

Caching and pre-rendering considerations

This chapter discusses performance considerations in IBM Workplace Web Content Management through the use of pre-rendering and caching strategies. It includes the following sections:

- ▶ Improving performance
- ▶ Caching options
- ▶ Pre-rendering options
- ▶ Implementing a caching or pre-rendering strategy
- ▶ Setting up caching
- ▶ Configuring data caching
- ▶ Configuring pre-rendering
- ▶ Custom caching with connect tags
- ▶ Considerations in developing a caching strategy
- ▶ Documenting the cache strategy
- ▶ WebSphere Dynamic Cache Service

8.1 Improving performance

A typical IBM Workplace Web Content Management implementation includes dynamically generated pages that often combine navigation elements with files (such as stylesheets or JavaScript™ scripts), images, and site content. Each of these components can have special security settings. As well, menus can be designed to show different content based on a user's profile.

While this flexibility is one of the many strength's of IBM Workplace Web Content Management, it can also lead to reduced performance. Consequently, it is important to analyze performance carefully and to implement tuning properly in any IBM Workplace Web Content Management implementation.

Because IBM Workplace Web Content Management runs as part of a WebSphere Portal environment, its performance is dependent on many settings that are outside of the IBM Workplace Web Content Management configuration. For example, IBM Workplace Web Content Management performance depends on hardware, Web server caching, WebSphere Application Server settings, and WebSphere Portal Server settings.

IBM Workplace Web Content Management offers caching and pre-rendering as solutions for improving performance within the IBM Workplace Web Content Management framework. Additionally, you can use *dynamic cache* to improve performance. By default, dynamic cache (via WebSphere Dynamic Cache Service) is enabled on an application server. It supports caching of servlet and JSP responses, WebSphere commands objects, Web services objects, and Java objects.

When an IBM Workplace Web Content Management server is configured to cache or pre-render content, it stores partial or complete copies of dynamically generated pages. By leveraging this stored content, the IBM Workplace Web Content Management server can deliver pages more quickly, because it can omit the page assembling process partially or completely.

Caching and page rendering differ in that *cached content* is added to the cache on an *as needed basis* (that is, when it is requested by a user and when it is not in the cache). Cached content is removed from the cache when it expires or when the cache is full and more recent content needs to be cached. In contrast, *page rendering* occurs as a *batch operation*. All of the content for a site is pre-rendered at roughly the same time.

You can configure both caching and pre-rendering in such a way that allows for dynamic components within a page (for example, to accommodate components with security or profile settings or connect tags that show live data).

8.1.1 Caching options

IBM Workplace Web Content Management offers many different caching options. However, all caching in IBM Workplace Web Content Management shares the following basic characteristics:

- ▶ Web pages are cached after a user requests the Web page. This strategy has two basic effects:
 - Generally, when a Web page that has not been cached is requested, it displays more slowly than a Web page that is delivered from the cache.
 - Cached pages can have broken links. For example, it is possible for a page with a menu to show links to content that has been expired or removed recently from the site.
- ▶ Cached Web pages expire and are removed automatically from the cache based on cache expiration settings.

IBM Workplace Web Content Management supports default expiration settings for caching throughout the site. In addition, IBM Workplace Web Content Management Advanced caching supports custom expiration settings for individual components in the site. You can adjust expiration settings to reduce the likelihood of broken links due to *stale* cached pages.

Basic caching

Basic caching is the simplest caching strategy that is available in IBM Workplace Web Content Management. Basic caching creates a single cached instance of each content object for all users — whether known or anonymous. Basic caching has the following characteristics:

- ▶ It cannot be used in sites that contain placeholders for dynamic content such as connect tags.
- ▶ It only supports sites where all content (including Library Components such as navigators and menus) are displayed identically for all site visitors.
- ▶ It does not allow connect tags (whether to IBM Workplace Web Content Management Library Components, Web pages, or database queries) to be configured to implement any other caching strategy.

Basic caching can be a successful strategy on a site where all content is static and where all content is displayed the same way for all users. Sites that are more complex should not employ basic caching.

Advanced caching

Advanced caching allows for far more flexibility than basic caching. However, this flexibility comes at a price. Configuring a site to use advanced caching can be more complicated and more time-consuming than configuring a site to use basic

caching. Also, advanced caching requires more disk space and more memory, and it can result in slower performance than basic caching. Advanced caching has the following characteristics:

- ▶ It supports dynamic content (such as content brought in with connect tags).
- ▶ It supports configuring dynamic content to use a different caching strategy from the default caching strategy. For example, if the content is the same for all users in a site, a site could be configured to use one site cache for all users. Then, any special menus that are drawn based on the user's profile could be coded to use special caching.
- ▶ It supports caching content in several different ways:
 - *Site caching*, which works much like basic caching, creates a single cache for all site visitors. Unlike basic caching, site caching supports adding caching parameters to connect tags so that parts of the site (such as menus) can be cached differently from most of the site.
 - *Session caching* creates a copy of each Web page in the session cache for each user's session. A cached page is created as the user visits the page. After the page is first cached, the user sees the cached page until starting a new session or until the page expires from the cache (whichever comes first).
 - *User caching* creates a copy of each Web page in the user cache for each user. Like session caching, a Web page is cached when the user visits a page and it expires. Unlike session caching, the Web page in the user cache lives beyond the current session.
 - *Secured caching* is used in sites that control access to Web pages and components based on the user's group memberships.
 - *Personalization caching* is used to share a cache among all users with the same *personalization profile*. Users with the same profile (categories and keywords) and group membership share a single cache.

Caching paradigms

IBM Workplace Web Content Management supports the following caching paradigms:

- ▶ *Default caching* is configured at the site level through the IBM Workplace Web Content Management configuration file (connect.cfg). It defines how any content that is not otherwise specified should be cached. In general, basic caching, site caching, and session caching are only used as your server's default Web content cache.
- ▶ *Custom caching* is configured in connect tags and URL requests. Custom caching overrides the IBM Workplace Web Content Management server's default caching strategy. Custom caching is possible only when no cache or

advanced caching (generally, site or session caching) is used as the site's default caching strategy. You can use custom caching to cache parts of a page. For example, a connect tag that shows a navigator might employ secured caching if links vary depending on a user's access. Similarly, a connect tag showing a menu might employ personalized caching if links vary depending on a user's profile.

Choosing between caching strategies

In general, basic caching is preferred over advanced caching for sites that show the same static content for all users. Advanced caching is preferred over basic caching when different users should see a site differently, when a site includes dynamic content, or when certain parts of a site need to be cached differently from most of the content in the site.

Note: Keep in mind that advanced caching requires more memory and disk space than basic caching. Also, basic caching can have much faster performance than advanced caching.

8.1.2 Pre-rendering options

IBM Workplace Web Content Management has two ways to deliver pre-rendered content: via an HTTP server or via IBM Workplace Web Content Management. In both cases, a site is pre-rendered as part of a regularly scheduled batch process. As such, pre-rendered sites are far less likely to show broken links (because all pages are rendered at roughly the same time). As well, pre-rendered sites have faster performance than cached sites on pages that have not yet been added to the cache.

Pre-rendering via an HTTP server

A site that is pre-rendered via an HTTP server is very similar to a site with basic caching, in that the content must be static and custom caching cannot be used. However, there are some significant differences between pre-rendering via an HTTP server and basic caching, as follows:

- ▶ Because all pages in a site are pre-rendered, pre-rendering generally requires more disk space than basic caching (which maintains copies of recently accessed pages).
- ▶ Because pre-rendering creates a copy of every page at roughly the same time, pre-rendering is less likely to result in broken links than basic caching (which caches pages as needed). As well, it is likely to result in higher performance, particularly when accessing less popular pages.

Table 8-1 lists the differences between a site that is pre-rendered via an HTTP server and site that is cached using basic caching.

Table 8-1 Pre-rendering via an HTTP server versus basic caching

	Basic caching	Pre-rendered site delivered via an HTTP server
Performance	Very fast	Extremely fast
Connect tag processing	No	No
Custom Caching	No	No
Memory Requirements	Low to Medium	Memory requirements depends on the Web server that is used.
Disk Requirements	Low to Medium	Potentially very high as the entire site must be able to fit on disk.
Unexpected Broken Links	Yes Because some pages might be cached at different times, there is a small chance that not all the links on a cached page are valid currently.	No The site is pre-rendered in a single batch, which greatly reduces the chances of inconsistencies in the site.

Pre-rendering via IBM Workplace Web Content Management

A site that is pre-rendered via IBM Workplace Web Content Management is very similar to a site with advanced caching, in that content can be dynamic. In addition, different caches can be used by different users and different caching strategies can be used for different site components.

Because pre-rendered pages are generated all at once — while cached pages are generated on an as-needed basis — pre-rendered pages are less likely to have broken links than cached pages. Web pages that are not in the cache display more slowly than pre-rendered pages.

Note: Because these options are very similar, it is generally necessary to test both strategies before deciding which is better for a site.

Table 8-2 lists the differences between a site that is pre-rendered via IBM Workplace Web Content Management and site that is cached using advanced caching.

Table 8-2 IBM Workplace Web Content Management versus advanced caching

	Advanced Caching	Pre-rendered site delivered via IBM Workplace Web Content Management
Performance	Fast when cached, but slower if the requested page has expired from the cache. (Because tag processing has a cost, this depends on how many connect tags a page contains.)	Fast, but because tag processing has a cost, this depends on how many connect tags a page contains.
Connect tag processing	Yes	Yes
Custom Caching	Yes	Yes
Memory Requirements	Medium to high	Medium to high
Disk Requirements	Medium to high	Medium to high
Unexpected Broken Links	Yes Because some pages might be cached at different times, there is a small chance that not all the links on a cached page are valid currently.	No The site is pre-rendered in a single batch, which greatly reduces the chances of inconsistencies in the site.

8.2 Implementing a caching or pre-rendering strategy

It is possible to develop a successful IBM Workplace Web Content Management site without implementing any caching or pre-rendering strategy. Nevertheless, it is important to test the performance of a site before rolling it to production in order to ensure that its production performance will be acceptable. As well, it is important to test a site's performance regularly, because changes to the site's design and user base can affect performance and can require changes to the caching or pre-rendering strategy.

The following steps are typically part of the successful implementation of a caching or pre-rendering strategy:

1. Develop a site.

Set up the site without implementing any caching. Ensure that it has a realistic representation of the content the site will provide, including any dynamic or personalized content.

2. Configure the environment.

Ensure that the new site has IBM-recommended hardware as well as IBM-recommended parameters for the following:

- WebSphere Application Server and WebSphere Portal Server tuning parameters (for IBM Workplace Web Content Management and WebSphere Portal)
- Network properties (for WebSphere Portal)
- Database parameters (for IBM Workplace Web Content Management and WebSphere Portal)
- HTTP Server tuning and caching parameters (for WebSphere Portal)
- Operating System settings (for IBM Workplace Web Content Management and WebSphere Portal)
- WebSphere Portal Service properties (for WebSphere Portal)
- If applicable, LDAP Server parameters (for IBM Workplace Web Content Management and WebSphere Portal)

3. Test performance for the site.

In this step, you want to test a site that is configured properly for production for two important reasons:

- To establish a baseline of performance.
- To determine whether it is necessary to enable any caching strategy.

It is highly preferable that you use an automated testing strategy so that stress-testing is measured properly. In addition, we recommend that you test ranges of suggested values in a few different ways to find the optimal settings for the environment.

Note: If testing reveals that the performance is acceptable, you might want to skip several steps and move right to rolling the site to production. Alternatively, you can continue with the following steps, further improving performance.

4. Analyze the site's content.

Is the content static, is it dynamic (that is, does it contain connect tags that show HTML from other Web pages, SQL result sets, or JSP components), or is it both? Are menus drawn differently based on the user's profile? Is content displayed and hidden using security?

5. Design a caching strategy.

If the performance test revealed that a caching strategy is necessary, the analysis of the site should suggest which of the four strategies would be most appropriate.

Note: Sometimes (particularly when choosing between advanced caching and pre-rendering via Web content management), it is necessary to implement and stress-test each strategy before making a final choice.

6. Implement caching in a development.

It can take quite a bit of time to implement a caching strategy. In addition, implementing a caching strategy can sometimes have unexpected results. It is important to first implement caching in a development environment to avoid disrupting your production environment. It is recommended that your development environment closely mirrors your production environment so that testing results are meaningful.

7. Test the cached site.

Perform functional testing (ensure that content is cached correctly). You should also perform stress testing to ensure that performance is acceptable.

8. Make adjustments as necessary.

Implementing a caching strategy often requires an iterative approach. So, you might need to perform steps 5, 6, and 7 until test results are satisfactory.

9. Implement caching strategy.

Implement all changes in the production environment.

10. Expect to revisit the caching strategy.

Healthy sites change over time. It often happens that characteristics of a pilot site (such as the absence of dynamic content) changes over time. Because sites change, it is important to keep in mind that any caching strategy might need to be updated over time as a site's content and design evolve.

Developing and implementing a caching or pre-rendering strategy requires an iterative process of designing, developing, and testing. While developing a strategy can be time-consuming, the work is rewarded with improved performance.

However, as you develop a caching or pre-rendering strategy, do not lose sight of the larger picture. Significant performance gains on IBM Workplace Web Content Management sites can often be found by using adequate hardware as well as by tuning other parts of the environment, including Web server caching, WebSphere Application Server settings, and WebSphere Portal Server settings. Configuring

the environment correctly, as discussed in step 2 on page 354, is critical to the success of an IBM Workplace Web Content Management site.

8.3 Setting up caching

Setting up IBM Workplace Web Content Management cache requires changes to the server's connect.cfg configuration file. The section provides a guide to the properties and attributes that you need to implement a cache strategy.

8.3.1 Understanding expiration parameters

All cached items have an expiration time that indicates when a cached item is eligible to be removed. An item is removed when the expiration time is reached, when the cache exceeds its memory capacity, or when a new request is made for the expired item. Expiration parameters are used to configure the expiration time. You can configure an item's expiration time to be either *relative* or *absolute*. A relative expiration time is calculated each time the cache is created. Relative expiration parameters are the most flexible and the most commonly used. Absolute expire times create a cache entry that expires at a specific time. When the absolute expiration time has passed, each time the cache is rebuilt, it expires immediately. Improper use of absolute parameters can cause unnecessary overhead, especially if the item is requested frequently.

To configure cache expiration, an absolute date and time or a relative period is set in the default cache configuration, as URL parameters and in connect tag attributes. Expiration parameters are not required on HURLs or in connect tags. If the expiration parameter is not included, the default cache configuration expiration time is used. These properties use a specific syntax for their values. Table 8-3 defines the syntax for the absolute and relative expiration settings.

Table 8-3 Syntax for absolute and relative expiration settings

Expiration type	Proper format and examples
Absolute	ABS {date and time} Date and time must conform to one of the following formats: Mon., 06 Nov. 2000 09:00:00 GMT Monday, 06-Nov.-00 09:00:00 GMT Mon. Nov. 6 09:00:00 2000 6 Nov. 2000 9:00 AM For example: ABS Mon., 29 May 2000 03:04:18 GMT

Expiration type	Proper format and examples
Relative	REL {integer}{units} Integer indicates whole number units of time. Units are represented by a single case-insensitive letter, as follows: - S: Seconds - H: Hours - D: Days - M: Months Examples: ► In connect tag: – REL 9000s – REL 7D ► In a URL: – REL+9000s – REL+7D

Tip: In general, using relative time periods is preferred over absolute dates. When an absolute date has past, any future requests for content build a cache that expires immediately after it is rendered.

For more information about the cache expiration setting, see the IBM Workplace Web Content Management Information Center at:

<http://www-128.ibm.com/developerworks/workplace/documentation/webcontentmanagement/>

8.3.2 Configuring the default cache

The first step in implementing a caching strategy is to configure the default cache. The cache configuration is divided into an IBM Workplace Web Content Management cache and a data cache. The IBM Workplace Web Content Management cache is the default cache for content that is stored within an IBM Workplace Web Content Management repository. External content, HTML, or SQL data that is referenced using connect tags is cached using the data cache. This section describes how to configure the IBM Workplace Web Content Management cache. You can find details about how to configure the data cache in 8.4, “Configuring data caching” on page 362.

Default cache is configured in the connect.cfg configuration file. The file is located in <WPS Root>/wcm/config directory.

- ▶ On Windows, you can find the file here:

C:\Program Files\WebSphere\PortalServer\wcm\config\connect.cfg

- ▶ On Linux®, you can find the file here:

/opt/WebSphere/PortalServer/wcm/config/connect.cfg

Before altering the connect.cfg file, decide if you will use basic caching, advanced caching, or pre-rendering. The properties that you modify for basic caching, advanced caching, and pre-rendering are quite different. Thus, it is important to know which cache you are implementing.

The caching properties are located throughout the connect.cfg file. Some properties are unique to a particular cache type. Other properties are repeated throughout the configuration but provide the same function. The following lists and describes those properties:

- ▶ **FlushExpiredInterval:** Controls how often the cache flushes expired items from disk and memory. Specify an integer to indicate the number of seconds between cache flushings.
- ▶ **WriteIndexInterval:** Controls how often the cache index is written to disk. Specify an integer to indicate the number of seconds. If 0 is specified, all changes are written to disk.
- ▶ **CacheDir:** The directory on the server where cache is stored.
- ▶ **MemCacheSize:** Indicates the number of items that are stored in the memory cache. If set to 0, the cache is not created.
- ▶ **DiskCacheSize:** Indicates the number of items that are stored in the disk cache. If set to 0, the cache is not created.

Note: The connect.cfg file is formatted in a syntax that is similar to XML. To help clarify which properties are being changed, the property names in the following sections are formatted to reflect the complete path to the property. For example, the property `DefaultCache` is found in the `BusinessLogic` section. The path to the value is `/BusinessLogic/DefaultCache`.

8.3.3 Understanding cache size

Items are stored normally in the cache until they expire. If the cache size limit is reached, then the cache removes items using a *first-in, first-out* rule. Older items are removed first to make room for new items. If the cache size is too small, the site performance can degrade due the constant swapping. Ideally, the memory and disk cache should be large enough to allow cache to expire normally. Be aware that adjusting the cache size affects how much memory and

storage resources are needed to run IBM Workplace Web Content Management. Consult with the system's administrator before changing these values.

8.3.4 Configuring basic caching

The following properties are used to configure basic caching:

- ▶ `/BusinessLogic/DefaultCache`: Enables basic caching as the default cache. To enable basic caching set value to `true`.
- ▶ `/BusinessLogic/DefaultCacheExpires`: Sets the default cache expiration. Items remain in cache until the expiration time has past or until the cache exceeds its maximum size. Set the value using the expiration syntax as discussed in 8.3.1, "Understanding expiration parameters" on page 356.

Example 8-1 shows a basic cache configuration in the `connect.cfg` file.

Example 8-1 Basic cache configuration in the connect.cfg file

```
...
<BusinessLogic>
  ...
  <DefaultCacheExpires value="REL 3D" />
  <DefaultCache value=true />

  <ModuleReponseCacheConfig>
    <FlushExpiredInterval value=120 />
    <WriteIndexInterval value=90 />
    <CacheDir value="../connect/module/" />
    <MemCacheSize value=1000 />
    <DiskCacheSize value=5000 />
  </ModuleReponseCacheConfig>
  ...
```

These settings enable IBM Workplace Web Content Management to use basic caching as the default for managed content. Not all properties need to be adjusted to enable basic caching. The `ModuleResponseCacheConfig` properties remain their default values. Usually, the default values are sufficient to begin testing your caching strategy.

Important: When using basic caching, JSP components do not function.

8.3.5 Configuring advanced cache

Configuring advanced caching requires more configuration than basic caching. Advanced caching permits default settings to be stored in the `connect.cfg` file and then to be overridden by including expiration and cache type attributes on connect tags. Selecting the advanced cache type is the first step in the

configuration. The typical choices are *site* or *session* cache types. Review “Advanced caching” on page 349 before selecting a default cache type.

The following properties are used to configure advance caching as the default cache:

- ▶ `/ModuleConfig/AJPE/ContentCache/DefaultContentCache`: Sets the default advance cache type for the site. Typical values are `site` or `session`.
- ▶ `/ModuleConfig/AJPE/ContentCache/ContentCacheExpires`: Sets the default expiry for all advance cache. Set the value using the expiration syntax as discussed in 8.3.1, “Understanding expiration parameters” on page 356.
- ▶ `/SessionCacheConfig/MemCacheSize`: Used only for `session` cache. The value indicates the number of items that are stored in cache for sessions.

Example 8-2 shows an advanced cache configuration in the `connect.cfg` file.

Example 8-2 Advanced cache configuration in the connect.cfg file

```
...
<ModuleConfig>
  <AJPE>
    ...
    <ContentCache>
      <DefaultContentCache value="Site" />
      <ContentCacheExpires value="REL 2D" />
      <FlushExpiredInterval value=120 />
      <WriteIndexInterval value=90 />
      <CacheDir value="../ilwcm/system/contentcache" />
      <MemCacheSize value=1000 />
      <DiskCacheSize value=5000 />
    </ContentCache>
  </AJPE>
  ...
</ModuleConfig>
...
<SessionCacheConfig>
  <MemCacheSize value=1000 />
</SessionCacheConfig>
...
```

Example 8-2 on page 360 configures IBM Workplace Web Content Management's default cache to use *advanced site caching*. In this example, the default cache sizes and cache directory remain. While testing the cache strategy, you might need to adjust the cache sizes to meet performance requirements.

Often when implementing advanced caching, there is a need to override the default cache. Connect tags can override the default cache by including cache

type and expiration attributes on the tag instance. Before using connect tags to override the default cache, there are some configuration steps.

First, it is a good practice to copy the default module property from the connect.cfg file and to give it a custom name for the connect tags to use. Using the default module works, but if you decide in the future to change the default module's implementation class, the connect tag entries might not function or might need to be re-written. In the connect.cfg file, locate the /BusinessLogic/Module/Default property. Copy it and rename it to /BusinessLogic/Module/ComponentRender.

Example 8-3 shows an example of using connect tags to override the default cache.

Example 8-3 Using connect tags to override the default cache to provide flexibility

```
...
<BusinessLogic>
...
  <Module>
...
    <Default class=com.aptrix.pluto.renderer.RendererModule remoteAccess=true
    autoLoad=false />
    <ComponentRender class=com.aptrix.pluto.renderer.RendererModule
    remoteAccess=true autoLoad=false />
...

```

Note: To verify the implementation class, refer to the IBM Workplace Web Content Management Information Center at:

<http://www-128.ibm.com/developerworks/workplace/documentation/webcontentmanagement/>

Depending on which version of IBM Workplace Web Content Management you are using, the implementation class name might begin with com.ibm.workplace instead of the existing Aptrix class names.

Second, by default IBM Workplace Web Content Management only allows a connect tag to be processed by known hosts. If IBM Workplace Web Content Management is accessed via an IP address or multiple host names, you need to update the configuration. There are two approaches to this issue. You can either disable this feature, or you can update the IBM Workplace Web Content Management configuration and list all applicable hosts and IP addresses. The excerpt shown in Figure 8-1 identifies the properties.

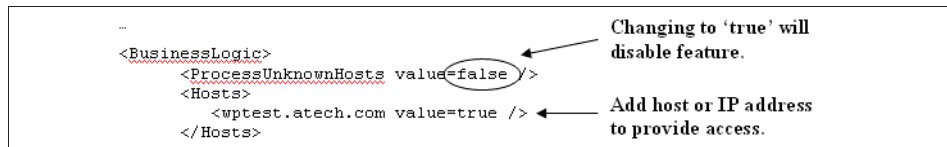


Figure 8-1 Configuring for unknown host

These settings enable advanced caching as the default cache and allow connect tags to provide custom caching. Using connect tags to override the default cache is discussed in 3.7.2, “Configuring the <Cacher> configuration options” on page 117.

8.4 Configuring data caching

An under used feature of IBM Workplace Web Content Management is the ability to include external content into a rendered page via connect tags. External content can be from a HTML file, database results, LDAP entries, or DB2 Content Manager federated content. The external content is inserted just like a component, but it is not retained in the IBM Workplace Web Content Management repository. The data cache is a separate cache from the Web content cache, and it allows external content to be cached on the IBM Workplace Web Content Management server. The external content is requested from its remote location on the first request and then stored in the data cache. Subsequent requests for the external content are retrieved from the cache until it expires. Data cache must be configured separately from default cache.

You can configure the data cache in the connect.cfg file or in connect tag attributes. You can configure only the cache for HTML and database results in the connect.cfg. You must specify cache parameters for all other external content types in connect tag attributes.

To enable the data cache in the connect.cfg file, locate the /Connector section. The /Connector/HttpConnector section has three properties that control HTML external data cache:

- ▶ /Connector/HttpConnector/DefaultCache: If set to true, HTML data is stored in the site cache if no cache type is specified in the connect tag or request.
- ▶ /Connector/HttpConnector/DefaultCacheExpires: The default expiration time for HTML data if no expiration is specified in the connect tag or request.
- ▶ /Connector/HttpConnector/OverrideCacheExpiryHeaders: If set to true, the content expiration and cache setting in the HTTP headers is ignored.

Example 8-4 shows an example configuration.

Example 8-4 Sample HTTP Connector settings

```
<Connector>
...
  <HttpConnector>
    ...
    <DefaultCacheExpires value="REL 1D">
    <DefaultCache value=true >
    <OverrideCacheExpiryHeaders value=false >
    ...
```

The `/Connector/SQLConnector` section has one property which control the external data cache. The `/Connector/SQLConnector/DefaultCache` property determines if the site cache should be used if no cache parameter is supplied on the connect tag or request. Example 8-5 shows an example configuration.

Example 8-5 Sample SQL Connection settings

```
<Connector>
...
  <SQLConnector>
    ...
    <DefaultCache value=true >
    ...
```

You do not need to configure the data cache in the `connect.cfg` file, but it is recommended. Because each request for external data is unique and can have its own update frequency, personalization, and security needs, you should optimize each instance according to the data's requirements. By including cache type and expiration parameters on a connect tag, you can customize each data request. For more information about connect tags, see 8.6, "Custom caching with connect tags" on page 368 and the IBM Workplace Web Content Management Information Center at:

<http://www-128.ibm.com/developerworks/workplace/documentation/webcontentmanagement/>

8.5 Configuring pre-rendering

Pre-rendering creates a snap-shot of the currently published site and can deliver the site via the IBM Workplace Web Content Management Servlet or an HTTP server. Pre-rendering is a batch process that generates the entire Web site each time it runs. Delivering a pre-rendered Web site via the IBM Workplace Web Content Management Servlet is enabled as the default module. To use an HTTP server, you need to configure pre-rendering as a stand-alone module. (Refer to 3.7,

“Static pre-rendering” on page 114 for more information about how to configure static pre-rendering.)

Some important factors to consider before enabling pre-rendering are:

- ▶ If configured as the default module, the local rendering portlet cannot be used.
- ▶ Connect tags are not processed. Instead content is rendered to the file system as though the connect tags did not exist. If using the IBM Workplace Web Content Management local rendering server, connect tags are processed when the page is invoked.
- ▶ JSP components do not function. Pre-rendering does not invoke JSP components.
- ▶ Sites are rendered using the security context of the `RenderUser` set in the configuration. A `RenderUser` tells the `cache` module to act as the specified user when processing content for pre-rendering. All content is pre-rendered as though the specified user was browsing the site. The user's security permissions and personalization profile effects which content is pre-rendered.
- ▶ Only live content that is accessible by the `RenderUser` is pre-rendered.
- ▶ File system path and file name conventions must be adhered to in the names of Sites, Site Areas, and Pages. Only ASCII characters are permitted. You cannot use the following characters: `/ \ : * ? " < > |`
- ▶ File system path and file names cannot exceed the operating system's maximum path length. For example, the Windows limit is 255 characters, while the Linux limit is 1024 characters.

These restrictions might not permit pre-rendering as viable caching solution for your Web site, but pre-rendering does offer extremely high performance gains. When configured as a stand-alone module, when the Web site is rendered to the file system, the HTML and resources can be distributed to other HTTP servers, removing even more load from IBM Workplace Web Content Management server.

8.5.1 Pre-rendering via an IBM Workplace Web Content Management local rendering server

The IBM Workplace Web Content Management local rendering server usually renders content as requested. When you enable pre-rendering, IBM Workplace Web Content Management saves the rendered content to the file system. The static content is then served from the local rendering server instead of building the content for each request.

To enable pre-rendering via the IBM Workplace Web Content Management local rendering server, you need to replace the default module's implementation class. In the connect.cfg file, locate the /BusinessLogic/Module/Default property and replace the class attribute value with com.aptrix.cacher.CacherModule, as shown in Example 8-6.

Example 8-6 Replacing the class attribute value with com.aptrix.cacher.CacherModule

```
<Default class=com.aptrix.cacher.CacherModule remoteAccess=true  
autoLoad=false />
```

Now that the default module uses the CacherModule to serve content, the content can be accessed on the local machine using the following URL:

http://localhost:9081/wps/wcm/connect/<The Site Name>

Important: When you use the CacherModule as the default module, you cannot use the local rendering portlet to access the content.

8.5.2 Pre-rendering via an HTTP Server

You can configure IBM Workplace Web Content Management to pre-render the contents of an entire site to static HTML files. When pre-rendered, any Web server can deliver the content. Pre-rendering via an HTTP server requires that you configure a stand-alone module to render the static files and that you set up a Web server to deliver the content.

Pre-rendering via HTTP is a little more complicated than the IBM Workplace Web Content Management local rendering server. In the connect.cfg file, uncomment the /BusinessLogic/Module/Cacher property and uncomment the /ModuleConfig/Cacher property block. Next, configure the cacher module to render the site to disk. Table 8-4 list the properties and their descriptions.

Table 8-4 Property descriptions for pre-rendering via HTTP

Property	Description
<DestDir value= "[ILWWCM_HOME]/ilwwcm/cacher" />	The base directory under which each site cache is created. One subdirectory is created for each site.
<TempDir value= "[ILWWCM_HOME]/ilwwcm/cacher/temp" />	The temporary DestDir that is required to build the site cache prior to moving the data over to the DestDir.
<Delay value="1" />	This property is used to set the time, in seconds, between requesting a page while caching.

