

Modifying the Content Server Interface
10g Release 3 (10.1.3.3.0)

March 2007

Modifying the Content Server Interface, 10g Release 3 (10.1.3.3.0)
Copyright © 2007, Oracle. All rights reserved.

Contributing Authors: Jean Wilson

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Table of Contents



Chapter 1: Overview

| | |
|------------------------------|-----|
| Introduction | 1-1 |
| About This Guide | 1-1 |
| Audience | 1-1 |
| Conventions | 1-2 |
| Customization Overview | 1-2 |
| What's New | 1-3 |

Chapter 2: Skins and Layouts

| | |
|-----------------------------------|-----|
| Introduction | 2-1 |
| Types of Skins and Layouts | 2-1 |
| Skins | 2-2 |
| Layouts | 2-2 |
| Selecting Skins and Layouts | 2-2 |
| Configuration Entries | 2-2 |
| Anonymous User Interface | 2-3 |

Chapter 3: Using the CreateLayout Component

| | |
|-----------------------------------|-----|
| Introduction | 3-1 |
| Installing the Component | 3-1 |
| Using Component Manager | 3-2 |
| Using Component Wizard | 3-2 |
| Completing the Installation | 3-3 |
| Uninstalling the Component | 3-3 |

Chapter 4: Customizing the Interface

| | |
|----------------|-----|
| Overview | 4-1 |
|----------------|-----|

| | |
|---|-----|
| About Dynamic Publishing | 4-2 |
| Benefits of Dynamic Publishing | 4-2 |
| Tables Used in CreateLayout. | 4-3 |
| CreateLayout_Layouts Table. | 4-3 |
| CreateLayout_PublishedWeblayoutFiles Table. | 4-4 |
| CreateLayout_PublishedStaticFiles Table. | 4-5 |
| Additional File Entries | 4-6 |
| Creating New Layouts | 4-6 |

Appendix A: Optimizing Published File Use

| | |
|---------------------------------------|-----|
| Overview | A-1 |
| Bundling Files. | A-1 |
| Referencing Published Files | A-4 |

OVERVIEW

INTRODUCTION

The information contained in this guide is based on Content Server 10gR3. The information is subject to change as the product technology evolves and as hardware and operating systems are created and modified.

Due to the technical nature of browsers, web servers, and operating systems, Oracle, Inc. cannot warrant compatibility with all versions and features of third-party products.

This chapter covers the following topics:

- ❖ [About This Guide](#) (page 1-1)
- ❖ [Customization Overview](#) (page 1-2)
- ❖ [What's New](#) (page 1-3)

ABOUT THIS GUIDE

This guide provides information about customizing the look and feel of the user interface. This includes modifying the layout and the colors (“skins”) used with the Content Server.





Audience

This guide is intended for content server administrators who want to provide alternate interface navigation and design options for users.

Conventions

The following conventions are used throughout this guide:

- ❖ The notation `<Install_Dir>/` is used to refer to the location on your system where the content server instance is installed.
- ❖ Forward slashes (/) are used to separate the directory levels in a path name. A forward slash will always appear after the end of a directory name.
- ❖ Notes, technical tips, important notices, and cautions use these conventions:

| Symbols | Description |
|---|---|
|  | This is a note. It is used to bring special attention to information. |
|  | This is a technical tip. It is used to identify information that can be used to make your tasks easier. |
|  | This is an important notice. It is used to identify a required step or required information. |
|  | This is a caution. It is used to identify information that might cause loss of data or serious system problems. |

CUSTOMIZATON OVERVIEW

The User Personalization settings in Content Server provide layouts and skins from which you can build quick customizations to the content server navigation and design. Included are a number of standard layouts with varying types of top and left-navigation, as well as skins with varying groups of images and color schemes.

This document discusses the CreateLayout component, available from the Samples directory on the support web site. That component demonstrates how to change layouts and skins and make new customizations available to your end users on their User Profile

pages. This permits your end users to apply the navigation and color scheme of their choice.



