7&mode=me
```

If the component is not in the ActivEra Portal, the script executes the URL directly.

```
<html>
<head>
<Common.Scripts.OwpWrapForm/>
</head>
<body>
<form name=test>
<field type=hidden name=mode value=me>
</form>
<a
href="javascript:{#componentname}OwpWrapForm(`http://www.mysite.com/
test.html?id=7', document.mode, false, `personalization')">Test</a>
</body>
</html>
```

Understanding the OwpWrapPage Script

The OwpWrapPage script calls the specified URL. The advantage of using this script instead of the showPersonalizationPage, showMaximizePage, showLinkPage, and showHelpPage functions is that the OwpWrapPage script already contains the name of the component. Therefore, you do not need to supply code within your servlet or template to extract this information. OwpWrapPage has the following format:

componentname`OwpWrapPage(url, addExtra, type)`

- *componentname* is the system name of the component (not its description).
- `url` is the URL of the object to which you would like to submit the form. You can include ?property-value properties in the URL.
- `addExtra` is a Boolean value specifying whether the extra fields should be added to the URL. The following is a list of the extra fields:
  - jsessionid (contains the Portal Builder Servlet's session ID string)

    This parameter allows components to dynamically generate relevant state-based content.

  - STATE (contains one of the following strings: MIN, MAX, PER, HLP, RES)

    This parameter aids in component language localization. MIN stands for minimized mode, MAX stands for maximized or link mode, PER stands for personalize mode, HLP stands for help mode, and RES stands for normal mode. These codes are subject to change in future releases; consequently, you might want to create your own vehicle for returning component mode.

  - LANGCODE (contains the language code string currently in use in the portal environment)

    This parameter allows a component to redirect to the PortalBuilderServlet when necessary.

  - PBSURL (contains the URL of the PortalBuilderServlet)

- `type` specifies the mode in which the URL target should be displayed. Valid types are Personalize, Maximize, Help, or Link. It also checks to make sure that the component is being run from the ActivEra Portal.

The script is already aware of the component name, so component name is not a required argument.

See Also

- *Wrapping URIs* for information about the URI functions

Understanding the OwpTargetedPage Script

The OwpTargetedPage script should be defined in the *onclick* event in the link. The script accepts any link assigned with one of the following targets and puts them in the Portal:

- _portal
- _portalnostuff
- _portalinframe
- _portalnostuffinframe

The inframe variations display the Enterprise Navigation Toolbar and the component each into their own frames, which allows a component that was not designed for the Portal to run smoothly within the Portal.

If none of these targets are indicated, the script allows the link to be processed with the original target and defaults to _top.

Consider the following example:

```
<a href= "http://www.myserver.com/index.html?id=1" target=
"_portalnostuff" onclick= "OwpTargetedPage (this)"> test</a>
```

This script sends the URL to the ActivEra Portal and tells the Portal not to send any extra parameters.

OwpTargetedPage has the following format:

componentname`OwpTargetedPage(myLink)`

- *componentname* is the system name of the component (not its description).
- `myLink` is the name of the link that the user clicks to activate the script.

The script is already aware of the component name, so component name is not a required argument.

Understanding the OneWorldComponentServlet Class

The OneWorldComponentServlet class is an abstract class designed to extend the functionality of the ComponentServlet class to provide the ability to acquire OneWorld environment and session information from the ActivEra Portal or by executing it directly. The OneWorldComponentServlet Class implements the standard service method for you. You, on the other hand, must provide some additional information not normally required by servlets. By deriving itself from the OneWorld component class, your component loads into the Portal faster then standard servlet components and use a minimum of bandwidth.

Extending the OneWorldComponentServlet Class

The following servlet is the minimum servlet required to extend the OneWorldComponentServlet class correctly:

```
package com.jdedwards.oneworld.owportal.components.helloWorld;

import com.jdedwards.oneworld.owportal.components.*;

class HelloWorldServlet extends OneWorldComponentServlet

{

protected void service(HTMLOWEnv owEnv, PrintWriter out,
HttpServletRequest req,

HttpServletResponse res)throws ServletException, IOException

{

out. println(getTitle());

}

public String getTitle()

{

return "Hello OneWorld";

}

public Boolean requiresOneWorld()

{

return false;

}

}
```

The service class must be implemented by every OneWorldComponentServlet. Because the standard service method is implemented in the parent class, this service method does not follow the Java servlet specification. For servlets that require an OneWorld environment, the PortalBuilderServlet must bypass the web server and instantiate the class directly. It will provide a PrintStream and an OWEnvironment to your class. You must not close the PrintWriter or instantiate a new one in your derived class; use the one that is provided to you.

The requiresOneWorld method tells the OneWorldComponentServlet Class or the PortalBuilderServlet whether a login and a OneWorld environment is needed. If this method returns false, you can expect the OneWorld Environment to be null. Use this method when you want to write a OneWorld component that does not use the database or session information but you still want the more robust interface and quicker instantiation of the OneWorld component mechanism.

All other aspects of this class are the same as the ComponentServlet class.

See Also

- *Wrapping URIs* for more information about the extra parameter, jsessionid
- *Understanding the ComponentServletClass*

Understanding the ComponentTemplateParser Class

The ComponentTemplateParser is a class that is used to parse HTML templates, which provides component servlets a simple and flexible method to display their data without having to embed the HTML inside of the Java code. You can use this class with both OneWorld components and standard Java servlets because it does not have any OneWorld dependencies. ComponentTemplateParser accomplishes this by offering a number of simple methods used to process such documents.

Constructing a Template Parser

To construct a template parser, provide ComponentTemplateParser with a string that contains the template to be processed. A copy of the provided string is stored within the ComponentTemplateParser class so that the original string will remain intact. The following is a sample instantiation of the ComponentTemplateParserClass:

```
package com.jdedwards.oneworld.owportal.components.helloWorld;
import com.jdedwards.oneworld.owportal.components.*;
class HelloWorldServlet extends ComponentServlet
{
protected void service(HttpServletRequest req,
HttpServletResponse res)throws ServletException, IOException
{
PrintWriter out = res.getWriter();
if (mypage==null)
myPage=loadTemplate("http://www.mysite.com/mytemplate.htm", req);
ComponentTemplateParser tp = new ComponentTemplateParser (mypage);
out.println(tp.toString());
out.close();
}
public String getTitle()
{
return "Hello OneWorld";
}
static String myPage;
}
```

You can pass a static variable to the ComponentTemplateParser because the contents of this string are copied into the ComponentTemplateParser during its instantiation; therefore, the class does not actually reference the static variable.

Understanding the toString Method

Like many classes in Java, the ComponentTemplateParser overrides the default toString() method. This method returns a string containing the contents of the template with all of its replacements. In the example shown in *Constructing a Template Parser,* the toString method is used to pass the template to the output stream.

Understanding the setString Method

This method sets the template parser's string. As with the constructor, the string stored in the class is actually a copy instead of a reference to the string provided by the parameter. The format for using this method is:

```
tp.setString("String");
```

Understanding the retrieve Method

This method retrieves the contents of data between two tags. Consider the following template code:

```
<Common> document.write("me");</Common>
```

When used on this template, myParser.retrieve("common") returns a string containing: document.write("me");.

The retrieve method also works on tags. If the tag is in the form <tag/> then the retrieve method will return an empty (not a null) string.

Using the encompass Method

Use this method to conditionally display a block of code. Consider the following template code:

```
<ieEncompass>
<script>
document.writeln("This is IE");
</script>
</ieEncompass>
```

If you were to use following call to the encompass method:

```
tp.encompass("ieencompass",true);
```

then Template Parser would return:

```
<script>
document.writeln("This is IE");
</script>
```

However, if you made this call to the encompass method:

```
tp.encompass("ieencompass",false);
```

then an empty string would be returned.

Understanding the replace Method

The replace method has two forms. The first form takes a tag name and a value. The second adds a Boolean value whereby you can tell the replace method to replace the first tag only. Simply put, this method replaces a tag with the method's value. Therefore:

```
tp.replace("myTitle", "My Component");
```

on the string:

```
<title><myTitle/></title>
```

would become:

```
<title>My Component</title>
```

Likewise, if the same statement were run on:

```
<title><myTitle>This is a test of</myTitle></title>
```

the string would still read:

```
<title>My Component</title>
```

To append a specific title to what was being encompassed in the tags, you would use code similar to the following:

```
String s = tp.retrieve("myTitle");
tp.replace("myTitle", s + " My Component");
```

Using the example above, this code would return a string that reads:

```
<title>This is a test of My Component</title>
```

By default, the replace method replaces every occurrence of a particular tag within the document. The retrieve method retrieves only the first occurrence. Running this code on a template such as this:

```
<title><myTitle>This is a test of</myTitle></title>
<body><myTitle>I like</myTitle></body>
```

would actually yield:

```
<title>This is a test of My Component</title>

<body>This is a test of My Component</body>
```

Obviously, this is not the intended result. Therefore, the second form of replace offers a Boolean value so that you can specify whether to replace only the first occurrence of an item. To process the template above correctly, the Java code should read as follows:

```
String s = tp.retrieve("myTitle");
while (s != null)
{
tp.replace("myTitle", s + "My Component", true);
s = tp.retrieve("myTitle");
}
```

In this code, the first occurrence of myTitle is retrieved. If the tag existed, then the first occurrence of the tag is replaced with the retrieved string and "My Component" is added to it. Then the next occurrence of the tag is retrieved and processed again. When there are no more myTitle tags to retrieve, then the string becomes null.

Understanding the replaceTag Method

Use the replaceTag method instead of the replace method when you only want to replace tags. Consider the following example template:

```
<html>
<head>
<title>Hello <mrman/></title>
</head>
<body>
Hello <boldIfFred><mrman/></boldIfFred>!
</body>
</html>
```

The purpose of this template is to print Hello Username in regular text unless the user name is Fred, in which case, the user name is printed in bold text. To perform this task, you would use the replaceTag method in the following manner:

```
tp.replace("mrman",username);
if (username.equals("Fred"))
tp.replaceTag("boldIfFred","strong");
else
tp.replaceTag("boldIfFred",null);
```

If the username was Fred then the <boldIfFred> and the </boldIfFred> tags would be replaced with and respectively. Otherwise, they would be removed entirely.

Understanding the parseKey Method

Use the parseKey method to replace keys. Key replacement has several advantages over tag replacement. First, the mechanism for replacing keys is simpler; thus, key replacement methods usually execute faster than tag replacement methods. Second, keys can exist and can be replaced while residing within tags, between <Script></Script> tags, and in comments, all of which are ignored by tag methods.

One disadavantage of key replacement is the fact that keys cannot encapsulate any data. Because there is no open key/close key syntax, the functions provided by the retrieve, replace, and encapsulate tag methods cannot be duplicated.

To illustrate how to use the parseKey method, consider a scenario in which you want to define a link to a graphic for a component. The graphic is located in the /images directory of your server. Defining the anchor as:

```
<a href="http://{#jas}/images/myimage.jpg">
```

and then replacing the key "jas" with the name of your server would render a fully qualified path to the image. The parseKey call that would perform this replacement would read:

```
tp.parseKey("jas", myJasServer);
```

If you derive your servlet from the ComponentServlet class and use the loadTemplate method that allows standard field replacements, the "jas" key will be automatically processed for you when you use this call.