Property	Description
<code><BusyDelay value="5" /></code>	This property is used to set the time, in seconds. It is used if executing within the BusyStart to BusyEnd period. Otherwise, the Delay setting is used.
<code><BusyStart value="9:00 am" /></code> <code><BusyEnd value="5:00 pm" /></code>	These settings determine the times between which the BusyDelay setting are used. Enter an absolute time.
<code><Password value="password" /></code>	When manually pre-rendering a site via a URL request, you are asked for a password. This password is set here. It is recommended that you change the default password of <i>password</i> to something more secure.
<code><OverWriteCache value="true" /></code>	If set to true, the pre-rendering overwrites files in the DestDir cache directory (then delete the unneeded files). This action results in a progressive change in site content that the user sees. If set to false, the pre-rendering renames the TempDir to the DestDir to achieve an instant update to a new version of the pre-rendered site. Do <i>not</i> use false should if you use an HTTP server to display the pre-rendered data because some HTTP Servers lock the data directories.
<code><RenderUser value="Anonymous" /></code>	This determines the user to be used to render the Web Content Management content. Either type "anonymous" or "administrator" or a specific Member Manager user or group name. The site is pre-rendered based on this User's security rights. If the user specified here does not have access to a particular component it will not be pre-rendered.
<code><CacherUrl value="http://[HOST]:[PORT]/wps/wcm/connect/" /></code>	The full URL to the CacherModule.
<code><ServletPath value="/connect" /></code>	The Servlet Path to the IBM Workplace Web Content Management servlet, not including the context path.
<code><DefaultContentName value="index.html" /></code>	This property sets the name of the default or home file of the site that is used when accessing the pre-rendered site. This setting normally is index.html.

Property	Description
<code><Sites value="Site1,Site2" /></code>	The sites within an IBM Workplace Web Content Management environment to cache are entered here, separated by commas.
<code><Interval></code> <code><Recurrence value="10" /></code> <code><StartDelay value="10" /></code> <code></Interval></code>	You can set the CacherModule to run after a recurring number of minutes. Recurrence is the recurring period in minutes for a recurring task. StartDelay is the delay in minutes prior to starting the first recurring task.
<code><Scheduled></code> <code><Times value="10:00am,12:00am,1:00pm" /></code> <code></Scheduled></code>	Alternately, you can set the CacherModule to run a certain times by entering a series of absolute times, separated by commas.

Note: If the pre-renderer is configured with `OverwriteCache` property set to `true`, all files located in the `DestDir` path that were not written by the pre-renderer are deleted. Therefore, only use the `DestDir` for storing rendered content.

In addition to configuring IBM Workplace Web Content Management for pre-rendering via HTTP, you must configure a Web server to serve the content. This Web server can be co-located with the IBM Workplace Web Content Management server, or it can exist on a remote server. IBM Workplace Web Content Management only renders content to a destination directory that is local to the IBM Workplace Web Content Management server. IBM Workplace Web Content Management cannot use other protocols (FTP, SCP, and so forth) to publish pre-rendered content. The destination directory can be a network mounted file system. Some operating systems, such as UNIX and Linux, support natively a wide variety of network file system protocols.

Regardless of where the Web server is located, IBM Workplace Web Content Management content contains path information that requires an alias entry in the Web server's configuration. The destination directory *must* be mapped to the alias `wps/wcm/connect`. This mapping ensures that links rendered in the IBM Workplace Web Content Management content are not broken.

8.6 Custom caching with connect tags

Connect tags are another function of the IBM Workplace Web Content Management engine that provide more control over the page rendering, caching, and dynamic content federation. This section discusses how you can use connect tags to implement custom caching. Connect tags do provide other features. However, those feature are beyond the focus of this section. To learn more about other features that connect tags provide, see the IBM Workplace Web Content Management Information Center at:

<http://www-128.ibm.com/developerworks/workplace/documentation/webcontentmanagement/>

Connect tag syntax is a replacement to IBM Workplace Web Content Management component tags (AptribxCmpnt and AptribxLibCmpnt) and provides attributes to override the site's default cache type and expiration settings. The connect tag syntax is as follows:

```
<connect mod="" srv="" {<Additional attributes>=""}>{Failure  
text}</connect>
```

Table 8-5 lists the available attributes and options.

Table 8-5 Attributes and options for connect tags

Attribute	Description
mod	Module used to display content. There are several valid modules, but for caching content in IBM Workplace Web Content Management, only the Default and ComponentRender options are needed. If this attribute is omitted, the Default module is used.
srv	Indicates the data provider for displaying content. The cmpnt option selects content or library components.
path	If retrieving a menu or navigator, this attribute supplies a path context (for example, /Site/Site Area/Content).
source	Specifies from where to retrieve the component. Valid options are: Library, Site, Site Area, and Content.

Attribute	Description
cache / contentcache	Overrides the site's default cache type, and indicates the cache type to use. Valid options are Site, Session, and None. These options are only valid when advance caching is used with the following options: User, Secured, and Personalized. Use the cache attribute with basic default cache. Use the contentcache attribute with advance or disabled default cache.
expires / contentcacheexpires	Overrides the site's default cache expiration and indicates a new expiration date or time period. Uses the expiration syntax as described in 3.7.2, "Configuring the <Cacher> configuration options" on page 117. Uses the expires attribute with basic default cache. Uses the contentcacheexpires attribute with advance or disabled default cache.
cmpntname	Name of component to retrieve content from

You can use connect tags as a replacement for the IBM Workplace Web Content Management component tags. The following examples replace the IBM Workplace Web Content Management component tags with connect tags and add caching attributes.

Example 1: Replacing the library HTML component

This sample uses the component tag:

```
<AptribLibCmpnt name="HTML - Home Page Navigation" />
```

This sample uses connect tags and adds caching attributes:

```
<connect mod="ComponentRenderer" srv="cmpnt" source="library"
cmpntname="HTML - Home Page Navigation" contentcache="site"
contentcacheexpires="REL 9000s" ></connect>
```

Example 2: Replacing the content component

This sample uses the component tag:

```
<AptribCmpnt context="current" type="content" key="Related Pages" />
```

This sample uses connect tags and adds caching attributes:

```
<connect mod="ComponentRenderer" srv="cmpnt" source="content"
  cmpntname="Related Pages" contentcache="personalized"
  contentcacheexpires="REL 2H" ></connect>
```

Example 3: Replacing the component reference in matching content of navigator or menu component

This sample uses the component tag:

```
<AptribCmpnt context="autoFill" type="content" key="Display Title" />
```

This sample uses connect tags and adds caching attributes:

```
<connect mod="ComponentRenderer" srv="cmpnt" source="content"
  path="<placeholder tag='sitepath' />/<placeholder tag='name' />"
  cmpntname="Display Title" contentcache="secured" contentcacheexpires="REL
  1D" ></connect>
```

Note: A connect tag does not have an *autoFill* option. To set the current path matched by a navigator or menu component, a connect tag *must* use placeholder tags in its path attributes.

Connect tags are the center piece of a custom caching strategy. These tags offer tremendous flexibility and performance improvements. However, they require discipline and practice. It is tempting to replace IBM Workplace Web Content Management component tags with connect tags throughout the site, without analyzing the site's performance first. Remember, connect tags are not processed in some circumstances. Be sure you use connect tags where it makes sense. When first using IBM Workplace Web Content Management, figuring out where to use connect tags should be driven by where the site's performance merits their use. Conduct a performance baseline, and then consider using connect tags to override the site's default cache.

8.7 Considerations in developing a caching strategy

The previous sections defined IBM Workplace Web Content Management caching capabilities and demonstrated how to configure and to use caching. This section provides tips and recommendations to help you develop a caching strategy. It includes the following sections:

- ▶ Identifying performance impacts
- ▶ Cache recommendations for components types
- ▶ Other caching systems that can effect performance

8.7.1 Identifying performance impacts

Presentation templates and components offer a great degree re-use. However, the number and complexity of the components that are included on presentation templates can affect the performance of the IBM Workplace Web Content Management page rendering process. Often, the same components are processed and inserted into multiple pages repeatedly. During the development of a caching strategy, it is important to understand the performance effects from component aggregation.

Identifying which components affect performance begins with the component or tag's function. The most compute intensive functions in IBM Workplace Web Content Management rendering are aggregation dynamic content inclusion and external content inclusion. Your caching strategy needs to consider the impact of the function before determining if and how an element should be cached.

Aggregation is the referencing of one IBM Workplace Web Content Management object within another. The process begins with the presentation templates that include components, fields, metadata, and connect tags. The aggregation continues as components include other components — generally called *composite* components. The composition depth can reach many levels deep, depending on the complexity of the component. A good cache strategy practice is to cache the composite component at the highest level possible (that is, closer to the reference in the presentation template is better).

In addition, you should consider how a composite component is re-used by other composite components. A composite component's security and personalization requirements also effect the decision of where to implement caching. If caching is implemented too low in the composition depth the cache might not save processing or memory, and if implemented too high in the composition depth, the cache can prevent dynamic content from updating appropriately.

Because composite components can include just about any other component or connect tag, these components can implement any of the caching types. Clearly defining the functionality for each element helps when the time comes to decide how to cache the element.

The dynamic functionality of IBM Workplace Web Content Management depends on the run-time state to determine what content to render. Evaluating dynamic functionality, achieved through components and connect tags, is one of the most compute expensive operations in IBM Workplace Web Content Management. Implementing caching on dynamic functionality improves performance significantly. The primary consideration for dynamic functionality is how dynamic must it be?

The update frequency of a dynamic element is defined in the site's information architecture. Often, the architecture defines how frequently a news feed should be refreshed for new content. However, elements such as the left-hand navigation or related links list are not defined. Because these elements are commonly represented as navigator and menu components, respectively, it benefits performance if these components are cached.

Dynamic functionality usually evaluates group memberships, category selections, keywords, or a personalization profile, and then delivers customized content. Selecting which cache type to use is a matter of the functionality that is required. For example, if the content needs to respond to the user's group memberships, then use the secured cache type. Dynamic functionality usually implements session, secured, user, or personalized cache types.

Repurposing external content in IBM Workplace Web Content Management by referencing it with a component or connect tag is an expensive operation. IBM Workplace Web Content Management retrieves content from a remote system and formats it for display within a presentation template. Functionality remains the primary concern. However, because a remote system is the source, a caching strategy must also consider the effect of the remote system that is temporarily unavailable. In this case, cache can mitigate the impact of temporary outages.

In addition, formatting the remote content can be expensive. It might require multiple requests to the remote system if only one attribute is retrieved at a time. When dealing with external content, it is preferable to format it and then cache the result when using a site cache type. Of course, if the external content responds to the current user's run-time state, then a more dynamic cache type is necessary.

8.7.2 Cache recommendations for components types

The following list includes our recommendations for implementing cache on some content and library component types:

HTML components

These components make excellent root containers for composite components. In addition to HTML, these components can contain IBM Workplace Web Content Management component tags and connect tags. As a result, HTML components are useful in aggregating other components via IBM Workplace Web Content Management component tags. It is a good practice to cache these composite components and to reference them with a connect tag, as shown in the following examples:

► HTML component: HTML — Home Page Navigation

```
<div>
  <AptribLibCmpnt name="NAV - Products" />
  <AptribLibCmpnt name="NAV - Services" />
  <AptribLibCmpnt name="NAV - Support" />
</div>
```

► Presentation Template — Home Page Layout (using basic caching)

```
<html>
...
<body>
  <connect srv="cmpnt" source="library" cmpntname="HTML - Home Page
  Navigation" cache="site" expires="REL 12H" >
</connect>
```

If a composite component needs different caching requirements on the components that it aggregates, then HTML and Rich Text components offer flexibility. Each connect tag can use different caching types and expiration parameters, as shown in the following examples:

► HTML — HTML, right-hand navigators:

```
<div>
  <connect srv="cmpnt" source="library" cmpntname="MENU - Hot Picks"
  cache="session" expires="REL 1H" ></connect>
  <connect srv="cmpnt" source="library" cmpntname="NAV - Site Tools"
  cache="site" expires="REL 24H" ></connect>
</div>
```

► HTML component— HTML, right-hand navigation

```
<div>
  <connect srv="cmpnt" source="library" cmpntname="MENU - Hot Picks"
  cache="session" expires="REL 1H" ></connect>
```

```
<connect srv="cmpnt" source="library" cmpntname="NAV - Site Tools"
cache="site" expires="REL 24H" ></connect>
</div>
```

Text components

Text components can contain static HTML and connect tags. If a connect tag is re-used throughout the site, instead of pasting the tag for each instance, the connect tag and its parameters can be saved in a text component and referenced using IBM Workplace Web Content Management component tags. This provides re-use, but if a parameter is changed, all occurrences are updated.

JSP components

Because JSP components can implement such a wide range of functionality, selecting a cache type is not always a clear choice. JSP components tend to deliver either custom formatting logic or integration with an external system. Determining a cache type usually depends on the level of dynamic behavior and whether the content is personalized.

Note: A JSP component is not rendered if a pre-rendering strategy is implemented.

Menu and navigator components

These two components are performance intensive functions of IBM Workplace Web Content Management. Menus tend to be far more performance-intensive than navigators, because of the nature of their more flexible content selection possibilities. How the site's menu and navigator components behave often is the primary factor for deciding between basic and advanced caching. If these components are required to respond to the current user's group memberships or profiling attributes, then an advanced cache strategy might be necessary. Typically, you can implement a site cache strategy and then override the cache for menus and navigators.

Personalization components

These components are new to IBM Workplace Web Content Management 5.1.0.1, and their behavior is similar to personalized menu components. Personalization components use personalization rules and content spots that are defined in WebSphere Portal's personalization engine, to aggregate and to deliver IBM Workplace Web Content Management and WebSphere Portal content.

Caching personalization components override the default cache and implement personalized cache type on an individual component basis. Unlike menus, which use IBM Workplace Web Content Management content or content that is

federated into IBM Workplace Web Content Management, personalization components display content that is aggregated in a portal personalization rule or content spot. Personalization rules and content spots can use IBM Workplace Web Content Management content but can also include external content.

Note: If connect tags are used within HTML, text, and Rich Text components, ensure that Process Connect Tags is selected on the presentation template that includes these components.

8.8 Other caching systems that can effect performance

Before implementing or testing a caching strategy, it is important to be aware of other caching systems which might effect IBM Workplace Web Content Management performance. If IBM Workplace Web Content Management is delivering content through WebSphere Portal, consult the portal administrator and determine if portal pages that deliver IBM Workplace Web Content Management content are configured to be cached by the Portal.

Note: The Portal's caching system is beyond to scope of this material. To learn more about Portal's cache system, consult the WebSphere Portal Information Center at:

<http://www-306.ibm.com/software/genservers/portal/library/index.html>

In addition to Portal's cache, review the cache settings on any Web servers that deliver IBM Workplace Web Content Management content that is configured with the WebSphere Application Server Plug-in. Basic caching should not be effected by the Web server's cache. However, if you are using an advanced cache strategy, ensure that the Web server's cache settings are configured to deliver secure and personalized content. Otherwise, performance can degrade because your Web server is caching too much or is rebuilding the cache constantly.

Be aware of how each system cache's content. Otherwise, unexpected results can appear when implementing and testing your cache strategy.

A caching strategy alone does not always improve performance sufficiently. We recommend that you follow WebSphere Application Server and WebSphere Portal Server performance tuning procedures in addition to implementing a caching strategy.

8.9 Documenting the cache strategy

Developing a cache strategy requires gathering information about the usability requirements and IBM Workplace Web Content Management implementation. Distilling all this information into a format to assist with site analysis can be a difficult task to manage. IBM Workplace Web Content Management does not provide an interface for tracking requirements and functionality. Producing cache strategy documentation is just another task on par with developing wire-frames, metadata schemas, or taxonomies.

Ideally, the cache strategy documentation should be included with the information architecture. The architecture should describe the functions of elements on the screen and include descriptions about how a user interacts with the elements. These elements are translated into templates and components in IBM Workplace Web Content Management.

After elements are realized as objects in IBM Workplace Web Content Management, the first step is to begin a catalog of components and presentation templates. The catalog can be simple spreadsheet workbook. Next, create a caching events timeline for each presentation template. You can create the events timeline using any flow chart application. If the site's architecture requires customized content to a session, user group, or personalization profile, it might be help to create diagrams with a timeline for each scenario. The following sections provide explanations of the documentation and examples of the information that you need to collect.

Before continuing, the volume and complexity of your documentation should reflect the size and complexity of the site. Some of the recommendations made in this section assume that the site in IBM Workplace Web Content Management is of sufficient size and complexity to warrant detailed documentation. Small sites might not require this level of documentation. Also, the cache type and expiration interval values use the same syntax as the values that are used in the connect tag cache and expires attribute.

8.9.1 Component and template catalog

The component and template catalog provides a list of all components in IBM Workplace Web Content Management and the presentation templates that use these components. Developing a catalog and updating it regularly is no small task. However, having a single reference to how the site's caching strategy is implemented is an invaluable resource. You can enter the list of library components and presentation templates manually, or you can export them programmatically via the IBM Workplace Web Content Management API.

Presentation template entries should include a unique entry ID, name, and description. Some of this data is stored in IBM Workplace Web Content Management as metadata to each presentation template. The unique entry ID is an ID that is unique for the catalog, as described in Table 8-6.

Table 8-6 Unique entry ID

Entry ID	Name	Description
T-01	Details Page Layout	HTML presentation template for Details authoring template. Default template for site.
T-02	Menu Page Layout	HTML presentation template for Menu authoring template. Defines Layout for Landing pages.

Each component entry in the catalog should include a unique entry ID, name, description, cache type, expiration interval, and a list of presentation template entry IDs. Some of this data is stored in IBM Workplace Web Content Management as metadata to each component. The cache type and expiration interval most likely is empty when you build the catalog. Make entries into the cache type and expiration when the site's default cache is overridden. Table 8-7 shows an example of a component entry.

Table 8-7 A component entry

Entry ID	Name	Description	Cache Type	Expiration Interval	Used in
C-01	NAV - Products	Displays all child site areas of the Products Site Area			T-02
C-02	Nav - Left Hand	Displays all child site areas of the Services site area	Site	REL 24H	T-02, T-03

Cache events timeline

After collecting the data for the component and template catalog, the information can be represented graphically on a timeline. Creating a visual timeline is another aid to understanding how the site's cache strategy functions. The data can be visualized in whatever format desired. Figure 8-2 uses a simple bar graph.

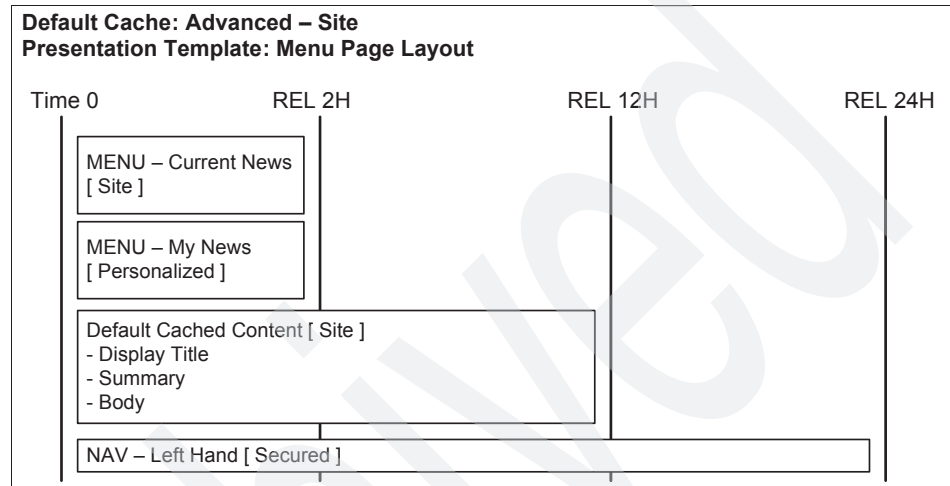


Figure 8-2 Example of a visual timeline in a bar-graph format

Figure 8-2 illustrates the cache refresh timeline for content that is rendered using the presentation template Menu Page Layout. The diagram defines:

- ▶ Time zero, which signifies the first time that the page is rendered. The page and components are stored in cache.
- ▶ Every 2 hours, the components MENU- Current News and MENU - My News expire from cache. The next time either of the components is rendered, it is cached. MENU - Current News implements site cache type, and MENU - My News implements personalized cache type.
- ▶ Every 12 hours, content that uses the default cache expires. The next time the page is accessed, it is cached.
- ▶ Every 24 hours, the component NAV - Left Hand expires from cache. The next time the component is rendered, it is cached. The component overrides the default cache and implements secured cache type.

Figure 8-2 depicts the case timeline for only one presentation template. You can expand this diagram easily to include multiple presentation templates and the cache dependences between them, as illustrated in Figure 8-3 on page 379.

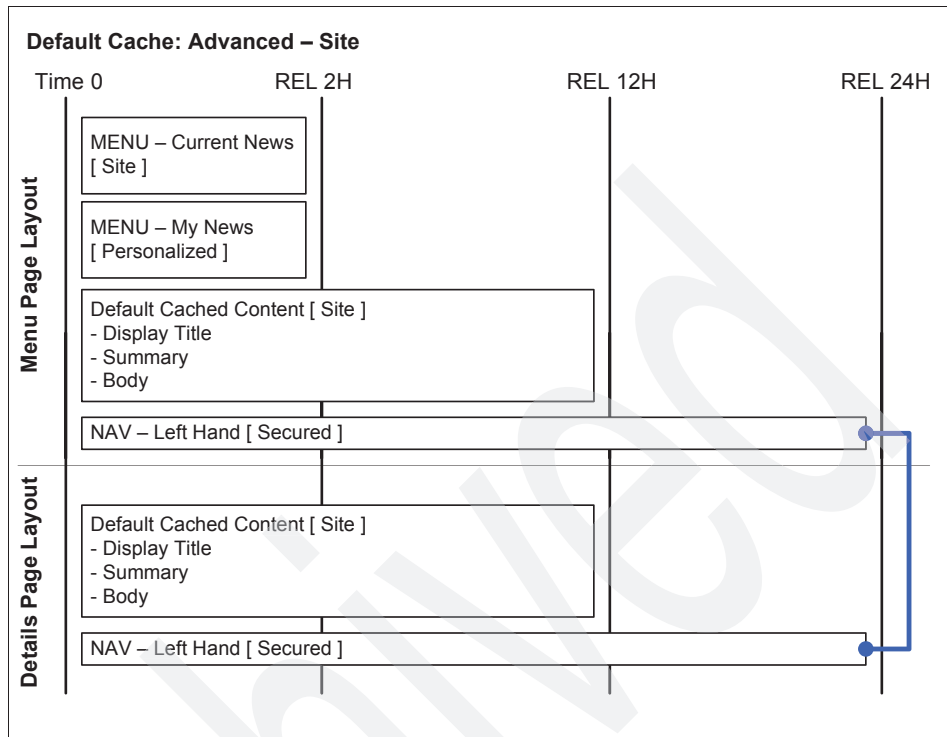


Figure 8-3 Two presentation templates and their cache dependences

Figure 8-3 expands the previous figure and includes a second presentation template. The cache refresh timeline is the same. However, this figure illustrates how the NAV - Left Hand component shares the same cache in both presentation templates.

Note: These diagrams only displayed relative times. You can also set the IBM Workplace Web Content Management cache to absolute times.

Time invested in developing a complete information architecture and supporting documentation pays off when it comes time to manage and to update the site. The cache strategy is just one aspect of the architecture to document. Creating visual representations of the caching strategy can communicate functionality to stakeholders and other interested parties who are not familiar with IBM Workplace Web Content Management.

8.10 WebSphere Dynamic Cache Service

Dynamic Cache (WebSphere Dynamic Cache Service) is a WebSphere service that is enabled on an application server by default. It supports caching of Servlet and JSP responses, WebSphere Commands objects, Web services objects, and Java objects. As an enterprise application on the WebSphere Application Server, the IBM Workplace Web Content Management uses this service.

To improve performance for an IBM Workplace Web Content Management site, there are a number of options available from the internal caching and pre-rendering to servlet caching within IBM Workplace Web Content Management. While this section provides a quick review of potential caching options, including IBM Workplace Web Content Management pre-rendering and IBM Workplace Web Content Management caching, it provides an overview of the WebSphere Dynamic Cache Service. It also provides a step-by-step guide to setting up WebSphere Dynamic Cache to work with IBM Workplace Web Content Management sites (portal and non-portal).

8.10.1 High performance (caching) options

This section serves as a brief review of the caching options and describes each option. It also introduces the WebSphere Dynamic Cache Service.

IBM Workplace Web Content Management pre-rendering

Consider the following before implementing an IBM Workplace Web Content Management pre-rendering strategy:

- ▶ Pre-rendering refers to a function within the IBM Workplace Web Content Management application that collates the design and content components and then stores them on a physical disk as distinct and complete HTML files.
- ▶ This option provides extremely fast response because the HTML files are placed as close as possible to the site visitors. For a portal site, use this option with Web clipping or other HTML portlets but not with the IBM Workplace Web Content Management rendering portlets.
- ▶ The IBM Workplace Web Content Management engine renders an entire site (based on the site framework that is selected in the configuration file) and cannot be broken down into partial pre-rendering. Pre-rendering is done by enabling the `<Cache class=...>` module in the `connect.cfg` file. The resulting HTML files can be stored on another device and used by other Web servers.
- ▶ This option cannot provide caching for dynamic caching (including IBM Workplace Web Content Management JSP component), personalized, or secured content.

For more information about this strategy, see 8.1.2, “Pre-rendering options” on page 351.

Figure 8-4 illustrates this strategy.

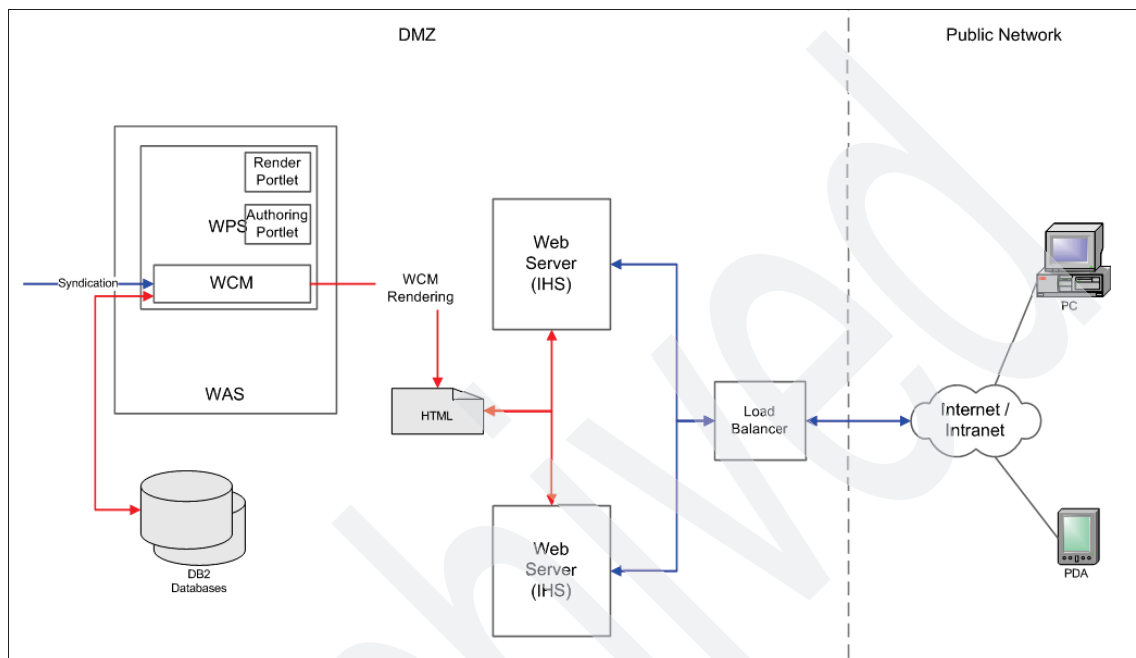


Figure 8-4 IBM Workplace Web Content Management pre-rendering strategy

IBM Workplace Web Content Management caching

Consider the following before implementing an IBM Workplace Web Content Management caching strategy:

- ▶ IBM Workplace Web Content Management caching stores content (after it has been rendered from various IBM Workplace Web Content Management components). Cache is stored when a page is requested.
- ▶ IBM Workplace Web Content Management Caching is set up through the cache configuration in the connect.cfg file. It requires IBM Workplace Web Content Management to act as a Web server because the caching file system is not in a true HTML format.
- ▶ Basic caching is similar to pre-rendering but offers a more flexible solution for a larger Web site with better currency of content. However, it cannot provide cache for dynamic, personalized or secured content.
- ▶ This option is also fairly fast but can only be used for non-portal sites.

- Advanced caching, which provides options to cache content selectively by session, users, and so forth, provides same output as the basic caching option.
- In addition, you can use connect tags to cache (or un-cache) components selectively within a Web page. This option is useful for a Web site that has content that is largely static with some dynamic components.

For more information about this strategy, see 8.1.1, “Caching options” on page 349.

Figure 8-5 illustrates this strategy.