Note: The skills required to create and modify skins or layouts include an understanding of HTML, Cascading Style Sheets, and JavaScript.

WHAT'S NEW

In version 10gR3 of the Content Server, layout files are published to the /weblayout directory instead of existing statically in that directory. Components can therefore alter files which are currently being published as well as publish completely new files. Many of the published files are created using the Content Server template infrastructure, so components can also use IdocS cript when creating or modifying weblayout files.

One benefit of this publishing method is the ability for administrators to simply disable co10gR3 changes to the weblayout files had to be undone manually by reverting to saved backups of those files.

SKINS AND LAYOUTS

INTRODUCTION

This section provides information about available skins and layouts provided by default with your Content Server. Skins and layouts provide alternate color schemes and alternate navigation designs.

This chapter covers the following topics:

- ❖ [Types of Skins and Layouts](#) (page 2-1)
- ❖ [Selecting Skins and Layouts](#) (page 2-2)
- ❖ [Configuration Entries](#) (page 2-2)

TYPES OF SKINS AND LAYOUTS

Several skins and layouts are provided by default with the Content Server. In addition, custom skins and layouts can be designed. When a user changes the skin or layout they change the look and feel of the interface. The user can select a skin and layout from the options provided on the My Profile page.



Note: Only administrators can create and make new or custom skins. See [Configuration Entries](#) (page 2-2) for additional information on setting the default look and feel of the user interface.

Skins

Skins define the “look” or color scheme of the layout (the default skin is Oracle). Custom skins and/or layouts can be designed or the existing skins can be modified.

Layouts

Layouts define the navigation hierarchy display (the default layout is Trays) and custom layouts can be designed. These layouts are provided:

- ❖ Trays—This layout with the standard Oracle skin is the default interface. High-level navigation occurs through the navigation trays.
- ❖ Top Menu—This layout provides an alternate look with top menus providing navigation.
- ❖ Classic—The Classic layout is included to provide compatibility with older (pre-5.5) browsers and only has one skin available.

SELECTING SKINS AND LAYOUTS



Important: This personalization functionality works with Internet Explorer 5.5+ or Netscape Navigator 7.0. The “classic” content server interface is available for users working with older browsers.

To change the interface used, follow these steps:

1. On the content server Home page, click **My Profile**.
2. On the content server My Profile page, select the desired skin and layout.
3. Click **Update** and view the changes.

CONFIGURATION ENTRIES

These values can be placed in the `<install_dir>/config/config.cfg` file to alter the default behavior for the Content Server instance:

- ❖ `LmDefaultLayout`—The name of the layout used by guests, and new users. The default is Trays, but it can be set to Top Menus or Classic.

- ❖ `LmDefaultSkin`—The name of the skin used by guests, and new users. The default is Oracle.
- ❖ `DisableAmberLayouts`—Set this to TRUE to completely disable the new layouts. This is helpful if you have many customizations or a large user base of older computers.

ANONYMOUS USER INTERFACE

The ExtranetLook component, available when installing Content Server, can be used to change the interface for those users who log in as anonymous. An example of this is when a content server-based website must be available to external customers without a login, but you want employees to be able to contribute content to that website.

You can customize your web pages to make it easy for customers to search for content, and then give employees a login that allows them to see the interface on login. To do this, modify the `ExtranetLook.idoc` file, which provides dynamic resource includes that can be customized based on user login. The Idoc file is checked into the content server repository so it can be referenced by the content server templates.

If the `IsWebServerPagesOnly` configuration variable is set to TRUE in the `<install_dir>/config/config.cfg` file, the ExtranetLook web server plug-in will deliver customized versions of pages created by the web server filter. It also disables cookie-based login functionality, which is discussed in the *Managing Security and User Access Guide*.

The following files in the `<install_dir>/data/users/` directory can be altered:

- ❖ `prompt_login.htm`
- ❖ `access_denied.htm`
- ❖ `report_error.htm`

To update the extranet pages after the Idoc file is changed and checked into the content server, click **Administration** then **Admin Applets**. Select the **Generate Filter Pages** option to generate the new filter pages. You must regenerate the filter pages any time the filter page is changed.

The remaining chapters in this guide contain information about the `CreateLayout` sample component, which demonstrates how to create new layouts and modify existing layouts for the Content Server.

This sample does not demonstrate how to create a new skin. It is designed to show developers how to write code that creates new layouts based on existing layouts.

USING THE CREATELAYOUT COMPONENT

INTRODUCTION

The CreateLayout sample component is available from the support web site. This component shows how to create new layouts for Content Server 10gR3.

Once installed and enabled, this component adds two new sample layouts to your Content Server interface: NewTopMenus and NewTrays. You can edit these samples as needed to create your own custom interface.

This chapter discusses the following topics concerning installation of the CreateLayout component:

- ❖ [Installing the Component](#) (page 3-1)
- ❖ [Completing the Installation](#) (page 3-3)
- ❖ [Uninstalling the Component](#) (page 3-3)

INSTALLING THE COMPONENT

You can install this component using one of two methods:

- ❖ [Using Component Manager](#) (page 3-2).
- ❖ [Using Component Wizard](#) (page 3-2).

Using Component Manager

Follow these steps to install the component using the Component Manager:

1. Select **Admin Server** from the Administration Menu.
The Content Admin Server page is displayed.
2. Click the instance name of the server where the component will be installed.
The Content Server *<instance-name>* page is displayed.
3. Click **Component Manager**.
The Component Manager page is displayed.
4. Click the **Browse** button and find the zip file that was downloaded and saved.
5. Highlight the component name and click **Open**.
6. Click **Install**. A message is displayed, detailing what will be installed.
7. Click **Continue** to continue with installation or **Cancel** to stop installation.
8. If you select **Continue**, a message appears after successful installation. You can choose one of two options:
 - To enable the component and restart the Content Server. You are returned to the Content Server *<instance-name>* page, where you can restart the server.
 - To return to the Component Manager, where you can continue adding components. When done, highlight the components you want to enable and click **Enable**. When finished enabling components, return to the Content Server *<instance-name>* page and restart the server.

Using Component Wizard

Follow these steps to install the component using the Component Wizard:

1. Start the Component Wizard by selecting **Start—Programs—Content Server—*<install_dir>*—Utilities—Component Wizard** (Windows) or by running the ComponentWizard script in the /bin directory (in UNIX).
The Component Wizard main screen and the Component List screens are displayed.
2. On the Component List screen, click **Install**.
The Install screen is displayed.
3. Click **Select**.

4. Navigate to the zip file that was downloaded and saved and select it.
5. Click **Open**.
The zip file contents are added to the Install screen list.
6. Click **OK**. You are prompted to enable the component.
7. Click **Yes**. The component is listed as enabled on the Component List screen.
8. Exit the Component Wizard.

Restart the Content Server.

COMPLETING THE INSTALLATION

After the component is installed and the Content Server has been restarted, you must make the component files available for use on the content server instance.

Select **Publish Static Layout Files** from the **Actions** menu in the **Administration** tray. This publishes the images needed by the new layout to the /weblayout directory.

After the publishing message appears indicating the publishing is complete, select **Publish Schema Configuration and Data** from the **Actions** menu in the **Administration** tray. In version 10gR3 of the Content Server, the list of available layouts on the User Profile page is powered by schema. By forcing a schema republish, this list is updated.

After the schema publishes, the new layouts are available on the User Profile page.


The next chapter discusses the procedure that was followed to create these new layouts.

UNINSTALLING THE COMPONENT

To uninstall a component, perform these steps using either Component Wizard or Component Manager:

1. Disable the component.
2. Restart the content server.
3. Click **Remove** or **Uninstall**.
4. Select **Publish Static Layout Files** from the **Actions** menu in the **Administration** tray. This removes all static files published by the component from the /weblayout directory.