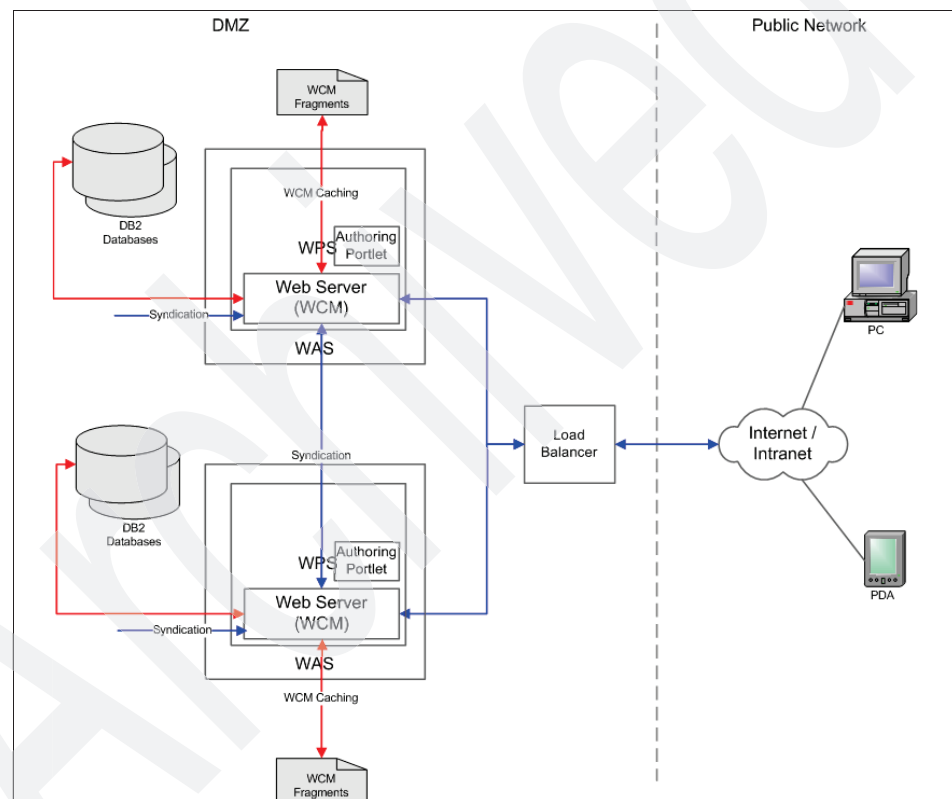


Figure 8-5 IBM Workplace Web Content Management caching strategy

Dynamic cache with servlet caching

Consider the following before implementing dynamic cache with servlet caching:

- ▶ This option represents an IBM Workplace Web Content Management caching solution that uses WebSphere Dynamic Cache Service.
- ▶ Servlet caching must be enabled on the WebSphere Application Server.
- ▶ For WebSphere Application Server V5, only one cache policy file is allowed per IBM Workplace Web Content Management instance. For WebSphere Application Server V6, it is possible to create multiple *servlet cache instances* to provide better flexibility and tuning of cache resources.
- ▶ This option offers a flexible and configurable solution based on WebSphere Application Server caching. You can configure a number of policy values to provide a tailored caching option, such as site structure, expiration, exclusion, and so forth.
- ▶ This option can cache static or dynamic content as fragments. Fragments can be set up to expire at different periods or to not cache at all.
- ▶ You can use this option for both portal (rendering portlets) and non-portal sites.
- ▶ This option can be configured to offload cache to disk if the memory cache is full (Disk Offload function).
- ▶ There is a limited cache memory management.
- ▶ In addition to the servlet caching, this caching solution can be further extended by using the Caching Proxy (part of Edge Components).

Figure 8-6 illustrates this caching strategy.

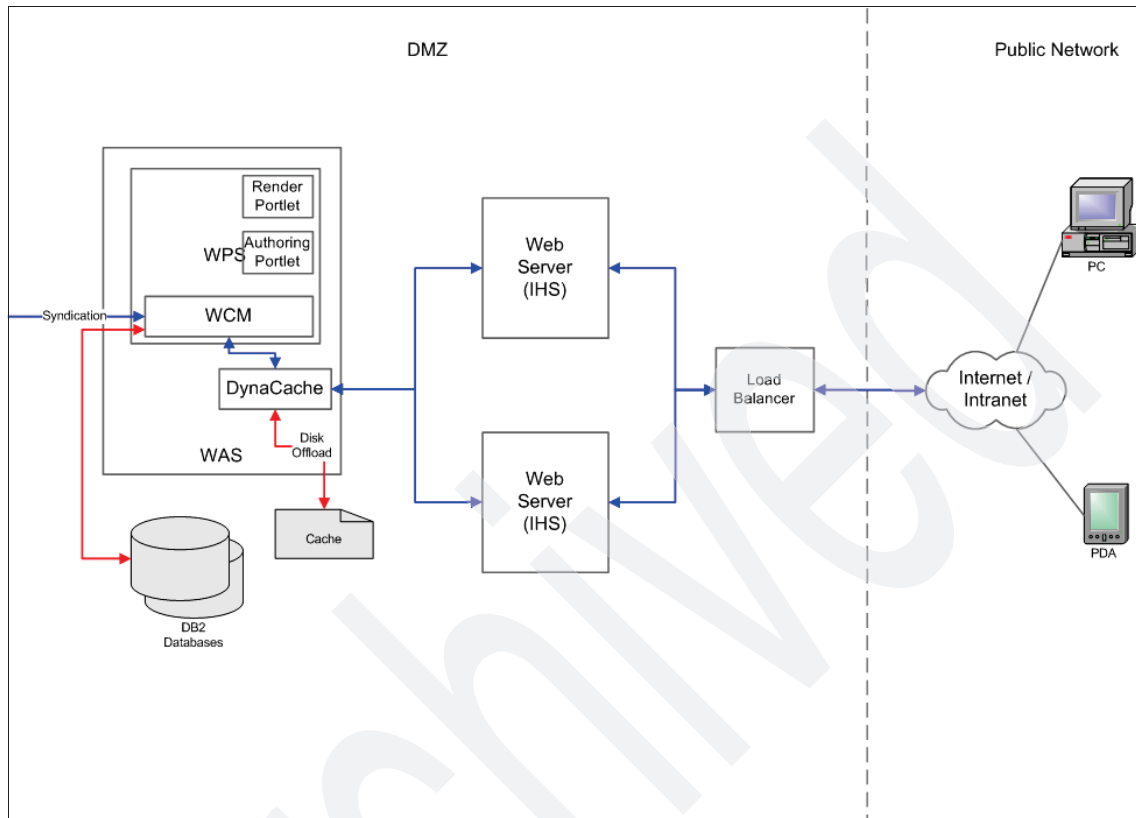


Figure 8-6 Servlet caching (dynamic cache)

Edge components

Consider the following before implementing edge components in your caching strategy:

- ▶ This option is an extension to the servlet caching.
- ▶ The caching proxy provides a caching solution, but is placed typically in front of the Web servers (and closer to the users).
- ▶ There are plug-ins to integrate with the directory to provide authentication and authorization.
- ▶ This method can also be cached at the edge (Web server level) if a plug-in is enabled. Invalidation options include timeouts, invalidation policy, or explicitly through the Edge Side Include API.

Figure 8-7 illustrates this caching strategy.

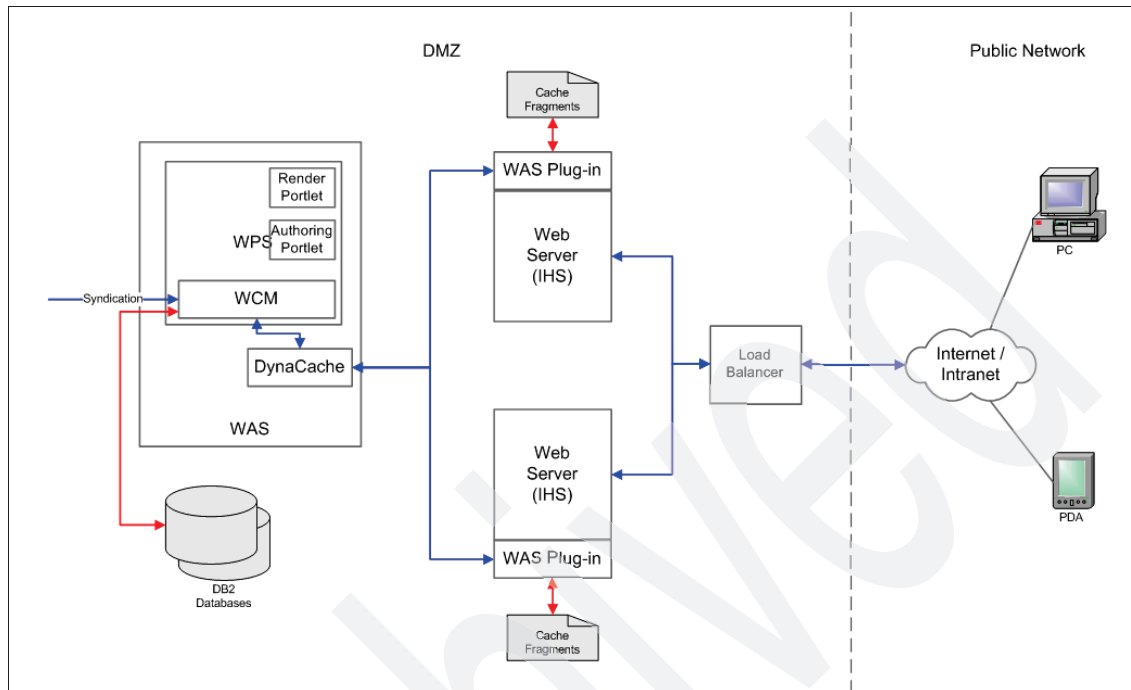


Figure 8-7 Dynamic cache with (Edge Side Include)

Dynamic cache with DistributedMap

DistributedMap is an interface that is provided by dynamic cache that applications use to cache and to share Java objects. This option is added for completeness, and we do not discuss it here. For more information about the DistributedMap option, refer to the WebSphere Application Server Information Center at:

<http://publib.boulder.ibm.com/infocenter/wasinfo>

8.10.2 Dynamic cache install and configuration

This section describes how to install and configure Dynamic Caching for WebSphere Portal. In addition to the description provided here, we recommend that you also refer to the WebSphere Portal Information Center at the following Web address for the latest information:

<http://www-306.ibm.com/software/genservers/portal/library/index.html>

Enabling servlet caching on a portal server

You must enable servlet caching on a portal server because IBM Workplace Web Content Management is a servlet by following these steps:

1. Go to the WebSphere Administration Console and select **Servers** → **Application Servers** → **WebSphere Portal**.
2. Select **Enable servlet caching** as shown in Figure 8-8.

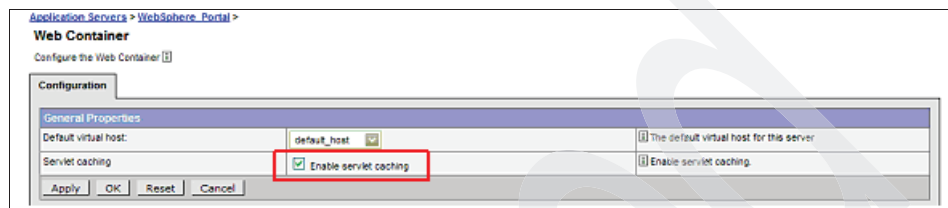


Figure 8-8 Enabling servlet caching

Note: This action requires that you restart the server. However, you should wait to restart the server until you have made all changes to the configuration.

Installing the dynamic cache monitor

The dynamic cache monitor is an installable Web application that displays simple cache statistics, cache entries, and cache policy information for servlet cache instances. The cache monitor is not installed on the WebSphere Application Server by default. To install it manually, follow these steps:

1. Locate the CacheMonitor.ear file (Figure 8-9). This file is usually located in the installableApps directory of the [WAS HOME] directory, for example C:\Program Files\WebSphere\AppServer\installableApps.

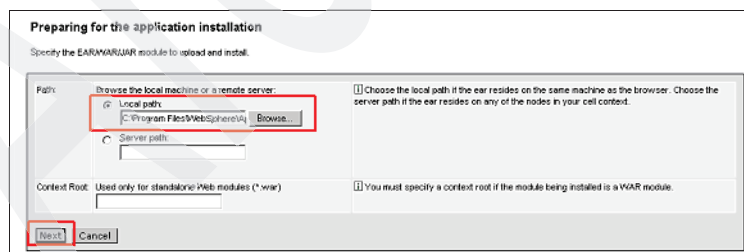


Figure 8-9 Locating the CacheMonitor.ear file

2. From the main WebSphere Application Server Administration Console, select, **Applications** → **Install New Application**.

3. Choose Local Path and use the Browse button to locate the Cache Monitor install file. Click **Next** to proceed.
4. Accept the default settings, as shown in Figure 8-10.

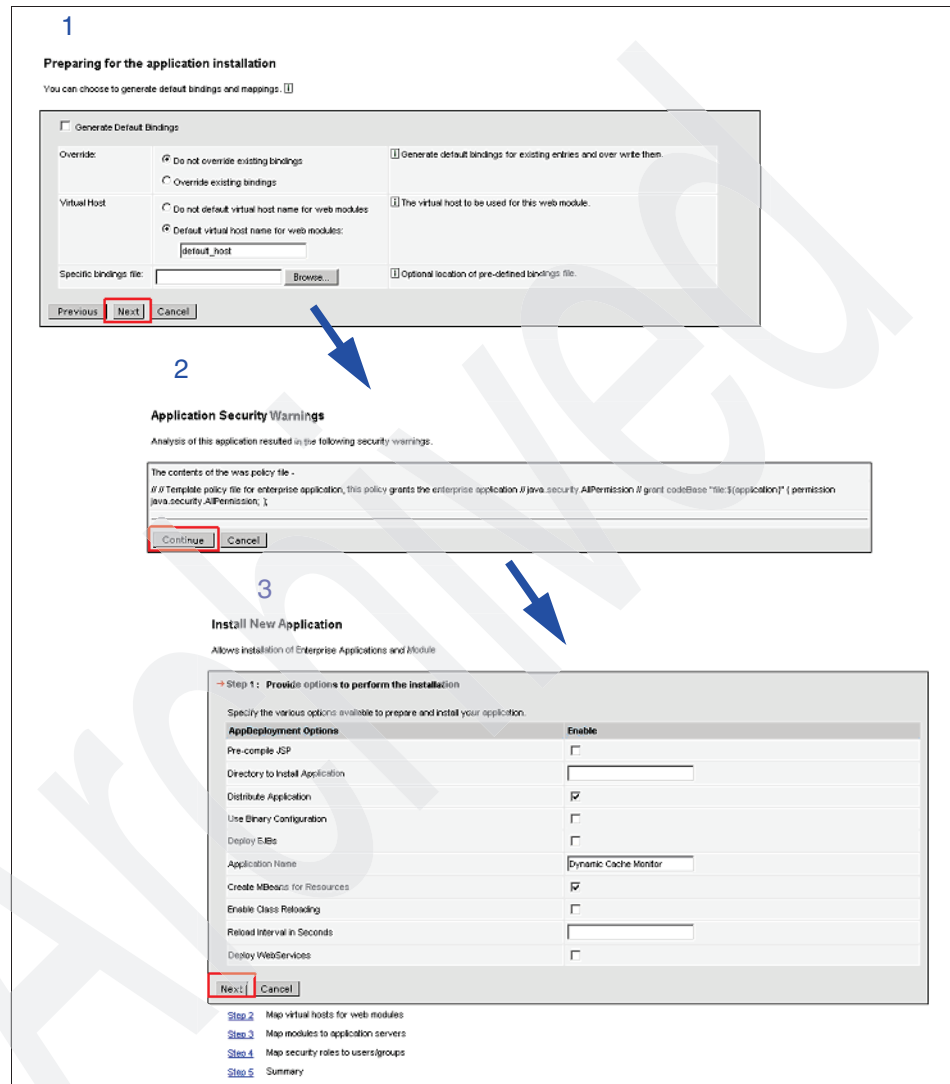


Figure 8-10 Default settings for installation screens

5. Map the cache monitor to a virtual host. Figure 8-11 on page 388 uses the default_host. (You can use the portal host name and port to access the cache monitor application.)

Install New Application
Allows installation of Enterprise Applications and Module

[Step 1](#) Provide options to perform the installation

Step 2: Map virtual hosts for web modules

Specify the virtual host where you want to install the Web modules contained in your application. Web modules can be installed on the same virtual host or dispersed among several hosts.

☐ Apply Multiple Mappings

To apply multiple mappings, follow the steps below:

1. Select one or more checkboxes in the table
2. Complete mappings and click APPLY

Virtual Host:

Web Module	Virtual Host
☒ Dynamic Cache Monitor	admin_host

[Step 3](#) Map modules to application servers
[Step 4](#) Map security roles to users/groups
[Step 5](#) Summary

Figure 8-11 Mapping the cache monitor to a virtual host

6. Map the Web modules to the servers to be monitored (for example, WebSphere_Portal and server1) as shown in Figure 8-12.

Install New Application
Allows installation of Enterprise Applications and Module

[Step 1](#) Provide options to perform the installation

[Step 2](#) Map virtual hosts for web modules

Step 3: Map modules to application servers

Specify the application server where you want to install modules contained in your application. Modules can be installed on the same server or dispersed among several servers.

Clusters and Servers:

☒ WebSphere cell=wp51vm/node=wp51vm/server=server1
☐ WebSphere cell=wp51vm/node=wp51vm/server=WebSphere_Portal

Module	URL	Server
☒ Dynamic Cache Monitor	CacheMonitor.war?WEB-INF/Arrob.xml	WebSphere cell=wp51vm/node=wp51vm/server=server1

[Step 4](#) Map security roles to users/groups
[Step 5](#) Summary

Figure 8-12 Mapping Web modules to the servers to be monitored

7. Set up security access to the Interface. In Figure 8-13, access is given to everyone. However, in the production environment, you might want to restrict this access to the Administrator group.

Install New Application
Allows installation of Enterprise Applications and Module

[Step 1](#) Provide options to perform the installation
[Step 2](#) Map virtual hosts for web modules
[Step 3](#) Map modules to application servers

→ **Step 4: Map security roles to users/groups**

Each role defined in the application or module must be mapped to a user or group from the domain's user registry.

Lookup users **Lookup groups**

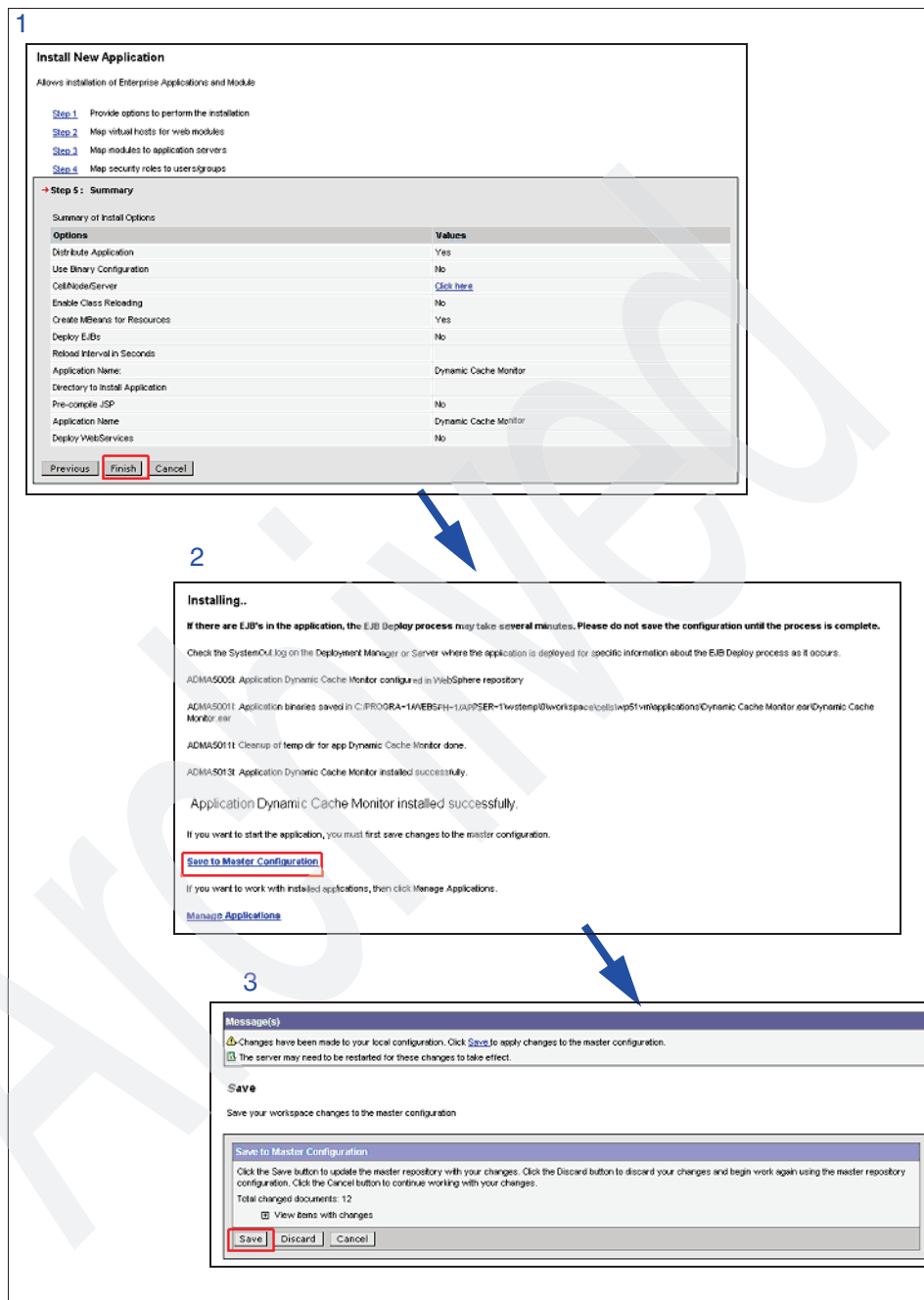
☐ Role	☐ Everyone?	☐ All Authenticated?	Mapped Users	Mapped Groups
☐ Administrator	☒	☒		

Previous **Next** Cancel

[Step 5](#) Summary

Figure 8-13 Setting up security access to the interface

8. Click **Finish**, **Save to Master Configuration**, and **Save** again, as shown in Figure 8-14 on page 390.



Configuring IBM Workplace Web Content Management cache policy

The cache policy is stored as an XML file (cachespec.xml) on the IBM Workplace Web Content Management installed sub directory, for example
<WAS_ROOT>\installedApps\<server_name>\ilwwcm_war.ear\ilwwcm.ear\WEB_INF\ Example 8-7 caches all content except for Search Results, Personalize, and Voice Mail Policy pages.

Example 8-7 Cache policy example

```
<?xml version="1.0" ?>
<!DOCTYPE cache SYSTEM "cachespec.dtd">
<cache>
  <cache-entry>
    <class>servlet</class>
    <name>com.presence.connect.ConnectServlet.class</name>
    <property name="store-cookies">false</property>
    <property name="save-attributes">true</property>
    <property name="consume-subfragments">true</property>
    <cache-id>
      <component id="" type="pathinfo">
        <required>true</required>
        <not-value>/Intranet/Search+Results</not-value>
        <not-value>/Intranet/Tools/Personalize</not-value>
      </component>
    </cache-id>
    <timeout>3000</timeout>
  </cache-entry>
</cache>
```

In this example:

- ▶ The com.presence.connect.ConnectServlet.class = container class for IBM Workplace Web Content Management caches everything except those pages that have been excluded explicitly.
- ▶ The exclusion value (in <not-value>) must contain the forward slash (/) at the beginning of the policy value. Otherwise, there is unpredictable behavior for the home page (error 500 - Server caught unhandled exception from servlet [WCM Framework]: OutputStream already obtained).
- ▶ A forward slash (/) at the end of a policy value to specify whether the children of the path is included on just the components at that level. For example:
<not-value>/Intranet/Tools/Personalize</not-value>

DynaCache interprets this as any objects and components with a URI that starts with /Intranet/Tools/Personalize/, such as the following:

```
<not-value>/Intranet/Home/IT/Procedures/Voice+Mail+Policy</not-value>
```

This example exclude only the Voice Mail Policy page from the cache.

Alternatively, you can use the policy in Example 8-8. This policy is particularly useful if the caching solution is required only for part of the site (or other site framework).

Example 8-8 Caching for only part of the site

```
<?xml version="1.0" ?>
<!DOCTYPE cache SYSTEM "cachespec.dtd">
<cache>
  <cache-entry>
    <class>servlet</class>
    <name>/connect/Intranet/Home/</name>
    <property name="store-cookies">false</property>
    <property name="save-attributes">true</property>
    <property name="consume-subfragments">true</property>
    <cache-id>
      <component id="" type="pathinfo">
        <required>true</required>
        <not-value>/Intranet/Search+Results</not-value>
        <not-value>/Intranet/Tools/Personalize/</not-value>
      </component>
    </cache-id>
    <timeout>3000</timeout>
  </cache-entry>
</cache>
```

In this example, The value for <name> must be in the following format:

```
/connect/[SITE]/[HOMEPAGE]/
```

Figure 8-15 illustrates an example site framework and its site areas.

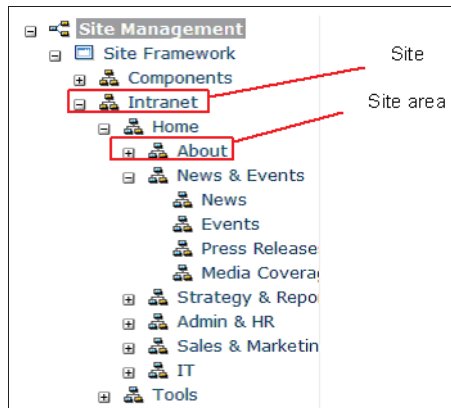


Figure 8-15 Site framework and site areas

Note: For more information about the definitions and use of IBM Workplace Web Content Management site and site areas, refer to 5.3.8, “Site framework” on page 170 and “Site area” on page 172. Alternatively, refer to the IBM Workplace Web Content Management Information Center at:

<http://pvcid.raleigh.ibm.com/ilwwcm/>

As the pages are accessed, the dynamic cache continues to cache page fragments. The cached fragments can be viewed via the cache monitor. Use the following URL to access the cache monitor:

`http://[portal_hostname]:[portal_port]/cachemonitor`

For example:

`http://wp51vm.ibm.com:9081/cachemonitor/`

Figure 8-16 illustrates the cache monitor interface.

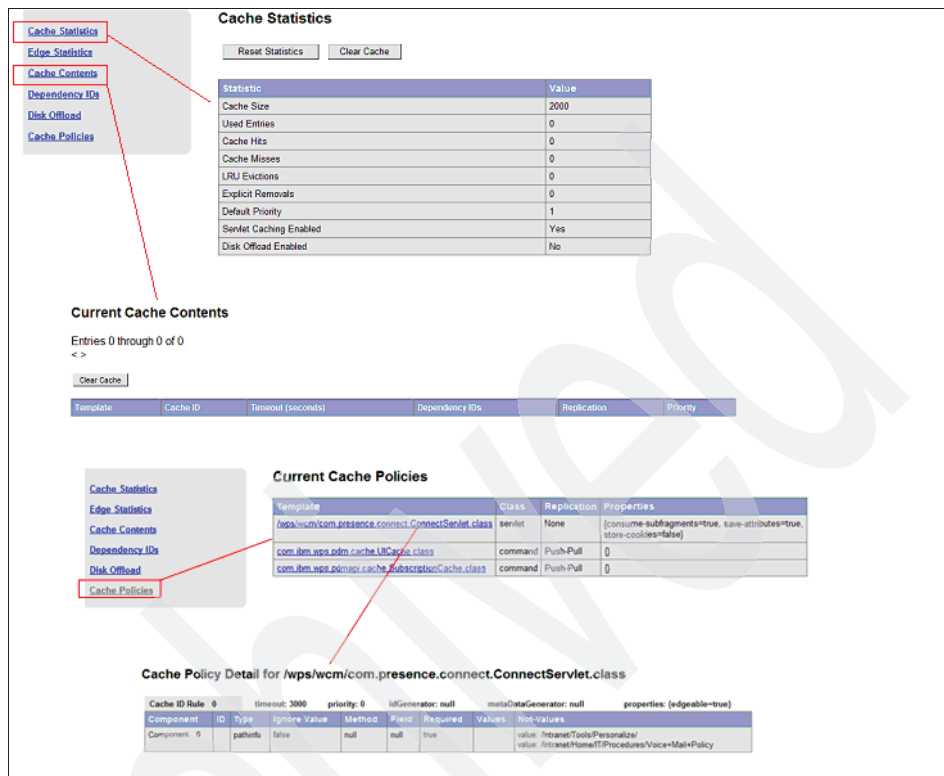


Figure 8-16 Cache monitor interface , including cache statistics and policies

<http://wp51vm.ibm.com:9081/wps/wcm/connect/Intranet/>

Then, the cache contents look similar to that shown in Figure 8-17.

[illegible]

Figure 8-17 Current cache contents

Each fragment can be refreshed or invalidated individually and manually as is the entire cache.

Installing Edge Side Include

By enabling the Edge Component, contents such as images and JSPs can be cached by the Edge Component, which is installed typically on the front-line Web Servers. You must first install the DynaCacheEsi.ear file so that the Edge components and statistics can be managed via the Cache Monitor. Follow these steps:

1. Locate the DynaCacheEsi.ear file. It is located usually in the installableApps directory of the [WAS HOME] directory (for example, C:\Program Files\WebSphere\AppServer\installableApps).
2. From the main WebSphere Application Server Administration Console, select **Applications** → **Install New Application**.
3. Choose **Local Path** and use the Browse button to locate the Cache Monitor install file.
4. Click **Next** to proceed.

Preparing for the application installation

Specify the EAR/VAR/JAR module to upload and install.