5. Restart the content server.

 **Note:** Uninstalling a component means that the content server no longer recognizes the component, but the component files are not deleted from the file system.

CUSTOMIZING THE INTERFACE

OVERVIEW

The Top Menu, Trays, and Classic layouts are included by default with the system. The first two layouts have four skins (Collegiate, Stellent, Stellent05, and Windows) which can be used. The Classic layout has no optional skins associated with it. The layouts are written in JavaScript and the “look” of the skins is created using Cascading Style Sheets.

You can modify layouts and skins by altering the template files provided with the Content Server or design new skins and layouts by creating components that can be shared with other users.

This chapter provides an overview of this process, using the CreateLayout sample component as an example. It includes these topics:

- ❖ [About Dynamic Publishing](#) (page 4-2)
- ❖ [Creating New Layouts](#) (page 4-6)

ABOUT DYNAMIC PUBLISHING

In previous versions of the Content Server, all JavaScript and CSS files in the /weblayout directory were static. The JavaScript files were pure JavaScript and the CSS files contained nothing but CSS. No files had IdocScript and the contents of these files never changed no matter what configuration variables were set by the Content Server administrator.

This made these files simple to view, edit, and understand, but they were also difficult to customize, modify, or override. Custom components could either replace the file completely, or they could override single JavaScript functions and CSS definitions by re-declaring them somewhere in the head or body of the HTML document.

Neither method is recommended. If replacing the file completely, other components will not be able to replace the same file without causing all of the changes made by the first component to be lost. Also, when a component replaces one of the static weblayout files, there is no way to undo the change except to revert to a saved backup file. Neither disabling nor uninstalling the component recovers the original version of the file, making it difficult to debug problems.

The second method of redefining functions is also not a good solution. When components re-declare JavaScript methods, it becomes difficult to determine which method is being called on a given web page. In addition, no two components can re-declare the same JavaScript method. The one declared last is always used, while the others do nothing because their change is ignored by the browser.

Benefits of Dynamic Publishing

A new method is now used to alter the layouts and skins. When the Content Server starts, or when the PUBLISH_WEBLAYOUT_FILES service is run, the PublishedWeblayoutFiles table in the std_resource.htm file is used to publish files to the /weblayout directory. To have your custom component use this publishing mechanism, create a template then merge a custom row which uses that template into the PublishedWeblayoutFiles table.

Other users who want to modify or customize your file can override your template or your row in the PublishedWeblayoutFiles table. If your template uses any resource includes, other users can override any of these includes or insert their own IdocScript code using the standard `super` notation. When your component is disabled, the file is no longer published or modified and the Content Server returns to its default state.

In addition to giving others an easy way to modify and add to your work, you can also construct these once static files using IdocScript. For example, you can have the files

change depending on the value of a custom configuration flag. You can utilize core Content Server objects and functionality by writing custom IdocScript functions and referencing them from inside your template.

Because this IdocScript is evaluated once during publishing, you cannot use IdocScript as you would normally do from the `<install_dir>/shared/config/resources/std_page.htm` file. When a user requests that file, it has already been created, so the script used to create it did not have any access to the current service's DataBinder or any information about the current user.

This does limit the type of IdocScript you can write in these files, so if you are writing CSS or JavaScript that needs information that dynamically changes with users or services, consider having the pages that need this code include the code inline. This increases the size of pages delivered by your web server and thus increases the amount of bandwidth used.

TABLES USED IN CREATELAYOUT

This section describes the tables used in the CreateLayout component, which provide new layouts demonstrating how this new method of updating the interface works.

In the CreateLayouts component, the `createlayout_resource.htm` file in `<install_dir>/custom/CreateLayout/resources/` contains the tables needed to add the NewTopMenus and NewTrays sample layouts.

This section describes the following information:

- ❖ [CreateLayout_Layouts Table](#) (page 4-3)
- ❖ [CreateLayout_PublishedWeblayoutFiles Table](#) (page 4-4)
- ❖ [CreateLayout_PublishedStaticFiles Table](#) (page 4-5)
- ❖ [Additional File Entries](#) (page 4-6)

CreateLayout_Layouts Table

The CreateLayout_Layouts table is merged in the LmLayouts table in the `std_resources.htm` file in the `<install_dir>/shared/config/resources` directory.

The new table defines two new layouts, NewTopMenus and NewTrays. These are based on existing layouts. This table has the following structure:

```
<@table CreateLayout_Layouts@>
<table border=1><caption><strong>
```

```

        <tr>
            <td>id</td>
            <td>label</td>
            <td>enabled</td>
        </tr>
        ...
    </table>
<@end@>

```

The columns are as follows:

- ❖ **id**: The identifier for the new layout.
- ❖ **label**: The text for the User Profile page, which describes the new layout.
- ❖ **enabled**: If set to 1, the layout becomes enabled and is available for users to select on the User Profile page. If set to 0, the layout is disabled.

CreateLayout_PublishedWeblayoutFiles Table

The CreateLayout_PublishedWebLayoutFiles table is merged into the PublishedWeblayoutFiles table in the std_resources.htm file, stored in the <install_dir>/shared/config/resources directory.

That table contains a list of published files and has the following structure:

```

<@table PublishedWeblayoutFiles@>
<table border=1><caption><strong>
    <tr>
        <td>path</td>
        <td>template</td>
        <td>class</td>
        <td>loadOrder</td>
        <td>doPublishScript</td>
    </tr>
    ...
</table>
<@end@>

```

The columns are as follows:

- ❖ **path**: A relative path inside the /weblayout directory where this file will be published.
- ❖ **template**: The Content Server template evaluated to produce this file.
- ❖ **class**: A colon-separated phrase to describe this file. This is usually the type of the file followed by a description of its functionality or use.

- ❖ `loadOrder`: The order in which these files will be created by the Content Server and, in the case of JavaScript and CSS, included on Content Server pages. Files with a lower `loadOrder` are created and loaded first.
- ❖ `doPublishScript`: IdocScript that, when evaluated, determines if this file is published. After the script is evaluated, if `doPublish` is set to true the file is published to the `/weblayout` directory. Set `<$doPublish = 1$>` for files that are always published.

This script can also be used for a simple check for an environment variable, an IdocScript function call, or for complex calls such as the following:

```
<$include determine_should_publish_web_resource$>
```

This calls a custom resource include that will eventually set the `doPublish` IdocScript variable. `<$fileTemplate$>`, `<$fileClass$>`, and `<$fileLocation$>` are all available from within this script.

In the `CreateLayout` sample, three entries in the `CreateLayout_PublishedWeblayoutFiles` table are used to push out the necessary JavaScript and CSS files for the `NewTopMenus` layout.

A new table entry is created for the `NewTrays` layout. Like the one created for the `NewTopMenus` layout, it is merged into the `PublishedWeblayoutFile` table in the `std_resources.htm` file. Because this is a `Trays`-based layout, more table entries are needed to define the different navigational aspects.

Publishing these entries can be turned off by changing the `Enabled` settings in the [CreateLayout_Layouts Table](#) (page 4-3) table to 0.

Another way to disable the publishing is to create a new configuration variable for use in the `<install_dir>/config/config.cfg` file on the Content Server instance. For example, a variable named `DisableNewTopMenusPublishing`, if set to `TRUE`, would disable the publishing of that particular layout.

To add more skins for use with layouts, specify additional `skin.css` files to push out to each skin directory for each layout.

CreateLayout_PublishedStaticFiles Table

The `CreateLayout_PublishedStaticFiles` table is merged with the `PublishedStaticFiles` table in the `std_resource.htm` file. This table lists the directories that should be published to the `/weblayout` directory in order to use the new layouts.

The rows in this table describe the files that need to be published for one skin for the `NewTopMenus` and `NewTrays` layouts. Additional entries can be added for new skins. The

files being published come from the `<install_dir>/shared/publish` directory. If the Content Server is updated, the new layouts are automatically updated, too.