Path:	Browse the local machine or a remote server:	i Choose the local path if the ear resides on the same machine as the browser. Choose the server path if the ear resides on any of the nodes in your cell context.
	☒ Local path: c:\apps\DynacacheEsi.ear Browse...	
	☐ Server path: 	
Context Root:	Used only for standalone Web modules (*.war)	i You must specify a context root if the module being installed is a VAR module.
<input checked="" type="button" value="Next"/> Cancel		

Figure 8-18 Installing the DynaCacheEsi.ear file

5. Accept the default settings as shown in Figure 8-19 on page 397.

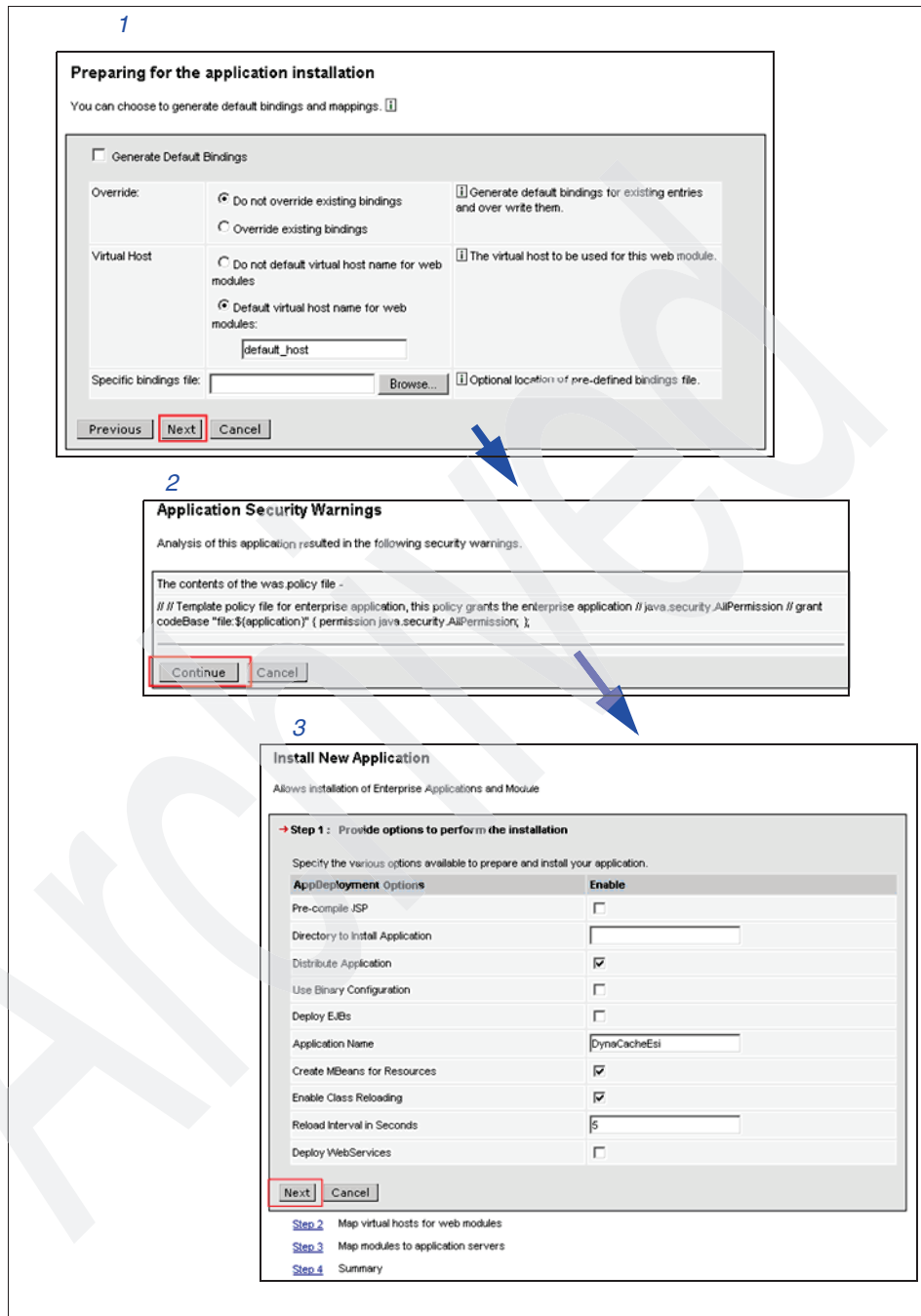


Figure 8-19 Installation default settings for DynaCacheEsi.ear

6. Map DynaCacheEsi.ear to a virtual host. In Figure 8-20, the default_host is used.

Install New Application
Allows installation of Enterprise Applications and Module

[Step 1](#) Provide options to perform the installation

→ **Step 2: Map virtual hosts for web modules**

Specify the virtual host where you want to install the Web modules contained in your application. Web modules can be installed on the same virtual host or dispersed among several hosts.

☐ Apply Multiple Mappings

To apply multiple mappings, follow the steps below.

1. Select one or more checkboxes in the table
2. Complete mappings and click APPLY

Virtual Host: default_host Apply

Web Module	Virtual Host
☒ DynaCacheEsi	default_host

Previous Next Cancel

[Step 3](#) Map modules to application servers
[Step 4](#) Summary

Figure 8-20 Mapping to a virtual host, selecting default_host

7. Map Web modules to the servers to be monitored (for example, WebSphere_Portal and server1), as shown in Figure 8-21.

Install New Application
Allows installation of Enterprise Applications and Module

[Step 1](#) Provide options to perform the installation
[Step 2](#) Map virtual hosts for web modules

→ **Step 3: Map modules to application servers**

Specify the application server where you want to install modules contained in your application. Modules can be installed on the same server or dispersed among several servers.

Clusters and Servers:

WebSphere: cell=wp51vm,node=wp51vm,server=server1
WebSphere: cell=wp51vm,node=wp51vm,server=WebSphere_Portal

Apply

Module	URI	Server
☒ DynaCacheEsi	DynaCacheEsi.war/WEB-INF/web.xml	WebSphere: cell=wp51vm,node=wp51vm,server=server1

Previous Next Cancel

[Step 4](#) Summary

Figure 8-21 Mapping Web modules to servers

8. Click **Finish**, **Save to Master Configuration**, and **Save** as shown in Figure 8-22.

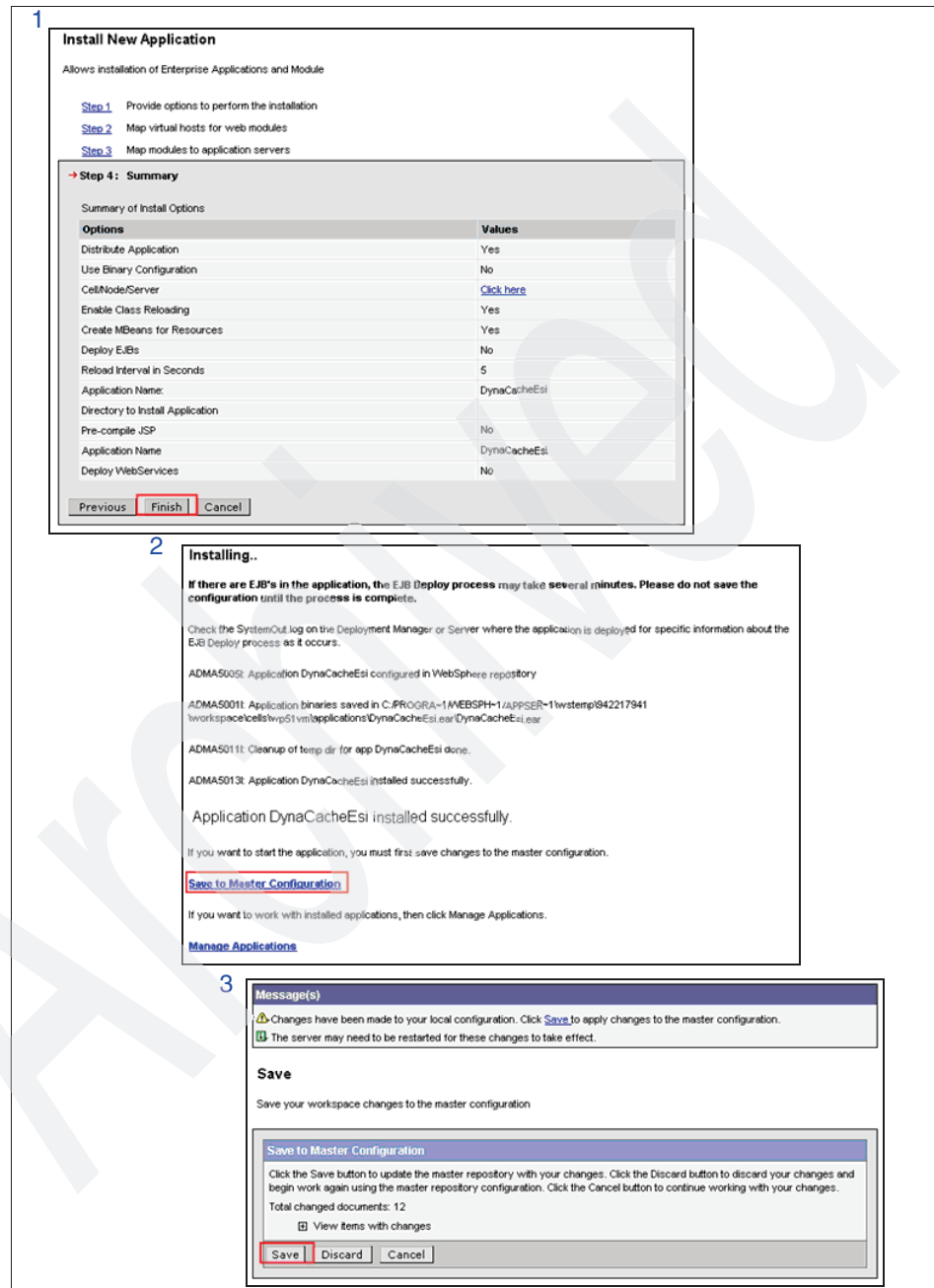


Figure 8-22 Finishing installation of DynaCacheESI.ear and saving

Enabling ESI components

You must enable the ESI components on the WebSphere Application Server plugin-cfg.xml. You must also amend the IBM Workplace Web Content Management cache policy to use the edge component.

This example caches fragments of the contents, such as images and JSPs by the Edge component (typically installed at the front line Web servers). To enable this component, the edge component is enabled on the plugin-cfg.xml, which is located in the [WAS_HOME]\config\cells directory, for example C:\Program Files\WebSphere\AppServer\config\cells.

The plugin-cfg.xml is generated by the server when the WebSphere Application Server plug-in is installed and the edge properties are there by default, as in the following:

```
<Property Name="EsiEnable" Value="true"/>
```

In addition, you should also set the following properties:

```
<Property Name="esiMaxCacheSize" Value="1024"/>
<Property Name="esiInvalidationMonitor" Value="true"/>
```

In this example:

- ▶ `esiMaxCacheSize` is the maximum size of the cache in 1 KB units. The default maximum size of the cache is 1 MB. If the cache is full, the first entry to be evicted from the cache is the entry that is closest to expiration.
- ▶ `esiInvalidationMonitor` specifies whether the ESI processor should receive invalidations from the application server. By default, it is set to false.

Example 8-9 shows the entries that plugin-cfg.xml should have.

Example 8-9 plugin-cfg.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Config ASDisableNagle="false" AcceptAllContent="false"
AppServerPortPreference="HostHeader" ChunkedResponse="false"
IISDisableNagle="false" IISPluginPriority="High" IgnoreDNSFailures="false"
RefreshInterval="60" ResponseChunkSize="64" VHostMatchingCompat="false">
  <Log LogLevel="Error"
Name="C:/PROGRA~1/WEBSPH~1/APPSER~1/logs/http_plugin.log"/>
  <Property Name="ESIEnable" Value="true"/>
  <Property Name="ESIMaxCacheSize" Value="1024"/>
  <Property Name="ESIInvalidationMonitor" Value="false"/>
  <VirtualHostGroup Name="default_host">
    .
    .
  </VirtualHostGroup>
</Config>
```

Example 8-10 adds the `EdgeCacheable` property to the `cachspec.xml` to indicate that the fragment has to be cached on the edge.

Example 8-10 `cachspec.xml`

```
<?xml version="1.0" ?>
<!DOCTYPE cache SYSTEM "cachspec.dtd">

<cache>
  <cache-entry>
    <class>servlet</class>
    <name>com.presence.connect.ConnectServlet.class</name>
    <property name="store-cookies">false</property>
    <property name="save-attributes">true</property>
    <property name="consume-subfragments">true</property>
    <cache-id>
      <component id="" type="pathinfo">
        <required>true</required>
        <not-value>/Intranet/Search+Results</not-value>
        <not-value>/Intranet/Tools/Personalize</not-value>
        <not-value>/Intranet/Home/IT/Procedures/Voice+Mail+Policy</not-value>
      </component>
      <timeout>3000</timeout>
      <property name="EdgeCacheable">true</property>
    </cache-id>
  </cache-entry>
</cache>
```

Archived

Advanced integration with WebSphere Portal

This chapter explains how to build more tightly integrated IBM Workplace Web Content Management and WebSphere Portal solutions. It includes the following sections:

- ▶ Overview of the WebSphere Portal integration
- ▶ Reasons for using Portal navigation
- ▶ Bridging the gap
- ▶ Beginning the implementation
- ▶ Implementing a custom display Portlet
- ▶ Managing links
- ▶ Using multilingual and personalized content display

Note: The integration methods that are described in this chapter are not a replacement for the publishing options that are outlined in previous chapters. The methods discussed here are an extension of the standard publishing. You can use them to implement more advanced functionalities.

9.1 Overview of the WebSphere Portal integration

The IBM Workplace Web Content Management implementation options that we have described so far come from a Web content management background. Thus far, we have used the WebSphere Portal framework mainly in the content creation process. We now discuss the options that you have to leverage the WebSphere Portal framework to its full extent.

From a developers point of view, WebSphere Portal is as a powerful framework on which to build Web applications. It provides 90% of the functionality that you need to build world-class Web applications. These features range from security, user-management, navigation, and integration APIs to Web content management. WebSphere Portal also removes from applications many of the design elements that are related to infrastructure and GUI support and allows the developer to focus on creating business functions in applications.

This chapter discusses how to build applications so that you mix and match content while also providing feature rich functionality. For example, you can have a product finder that interacts with your IBM Workplace Web Content Management content, or you can build Portlets that handle multi-lingual content display.

Before we jump into coding, however, we want to review the following important concepts:

► Navigation

Chapter 5, “Key concepts and terminology” on page 141 describes the function of sites and site areas. Sites and site areas structure the content inside the IBM Workplace Web Content Management repository and build the site navigation.

WebSphere Portal maintains a content structure as well. This structure is part of its *composition topology* and, therefore, is a fundamental concept of WebSphere Portal. The Portal holds a tree-structure of its contents, which is called the *Content-Model*. Elements of the Content-Model represent pages with content, labels, or URLs. Each single page consists of layout information (containers) and content (Portlets). Portal pages are the containers for the Portlets. One of the main purposes of this Content-Model is to build the Portal navigation. The composition topology is closely tied to the security model.

► Review on Portlets

A *Portal* provides a consistent view of Portal applications (*Portlets*). The Portal allows the administrator to define specific sets of applications, which are presented to the user in a single page context. The Portlets themselves are more than simple views of existing Web content. A Portlet is a complete

application that has multiple states and view modes, plus event and messaging capabilities.

Portlets run inside the Portlet *container* of a Portal *server*, similar to a servlet running on an application server. The Portlet container provides a runtime environment in which portlets are instantiated, used, and finally destroyed. Portlets rely on the Portal infrastructure to access user profile information, participate in window and action events, communicate with other portlets, access remote content, look up credentials, and store persistent data.

The user interacts with Portlets through *Action-URIs*. By clicking a link inside the Portlet, the user can perform certain actions, for example, changing the display view. Internally, the Portlet stores persistent information about its current state. One Portal page aggregates many different Portlets at the same time, which is an important issue to consider when you design and work with Portals.

9.2 Reasons for using Portal navigation

The content-model is an important component in the WebSphere Portal architecture. To take full advantage of the Portal framework and its functionality, you should consider using the Portal Content-Model to build the site structure and the navigational schema. From an implementation point of view, using the Content-Model means the following:

1. You have to create one Portal page for every navigation entry in the site. Our sample site uses a head navigation and a left-hand navigation pane. (See Figure 9-1 on page 406.) Of course, you can build other navigation schemas as well. For pages that do not contain any content, you can use labels.

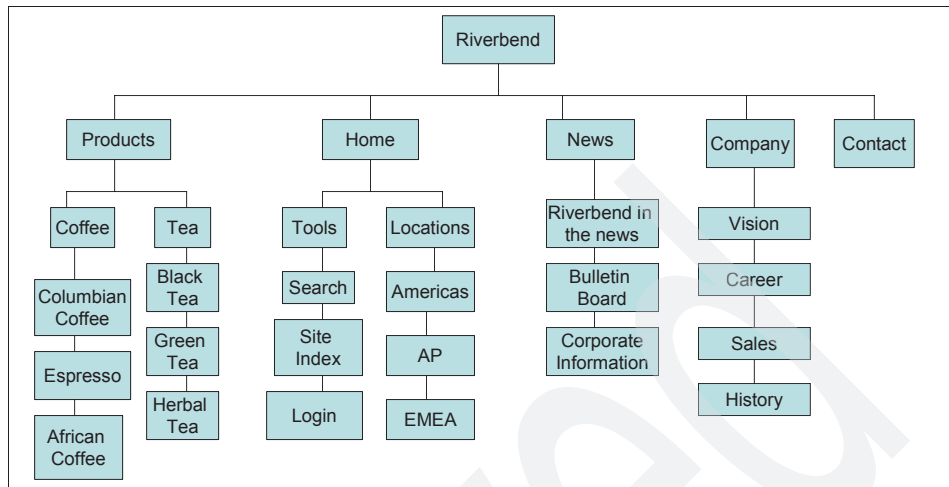


Figure 9-1 Each page and navigation element represents one Portal page

2. You need to create the Portal pages in a tree hierarchy that represents the hierarchy of the navigation, as shown in Figure 9-2.

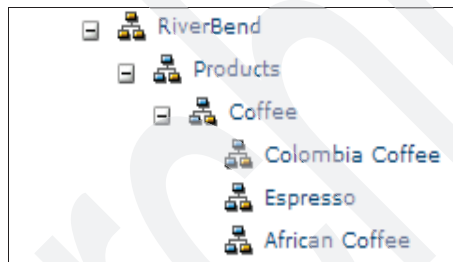


Figure 9-2 Illustration of the Portal navigation hierarchy

3. You need to implement the navigation itself using the themes and skins of the Portal.

This implementation starts outside of the IBM Workplace Web Content Management environment. Thus, it is helpful for you to familiarize yourself with basic Portal administration and customization.

9.2.1 Understanding the layout of the Portal page

Before customizing your Portal site, it helps to understand the underlying structure of the Portal as it is determined by the Portal JSPs. The Portal page is composed of JSPs for screens, themes, and skins that are created typically by the Web designer of the Portal.

These JSPs reside in the corresponding /screens, /themes, /skins directories under the *was_profile_root*/installedApps/cellname/wps.ear/wps.war directory. Within this location, subdirectories for markup, locale, and client types are used to support Portal aggregation.

- ▶ Screens

Screens represent the area of the Portal that typically displays portlets (that is, the Home screen) but can also display other content in its place (for example, a log in form or error message). Screens are selected from navigation icons in the theme.

- ▶ Themes

Themes provide the navigation, appearance, and layout of the Portal, including colors, fonts, and images outside of the portlet content area (the Home screen).

- ▶ Skins

Skins represent the border rendering around components, such as row containers, column containers, or portlets. Skins are installed independently from themes. However, the administrator can set a default skin for a theme.

The starting place for building the Portal page is Default.jsp in the /themes directory. For more information, see the Information Center at:

http://publib.boulder.ibm.com/pvc/wp/510/ent/en/InfoCenter/wps/dgn_lyt.html

In the following sections, we assume that you have built the site, site areas, and input templates as described in Chapter 7, “Building the River Bend Web site with IBM Workplace Web Content Management” on page 253 and that you also have created some content. We now discuss how to integrate the IBM Workplace Web Content Management content with the WebSphere Portal Content-Model and that navigation that you have created.

9.3 Bridging the gap

How do the content documents know where they should get displayed inside of the Portal? Obviously, there is a gap between the WebSphere Portal Content-Model and the site and site areas that you have built. To bridge this gap, you can use *mapping schemas*. There are two types of mapping schemas: explicit schemas and implicit schemas.

9.3.1 Explicit mapping

Explicit mapping allow either the Portal administrator or the content authors to control where content is shown inside of the Portal. You have two options:

- ▶ You can have the administrator place the IBM Workplace Web Content Management local rendering Portlet on a Portal page. Then, you can configure which document from the IBM Workplace Web Content Management repository should be shown. Read 9.4.1, “Explicit mapping approach” on page 414 for more information about this option.
- ▶ You can allow the content author to assign a document to a Portal page. In this case, the content author selects a Portal page during the authoring process. The input template might offer the content author a list with all Portal pages from which to choose. Read 9.4.2, “Making the implementation more functional” on page 416 for more information about this option.

Figure 9-3 on page 409 illustrates the two different structures that you have to map together.

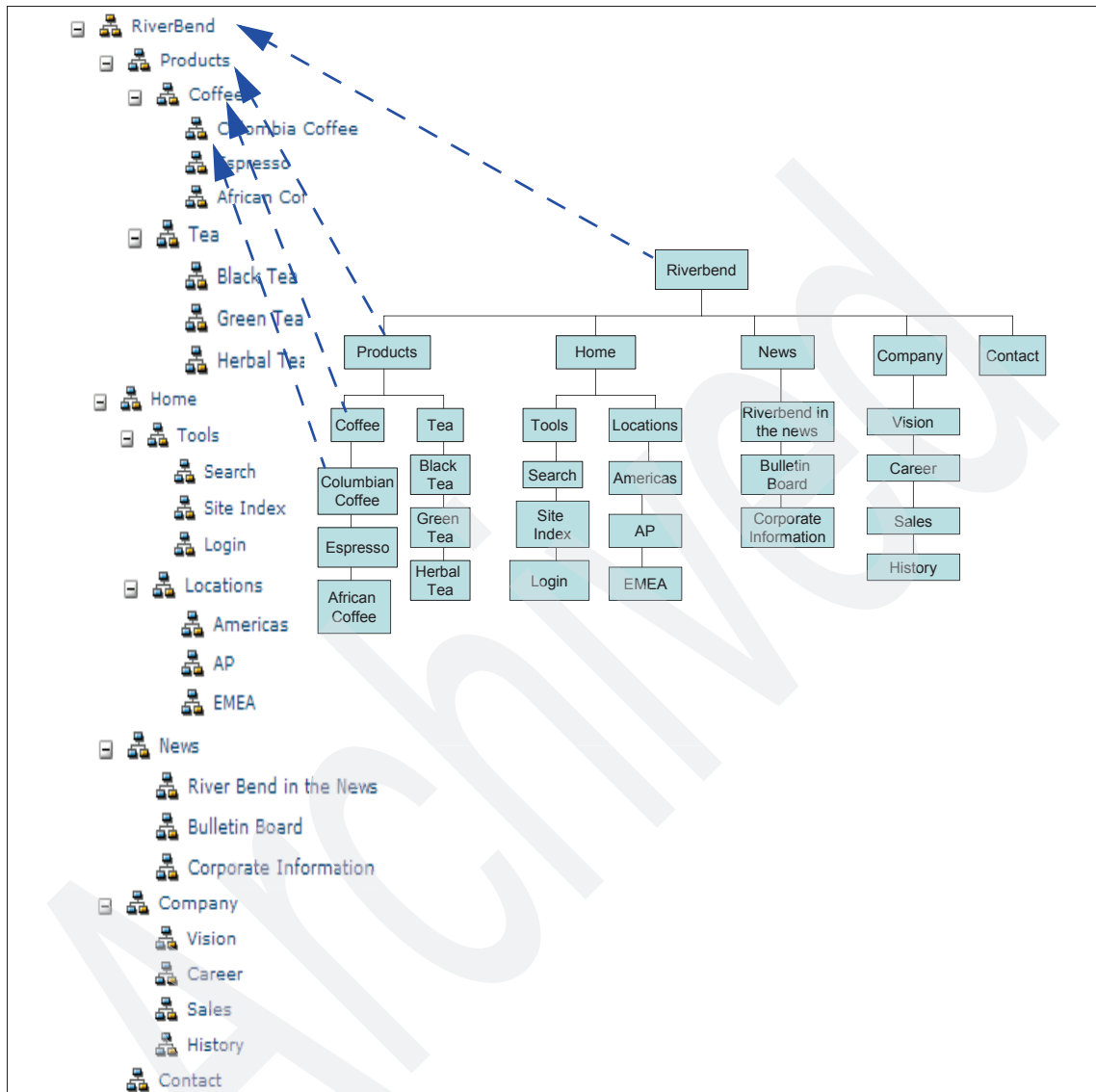


Figure 9-3 Site areas structure and Portal content model (page hierarchy)

On the left-hand side, the figure shows the site areas of IBM Workplace Web Content Management, and on the right-hand side, the figure shows the Portal page hierarchy.

The content author assigns a target Portal page explicitly for the content that is displayed. Figure 9-4 illustrates the resulting mapping schema.

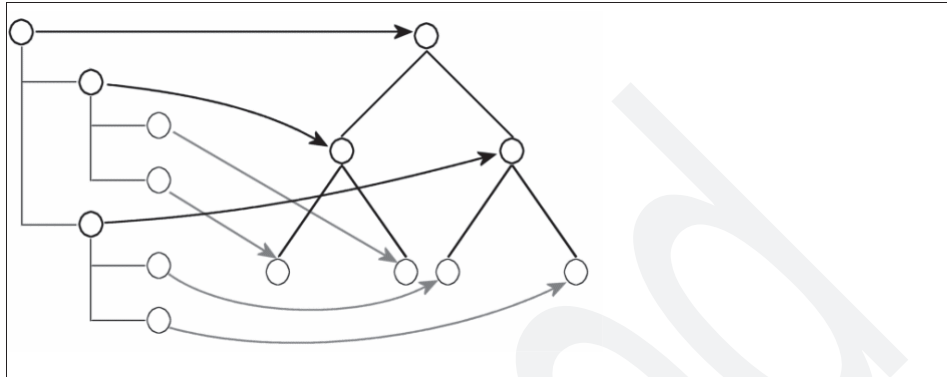


Figure 9-4 Explicit mapping between site areas and Portal page hierarchy

The mapping information is then stored inside of the content document.

9.3.2 Implicit mapping

With implicit mapping, you create rules that are based on a content taxonomy. For example, if `product=Coffee` then `Portal page= riverband.products.coffee`. The mapping rule is then evaluated by the Portlet code to decide what content document to display. A good way to work with implicit mappings is to use the rules engine of the Portals personalization engine. For more information, refer to the Information Center at:

http://publib.boulder.ibm.com/pvc/wp/510/ent/en/InfoCenter/pzn/pzn_demooverview.html

9.3.3 Choosing structures that make mapping easy

From our experience with customer engagements we have learned that we need to think about the IBM Workplace Web Content Management structure and Portal structure *before* beginning any implementation. To choose structures that make the mapping easy, you have the following options:

► Use Portlet menus

Portlet menus allow a Portlet to add menu entries to the navigation tree of the Portal (Figure 9-5 on page 411). This functionality allows a custom IBM Workplace Web Content Management Portlet to extend the Portal navigation dynamically. This way, the IBM Workplace Web Content Management content can add its own sub navigation entries.

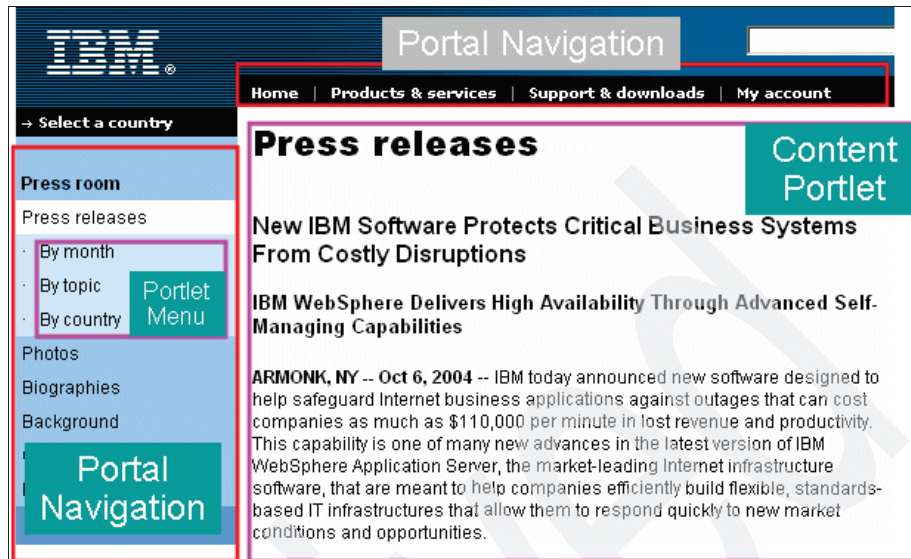


Figure 9-5 Example showing Portlet menu

The added Menu items are positioned automatically in the navigation tree after the page that contains the custom Portlet. The Portlet menu entries do *not* belong to the internal Portal content model.

Figure 9-6 illustrates Portlet menus that use the context of the River Bend site, as described in Chapter 7, “Building the River Bend Web site with IBM Workplace Web Content Management” on page 253.



Figure 9-6 Portlet menu example within the context of the River Bend site

The Portlet menu nodes contain a URL that either points to the Portlet itself or to an external page. When the menu nodes refer to the Portlet, the URL includes an action string on which our Portlet code can react (for example, to display a different IBM Workplace Web Content Management content). To add menu entries to the Portal navigation tree, the Portlet must implement the MenuProvider interface.

Note: The implementation of Portlet menus and dynamic menus is described very well in the WebSphere Portal Information Center at:

<http://publib.boulder.ibm.com/pvc/wp/510/ent/en/InfoCenter/wps/wpsbsmenu.html>

For more information and examples for using portlet menus, see the Javadoc in the `com.ibm.wps.portlet.menu`, `com.ibm.wps.portlets.menu`, and `com.ibm.wps.portletservice.portletmenu` packages. The Javadoc includes some very detailed examples of how to implement portlet menus.

► Develop context navigation Portlets

Another way for the IBM Workplace Web Content Management Content to extend the navigation schema is via context navigation Portlets. In this case, the primary content can contain links with further reading or, based on a taxonomy, you query related content and display that inside of the context navigation Portlet, as shown in Figure 9-7.

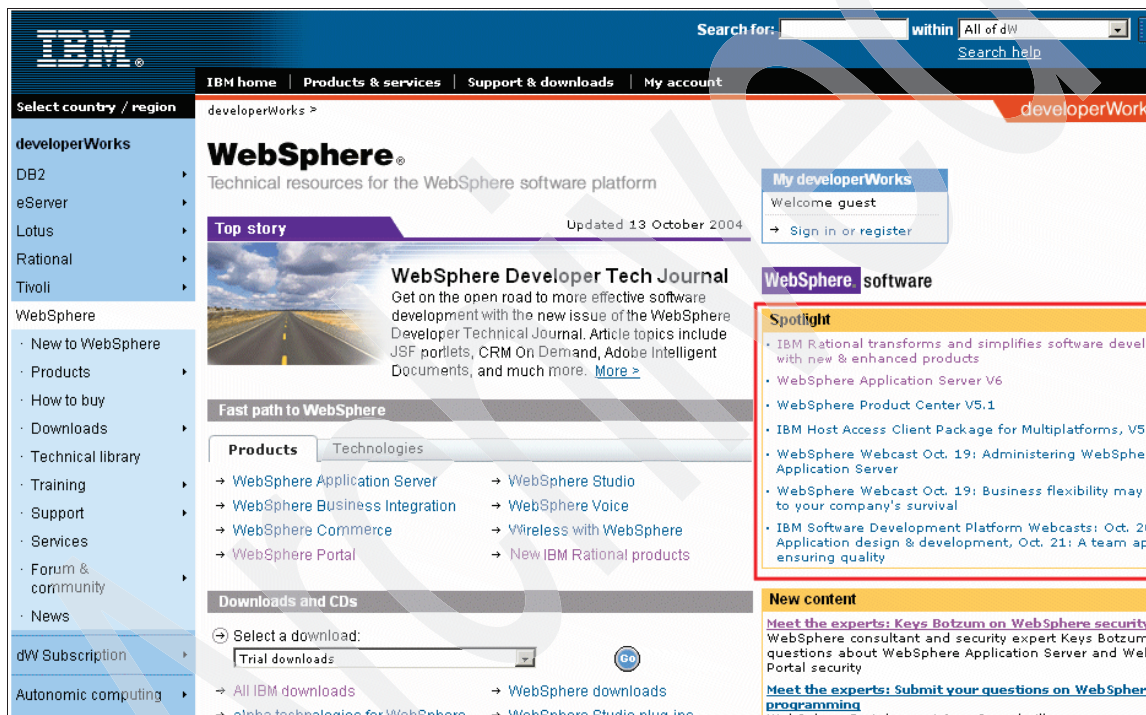


Figure 9-7 Example of a context navigation Portlet

Alternatively, Figure 9-8 illustrates a context navigator that uses the River Bend site, as described in Chapter 7, “Building the River Bend Web site with IBM Workplace Web Content Management” on page 253.



Figure 9-8 Contextual navigation within the context of the River Bend site

9.4 Beginning the implementation

Now that we have explained some of the concepts involved in the implementation, we discuss how to begin the implementation in this section.

9.4.1 Explicit mapping approach

With the explicit mapping approach, the administrator assigns one IBM Workplace Web Content Management content to every Portal page explicitly. Therefore, you need to place the IBM Workplace Web Content Management local rendering Portlet on a Portal page and configure it to display one content page. However, the local rendering Portlet does not support an edit mode. It only

has a configuration mode. Thus, you have to make copies of the Portlet so that it displays different content on different Portal pages.

Working in configuration mode

Figure 9-9 shows different variations of a Portlet as it is created, placed on a page, and accessed by users. Notice that the first two steps involve the use of persistent data.

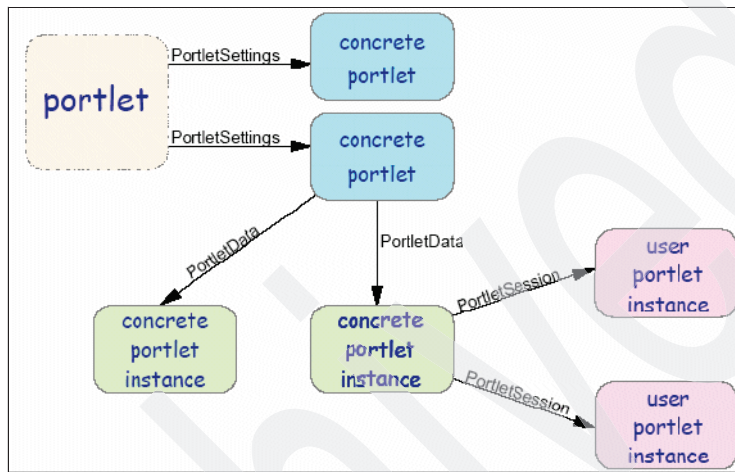


Figure 9-9 Variations of a portlet

► Config mode

A concrete parameterization of a Portlet object is referred to as a *concrete* Portlet. The settings of a concrete Portlets can change at any time (for example, when administrators modify Portlet settings using the config mode of a Portlet). To have a Portlet with a different parameterization, you have to make a copy of the Portlet or implement an edit mode.

► Edit mode

In edit mode, the Portlet is placed on a page by a user or an administrator. This action creates a concrete Portlet instance, which is a concrete Portlet parameterized by a single PortletData object. This configuration is done using the edit mode of the Portlet.

The scope of this configuration depends on the scope of the page on which the Portlet is placed:

- If an administrator puts a Portlet on a group page, then the edit configuration is shared for the group of users. This holds true for a group of users who have view access to the page. However, if users have edit access to the Portlet on a group page, then a new concrete Portlet instance is created for each user

that edits the Portlet. In this case, PortletData (Edit Mode) contains data for each user that edits the Portlet.

- If a concrete portlet is put on a user's page, the PortletData contains data for that user.

In our scenario, we assume that users have only view access to pages.

Conclusions for the explicit mapping approach

So, when considering the explicit mapping approach, do the following:

1. Make n -copies of the local rendering Portlet, where n is the number of Portal pages.
2. Implement a simple content display Portlet that has an edit mode and that stores the configuration information in the Portlet data.

For now, you can copy the local rendering Portlet, configure every instance, and then place them on the Portal pages. This approach might sound awkward. However, it allows you not just to use one local rendering Portlet per page but to have pages with different Portlets and page structure. So, you gain flexibility to leverage the Portal framework to build (for example, composite applications that go alongside with content from the IBM Workplace Web Content Management).

9.4.2 Making the implementation more functional

Now we discuss how to make the IBM Workplace Web Content Management and WebSphere Portal integration more functional. At the same time, we also want to reduce the administrative overhead that is involved with the use of the IBM Workplace Web Content Management local rendering Portlet. The solution is the implementation of a custom IBM Workplace Web Content Management display Portlet. (The actual implementation is discussed in detail in 9.5, "Implementing a custom display Portlet" on page 418.)

Before we talk about the implementation of the Portlet, let us first look at the authoring process. Having the administrator define what content is displayed on a Portal page might not be desirable. Usually, the content author is responsible for this action. To make this implementation work, we have to allow the content author to select the Portal page where content should be displayed. Thus, we need to extend the content creation template to allow the content author to choose from all available Portal pages. Internally, we store the unique-name of the page in case the page title changes.

This implementation can be done in several ways:

- ▶ You can maintain a list of all Portal pages manually, which implies an administrative overhead.
- ▶ You can use XMLAccess to query the Portal navigation. This query can be done periodically and might result in using a JavaScript library to populate the list.
- ▶ You can use the Model SPI (system programming interface) to get a list of all Portal pages.

In the following sections, we discuss more about the XMLAccess approach and the Model SPI approach.

Using the XML configuration interface (XMLAccess)

The XML configuration interface provides a batch processing interface for Portal configuration updates. It allows you to export or import an entire Portal configuration or parts of a configuration (for example, export specific pages to an XML file). You can then re-create the exported configuration from such a file on another Portal.

You access the XML configuration interface using a command line tool. This command line client is a small, separate program that connects to the server using an HTTP connection. It is, therefore, possible to configure the Portal remotely.

Tasks that you can perform with the XML configuration interface include:

- ▶ Exporting and importing the complete Portal configuration
- ▶ Exporting or creating sub-trees of the content-model
- ▶ Exporting or creating single page definitions
- ▶ Exporting or creating URL mappings

There are sample scripts available that ship with the Portal server in the `\WebSphere\PortalServer\doc\xml-samples` directory.

For example, `ExportSubTree.xml` allows you to export a sub-tree of the content hierarchy. The export result from this script is exactly what we would use to build the Portal page list.

Using the Model SPI

The Model SPI is an API that is used by Portlets and Portal applications to obtain read-only information about the Portal infrastructure. A number of models are available that can be queried, such as the Portal navigation model, content hierarchy, or the layout of a page.

Elements of the sub package `com.ibm.portal.content` define how content is represented in the Portal. The two main models of this package are the *ContentModel* and the *LayoutModel*. The *ContentModel* defines a tree structure for content elements (such as pages and labels). It is used to group these logically.

Note: You can find more information in the WebSphere Portal Information Center at:

http://publib.boulder.ibm.com/pvc/wp/510/ent/en/InfoCenter/wps/dgn_modelovw.html

9.5 Implementing a custom display Portlet

Unfortunately, the local rendering Portlet cannot use the *content to Portal page* assignments. At this point, you have to implement a custom IBM Workplace Web Content Management content display Portlet. For information about how to implement a custom content display Portlet, see 12.5.1, “Developing a simple Portlet” on page 501.

The custom Portlet can now use the Portal page assignment that the content authors created. It just has to decide what content document it should display as its primary content. This can be decided in the following ways:

- ▶ The Portlet queries the IBM Workplace Web Content Management backend for a document that contains the same unique Portal page name as the Portal page on which the Portlet is currently placed. You can use the Model SPI to get the unique name of the current Portal page.
- ▶ What if it finds more than one content document that is assigned to the current Portal page? In this case, you have to implement a rule that selects the primary content to be displayed (for example, based on document fields or publishing dates).
- ▶ You could also query for the primary content based on a taxonomy.

9.6 Managing links

So far our solution depends on the Portal navigation for the user to navigate the site. We have not yet thought about links inside of our content or within our

presentation templates. First, there are different types of links that need to be supported:

► **Inside of the content documents**

Typically, you want the content author to manage the links in the IBM Workplace Web Content Management. For example, while creating a new article, the author wants to link to a specific piece of content. When the user clicks the link, the content document is rendered on the same Portal page.

► **Across Portal pages**

Sometimes the IBM Workplace Web Content Management document to which you have a link lives on a different Portal page. In that case, the link should take us to the correct Portal page and should show the content document to which we linked.

► **To Portal pages (pages with no Content Portlets)**

In addition, we want to link to Portal pages that might have no content on them (for example, they host a feedback Portlet or other Portlet application).

Additionally, the user has a certain perspective and expectation of the functionality that links should support. For example:

- The user's navigational experience of a Web site — rather than an application — includes a *working back button*.
- Links should be able to be bookmarked, technically stable, and able to be sent in an e-mail.

Before we start to implement link management, let us first get a better understanding of how links work inside of a Portal framework.

9.6.1 Understanding links inside of the Portal framework

At first glance, the navigation inside of a Portal is comparable to the navigation of a simple Web page. This is true as long as the user navigates between Portal pages. However, the situation gets more complex as soon as the user interacts with a Portlet (for example, the local rendering Portlet). From the browser's point of view, interactions with a Portlet result in the loading of a new page. This observation is correct in that the URL changes. From a user's point of view, the user seems to be on the same page. Only the small area of the Portlet itself changes.

This usage pattern of the Portal becomes interesting as soon as the user presses, for example, the back button. Starting with WebSphere Portal Version 5.1, the Portal introduces a new concept called *navigational state handling*. This concept implies that users can now use the back button of the browser to navigate back through the recent history of the pages that they visited.

When a user navigates backwards using the back button, the Portal restores the views of pages that the user visited recently. This behavior affects the following aspects of the pages:

- ▶ The sequence by which the user navigated through the Portal and selected the pages.
- ▶ The expanded or collapsed state of the tree hierarchy that the navigation shows.
- ▶ The lateral scrolling position of the register tabs that users can use to switch between pages.
- ▶ The information of the user's default selection for a label. For example, this can be which page is selected for a label by default for that user. If the user selects a different page from that label, the Portal displays that page as the default page for that label from then on.
- ▶ The portlet window information that distinguishes between different instances of the same portlet on different pages.
- ▶ Modifications of the view or window state of a portlet, such as the minimized, maximized, normal, or default state. For example, if a user views a portlet in its default view state and then maximizes the portlet, clicking the back button returns the portlet to its previous state (that is, the portlet is displayed with its default size).
- ▶ The portlet modes, such as view, edit, or configure modes.
- ▶ The information as to whether a portlet is displayed in solo state, and which portlet is displayed in solo state.
- ▶ The information as to whether the Show tools icons are displayed, for example the wrench icon for configuring the portlet.
- ▶ If the portlet complies with the JSR 168 standard, the view state of the portlet is preserved as it is determined by the render parameters of the portlet.

For details about state changes to JSR 168 compliant portlets, refer to *Using the Back button with JSR 168 portlets* in the WebSphere Portal Information Center at:

http://publib.boulder.ibm.com/pvc/wp/510/ent/en/InfoCenter/wps/adbakbut.html#jsr168_portlets

Thus, using the back button returns to a previously visited page, and then clicking a link on that page that has not been followed before correctly preserves any settings. This behavior is important to remember for the link management discussion later on.

Based on the requirements, our links need to be stateful (that is, they must contain all the information to recreate the currently displayed Portal page, which is what the user sees in his browser).

9.6.2 Implementing the links using 2-Phase links

To implement stateful links, we use the concept of 2-Phase links. 2-Phase links are links to a Portal page that contain the following information:

- ▶ In Phase 1, links change to a Portal page.
- ▶ In Phase 2, the custom IBM Workplace Web Content Management Portlet picks up information about the target document that it displays from an URL parameter.

To implement links, it might seem best to use user-friendly URLs. However, this approach might be not the best option.

User friendly URLs

URL mapping allows Portal administrators to create constant user-friendly URLs and to map them to Portal pages. As administrators create the URLs, they define human readable names for them. These names can be easily remembered and are, therefore, more user friendly. The self-defined URLs can be published externally and, thereby, can be made available to Portal users. For example, for the River Bend site, we could create a user-defined URL called *products/coffee* for the page on which we advertise our different coffee brands. This URL can then be appended to the Portal prefix of our Portal site (for example <http://www.riverband.com/wps/portal/products/coffee>).

Clicking such a mapped URL from outside of the Portal takes the user to the desired Portal page. Users can also combine several mapped contexts into the representation of a full valid URL, type that full mapped URL into the address field of the browser, and thereby get to the Portal page.

Although this linking sounds like the ideal concept to link across Portal pages, there is a drawback to using user-friendly URLs.

Portal URLs encode navigational state information about the Portal (for example, the user's currently selected page) and about the Portlets on a page (for example, the window state of the portlet) in a serialized form. Encoding navigational state information in the URL is used by the Portal server to support use of the browser's back button. Be aware that using user-friendly URLs interferes with the navigational state handling of the Portal and your Portlet applications — especially JSR168 Portlets that make use of rendering parameters. All previous navigational state information is lost if you use user-friendly URLs. Therefore, you should use user-friendly URLs *only* as a starting point into the Portal.

You can find more information about this topic in *Portal navigation and Back button behavior* in the WebSphere Portal Information Center at:

<http://publib.boulder.ibm.com/pvc/wp/510/ent/en/InfoCenter/index.html>

This drawback is the reason why we implement link management differently. We use the Portal API to create working links properly. Technically, you can create links using the `CreateUrlCommand` class.

Note: The `CreateUrlCommand` class is the equivalent to the `URLGeneration` tag.

The `CreateUrlCommand` class allows you to create URLs to the following target resources:

- ▶ Pages or labels
- ▶ Portlets, with the option of adding render or action parameters (that's what we will need)
- ▶ Specific resources, such as an image file in the Portal
- ▶ Protected resources such that the URL is generated on the condition that the user has sufficient permissions

9.6.3 Implementing the link management

As a prerequisite for the link management, you also want to allow users to create links easily. Therefore, you need to implement a Link Helper Portlet. This Portlet allows the user to choose a content document from the IBM Workplace Web Content Management repository and then the Portlet creates the link. Here is how it works:

1. The Link Helper Portlet offers the user a list of all IBM Workplace Web Content Management content pages.
2. The IBM Workplace Web Content Management API queries for the linked to document and gets the unique-name of the Portal page to which the content is assigned.
3. The Link Helper Portlet shows the built link to the user, for example, `JavaScript:link2content([unique-name], [site are id])`.

The computed JavaScript links need to contain the information of the Portal page ([unique-name]) of the linked to document. Otherwise, you are not able to change the Portal page accordingly.

Important: Notice that we do not render the final links at this time. This rendering is done during the rendering phase of the custom IBM Workplace Web Content Management display Portlet. *This delay in rendering is necessary, because the Portal links contain the navigational state of the current user and are, therefore, user specific, which means that the links need to be created on a per user basis.* This can only be done at run time.

Thus, we use a JavaScript function as a place holder to allow us to defer the link resolution to the rendering phase. By using JavaScript, we are also able to build the link list during the publishing process. This is very important, because building the link list involves content rendering and HTML parsing. Because these tasks are processing intensive tasks, we do not want to do them when the content is displayed.

9.6.4 Building the links

Initially, we have to identify what links are contained in our content documents. The issue is that these links can be embedded in Rich Text components or in the presentation template. To build a comprehensive list with all links, we have to render the content and then parse out all occurrences of the `link2content()` JavaScript function. With that information, we can then build a link list of all linked to documents and store this list in an IBM Workplace Web Content Management document field.

To implement this functionality, we have to extend the Customizable Template Portlet, as discussed in 13.2, “Using the Customizable Template Portlet” on page 535. After the users saves the content, we implement some code that renders the content, that then does the parsing, and finally that builds and saves the link list.

The link list is then used during the rendering phase of our custom IBM Workplace Web Content Management Portlet. There, we create the Portal links for the individual user using the `CreateUrlCommand` class. This class allows us to build links across Portal pages and to add URL parameters as well. The resulting links are stored in a string that represents a JavaScript array. Why a string? Well, we are still in the controller code of our Portlet at this time. So, we wrap the JavaScript array into a string to be used later on the JSP page that renders the content. The JavaScript array has two dimensions and can be used similar to a `HashMap`. As the key, we use the combination of unique page name plus document-ID. The value is the resulting link from the `CreateUrlCommand`. The string that contains the JavaScript array is then passed into the `view.jsp` and allows the `link2content` JavaScript function to resolve the links easily for us.

Important: The URL parameter is very important, because it allows us to implement phase 2 of our link management. We are going to add the document-id of the linked to document as an URL parameter.

Example 9-1 from the view.jsp shows how this implementation works.

Example 9-1 Sample from the view.jsp