The final row in the `CreateLayout_PublishedStaticFiles` table provides a way to override files already stored in the `NewTrays/Stellent05` directory. Because the `loadOrder` entry on this row is greater than the `loadOrder` on the previous rows, files published by this entry override any files published by the previous rows.

A row like this can also be used to add new files to the directory. If you put a file into the `<install_dir>/custom/CreateLayout/resources/images/NewTrays/Stellent05` directory it is pushed out to the `NewTrays/Stellent05` layout in the `/weblayout` directory. This makes it easy to update the customized layout.

Additional File Entries

The `custom_trays_stellent_five_skin_variable_overrides` resource include can be used to customize `IdocScript` variables used with any `Trays Stellent05` skin. It is included near the top of the `TRAYS_STELLENT05_SKIN` template file and can be used to modify often-used variables. The example in the `CreateLayout` component modifies the color of the menu bar.

The `custom_global_skin_definition` resource include adds a global CSS style declaration to every `skin.css` file published by the Content Server. The example in the `CreateLayout` component added a new style, “`bigBoldAndRed`” to every layout.

CREATING NEW LAYOUTS

This section describes the general steps needed to create and publish new layouts.

1. Merge a table into the `LmLayouts` table in `<install_dir>/shared/config/resources/std_resources.htm` to define the new layout. Define the layout ID, label, and whether it is enabled (set to 1) or not.
2. Merge a table into the `PublishedWeblayoutFiles` table in `<install_dir>/shared/config/resources/std_resources.htm`. This new table describes the files that will be created from Content Server templates and then pushed out to the `/weblayout` directory. Specify the necessary `skin.css` files to push out to each skin directory.
3. Merge a table with the `PublishStaticFiles` table in `std_resources.htm`. This lists the directories that contain files, such as images, that should be published to the `/weblayout` directory.



OPTIMIZING PUBLISHED FILE USE

OVERVIEW

You can direct the Content Server to ‘bundle’ published files together so they can be delivered as one, thus minimizing the number of page requests to the server. In addition, you can optimize file use by referencing published pages using IdocScript.

This section discusses the following topics:

- ❖ [Bundling Files](#) (page A-1)
- ❖ [Referencing Published Files](#) (page A-4)

BUNDLING FILES

Static weblayout file contents are cached on client machines and on web proxies, significantly lowering the amount of server bandwidth they use. Therefore, best practice indicates that these types of files should be used wherever possible.

However, each static weblayout file requested by the client’s browser requires a round-trip to the server just to verify that the client has the most up-to-date version of this file. This occurs even if the file is cached. Therefore, as the number of these files grows, so does the number of pings to the server for each page request.

To help minimize the number of round-trips, the Content Server can bundle multiple published files together so they are delivered as one. This feature can be disabled by setting the following configuration in the server’s `<install_dir>/config/config.cfg` file:

```
BundlePublishedWeblayoutFiles=false
```

Bundling is accomplished using the `PublishedWeblayoutBundles` table in the `std_resources.htm` file.

```
<@table PublishedWeblayoutBundles@>
<table border=1><caption><strong>
  <tr><td>class</td><td>bundlePath</td><td>loadOrder</td></tr>
  <tr>
    <td>javascript:common</td>
    <td>resources/layouts/commonBundle.js</td>
    <td>10</td>
  </tr>
  ...
</table>
<@end@>
```

The columns in this table are as follows:

- ❖ `class`: This refers to the same column in the `PublishedWeblayoutFiles` table and is used to determine which files are placed in which bundle.
- ❖ `bundlePath`: The eventual location where the bundle will be published. This path is relative to the `/weblayout` directory.
- ❖ `loadOrder`: The order in which this bundle should be loaded on Content Server pages. Bundles with a lower `loadOrder` are loaded first.