```
<script language="JavaScript" type="text/javascript">
<!--
<%=portletRequest.getAttribute(WXMCustomPortlet.JS_LINK_ARRAY)%>
function link2content(unique_page_name, document_id)
{window.location.href=links[unique_page_name+ document_id];
}
//-->
</script>
```

Note: Notice that we have chosen this approach for performance reasons. Alternatively, you could render the content presentation at run time and replace the links inside of the HTML. This approach, however, is not good for the performance of the Portlet. Thus, we recommend that you not do it.

Important: It is best practice to do all the processing intense operations during the save and publishing cycle to ensure that you get optimal performance at run time.

9.6.5 Walking through of the linking process

This section contains a sample scenario that walks you through the link creation and link execution processes.

Scenario

We have two IBM Workplace Web Content Management documents. One is a news article, and the other is a product information (perhaps for a new coffee brand). The news article's primary Portal page is the news section, and the product information document belongs to the product section of the Portal.

The content author now wants to create a link to the product information from within the news article announcing the new product. Using the Link Helper Portlet, the content author chooses the product information as the destination document for the link. Internally, the Link Helper Portlet queries for the product information document and gets the unique-name of the assigned Portal page. It

then builds the JavaScript link. The computed URL might look similar to the following:

```
JavaScript: link2content([unique portal page name], [Workplace Web Content Management document id])
```

When the content is saved, the unique Portal page name and IBM Workplace Web Content Management document ID are extracted and saved into a link list field. At run time, this link list is then used to create the user specific Portal links. If the user clicks the JavaScript link, phase one of the link management begins, and the Portal changes to the Portal page of the product area. Then, phase two is activated, and the custom IBM Workplace Web Content Management Portlet picks up the URL parameter and displays the content document to which we have linked.

9.7 Using multilingual and personalized content display

Earlier in this chapter, we discussed how a Portlet finds the primary content to display when a user navigates to a page and the Portlet has to decide what IBM Workplace Web Content Management content to display. We achieved this by querying the IBM Workplace Web Content Management backend for the unique-name of the Portal page. This concept worked well, but has potential for more.

Imagine that you have to write product documentation for a global corporation. You are responsible for the English versions of the document, but there are other colleagues who would do documentation in other languages. In addition, every product documentation uses different tailoring for sales, management, and technical audiences. How do you organize all these versions?

Well, the easiest way is to use folders in a shared file system. For each product, you create a folder to hold all the different versions that are available for that product. To keep the languages organized, you use descriptive extensions in the file names (for example _DE for German, _EN for English, and so forth).

So far, this approach sounds reasonably easy to manage. However, how do we organize the content in IBM Workplace Web Content Management. You can use site areas to logically organize language versions and personalized (tailored) versions in a simple way by using the site areas as a folder structure. Thus, every content page has its own site area and shares that site area with corresponding (language or personalized) content versions.

The custom Portlet still uses the unique-name to query the content. However, now this content might exist in alternative versions. From the content document, we can get the information regarding its corresponding site area. Then, from the

site area, we get access to all content documents (which are children of this site area). Finally, we can implement rules that decide what content should be displayed to the user, as follows:

- The user has defined German as his preferred language in the browser. The custom Portlet calls `request.getLocale` to get the user's language and uses this information to iterate through all the IBM Workplace Web Content Management content documents in the site area. The Portlet finds the site area that ends with the appropriate extension for the German language content. Alternatively, the Portlet can also get the user's language preference from the user's profile data or from `request.getHeader("Accept-Language")` = directly from the browser.

What happens if the document is not available in the user's preferred language? In that case, the custom IBM Workplace Web Content Management Portlet displays the document in the site's default language.

Alternatively, you could define that the Portlet always display content in the site's default language. If alternative languages are available, the Portal could offer a link to switch the language display. The information that an alternative language was requested then can be passed as a URL parameter to the Portlet.

- The user belongs to a the management user group. You might want to show different content on the Home page for members of the management team than for regular staff. In this case, you can take the same approach as you did for a user with a language preference other than the default language, except that you match the groups to which the user belongs with document attributes. Then, compare this information with an audience attribute inside of the IBM Workplace Web Content Management content document. You can use the PUMA SPI to access the profiles of a Portal user or group.

PUMA SPI

The PUMA SPI provides interfaces for accessing the profiles of a Portal user or group. It is used to find, create, modify, and delete users and groups. Also, profile information about the currently logged in user can be retrieved. You can find more information about this in *PUMA SPI overview* in the WebSphere Portal Information Center at:

<http://publib.boulder.ibm.com/pvc/wp/510/ent/en/InfoCenter/wps/wspuma.html>

The PUMA SPI is a very simple but powerful concept that can be used for many different purposes.

Note: Do *not* query the IBM Workplace Web Content Management backend every time a user makes a request. Doing so is usually not necessary and is not good for performance. Instead, consider a caching strategy, as described in Chapter 8, “Caching and pre-rendering considerations” on page 347.

If you are concerned about caching old values, implement a servlet that can delete entries from the cache. You can use the IBM Workplace Web Content Management document ID as the key. Then, in your input template, embed an image link to your servlet and pass it the document ID as an input parameter.

9.7.1 Ensuring link integrity

For any link management implementation, it is also important to think about link integrity. What happens when a linked document gets deleted or expires? Well, then the links in all documents that link to the deleted document become invalid. Therefore, it is necessary to check whether a document has references to it before deleting or expiring the document. This is a functionality that is usually implemented in the IBM Workplace Web Content Management engine.

Because we have implemented the link management ourselves, we need to implement some link checking mechanism on our own also. Remember that during the save operation of an IBM Workplace Web Content Management document, we build a link list with references to all documents to which the content links. We can now use this link list to build a link checking process. We can then run this link checking on a periodic schedule to check for broken links. The result might be a list of documents that contain broken links that are displayed in an administrative Portlet. (A detailed description about how to implement link checking is outside of the scope of this book.)

9.7.2 Other linking functionalities

Other functionalities that might be useful for a custom IBM Workplace Web Content Management Portlet include:

- Linking to a non-content page

Sometimes, you might want to link to a Portal page that hosts just a Portlet or Portal application but that contains no content. The easiest way for content authors to link to a non-content page is to do the following:

- a. Create an empty document in IBM Workplace Web Content Management.
- b. Assign the empty document to a Portal page (Portal unique name).
- c. Link to the document.

Note: Even if there is no Content Portlet on the page, the URLGeneration class changes the Portal page for us. We do not recommend that you use a user-friendly URL. Refer to “User friendly URLs” on page 421 for more information.

- Making the Content Portlet hide if no URL parameter or Portlet Data default is present

This option is useful if you want another Portlet to control the behavior of the custom IBM Workplace Web Content Management Portlet (for example, a product finder).

Go to <http://www.aofoundation.org> and click **Innovations**. You see a skeleton that functions as a content navigator. If you click a bone segment, it displays product thumbnails from the IBM Workplace Web Content Management repository. If you then click a product, the hidden content Portlet becomes visible and the skeleton hides instead.

You can implement this behavior by configuring the custom IBM Workplace Web Content Management Portlet to display no default content. Instead, it looks for the URL document-id parameter. When the user clicks a product thumbnail, the skeleton Portlet places the document ID into the URL. The custom IBM Workplace Web Content Management Portlet then picks up this information and displays the desired product document. The hiding of the skeleton Portlet is implemented in the same way — just the other way around. The skeletonHide=true parameter is present in the URL.

You cannot use Portlet messaging to implement this behavior because you cannot reproduce the same result from a bookmark or link in an e-mail.

Search functionality for IBM Workplace Web Content Management

Managing your Web content in a structured environment such as IBM Workplace Web Content Management is a good step towards proper management of the content. However, the content is not very helpful if visitors cannot locate the information that they need. This chapter discusses the different ways that you can implement the search functionality so that visitors can find content easily.

This chapter includes the following sections:

- ▶ Providing multiple techniques to search content
- ▶ Using the IBM Workplace Web Content Management search module
- ▶ Using the IBM Workplace Web Content Management API
- ▶ Using the WebSphere Portal search engine
- ▶ Using external and third-party search products

10.1 Providing multiple techniques to search content

If your site is large or if it contains a wide variety of content, visitors expect the site to provide one or more techniques for searching the content. IBM Workplace Web Content Management 5.1 provides multiple techniques for searching its content repository, which include the following:

- ▶ IBM Workplace Web Content Management integrated Search Module
- ▶ IBM Workplace Web Content Management API
- ▶ WebSphere Portal Search
- ▶ Third-party search products

Each search option has its advantages and disadvantages. In many situations you need a combination of these capabilities to meet your specific requirements.

Tip: As you read through this chapter, do not think of each option as exclusive of the others. Look at ways that each is suited to solve a specific problem.

10.2 Using the IBM Workplace Web Content Management search module

You can use the IBM Workplace Web Content Management search module to create a basic search function on content objects, keywords, and categories that are stored in an IBM Workplace Web Content Management content repository. In addition, you can combine queries with and/or operators to create a more specific query.

The search module's functionality is independent from the data repository type that you use to store your IBM Workplace Web Content Management objects. Thus, no matter which content repository you choose (Cloudscape, DB2, or DB2 Content Manager), the search results are the same.

At a high level, here is how the IBM Workplace Web Content Management search module works:

1. The site developer creates a search form and formats how the search results appear using HTML. The developer saves the HTML code in a combination of IBM Workplace Web Content Management components, such as presentation templates, HTML components, and content items.
2. A site visitor performs a search by submitting one or more words in the search form. The search module compares the words against the search index and lists content pages that match the search words as search results.

10.2.1 Security and the search module

The search module takes into account the security settings on content during a search. A search returns only the content that the requestor is allowed to see. Searching external content (for example, content in DB2 Content Manager) is not supported unless you include the external content in your Web page using the Federated Content Component.

10.2.2 The search module index

When you enable the search module, the search module creates a search index. This index is refreshed at regular intervals based on the settings in the `<WP_Root>/wcm/config/aptrixsearch.properties` configuration file. The configuration file contains a variety of settings, including four that directly effect the search index, as shown bold in Example 10-1.

Example 10-1 Search Index related settings from `aptrixsearch.properties`

```
# the directory where IBM Workplace Web Content Management search
# should store its hit indexes
search.directory=D:/WEBSPH~1/PORTAL~1/wcm/ilwwcm/search

# the hit index refresh interval in minutes, optional
refresh.interval=120

# the hit index refresh times, optional
# note that these times must match the format of your locale.
#refresh.time=18:00:00;04:00:00

# the username to use when indexing ilwwcm (optional)
# choices are Anonymous, Administrator or a specific user or group common name.
ajpepageserver.user.name=wpsadmin
```

The `refresh.interval` and `refresh.time` parameters control how often the index is updated. Content does not appear in a search result until it is added to the index. However, the indexing process can effect the performance of your IBM Workplace Web Content Management server. The more often your content changes, the more frequently you should index.

The IBM Workplace Web Content Management search engine index is stored on the file system of the IBM Workplace Web Content Management server at `<WP_Root>/wcm/ilwwcm/search/hitIndex.dat`.

Note: The search application returns no-results until it creates the index.

If necessary, you can limit what text is indexed by creating a search filter using a presentation template. For example, you can exclude text from navigators and menus.

For complete details about configuring IBM Workplace Web Content Management and building the necessary assets to use the IBM Workplace Web Content Management search module see Chapter 7, “Building the River Bend Web site with IBM Workplace Web Content Management” on page 253, and in particular, 7.24.3, “Modifying a presentation template for the Search page” on page 329.

10.3 Using the IBM Workplace Web Content Management API

The IBM Workplace Web Content Management API provides access to the content within the content repository. From a technical perspective, it is feasible to use this API to build a search engine for IBM Workplace Web Content Management content. However, one should not embark on a process simply because it is possible. There are issues that you need to address if you build your search engine using this method, including the following:

- ▶ Consider the caching method that you use and the performance of the site. A large site requires a very strong caching algorithm to give the search engine a reasonable performance.
- ▶ Consider the security that the site uses. You need to ensure that content is visible only to the appropriate people.

If the standard IBM Workplace Web Content Management search module does not meet your needs, you can build your own search engine. However, due to the level of effort that is required to write a good search engine, we do not recommend this approach. Instead, we recommend that you look at existing search products that can search your IBM Workplace Web Content Management content repository.

10.4 Using the WebSphere Portal search engine

The WebSphere Portal search engine is made up of several components:

- ▶ Administrative Portlets that allow you to define search parameters, for example, the search scope.
- ▶ A crawler component that works similar to an Internet spider and that indexes the site.
- ▶ Search collections that represent index information and allow searching capabilities.
- ▶ Portlets that allow a user to search inside of a search collection.

In configuring the WebSphere Portal search engine, you can specify a starting point for the crawler. The crawler then follows links from one page to the next, so that it can gather information and learn about a Web site infinitely. The WebSphere Portal search engine crawler then stores the gathered information inside a collection for searching. This crawl can happen to two types of content sources:

- ▶ WebSphere Portal sites
- ▶ Regular Web sites

The difference between these two types of content is the way that the crawler follows links inside of the sites.

10.4.1 Crawling a WebSphere Portal site

When crawling a WebSphere Portal site, the crawler fetches and indexes all pages with Portlets to which it has access rights. This way, you enable visitors to search those pages. You can define which Portlets on which pages are searchable by granting the required access permissions to the crawler user.

To help the crawler index a WebSphere Portal site in an efficient way, the Portal provides the crawler with a *seed-list* page. This page contains information about all Portal pages and Portlets that the crawler should index.

Technical background to the seed-list indexing method

If you browse the content model of your Portal, you find a Portal page named `wp.search.seedlist`. This page hosts a Portlet named Manage Seed List. The seed-list Portlet builds an XML structure that contains references to all Portal pages and Portlets. To build the XML structure, the seed-list Portlet traverses the content model for the Portal. The WebSphere Portal search engine crawler then uses this XML data to index the Portal site.

The XML structure looks similar to that shown in Figure 10-1.

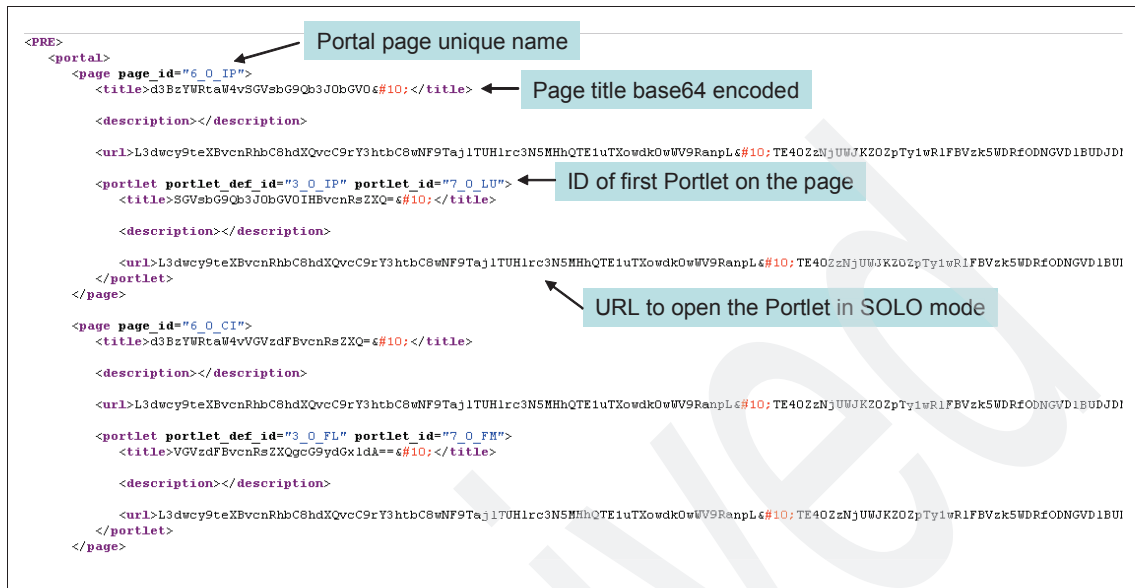


Figure 10-1 XML structure returned by the seed-list Portlet

All the information that is rendered by the seed-list Portlet is base64 encoded. The XML structure contains URLs to all Portal pages as well as URLs to display every single Portlet in *SOLO* mode, meaning without the surrounding navigation. The WebSphere Portal search engine uses the title and description information as metadata in the indexing process. The WebSphere Portal search engine crawler now traverses the XML tree, calls all the links, and indexes the resulting pages.

A drawback to the seed-list indexing method is that the crawler does not follow any links inside of a Portlet because the links could be actions potentially. By following the links, the crawler would execute these actions, which could result in unexpected content creation or other unwanted data processing. In WebSphere Portal 5.1, the crawler cannot distinguish between navigational and action links. Therefore, it does not follow links inside of a Portlet at all. You should keep this in mind for your IBM Workplace Web Content Management content indexing.

In the case where you want to index a regular Web site, you can provide the crawler with a starting URL and with information about how deep you want to index the site. The depth is defined by the Levels of links to follow parameter. This parameter tells the crawler at what level to stop indexing. If you index regular Web sites, you should also consider including a filter that tells the

crawler *not* to follow links that are outside of the current domain. Otherwise, it might find a link to an external site and might start to index that site.

10.4.2 Considerations for using the WebSphere Portal search engine

Before using the WebSphere Portal search engine to search IBM Workplace Web Content Management content inside of our WebSphere Portal site, you should consider the implications of using this method.

For example, imagine that you have created a Portal page that holds an IBM Workplace Web Content Management navigation portlet and a display Portlet, similar to that shown in Figure 10-2.



Figure 10-2 Portal page with an IBM Workplace Web Content Management navigation and display Portlet

In this example, the WebSphere Portal search engine uses the information from the seed-list Portlet to index this Portal page. The seed-list XML structure contains two references for this portal page — one for each Portlet on the page.

Each entry allows the crawler to open each Portlet in SOLO mode and to index the resulting HTML. The result is only two entries in the search collection. As title,

it uses the title of the Portlet, not the IBM Workplace Web Content Management document title. The crawler also does not follow links inside of the Portlets. So, there is no way for the crawler to index all the IBM Workplace Web Content Management content documents that are linked in the IBM Workplace Web Content Management navigator.

10.4.3 Crawling a regular Web site

The next logical option to index the IBM Workplace Web Content Management content is to have the crawler index the Portal as a regular Web site. In this mode, the crawler actually indexes all of the IBM Workplace Web Content Management documents, with the following limitations:

- ▶ All IBM Workplace Web Content Management documents that are displayed on the same Portal page have the same metadata. Thus, all documents appear under the name of the Portal page inside the index. Why? The WebSphere Portal theme takes the title of the Portal page and renders it as a <title> tag in the head section of the HTML. This tag is what the crawler picks up and uses as the page title for the index. Because all of the content documents are shown on the same page, *they all appear under the same name*, presenting a significant limitation of this search method.
- ▶ The crawler follows all links. However, it cannot distinguish between navigational links that display additional content and Portlet actions. This behavior can result in unexpected actions inside Portlets. You need to be very careful if you choose this indexing method for Portal sites.

10.4.4 Crawling using a custom method

This section discusses ways to overcome the limitations of these search methods. The idea is to build a custom seed-list Portlet for IBM Workplace Web Content Management content. In this section, we call this Portlet the *Site-Summary Portlet*. The Site-Summary Portlet uses the IBM Workplace Web Content Management API to build a list of all IBM Workplace Web Content Management content documents of the site. In addition, it manipulates the metadata that is picked up by the WebSphere Portal search engine crawler. This method allows us to pass the IBM Workplace Web Content Management document title and other metadata to the crawler.

Building the Portlet

To creating the custom crawling approach within the portlet, follow these steps:

1. Use the IBM Workplace Web Content Management API to build a list of all IBM Workplace Web Content Management documents, as described in Chapter 12, “Working with the application programming interface” on page 469.
2. Create a link-list to all the IBM Workplace Web Content Management content documents by leveraging the 2-Phase linking concept, as described in 9.6.2, “Implementing the links using 2-Phase links” on page 421. This method is based on the Portal's URLGeneration tag. It allows us to create links to a Portal page and to add additional information to the URL.

We use the 2-Phase linking as follows:

- a. The URLGeneration tag creates a link to the portal page that hosts the IBM Workplace Web Content Management local rendering Portlet.
- b. We add the information about what IBM Workplace Web Content Management document that we want to display to the URL, as shown in Example 10-2.

Example 10-2 Code that shows the URL generation

```
<wps:urlGeneration contentNode="[PORTAL_UNIQUE_PAGE_NAME]">
  <wps:urlParam name=WCM_GLOBAL_CONTEXT"
value="[WCM_CONTEXT_ROOT_PATH]+[WCM_SERVLET_PATH]+[CONTENT_PATH]"/>
  <wps:urlParam name="TITLE" value="[IWWCM_DOCUMENT_TITLE]"/>
  <% genURL = wpsURL; %>
  <a href = "<% wpsURL.write(out); %>">[LINK_LABEL]</a>
</wps:urlGeneration>
```

- c. This information is passed in the WCM_GLOBAL_CONTEXT URL parameter that tells the local rendering Portlet what IBM Workplace Web Content Management document to display.

Note: Make sure that your local rendering Portlet is configured to listen to the passed URL parameters. In the config-mode specify the following:

Receive Links From: 'Other portlets and this portlet'

The resulting links look similar to this:

...?WCM_GLOBAL_CONTEXT=/wps/wcm/connect/Products/Coffee/Columbia Coffee

- d. We add a TITLE parameter that passes the IBM Workplace Web Content Management document title to the portal theme.

3. To make use of the TITLE parameter in the URL, change the head.jsp of the Portal and pick up the TITLE parameter to render it to the HTML head as the <title> tag, as shown in Example 10-3.

Example 10-3 Page title contained in the document title

```
<head>
  <title><%request.getParameter["TITLE"]%>/title>
  ...
</head>
```

Then, if the WebSphere Portal search engine crawler follows the link, the page title is the IBM Workplace Web Content Management document title.

If the WebSphere Portal search engine crawler follows the links in the link-list, the crawler goes to the Portal page with the IBM Workplace Web Content Management rendering Portlet. In addition, the URL contains the ID parameter that tells the IBM Workplace Web Content Management rendering Portlet what document it should display.

4. Create a Portal page and place the Site-Summary Portlet on it. To allow the WebSphere Portal search engine crawler to access the page easily, create a user-friendly URL for the page.
5. Add a content source to the search collection that uses the following parameters:
 - a. Crawl source type: Web site
 - b. Collect documents linked from this URL: [your user-friendly URL]
 - c. Levels of links to follow: 1 level
6. Run the WebSphere Portal search engine crawler to build the index.

If you now use the Search Center or Search & Browse Portlet to search the IBM Workplace Web Content Management search collection, you find that the search result shows the proper IBM Workplace Web Content Management document titles. If you click a link, the link takes you to the correct Portal page and also opens the IBM Workplace Web Content Management document that you requested.

10.4.5 Using multiple Portal pages to display content

If you use multiple Portal pages to display IBM Workplace Web Content Management content, you have to implement some additional logic inside the Site-Summary Portlet. In this case, the following scenarios are possible:

- ▶ You have multiple Portal pages that host the IBM Workplace Web Content Management navigation and local rendering Portlet.

In this case, you need to build a map that tells the Site-Summary Portlet which Portal pages displays which IBM Workplace Web Content Management site area. Table 10-1 shows an example.

Table 10-1 Example map for the Site-Summary Portlet

This site area	Maps to this Portal page
Riverbend/Products	Products
Riverbend/Tea	Tea

The map information is then used together with the `URLGeneration` tag to create links to the corresponding Portal page.

- ▶ You have used the explicit or implicit mapping schema approach from 9.3, “Bridging the gap” on page 408.

In this case, you just re-use the mapping code from the custom IBM Workplace Web Content Management display Portlet to calculate the corresponding Portal page for your IBM Workplace Web Content Management content.

10.4.6 Considering outside search engines

You should also consider how external search engines, such as Google, locate your IBM Workplace Web Content Management content. To make it easy for external search engines to index your site as well, you can present the Site-Summary Portlet to them for indexing too. The technique to achieve this is called *cloaking*. Cloaking is the process of delivering one version of a page to one user and a different version of a page to another user, such as a search engine. There are three ways of cloaking:

- ▶ IP delivery, where the IP addresses of spiders are recognized at the server and handled accordingly.
- ▶ User-Agent delivery, where the spiders' User-Agents are recognized
- ▶ A combination of IP delivery and User-Agent delivery.

A simple way to implement cloaking is to use the `mod_rewrite` module on the IBM HTTP server. The re-write module can look for the User-Agent and can re-direct Google to the Site-Summary Portal page.

10.5 Using external and third-party search products

This section highlights other tools and technologies for searching IBM Workplace Web Content Management content beyond the built-in IBM Workplace Web Content Management Search feature and the WebSphere Portal search engine. While IBM Workplace Web Content Management and WebSphere Portal provide search capabilities within their standard features, IBM and other vendors offer search specific products with more comprehensive features and functions.

Third-party search products include:

- ▶ IBM WebSphere Information Integrator OmniFind™ Edition
http://www-306.ibm.com/software/data/integration/db2ii/editions_womnifind.html
- ▶ Apache Lucene
<http://lucene.apache.org/>
- ▶ Verity Ultraseek
<http://www.verity.com/products/ultraseek/>

Each search product has different features, functionality, support, IBM Workplace Web Content Management integration, and cost. Furthermore, every IBM Workplace Web Content Management project has a different set of goals and requirements. Which third-party product you choose is based upon the specific requirements for your project and organization.

10.5.1 Key considerations when select a third-party search product

While there are numerous factors that can enter into a decision on which product best suits your needs, most Web content project have a common set of core considerations, as follows:

► How content is delivered

One of the most important issues that you need to analyze is where you will deliver your content:

- Stand-alone Web site (via an IBM Workplace Web Content Management rendering server)
- WebSphere Portal (via rendering portlets or custom portlets)
- Stand-alone Web site **and** WebSphere Portal

If you are delivering content in a single environment, the goal would be to find a search engine that performs best for that scenario. If you want to deliver content both stand-alone and within WebSphere Portal, then you have a more complicated search requirement. Some search products work well in all scenarios, while others work well in less complicated scenarios.

► Return on investment (ROI)

At some level, all business decisions have to look at cost. There is a cost to implementing a search engine, and there is a cost associated with people not finding the content for which they are looking on your site. When considering which search engine you need for your environment, consider the following:

– Cost

The purchase price of the software can be significant. Most products require some form of maintenance for ongoing support and upgrades. Make sure that you understand all of the costs that are associated with the initial purchase and the ongoing maintenance of a given solution.

– Time to implement

When you purchase a search product, you need to install, configure, and implement the solution to work in your environment. This time cost is often difficult to define, but underestimating this effort can lead to flawed ROI numbers.

When you take these (and other) factors in total, it is possible that the ROI is not high enough to invest in a third-party search product.

Note: For further details on IBM WebSphere Information Integrator OmniFind Edition, see *WebSphere Information Integrator OmniFind Edition: Fast Track Implementation*, SG24-6697, which is available at:

<http://publib-b.boulder.ibm.com/redpieces/abstracts/sg246697.html?Open>

Personalization rules for IBM Workplace Web Content Management

In WebSphere Portal implements rule-based content delivery through a feature known as *Personalization*. Personalization lets Web sites customize content automatically for each user. Personalization can recognize a specific user based on a profile. It can also determine characteristics of a user based on previous purchases, products, or pages viewed and then select content that is appropriate for that profile.

This chapter focuses on the new personalization integration with IBM Workplace Web Content Management that is available in WebSphere Portal 5.1.01. This new feature allows Personalization rules to be created based on the basic attributes of IBM Workplace Web Content Management content. It includes the following sections:

- ▶ Overview of the personalization feature
- ▶ Installing and configuring WebSphere Portal 5.1.01
- ▶ Enabling WebSphere Portal personalization

Important: This functionality requires WebSphere Portal 5.1.0.1. Although this chapter describes how to do update to this version, you might also want to refer to the following:

[http://www-12.lotus.com/ldd/doc/uafiles.nsf/docs/wcm5101/\\$File/wcm-5-1-0-1-releasenotes.htm](http://www-12.lotus.com/ldd/doc/uafiles.nsf/docs/wcm5101/$File/wcm-5-1-0-1-releasenotes.htm)

11.1 Overview of the personalization feature

This section summarizes some basic terminology and concepts that are needed to implement the basic scenario in this chapter. You can find complete information about the rich set of capabilities that the new personalization feature provide in the WebSphere Portal Information Center, which is available at:

<http://www-306.ibm.com/software/genservers/portal/library/index.html>

Figure 11-1 illustrates the components that we use in our scenario.

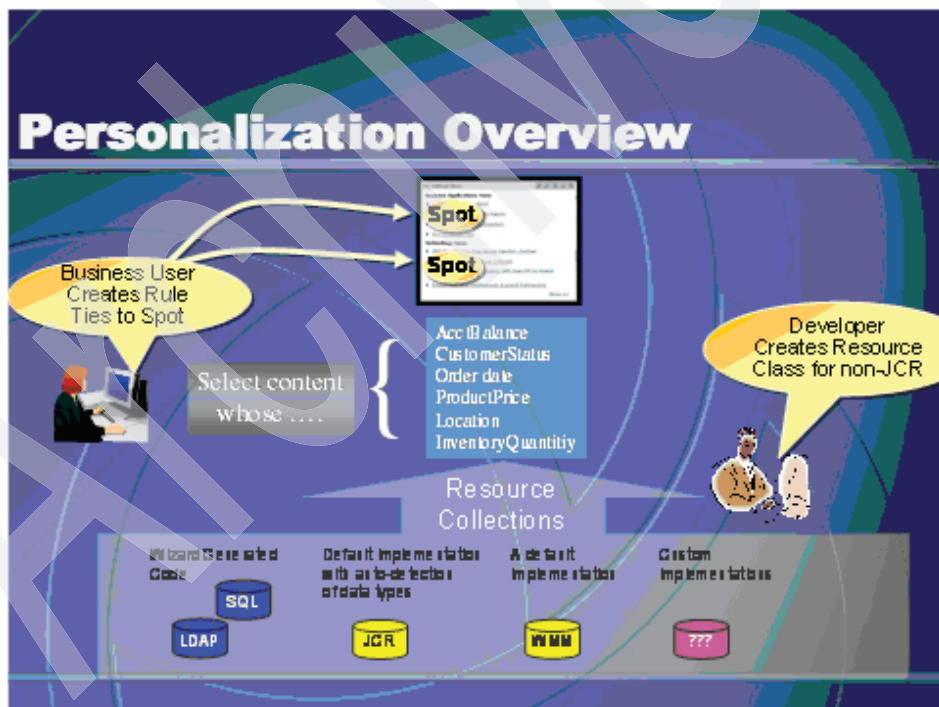


Figure 11-1 Overview of the personalization features

A Portlet can have a designated *spot* for rule-based content. For each spot, a rule is selected. All rules are created based on the resource attribute definition. These attributes are defined in a *resource collection*. The resource collection is a set of Java classes that retrieve content that is requested by the rule from the defined repository source. This backend source might be LDAP or SQL databases, the Portal Java Content Repository (JCR), WebSphere Member Manager, or any other custom collection that is coded for the personalization API. The JCR resource collection is provided with WebSphere Portal as well as WebSphere Member Manager. There are wizards available in WebSphere Application Developer and RAD6 that guide developers in creating classes for other LDAP and relational database resources. A guide of how to create these resources is described fully in the WebSphere Portal Information Center.

For our scenario, we use the personalization sample resources and the JCR. We do not create any classes using the wizard.

Three types of rules are used to target select content to a certain group of users:

- ▶ The *profile* rule can be made on any user resource. A user's profile is best described as a named group that is based on user characteristics. For example, a user is profiled as a *premium* user if the amount of the user's mortgage is over a certain amount. Another example could be that if a user's order amount is over a certain amount, then that user is profiled as a *gold* user. In addition, an employee could be profiled as non-management if the employee's `isManager` attribute is set to `false`.
- ▶ A *select* rule chooses content based on the attributes of the content. For example, the `GoldOffers` rule selects products whose retail price is greater than a certain amount. With IBM Workplace Web Content Management content, you can also select content with a specific category. For example, a `NonManager` rule gets content that is created with the `HRNews` template that has a category of non-manager.
- ▶ A *binding* rule ties the profile and select rules together. So, premium users see `PremiumOffers`, gold customers see `GoldOffers`, and non-manager employees see `NonManagement` news. You do not have to coordinate the naming of the profile with the select rule name. However, this convention helps when creating the binding rule.

Any of these rules can be tied to a content spot. To enable your environment to construct these rules, you need to install and configure personalization and IBM Workplace Web Content Management.

11.2 Installing and configuring WebSphere Portal 5.1.01

To upgrade an existing WebSphere Portal 5.1 system, you use the Portal Update Installer and the fix pack. Download the WebSphere Portal Version 5.1.0.1 cumulative fix and the WebSphere Portal update installer from the following Web site:

<http://www.ibm.com/software/genservers/portal/support/>

The following files are required:

- ▶ Cumulative fix file: WP_PTF_5101.jar
- ▶ WebSphere Portal update installer: PortalUpdateInstaller.zip

Use the instructions located in the WebSphere Portal Information Center to install this fix:

<http://publib.boulder.ibm.com/pvc/wp/510/ent/en/InfoCenter/index.html>

Search on 5.1.0.1 to find the information that you need.

11.3 Enabling WebSphere Portal personalization

If you want to use personalization rules on IBM Workplace Web Content Management content, then you must configure IBM Workplace Web Content Management to *shadow* some of its content in the Java content repository. This repository is based on the JSR 170 standard and is the base repository direction for IBM Enterprise Content Management.

11.3.1 Changing the connect.cfg file

To write rules for IBM Workplace Web Content Management content with WebSphere Portal personalization, you must enable a JCR shadowing within Web Content Management. To do this, follow these steps:

1. Stop the IBM Workplace Web Content Management Application.
2. Edit the connect.cfg file that is located in the /PortalServer/wcm/config directory.
3. Find the following line (located before the <LogManager> section), and change the value to true.

```
JCRRepository value=true />
```

4. Restart IBM Workplace Web Content Management.

Note: If LDAP has been enabled, you need to edit the following values within the <JCRConnector> element in the connect.cfg file to include the following values:

```
<DefaultUsername value="uid=wpsadmin,o=default organization"/>
<DefaultPassword value="wpsadmin"/>
```

You can find further information about LDAP settings in the LDAP user registry topic in the WebSphere Portal Information Center.

Note: In i5/OS® installations, IBM Workplace Web Content Management files are located in the WebSphere Portal user root directory, not in the /PortalServer/wcm/ directory. For example, for WebSphere Application Server 5.1.1.3, you can find the files in this directory:

```
/QIBM/UserData/WEBAS51/Base/instance name/PortalServer51
```

For WebSphere Application Server 6, you can find the files in this directory:

```
/QIBM/UserData/WebSphere/AppServer/V6/Base/profiles/
instance name/PortalServer51
```

11.3.2 Using personalization rules on IBM Workplace Web Content Management Content

To write rules you need to define a resource collection. To create a resource collection, do the following:

1. Log in to WebSphere Portal.
2. Navigate to the Personalization Workspace.
3. Click **New** → **Resource Collection**.
4. Enter a name for the collection (for example, *IBM Workplace Web Content Management Resource Collection*).
5. Specify the following settings under Standard Resource Collection, as shown in Figure 11-2 on page 448:
 - Resource Class: com.aptrix.pluto.pzn.resources.WCMResource
 - Manager Class: com.aptrix.pluto.pzn.resources.WCMResourceManager
 - Domain Class: com.aptrix.pluto.pzn.resources.WCMResourceManager

Be sure that there are no trailing spaces in these fields.

The screenshot shows the 'Personalization Editor' window with tabs for 'Workspace', 'Personalized List', and 'Demo'. The 'Edit Collection *' tab is active, showing 'Document Info' for a 'WCM Resource Collection'. The collection is located in 'PznWCM' and was last modified on May 11. The description is 'WCM Resource Collection'. Under 'Collection Type', 'Standard Resource Collection' is selected. The 'Node Type' is 'Portal Category'. The 'Resource Class', 'Manager Class', and 'Domain Class' are all set to 'com.aptrix.pluto.pzn.resources.WCMResourceManager'. The 'Translator Class' is empty. Under 'Locate the current resource:', 'No current resource' is selected. 'Save' and 'Cancel' buttons are at the bottom.

Figure 11-2 Specify settings for the Standard Resource Collection

6. Select **Save**.

11.3.3 Populating the JCR with existing content information

Since the integration of IBM Workplace Web Content Management, content for rules relies on the IBM Workplace Web Content Management rule attributes to be stored in the JCR. Because this feature is optional, you need to populate the JCR with your IBM Workplace Web Content Management content attributes initially. After you populate the JCR and configure IBM Workplace Web Content Management to shadow content in the JCR, any new content is shadowed automatically.

To execute this one-time step run the following:

`http://yourserver/wps/wcm/connect?MOD=JCRModule&action=populate`

This module created in this command is a requirement for the resource collections definition that is described in the previous section, because the classes are used to access the JCR.

Your IBM Workplace Web Content Management content is now shadowed in the JCR, and you are ready to write some rules.

11.3.4 Creating a personalization rule

To create a personalization rule, write a select rule that retrieves all the content that is created using the DeptNews authoring template. You can find a guide to creating this content at:

http://www.ibm.com/developerworks/websphere/library/techarticles/0503_smit/0503_smit.html

1. Log in to WebSphere Portal.
2. Navigate to the Personalization Workspace.
3. Select **New** → **Rule**.
4. Enter a name for the new rule (for example, Get WCM Content).
5. Select **Resource.Attribute**.
6. Select the Resource Collection that you created earlier.
7. Select the Web Content Management metadata that you would like personalization to use as the search criteria (for example, Template), as shown in Figure 11-3.

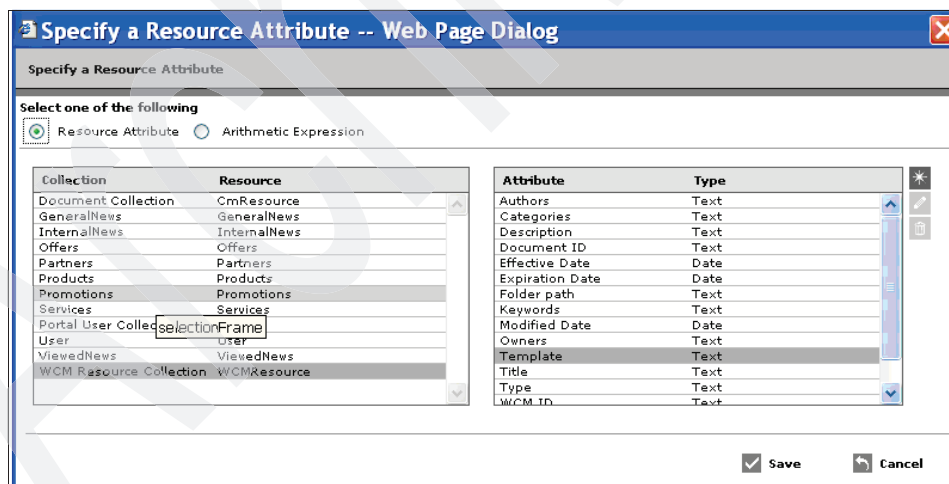


Figure 11-3 Specifying a resource attribute

8. Select **Value** and enter in a value for which you want to search (for example, your authoring template name), as shown in Figure 11-4.

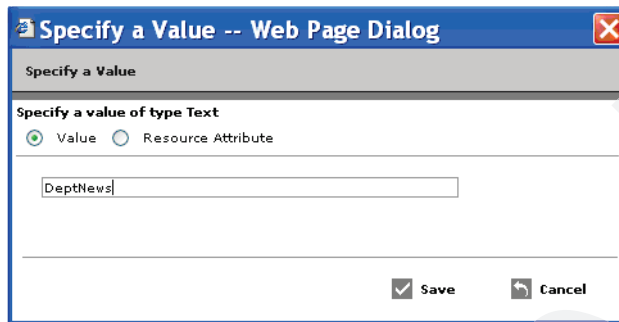


Figure 11-4 Entering a value to search for

9. Select **Save**.
10. To see that content is returned from this rule, select the Preview tab, as shown in Figure 11-5. The preview tab displays the IBM Workplace Web Content Management internal ID of the content and shows you that you do have content returned from your rule.

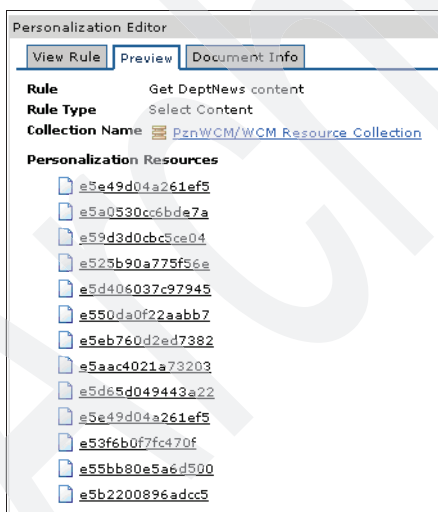


Figure 11-5 The Preview tab in the Personalization Editor

This Personalization Rule can now be displayed in a Personalization List Portlet, referenced in an IBM Workplace Web Content Management component, or included within a Content Spot on a Portal page.

11.3.5 Configuring the Personalized List portal

The Personalized List Portlet is installed with WebSphere Portal and can be added to any Portal page. A portal page is configured within the Personalization page with this list Portlet already on it when WebSphere Portal is installed. It has many configuration options that enable you to use a rule to show content quickly. There are two configuration modes:

- ▶ One that configures the Spot information
- ▶ One that configures the Display Options

Initially the Configure Spot is displayed, as shown in Figure 11-6.

Workspace Personalized List Demo

Personalized List

Configure the Personalization Rule Portlet

Configure Spot | [Display Options](#)

Which Personalization resources are retrieved

☐ Show all resources of the following type
Document Collection

☐ Show the results from content spot
PznDemo/GeneralNewsSpot

☒ Execute the following rule
PznWCM/Get DeptNews content

How details for each resource are displayed

☒ Show details in this portlet

☐ Open object in Personalization editor

☒ Generate details page. Configure on the Display Options page

☐ Use an existing JSP in the portlet WAR
JSP Location:

Ok Cancel Reset Refresh

Figure 11-6 The Configure Spot configuration mode

To configure the Personalization List Portlet, do the following:

1. Select the rule that you just created as the rule to execute (see Figure 11-7).
2. Click **Display Options**.
3. Choose the Fixed Title attribute for the Title Attribute and choose the Fixed Description attribute for the Detail Attribute.
4. Select the Fixed Folder Path for the Categorization Attribute. This groups the content under the site area structure.
5. Click **OK**.

Personalized List

Configure the Personalization Rule Portlet

[Configure Spot](#) | **Display Options**

Choose the attributes for the portlet

Title Attribute

☒ Fixed

☐ Dynamic

Each resource displayed will be identified by a link with this attribute value as its text.

Detail Attribute

☒ Fixed

☐ Dynamic

When generating a Details page (instead of using an existing JSP) this attribute will become the page's body.

Categorization Attribute

☒ Fixed

☐ Dynamic

If categorization is enabled then resources will be grouped into categories by the value of this attribute.

Options affecting resource layout

☒ Group resources by their categorization attribute?

Categories should be in order

☒ Display the title for the resource being viewed on the Details page?

Alter the CSS Style Classes used to display attributes

Title attribute style
Default is rulePortletDefaultTitleProperty

Detail attribute style
Default is rulePortletDefaultDetailProperty

Category attribute style
Default is rulePortletDefaultCategoryProperty

Category list style
Default is rulePortletDefaultListStyle

The default CSS styles for each attribute are located in the file /html/rulePortlet.jsp inside the portlet WAR. If you wish to create new styles you may add them to this file or to the currently running theme style sheet.

Figure 11-7 Display options for the Personalized List Portlet

Your rule content is returned (as shown in Figure 11-8).

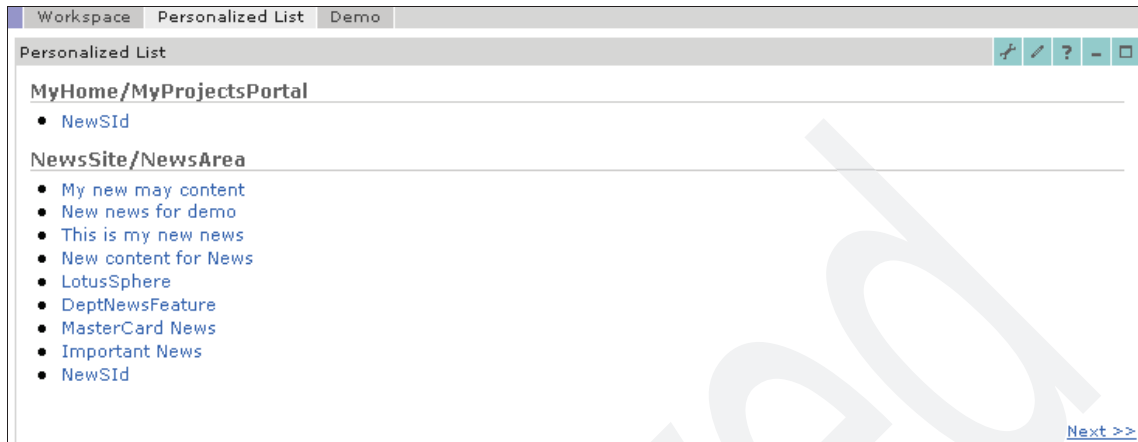


Figure 11-8 Personalized List content

When you select a title, you get the detail attribute displayed for that title. Note that this list does not use the site area and presentation template to render the content. Instead, it uses just the attribute that you defined in the Portlet configuration.

11.3.6 Including rules on presentation template

A IBM Workplace Web Content Management presentation template lets you format the content as you need it. The new personalization component has been added to the list of IBM Workplace Web Content Management library components. You can select this component when configuring the IBM Workplace Web Content Management local rendering Portlet, or you can reference it in a presentation template.

To reference the personalization component in a presentation template, do the following:

1. In the IBM Workplace Web Content Management author user interface Portlet, select **New** → **Personalization Component** and click **OK**.
2. In the Identification dialog box, enter a name and a description for the personalization component (Figure 11-9).

Identification

* Name:

Description:

Type: Personalization Component

Authors: wpsadmin

Owners: wpsadmin

Personalization Component

Name:

Title:

Description:

Author:

Date modified:

Figure 11-9 New personalization component

3. Select search and find the new personalization component that you created, as shown in Figure 11-10.

Web Content Authoring

Content Spot Chooser.

Select a personalization Content Spot or Rule.

Type: /PznWCM/Get DeptNews content

Name. ^	
ibmcm:author	uid=wpsadmin,o=default organization
ibmcm:countryCode	
ibmcm:creator	uid=wpsadmin,o=default organization
ibmcm:deleted	false
ibmcm:description	
ibmcm:effectiveDate	
ibmcm:expirationDate	1970-01-25T20:31:23.647Z
ibmcm:language	
ibmcm:lastModifier	uid=wpsadmin,o=default organization
ibmcm:name	Get DeptNews content
ibmcm:title	Get DeptNews content
ibmcm:type	
ibmpzn:collectionType	/PznWCM/WCM Resource Collection
ibmpzn:contents	<?xml version="1.0" encoding="UTF-8"?><

Figure 11-10 Listing of Personalization Components

4. Key in the heading and detail HTML tags as shown in Figure 11-11.

Header:

Import Export

Here's are the department news items:

 <table>

Component Design for each matching Content:

Import Export