In the previous example, files of the `javascript:common` class are published to a single bundle located at `resources/layouts/commonBundle.js`. The contents of all bundled files that match this class are appended together to form a single file to be stored at that location.

The `class` column in the `PublishedWeblayoutFiles` and `PublishedWeblayoutBundles` tables is a colon-separated classification. In the following example, two different bundles overlap. `food` accounts for all three published weblayout files, while `food:fruit` accounts for two of the three and `food:vegetable` accounts for the third.

Any given weblayout file can only be published into a single bundle, so `food:fruit` will contain both `APPLE` and `PEAR`, while `food` picks up the leftover `CARROT`. The server checks each file to be published then looks for the most specific bundle in which to place it. If no bundle exists, it is published as a single file.

The order in which files are included in a bundle is determined through the `loadOrder` column in the `PublishedWeblayoutFiles` table. Files with a lower `loadOrder` are placed earlier in the bundle.

```
<@table PublishedWeblayoutFiles@>
<table border=1><caption><strong>
```

```

<tr>
  <td>path</td>
  <td>template</td>
  <td>class</td>
  <td>loadOrder</td>
  <td>doPublishScript</td>
</tr>
<tr>
  <td>resources/apple</td>
  <td>APPLE</td>
  <td>food:fruit:apple</td>
  <td>10</td>
  <td><$doPublish = 1$></td>
</tr>
<tr>
  <td>resources/pear</td>
  <td>PEAR</td>
  <td>food:fruit:pear</td>
  <td>20</td>
  <td><$doPublish = 1$></td>
</tr>
<tr>
  <td>resources/carrot</td>
  <td>CARROT</td>
  <td>food:vegetable:carrot</td>
  <td>10</td><td><$doPublish = 1$></td>
</tr>
</table>
<@end@>

<@table PublishedWeblayoutBundles@>
<table border=1><caption><strong>
  <tr><td>class</td><td>bundlePath</td><td>loadOrder</td></tr>
  <tr>
    <td>food:fruit</td>
    <td>resources/fruit</td>
    <td>20</td>
  </tr>
  <tr>
    <td>food</td>
    <td>resources/food</td>
    <td>10</td>
  </tr>
</table>
<@end@>

```

REFERENCING PUBLISHED FILES

Most published files (both bundled and unbundled) need to be directly referenced from within HTML to be included in a page. It can therefore be difficult to know exactly which files to include for a given situation, especially when bundling can be enabled or disabled by server administrators. A simple IdocScript method can be used to easily and transparently include all of the files you need on a given page.

For example, if you write a page that includes all files associated with the `javascript:common` bundle (as described previously) then do not write HTML that includes all of the files mentioned in the first table in addition to the bundle mentioned in the second, the server is asked for each file. This negates the purpose of bundling because the server is pinged for each file whether it actually exists or not.

To correctly include these files on a page, use the following IdocScript and include it from somewhere within the HEAD of the page:

```
<$exec createPublishedResourcesList("javascript:common")$>
  <$loop PublishedResources$>
    <script language="JavaScript"
      src="<$HttpWebRoot$><$PublishedResources.path$>" />
    </script>
  <$endloop$>
```

This code fragment includes all `javascript:common` files even if bundling is switched off. If `javascript` instead of `javascript:common` is passed, all files whose class starts with `javascript` are included.

This `PublishedResources` result set is sorted by `loadOrder` so files and bundles with the lowest `loadOrder` are included first. Files with a greater `loadOrder` can override JavaScript methods or CSS styles that were declared earlier.

THIRD PARTY LICENSES

OVERVIEW

This appendix includes a description of the Third Party Licenses for all the third party products included with this product.

- ❖ [Apache Software License](#) (page B-1)
- ❖ [W3C® Software Notice and License](#) (page B-2)
- ❖ [Zlib License](#) (page B-3)
- ❖ [General BSD License](#) (page B-4)
- ❖ [General MIT License](#) (page B-5)
- ❖ [Unicode License](#) (page B-5)
- ❖ [Miscellaneous Attributions](#) (page B-7)

APACHE SOFTWARE LICENSE

```
* Copyright 1999-2004 The Apache Software Foundation.  
* Licensed under the Apache License, Version 2.0 (the "License");  
* you may not use this file except in compliance with the License.  
* You may obtain a copy of the License at  
*   http://www.apache.org/licenses/LICENSE-2.0  
*
```

- * Unless required by applicable law or agreed to in writing, software
- * distributed under the License is distributed on an "AS IS" BASIS,
- * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
- * See the License for the specific language governing permissions and
- * limitations under the License.

W3C® SOFTWARE NOTICE AND LICENSE

- * Copyright © 1994-2000 World Wide Web Consortium,
- * (Massachusetts Institute of Technology, Institut National de
- * Recherche en Informatique et en Automatique, Keio University).
- * All Rights Reserved. <http://www.w3.org/Consortium/Legal/>
- *
- * This W3C work (including software, documents, or other related items) is
- * being provided by the copyright holders under the following license. By
- * obtaining, using and/or copying this work, you (the licensee) agree that
- * you have read, understood, and will comply with the following terms and
- * conditions:
- *
- * Permission to use, copy, modify, and distribute this software and its
- * documentation, with or without modification, for any purpose and without
- * fee or royalty is hereby granted, provided that you include the following
- * on ALL copies of the software and documentation or portions thereof,
- * including modifications, that you make:
- *
- * 1. The full text of this NOTICE in a location viewable to users of the
- * redistributed or derivative work.
- *
- * 2. Any pre-existing intellectual property disclaimers, notices, or terms
- * and conditions. If none exist, a short notice of the following form
- * (hypertext is preferred, text is permitted) should be used within the
- * body of any redistributed or derivative code: "Copyright ©
- * [\$date-of-software] World Wide Web Consortium, (Massachusetts

* Institute of Technology, Institut National de Recherche en
* Informatique et en Automatique, Keio University). All Rights
* Reserved. <http://www.w3.org/Consortium/Legal/>
*
* 3. Notice of any changes or modifications to the W3C files, including the
* date changes were made. (We recommend you provide URIs to the location
* from which the code is derived.)
*
* THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS
* MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
* NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR
* PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE
* ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.
*
* COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR
* CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR
* DOCUMENTATION.
*
* The name and trademarks of copyright holders may NOT be used in advertising
* or publicity pertaining to the software without specific, written prior
* permission. Title to copyright in this software and any associated
* documentation will at all times remain with copyright holders.
*

ZLIB LICENSE

* zlib.h -- interface of the 'zlib' general purpose compression library
version 1.2.3, July 18th, 2005

Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly jloup@gzip.org

Mark Adler madler@alumni.caltech.edu

GENERAL BSD LICENSE

Copyright (c) 1998, Regents of the University of California

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

"Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

"Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

"Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.



A

anonymous user interface, 2-3

B

bundled files

 example, A-2

 order of, A-2

BundlePublishedWebLayoutFiles, A-2

bundling files, A-1

C

class column, A-2

component installation

 publishing files, 3-3

 schema publishing, 3-3

component manager, 3-2

component wizard, 3-2

customizing resource files, 4-3

E

extranet pages, 2-3

ExtranetLook component, 2-3

F

file usage, A-4

files

 ExtranetLook, 2-3

I

installation

 component manager, 3-2

 component wizard, 3-2

IsWebServerPagesOnly variable, 2-3

L

Layouts, 2-2

 Classic, 2-2

 Top Menu, 2-2

 Trays, 2-2

M

My Profile, 2-2

O

Overview, 1-2

 Audience, 1-1

 Conventions, 1-2

P

published files, 4-3, 4-4

PublishedResources, A-4

PublishedWebLayoutFiles table, 4-3, 4-4

publishing resource files, 4-2

R

resource files

 customization, 4-3

S

Skins, 2-2

T

Types of skins and layouts, 2-1

U

uninstalling components, 3-3

W

web resource publishing, 4-2