```
<tr>
<td>
<A HREF='<Placeholder tag="href"/>'><ApatrixCmpnt context="autoFill"
key="NewsTitle" type="content"/></A><br>
</td>
<td>
<WorkflowCmpnt context="autoFill" field="publishdate" format="MMMM dd,
yyyy" type="content"/>
</td>
</tr>
```

Footer:

Import Export

</table>

Figure 11-11 Heading and detail HTML tags

5. Save and close.

Next, you imbed this component in a presentation template as follows:

1. Copy an exiting detail presentation template (such as NewsDetail). Save this presentation template as *NewsDetailPzn* (Figure 11-12 on page 456) after including the following tag:

```
<ApatrixLibCmpnt name="DeptNewsList"/>
```

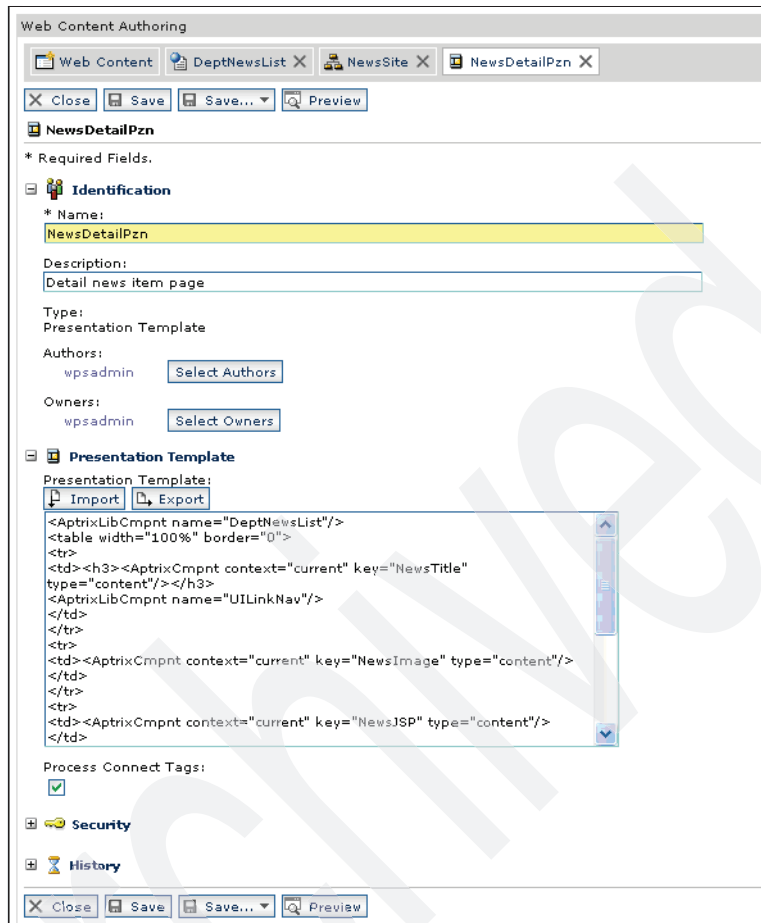


Figure 11-12 Saving the NewsDetailPzn presentation template

2. Click **Preview** to see the presentation template with the site area that matches your content (as shown in Figure 11-13).



Figure 11-13 Previewing the new presentation template

You have now used a rule to render IBM Workplace Web Content Management content. Next, you add a user profile and binding rules to direct specific content to certain users based on the attributes of the user.

11.3.7 Creating rules

A profile rule needs a user resource. You can use the portal resource, but it is more common to use some sort of database repository that contains employee or customer information. The personalization demo provides a user resource and populates a database with user attributes. The portal user is connected with these attributes via a translator class.

To create rules:

1. Create a *profile* rule named *UserType*, as shown in Figure 11-14. This rule classifies the user as a manager if the role attribute is set to manager.

The screenshot shows the 'Personalization Editor' window with the 'View Rule *' tab selected. The rule name is 'UserType' and it is located in 'PznWCM'. The 'Content Type' is set to 'Profiler'. The 'Rule Type' is set to 'Profiler'. The rule logic is defined as follows: 'UserType is' followed by a list of conditions. The first condition is 'Manager' when 'current User.Role is equal to Manager'. The second condition is 'Otherwise NonManager'. There are links for 'add Condition' and 'add Profile'.

Figure 11-14 Creating a profile rule

2. Create a *select* rule that selects new content for managers called *Manager News*, as shown in Figure 11-15.

The screenshot shows the 'Personalization Editor' window with the 'View Rule *' tab selected. The rule name is 'ManagerNews' and it is located in 'PznWCM'. The 'Content Type' is set to 'com.aptrix.pluto.pzn.resources.WCMResource'. The 'Rule Type' is set to 'Select Content'. The rule logic is defined as follows: 'Select content' followed by a list of conditions. The first condition is 'whose PznWCM/WCM Resource Collection.Template is equal to DeptNews and'. The second condition is 'whose PznWCM/WCM Resource Collection.Categories is equal to Manager'. There are links for 'add Condition', 'order as is', and 'show all items'.

Figure 11-15 Creating a select rule

3. Exclude the manager items from the Dept News for non-managers by changing the rule that to match the rule shown in Figure 11-16.

The screenshot shows the 'Personalization Editor' window with three tabs: 'View Rule *', 'Preview', and 'Document Info'. The 'View Rule *' tab is active. The rule name is 'Get DeptNews content' and it is located in the 'PznWCM' folder. The 'Description' field is empty. The 'Content Type' is 'com.aptrix.pluto.pzn.resources.WCMResource'. The 'Rule Type' is 'Select Content'. The rule conditions are listed below:

- ☒ whose [PznWCM/WCM Resource Collection.Template](#) is equal to [DeptNews](#) and
- ☒ whose [PznWCM/WCM Resource Collection.Title](#) includes [New](#) and
- ☒ whose [PznWCM/WCM Resource Collection.Categories](#) is not equal to [manager](#)

Below the conditions, there are links for [add Condition](#) and [order by](#). The 'order by' section shows 'Title ascending then Effective Date descending' and a link for [show all items](#).

Figure 11-16 Excluding the manager items

4. Tie this together by creating a binding rule that displays the Get DeptNews for both managers and non-managers and that displays manager content for managers (as shown in Figure 11-17 on page 460). A binding rule can use multiple profile names to deliver the targeted content to those users conditionally.

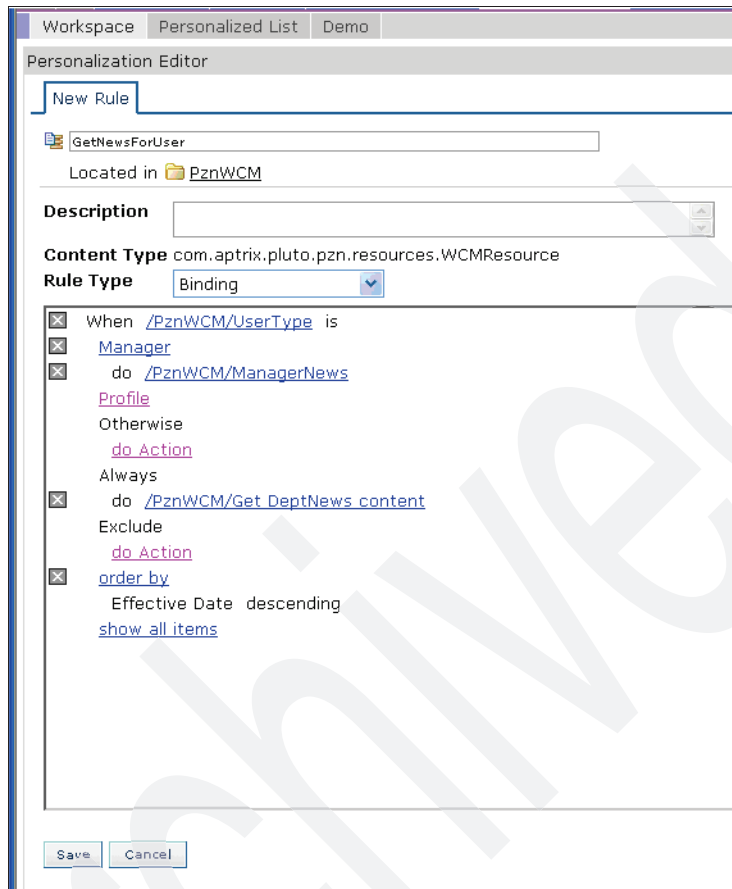


Figure 11-17 Creating a binding rule

Now all the rules are in place, and you can create some content to ensure that managers and non-managers see different news items.

11.3.8 Creating IBM Workplace Web Content Management content using a profile

News items can have a profile that is associated with that item. Profile assignments are selected from a list of taxonomy categories. These categories are created easily using the IBM Workplace Web Content Management authoring user interface (as shown Figure 11-18).

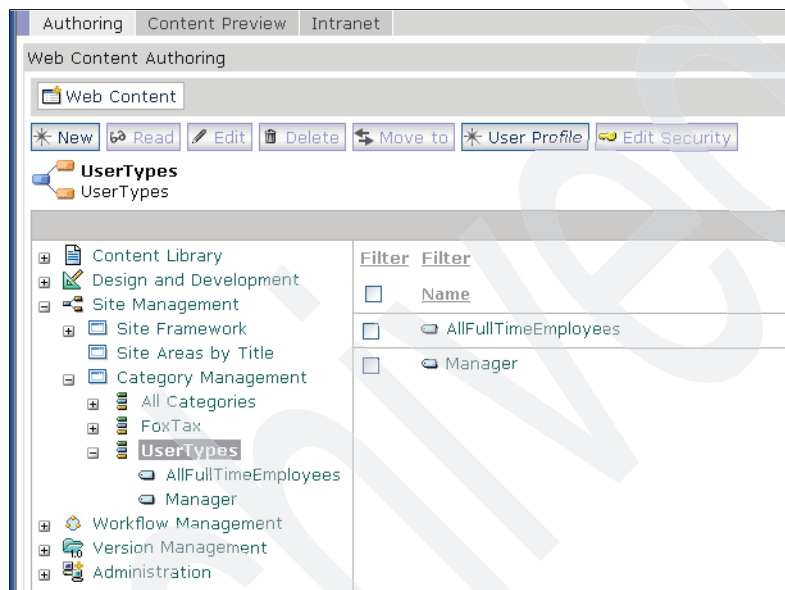


Figure 11-18 Authoring user interface

For our scenario, we have two categories under the UserTypes taxonomy: AllFullTimeEmployees and Manager. After you have created the categories, you can create a new DeptNews item and select Manager for the Profile, as shown in Figure 11-19.

Web Content Authoring

Web Content News for Managers X

News for Managers was saved successfully.

X Close Save Save... Preview Component Manager Apply Authoring Template

News for Managers

* Required Fields.

Identification

* Name:
News for Managers

Description:
News for managers Description

Type:
Content

Authors:
wpsadmin Select Authors

Owners:
wpsadmin Select Owners

Profile

Categories:
/UserTypes/Manager Select Categories

Keywords:

Site Area Information

Content

NewsBody

Format Font Size b i u

Figure 11-19 Creating a new item

Next, change the DeptNewsList Personalization component that you created previously to use the new Binding Rule, as shown in Figure 11-20.

The screenshot shows the 'Web Content Authoring' window with two tabs: 'Web Content' and 'DeptNewsList'. The 'DeptNewsList' tab is active, displaying the configuration for a Personalization Component. The interface includes buttons for 'Close', 'Save', 'Save...', and 'Preview'. The component is titled 'DeptNewsList' and is marked as a 'Required Field'. Under the 'Identification' section, the 'Name' is 'DeptNewsList', the 'Description' is 'PZN rule for list of Dept News', and the 'Type' is 'Personalization Component'. The 'Authors' and 'Owners' are both listed as 'wpsadmin', with 'Select Authors' and 'Select Owners' buttons next to them. The 'Personalization Component' section shows the 'Name' as '/PznWCM/GetNewsForUser' with 'Search', 'Open', and 'Clear' buttons. The 'Title' is 'GetNewsForUser', the 'Description' is '--', the 'Author' is 'uid=wpsadmin,o=default organization', and the 'Date modified' is 'Jun 25, 2005 3:07:53 PM EDT'. The 'Header' section has 'Import' and 'Export' buttons. A text area at the bottom contains the text 'Here's are the department news items:' followed by HTML tags '
' and '<table>'. A large, faint 'APR 2005' watermark is visible across the center of the image.

Figure 11-20 Using the new binding rule

We you have the rules and the content ready, next you need to add a Portlet to a page and configure the Portlet to the new presentation page.

11.3.9 Configure the Portlet in personalization demo

The personalization demo is a great example that comes with WebSphere Portal 5.1. (See the Information Center for instructions on how to install these portlets.) To configure the Portlet:

1. On the personalization demo page, add the content viewer Portlet, as shown in Figure 11-21.

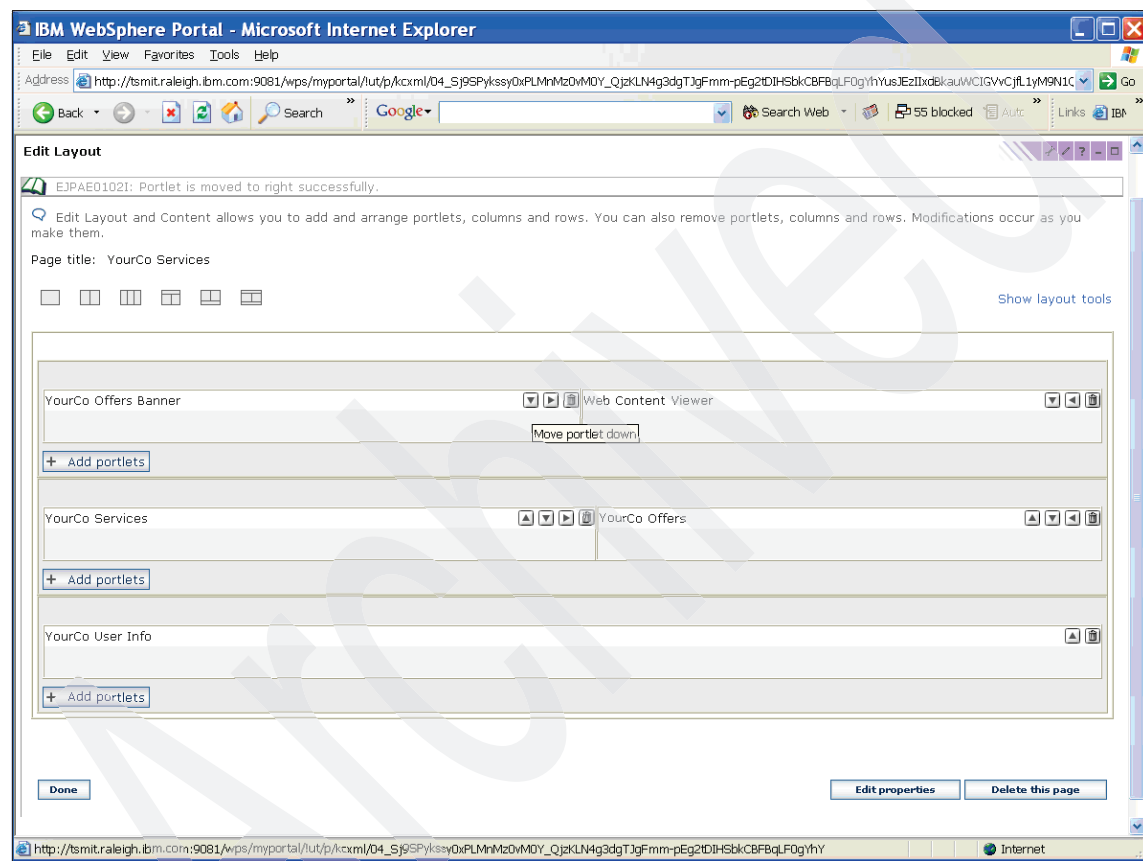


Figure 11-21 Adding the content viewer Portlet

2. Configure the Portlet to map to the PageDesign that you created that has the personalized list and detail, as shown in Figure 11-22

The screenshot shows the configuration interface for a portlet titled "News just for you". The interface is divided into several sections:

- Content Type:** Radio buttons for "Content" (selected), "Library Component", and "Content Component".
- Content:** A text field containing "/NewsSite/NewsArea" with an "Edit" button.
- Alternate Presentation Template:** A text field containing "NewsDetailPzn" with "Edit" and "Clear" buttons.
- Links:**
 - Broadcast Links To:** Radio buttons for "None", "This page" (selected), and "Another page".
 - Select a target portal page:** A dropdown menu showing "My Portal".
 - Receive Links From:** Radio buttons for "None", "This portlet" (selected), and "Other portlets and this portlet".
- Settings:**
 - Portlet Display Title (Optional):** A text field containing "News just for you".

At the top of the configuration area are buttons for "Ok", "Cancel", "Apply", and "Help".

Figure 11-22 Mapping to the PageDesign

3. Test the new rules. There are a number of IDs that are created by the demo installation task, as shown in Figure 11-23. For example, Scott is not a manager, and Marge is a manager.

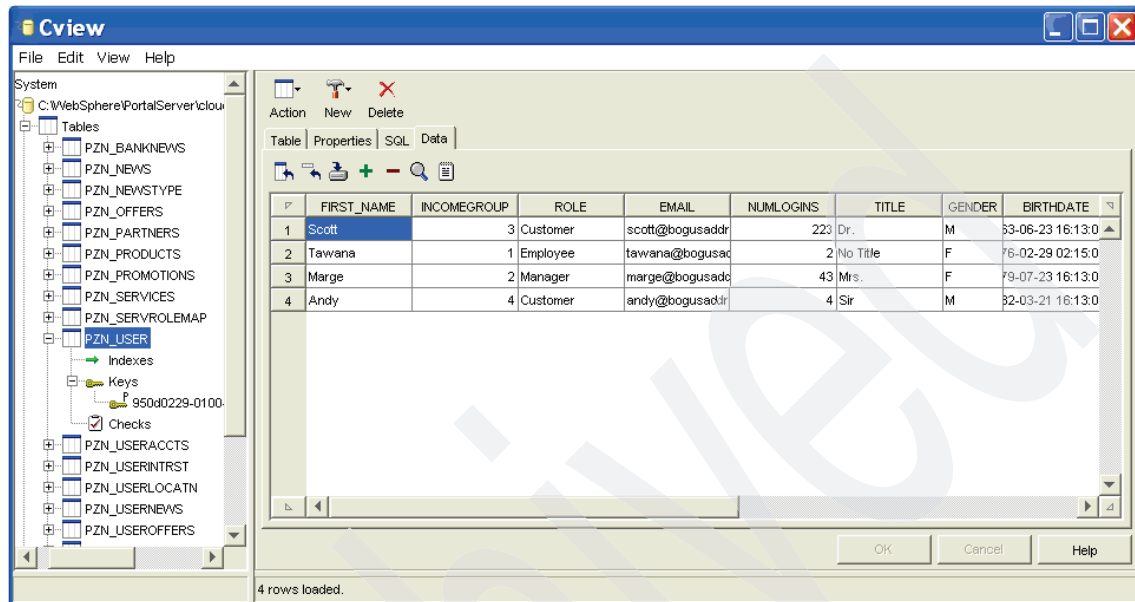


Figure 11-23 IDs created by the demo installation task

Log in as Scott and navigate to the Personalization Demo. You should see all non-manager news content.

Log in as Marge, and you see your additional item for managers (Figure 11-24).

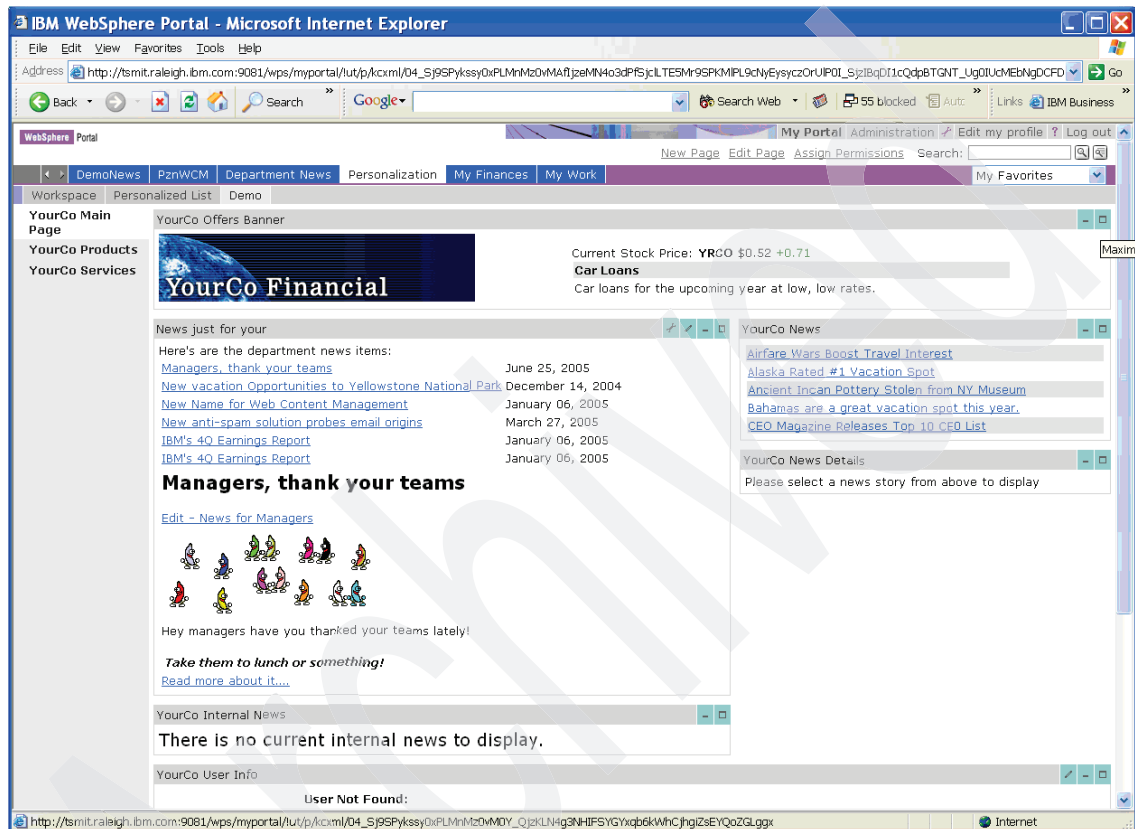


Figure 11-24 Content visible to managers

11.3.10 Including the rule in JSP

Yet another way to use this rule is to include it in a JSP by including a content spot. The Personalization Demo has many examples of rules in a JSP. There are two ways to code a content spot: *explicitly* generated or *dynamically* generated. The rule that is associated with a the spot is specified in the Personalization Workspace, as shown in Figure 11-25 on page 468.

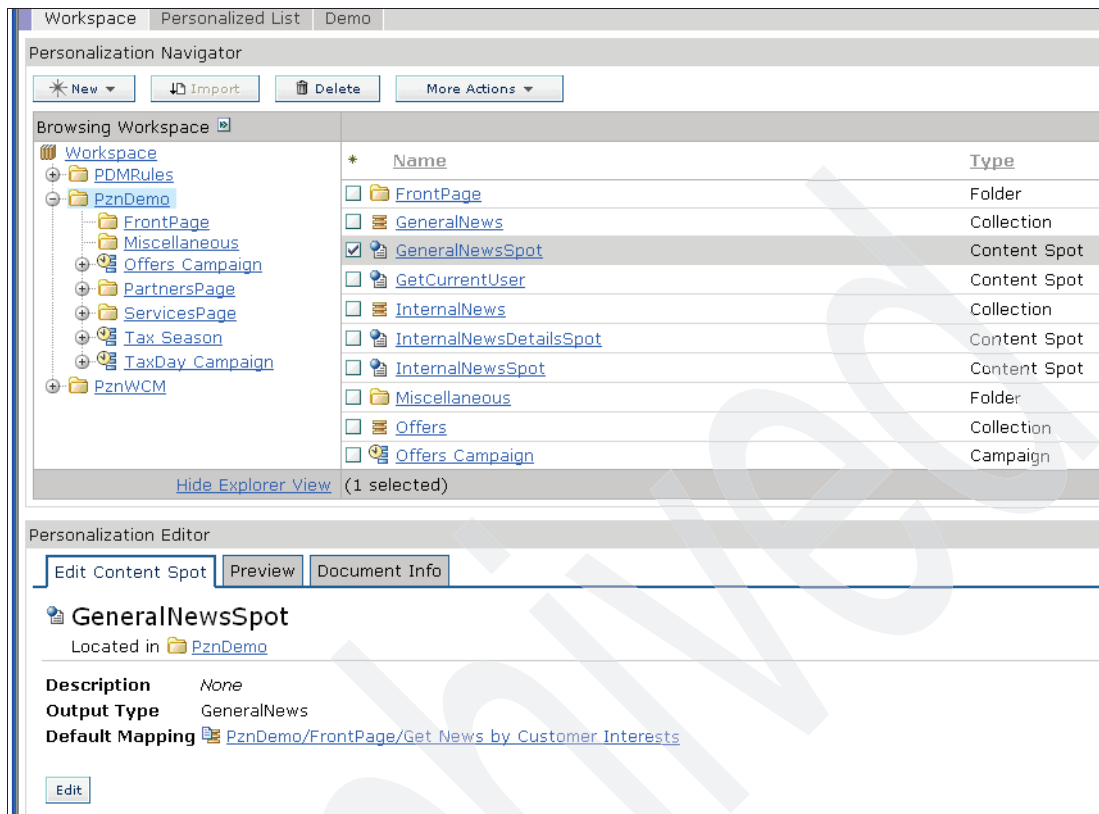


Figure 11-25 The Personalization Workspace

The JSP code remains static. Business users can vary the rules by creating a new rule and tying that rule to the spot, without a developer changing the site.

Working with the application programming interface

This chapter provides details on the IBM Workplace Web Content Management API. It provides you with the ability to create custom Portlets and JavaServer Pages using IBM Workplace Web Content Management functionality.

This chapter includes the following sections:

- ▶ Introduction to the API
- ▶ Components of the IBM Workplace Web Content Management API
- ▶ Extending your Web site with JSP support
- ▶ Setting up the development environment
- ▶ Examples of API development
- ▶ Deployment considerations

12.1 Introduction to the API

The IBM Workplace Web Content Management Java application program interface (API) allows you to access Web content management functionality from custom code and applications. With the IBM Workplace Web Content Management API, you can create custom portlets, servlets, and JSP pages that use IBM Workplace Web Content Management functionality. It provides access to important items during the content creation and rendering process, including content, sites, site areas, taxonomies and categories, and workflows and library components. The IBM Workplace Web Content Management API provides a range of search iterators to access these items. For example, you can find items by their item ID or by a given name and type. The rendering process is achieved through a workspace object which is the entry point into the repository for a particular user.

The current version of the IBM Workplace Web Content Management API does not allow read and write access to all items of the Web content management system. Some functionality is excluded to avoid potential damage to the data integrity or to avoid exposing functionality which can be changed significantly in future releases of the product.

Restriction: You cannot create, update, or delete workflows, workflow stages, workflow actions, taxonomies, categories, syndicators, subscribers, presentation templates, authoring templates, and library components with the IBM Workplace Web Content Management API (with the exception of file resource, HTML, and image library components).

12.1.1 Use cases for the API

In addition to the integration of Web content management functionality within custom portlets, you can use the IBM Workplace Web Content Management API in multiple scenarios as shown in the following examples:

- ▶ Bulk resource import

You can use the IBM Workplace Web Content Management API to create a bulk resource import of a large number of files or images into IBM Workplace Web Content Management.

- ▶ Import site form HTML

After some initial preparation work in IBM Workplace Web Content Management, you can use the IBM Workplace Web Content Management API to import a flat HTML site. The development of the import process and the preparation work can be as complex as necessary. For example, the

importer could break the pages into different components or could reference library components.

- ▶ Import site from third-party Web content management system

Similar to the import of a flat HTML site, you can use the IBM Workplace Web Content Management API to migrate an existing Web site from a third-party Web content management system. In addition to the raw content, this scenario also includes the migration of additional metadata such as authors, keywords, or descriptions. You can also migrate other content stores (for example, Lotus Domino databases).

- ▶ Custom site export site to HTML or XML

Similar to the pre-rendering (Cacher) functionality of IBM Workplace Web Content Management, you can use the IBM Workplace Web Content Management API to render an entire site or subset of pages. The output can be generated in any format, such as HTML or XML, to perform custom rendering transformation.

- ▶ External workflow applications

A more complex example for the usage of the IBM Workplace Web Content Management API is the extension of the built-in workflow capabilities through an external workflow application. The external workflow application can push documents based on rules around defined workflow stages. On the other side, external applications can poll documents from their workflow stage and perform some action before moving documents to the next stage.

12.2 Components of the IBM Workplace Web Content Management API

This section describes the core class and interfaces of the IBM Workplace Web Content Management API. It introduces the basic objects and components of IBM Workplace Web Content Management and then connects these objects and components with the corresponding application programming interfaces.

When you use IBM Workplace Web Content Management to develop a Web site, you act as a registered user. All activities that you perform are associated with this user. Thus, you must have the authority to perform certain functions as this registered user. In the IBM Workplace Web Content Management API, this authorized access is represented by a *Workspace* object. The *Workspace* is the heart of the API. Objects are created, saved, deleted, and searched for in the *Workspace* object. A *Workspace* is basically an interface to IBM Workplace Web Content Management that is associated with a user. Using a *Workspace* object, you can perform operations as that authorized user.

Figure 12-1, Figure 12-2 on page 473, and Figure 12-3 on page 473 show an architectural overview of the IBM Workplace Web Content Management API interface hierarchy.

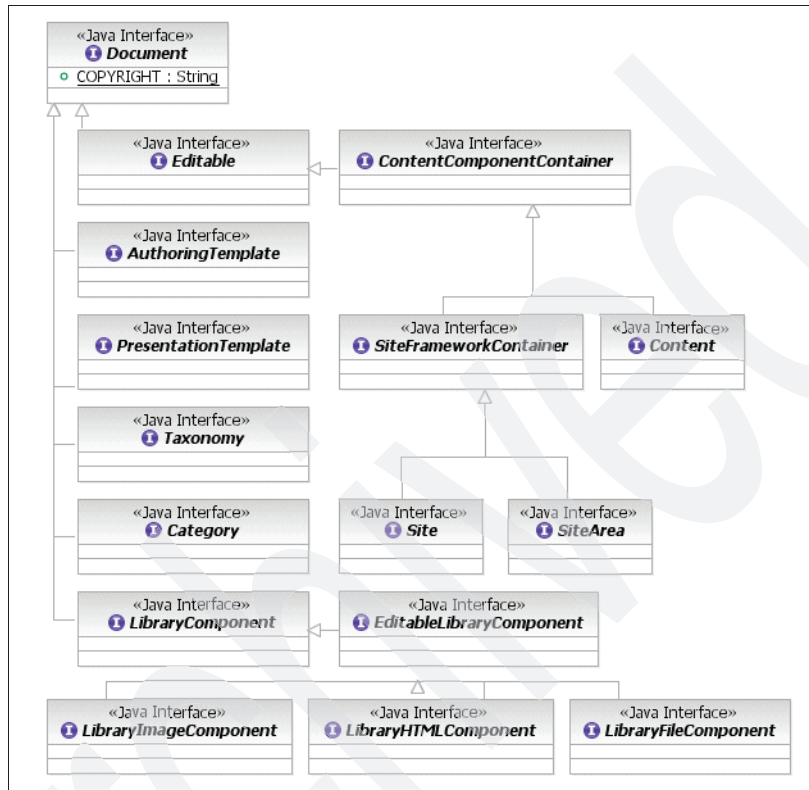


Figure 12-1 Architectural overview , the API interface hierarchy - Part 1

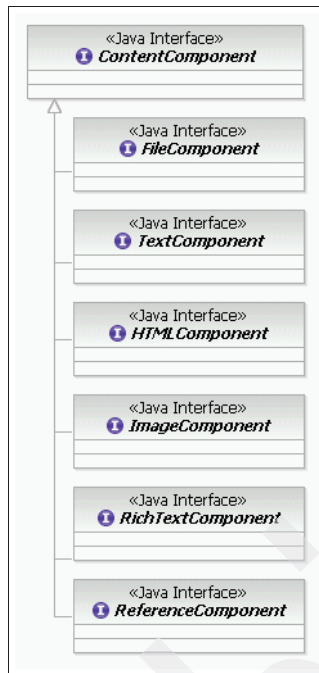


Figure 12-2 Architectural overview, the API interface hierarchy - Part 2



Figure 12-3 Architectural overview, the API interface hierarchy - Part 3

The IBM Workplace Web Content Management API design follows a concept that each interface has only one super-interface. Because of the design decision, the interface Editable and EditableLibrary components contain the same methods. The IBM Workplace Web Content Management API does not support native remote access to the IBM Workplace Web Content Management server. It

only supports Java code that runs in the same Java Virtual Machine as IBM Workplace Web Content Management itself.

Note: Only one class is provided in the IBM Workplace Web Content Management API — the WCM_API class. The remaining classes are interfaces.

To read about the complete set of features for the IBM Workplace Web Content Management API, review the JavaDocs that are available via the API. The JavaDocs are located in the following folder on your IBM Web Content Management server:

```
<Drive>:\<Path>\WebSphere\AppServer\installedApps\<Node>\wcm.ear\
ilwwcm.war\webinterface\api-javadoc
```

You can find further information about the IBM Workplace Web Content Management API in the section *Advanced Features Guide - The Web Content Management API* in the WebSphere Portal Information Center at:

<http://www-128.ibm.com/developerworks/workplace/documentation/webcontentmanagement/>

12.2.1 The WCM_API class

The WCM_API class is the starting point of when you use the IBM Workplace Web Content Management API. This class provides a method to get the repository.

12.2.2 The repository interface

The repository interface provides methods for obtaining a Workspace. A Workspace represents a view of the repository for a specified user. The repository interface provides the following methods to obtain the workspace:

- ▶ `getWorkspace(java.security.Principal p_principal)`
Returns a Workspace object that represents a view of the repository for the specified principal.
- ▶ `getWorkspace(java.lang.String p_username, java.lang.String p_password)`
Returns a Workspace object that represents a view of the repository for a user with the specified credentials.
- ▶ `getAnonymousWorkspace()`
Returns a Workspace object that represents a view of the repository for the Anonymous user.

- ▶ `endWorkspace()`

Closes the current Workspace when the user is finished with it.

The following code shows an example of how to obtain a repository instance from the `WCM_API` class:

```
Repository repository = null;  
repository = WCM_API.getRepository();
```

12.2.3 Workspace interface

A Workspace represents a view of the IBM Workplace Web Content Management repository for a particular user. Most of the important functions are provided within the Workspace interface. Operations that are available in the Workspace include:

- ▶ Searching for a document.

The IBM Workplace Web Content Management API provides two kinds of search methods.

- You can search for documents by providing a detailed search criterion. The criterion includes template ID, siteArea ID, category ID, and keyword.
- You can search for documents by several provided `findby` methods, including `findbyName`, `findbyType`, `findContentByCategory`, and so on.

- ▶ Creating new objects of editable types, such as content, site, component, and so on.
- ▶ Rendering given context or rendering a given component using specified context.
- ▶ Saving and deleting editable objects.

To get a Workspace object, you must request one from the repository singleton. The Workspace object is the entry point into the IBM Workplace Web Content Management repository. For example, to get a Workspace object for a registered user, use the following call:

```
WCM_API.getRepository().getWorkspace(username, password)
```

Note: If the user is not recognized as an IBM Workplace Web Content Management user or if the user cannot be authenticated, an `OperationFailedException` is thrown.

Within the portal context, the current user could be retrieved by the portal request, as shown in Example 12-1.

Example 12-1 Access workspace object for a portal user

```
Workspace ws = null;
Repository repository = null;
try {
    repository = WCM_API.getRepository();
    ws = repository.getWorkspace((java.security.Principal)request.getUser());
} catch (ServiceNotAvailableException e) {
    e.printStackTrace();
} catch (OperationFailedException e) {
    e.printStackTrace();
}
```

12.2.4 Document object

A document represents an object or an entity in the repository. A document is the parent interface of most other entity interfaces, such as content, category, site, and so on. In most cases, you only use a document's subinterfaces. The interface contains methods to access general attributes of the repository entity, as follows.

- ▶ `getName()`
Returns the name of the document object
- ▶ `getID()`
Returns the document ID of the document object
- ▶ `getDescription()`
Returns the description of the document object.

The document interface also contains several methods to retrieve the members of different security level, as follows:

- ▶ `getAuthors()`
- ▶ `getDeleteAccessMembers()`
- ▶ `getEditAccessMembers()`
- ▶ `getLiveAccessMembers()`
- ▶ `getOwners()`
- ▶ `getReadAccessMembers()`

Each method returns a string array that contains the members. If the document does not have the requested access right specified, the methods return an empty array.

Note: The created documents do not exist in the repository until they are saved.

12.2.5 DocumentID interface

Each entity in the repository has a unique identifier, called *DocumentID*, that uniquely represents this entity. Different entities can have the same name, but they must have different IDs. The DocumentID is represented in the IBM Workplace Web Content Management API by the DocumentID interface. This interface contains the following methods:

- ▶ `getName()`
Returns the name of the document with this DocumentID.
- ▶ `getType()`
Returns the type of the document with this DocumentID.
- ▶ `isOfType(com.ibm.workplace.wcm.api.DocumentType p_type)`
Returns true if, and only if, the type of the document that is represented by this DocumentID is the same as the DocumentType argument.
- ▶ `toString()`
Returns a string representation of this DocumentID.

Attention: The DocumentID format is not guaranteed to be consistent between releases. Do not store the string representation and use later.

12.2.6 DocumentType and DocumentTypes interfaces

A *document type* represents a kind of document. The document type can be content, category, site, or any of the components. In the IBM Workplace Web Content Management API, the *DocumentType* and *DocumentTypes* interfaces handle document types. The DocumentTypes interface maintains a list of all available document types, and a DocumentType object is an individual document type.

The DocumentType interface contains the `toString()` method, which returns a string representation of the object's class name. The string values are stored as constants in the DocumentTypes interface.

12.2.7 RenderingContext interface

In IBM Workplace Web Content Management, it is a basic requirement to render content or components in a Web page. Due to the dynamic presentation feature when rendering components, you need to get the context for rendering first. The RenderingContext interface represents the context of rendering that contains information that is required to present a renderable object. The contained information includes the current content, page design and site, the servlet path and the application path, the rendered content, and so on.

The source code fragment shown in Example 12-2 includes all the steps that are necessary to render a content page. This example assumes the existence of a request and response such as within a servlet or JSP.

Example 12-2 RenderingContent method

```
Workspace workspace = null;
RenderingContext context = null;
workspace = WCM_API.getRepository().getWorkspace("user", "password");
HashMap map = new HashMap();
context = workspace.createRenderingContext(request, response, map);
context.setRenderedContent("/Intranet/IT Home/News and Events/News/");
String content = workspace.render(context);
```

12.2.8 Editable components of the interfaces

This section discusses the editable components of the interfaces. In IBM Workplace Web Content Management, there are two kinds of editable components: *content components* and *library components*.

Content components are components that are stored in content, site, or site area components. *Library components* are static components that are stored directly in the component library.

Content components and library components are both retrieved by the page design. However, when you retrieve library components in a page design, the components that are retrieved are known at all times because they are directly referenced. As for content components, the actual component that is rendered is dependent upon the current content or site, because they are referenced dynamically when content is rendered.

ContentComponentContainer interface

The ContentComponentContainer interface is the root interface for the content component container hierarchy. A ContentComponentContainer represents a container for ContentComponent objects.

The `ContentComponentContainer` interface includes methods for retrieving and setting `ContentComponent` objects. `ContentComponents` can be added to site, sitearea, and content objects. The `ContentComponentContainer` interface provides basic functions such as the updating and retrieving of content components from their container.

The following list shows examples for methods that are included within the `ContentComponentContainer` interface:

- ▶ `componentIterator()`
Returns an iterator of all `ContentComponent` objects in this `ContentComponentContainer`.
- ▶ `getComponent(java.lang.String p_name)`
Retrieves a `ContentComponent` with a name specified by the argument from this `ContentComponentContainer`.

In the hierarchy that is below the `ContentComponentContainer` interface, the API implements interfaces for the content and the site framework.

Content objects

Content objects store values that can be referenced in other IBM Workplace Web Content Management objects. These values include ID information, profiling, security, and components. Content is represented by the content interface. You can create and save content objects through the API. You can also retrieve existing content object to be edit and save them. The content interface provides methods to access the items workflow properties, including moving the item to the next stage in the workflow, creating a draft, checking its published or expired status, and so forth. The following list shows a few examples for most common methods of the content interface:

- ▶ `getAuthoringTemplateID()`
Returns the ID of the `AuthoringTemplate` that was used to create this `Content` document.
- ▶ `nextWorkflowStage()`
Move this `Content` object into the next stage in its workflow.
- ▶ `getWorkflowId()`
Returns the ID of the workflow associated with this content.
- ▶ `setKeywords(java.lang.String[ ] p_keywords)`
Sets the keywords for this `Content` document.

Refer to the API documentation for a complete list of methods for the content interface.

SiteFrameworkContainer interface

The SiteFrameworkContainer interface implements the ability to organize and build a site framework. A site framework is a hierarchy consisting of site, site area, and content objects. The site and site area extends the SiteFrameworkContainer interface to provide a uniform interface when organizing and building the site framework.

The following list shows a few examples for most common methods of the SiteFrameworkContainer interface:

- ▶ `setDefaultContent (...) / getDefaultContent()`
Sets or gets the DocumentID of the default content for this site or site area.
- ▶ `addTemplateMapping(...) / getTemplateMapping(...)`
Adds or gets a mapping for the specified authoring template and presentation template.

Refer to the API documentation for a complete list of the methods for this interface.

Example 12-3 iterates through all site area's within all sites and adds names to an ArrayList.

Example 12-3 SiteArea ArrayList

```
DocumentIdIterator siteIterator = ws.findByType(DocumentTypes.Site);
Site siteDocument = null;
ArrayList contentPaths = new ArrayList();
while (siteIterator.hasNext()){
    try {
        siteDocument = (Site)ws.getById((DocumentId) siteIterator.next());
    } catch (DocumentRetrievalException e1) {
        e1.printStackTrace();
    } catch (AuthorizationException e1) {
        e1.printStackTrace();
    }
}

DocumentIdIterator siteareaIterator = siteDocument.getChildren();
SiteArea siteareaDocument = null;
while (siteareaIterator.hasNext()){
    try {
        siteareaDocument =
        (SiteArea)ws.getById((DocumentId)siteareaIterator.next());
        contentPaths.add(siteareaDocument.getName());
    } catch (DocumentRetrievalException e2) {
        e2.printStackTrace();
    } catch (AuthorizationException e2) {
        e2.printStackTrace();
    }
}
```

```
}  
}  
}
```

LibraryComponent interface

For library components, the IBM Workplace Web Content Management API provides a `LibraryComponent` interface to represent all the library components. Also, there are `LibraryFileComponent`, `LibraryHTMLComponent`, and `LibraryImageComponent` interfaces as follows:

- ▶ `LibraryFileComponent`

A `LibraryFileComponent` can contain one file. The contents of the file are dealt with as raw byte content. This interface provides methods to manipulate the file.

- ▶ `LibraryHTMLComponent`

A `LibraryHTMLComponent` can contain HTML. This interface provides methods to retrieve and set the HTML contained within this component.

- ▶ `LibraryImageComponent`

A `LibraryImageComponent` can contain one image. The contents of the image file are dealt with as raw byte content. This interface provides methods to manipulate the image file and the attributes to use when displaying the image.

Other library components, such as the navigator and menu components, do not have corresponding interfaces in the API.

ContentComponent interface

The `ContentComponent` interface represents the root interface in the content component hierarchy. `ContentComponent` marks a component that can be held within a `ContentComponentContainer`. A `ContentComponent` can exist only on content, site or site area objects. A `ContentComponent` must live in a `ContentComponentContainer`. It cannot exist in the repository as a separate object.

A content component has a one-to-one relationship with a `ContentComponentContainer` object, and consequently, with an object that can contain a `ContentComponentContainer`.

The `ContentComponent` interface includes the following methods:

- ▶ `getContainer()`

This method returns the `ContentComponentContainer` object that contains this `ContentComponent`.

- ▶ `getName`

Returns the name of this `ContentComponent` object.

The interface acts as super-interface for the content components. The IBM Workplace Web Content Management API provides the following content components:

- ▶ `File`
- ▶ `HTML`
- ▶ `Image`
- ▶ `Reference`
- ▶ `RichText`
- ▶ `Text`

Like the `ContentComponent`, the content components cannot be stored as a separate entity in the repository.

Each of the subinterfaces contains different `get` and `set` methods to manipulate the data that is contained within this component. For example the `TextComponent` interface contains the following methods:

- ▶ `getText()`
- ▶ `setText()`

Refer to the IBM Workplace Web Content Management API JavaDocs for a complete reference of the subinterface methods.

Note: The `JSPComponent` and `Federated` content components are not provided through the IBM Workplace Web Content Management API.

12.2.9 Read-only component interfaces

In addition to the components that you can edit, the IBM Workplace Web Content Management API contains read-only interfaces that you can use access to certain components. This sections describes those component interfaces that you cannot edit.

Note: You cannot create, edit, or delete the objects that this section discusses through the IBM Workplace Web Content Management API. They must already exist in the repository. You must create them manually through the authoring portlet.

AuthoringTemplate

The AuthoringTemplate subinterface represents an authoring template. This template is the basis of a content object. A new content object derives its default attributes from this template. The AuthoringTemplate subinterface inherits methods from the Document interface and does not contain its own methods.

PresentationTemplate

A PresentationTemplate is a template for how a Web page or the content in the Web Content Viewer portlet looks when rendered. Like the AuthoringTemplate, the PresentationTemplate inherits its methods from the Document interface and does not implement its own methods.

Category

The Category interface provides methods that aid in traversing a category tree. The interface extends the Document interface and includes the following additional methods:

- ▶ `getParent()`
Returns the DocumentID of the parent of this Category object.
- ▶ `getChildren()`
Retrieves the immediate children of the Category object.
- ▶ `getAllChildren()`
Returns an depth first iterator of DocumentID objects of the immediate child categories of this Category object.

For example, let us assume that our category tree contains the structure shown in Figure 12-4.

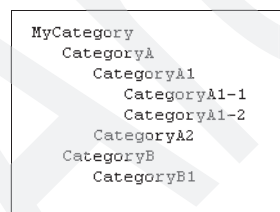


Figure 12-4 Sample category tree

If we make a call to the `getAllChildren()` method of the object that represents `MyCategory`, we get an iterator that contains the following categories:

- ▶ `CategoryA`
- ▶ `CategoryA1`
- ▶ `CategoryA1-1`
- ▶ `CategoryA1-2`
- ▶ `CategoryA2`
- ▶ `CategoryB`
- ▶ `CategoryB1`

A call of the `getChildren()` method of the same object returns an iterator that contains categories `CategoryA` and `CategoryB`.

Attention: The returned iterator only contains objects that have a least read access for the current user.

Taxonomy

The Taxonomy interface is very similar to the Category interface. The only exception is that a taxonomy object does *not* have an ancestor. Therefore, the interface does not contain a `getParent()` method.

Workflow

The Workflow interface represents a Workflow within in repository. A Workflow consists of `WorkflowStages` in a specific order. The Workflow controls access to and manages the progression of IBM Workplace Web Content Management objects through the specified `WorkflowStages`. The interface contains the following methods to iterate through the Workflow stages:

- ▶ `getStagesIterator()`
Returns an iterator of `DocumentID` objects of the `WorkflowStages` in this Workflow.
- ▶ `getNumStages()`
Returns the number of stages in this Workflow.

WorkflowStage

A `WorkflowStage` represents a step of a Workflow definition. The interface inherits its methods from the Document interface and does not implement its own methods.

12.2.10 UserProfile

The UserProfile interface allows access to user related information for a user who is logged in and validated. The interface provides the following methods:

- ▶ `getCategories()`
Returns a string array of the categories associates with this user.
- ▶ `getKeywords()`
Returns a string array of the keywords associated with this user.
- ▶ `getUsername()`
Returns the username of the user in this set of credentials.

12.2.11 Iterator interfaces

The IBM Workplace Web Content Management API provides the following interfaces to iterate through documents:

- ▶ `DocumentIdIterator`
- ▶ `ContentComponentIterator`

Both interfaces extend the `java.util.Iterator` interface and contain the following methods:

- ▶ `hasNext()`
Returns `true` if the iteration has more elements.
- ▶ `next()`
Returns the next element in the iteration. All objects returned from this iterator are instances of `DocumentID`.

Attention: The `remove()` method of the `java.util.Iterator` interface is not implemented in both interfaces and throws an `UnsupportedOperationException`.

12.3 Extending your Web site with JSP support

If you are planning to install IBM Workplace Web Content Management, chances are that you have existing Web resources (such as applications or business logic) that you want to integrate into your Web site. IBM Workplace Web Content Management provides a number of new features for developers to extend the capabilities of their Web content management implementations. This release includes the support for JavaServer Page (JSP) technology. JSPs enable you to

develop dynamic Web pages that you can integrate with existing business logic by using embedded Java code in HTML files. IBM Workplace Web Content Management provides JSP technology through the JSP component and a JSP tag library.

Designing for Portlets

When designing your IBM Workplace Web Content Management portlet applications, you generally use the MVC Model 2 architecture pattern. Like the servlet MVC implementation, the view is traditionally implemented using JSPs.

The JSP uses custom tags, which are user-defined JSP language elements that encapsulate reusable Java code. JSP tags are used to access components and content from IBM Workplace Web Content Management content easily. You can use the IBM Workplace Web Content Management JSP tag library in stand-alone JSP pages, which also includes JSP used within a portlet life cycle and IBM Workplace Web Content Management JSP components.

12.3.1 JSP components

The JSP component references a JSP from within the IBM Workplace Web Content Management framework. This reference provides site developers with a number of capabilities. For example, you can use this new JSP component to integrate existing JSPs into a IBM Workplace Web Content Management Web site. Therefore, if you have existing JSPs that are used for generating dynamic content or business logic, you can integrate these JSPs into IBM Workplace Web Content Management. In addition, the JSP can serve as a container for additional business logic or can reference a servlet or bean. Thus, you can use the JSP to contain IBM Workplace Web Content Management custom extensions, such as integration with other applications (backend data, transactional systems, or even Lotus Domino).

Note: All JSPs referenced using the new JSP component need to reside within the IBM Workplace Web Content Management application. Therefore, the JSP file needs to reside under the Web-app directory of the WebSphere Application Server on which the JSP is running.

Note: The JSP must be stored within the following directory of your IBM Workplace Web Content Management server:

`../AppServer/installedApps/[node-name]/wcm.ear/ilwwcm.war/jsp`

You might also need to store the JSP in the client war directory of the servlet or portlet that calls the JSP if you are using the IBM Workplace Web Content Management API.

Note: Syndication does not move the JSP that is referred to in a JSP component. Only the component itself is moved. You need to store a JSP on both the subscribing *and* the syndicating servers.

12.3.2 JSP tag library

In addition to the JSP component, IBM Workplace Web Content Management includes a JSP tag library that encapsulates the core functionality common to the IBM Workplace Web Content Management API. IBM Workplace Web Content Management ships with a library of JSP tags that you can use to display IBM Workplace Web Content Management components and content from an external JSP. These tags provide new menus in existing JSPs and applications, navigational components, or content. As with the JSP component, all JSPs must reside within the IBM Workplace Web Content Management application.

The JSP tag library is named `wcm.tld` and contains the following tags:

- ▶ `initworkspace`
- ▶ `setExplicitContext`
- ▶ `setContext`
- ▶ `content`
- ▶ `contentComponent`
- ▶ `libraryComponent`

The JSP tag library is located in the `\wcm.ear\ilwwcm.war\WEB-INF\tld` directory below the `installedApps` directory of the WebSphere Application Server. You can reference the library as shown in the following example:

```
<%@ taglib uri="/WEB-INF/tld/wcm.tld" prefix="wcm" %>
```

The tag library is not included in the Portlet projects that are created using the RAD Portlet Tools. For this reason, you need to add the `wcm.tld` file to portlet projects manually to avoid broken link warning messages during the development of JSP.

The initworkspace tag

The initworkspace tag is used to set the initial workspace. Each workspace is associated with a specified user. Therefore, the initworkspace tag contains parameters that provide a user name and password, as shown in Example 12-4.

Example 12-4 JSP tag library, initworkspace tag

```
<wcm:initworkspace username="<username>" password="<password>" >
Could not init workplace
</wcm:initworkspace>
```

If you use the JSP tag library in the WebSphere Portal context, you should use the `getUser()` property of the `PortalRequest` to get the current user name, as shown in Example 12-5.

Example 12-5 The initworkspace tag retrieves the user name from the portalRequest

```
<wcm:initworkspace user="<%= (java.security.Principal)portletRequest.getUser()
%>" />
```

The setExplicitContext tag

After you have initialized the IBM Workplace Web Content Management workplace, you then need to use the explicit context tag, `setExplicitContext`. This tag tells the JSP and application server the location of your IBM Workplace Web Content Management instance, as shown in Example 12-6.

Example 12-6 setExplicitContext tag syntax

```
<wcm:setExplicitContext wcmWebAppPath=" " wcmServletPath=" " path=" " >
[Error Message]
</wcm:setExplicitContext>
```

For our example, the JSP and IBM Workplace Web Content Management are located on the same server, because currently, the tag library is implemented for local Web container access to Web content management components. Therefore, our settings are as shown in Example 12-7.

Example 12-7 setExplicitContext within a stand-alone JSP

```
<wcm:setExplicitContext wcmWebAppPath="http://localhost:9080/content"
wcmServletPath="/connect" path="/Intranet">Error in explicit path to WCM
server</wcm:setExplicitContext>
```

When we use the JSP tag as a View component within a Portlet, it usually does not make sense to use static path information. Instead of static information, you should use Portal constancies, as shown in Example 12-8.

Example 12-8 setExplicitContent within WebSphere Portal context

```
<% String appPath =  
"http://" + portletRequest.getServerName() + ":" + portletRequest.getServerPort() + "/w  
ps/wcm"; %>  
  
<wcm:initworkspace user="<%= (java.security.Principal)portletRequest.getUser()  
%>" />  
<wcm:setExplicitContext wcmServletPath="/connect" wcmWebAppPath="<%=appPath%>"  
path="/RiverBend/Company">
```

Notice that Example 12-8 includes the `wcmWebAppPath` and `wcmServletPath` parameters. These parameters are optional. They are used to specify the servlet and context root of the IBM Workplace Web Content Management Web site. These parameters are important for the rewriting of URLs when the JSPs are located in a path that is different from the Web content. If you include `wcmWebAppPath`, you must also specify `wcmServletPath`.

The setContext tag

The `setContext` tag is very similar to the `setExplicitContext` tag. The `setExplicitContext` tag takes the content path as one of its parameters, while the `setContext` tag looks for the content path in the specified location (query, request, or session) with the specified key. The content path value that is either specified in the tag or retrieved from the request attributes or parameters or session is the same for either tag, as shown in the following example:

```
<wcm:setContext location=" " wcmWebAppPath=" " wcmServletPath=" " param=" " >
```

The content tag

The `Content` tag renders the content based on the current context. An optional parameter is provided to set a specified page design. The default page design that is associated with the content is used if the page design parameter is not set.

The following example renders the current content by using its default page design:

```
<wcm:content pageDesign=" " >
```

The contentComponent tag

The `contentComponent` tag is similar to the `ApatrixCmpnt` tag. The tag renders a Component from the current site, site area, or content object. The tag includes the two parameter types and names. Both parameters are equal to the `ApatrixCmpnt` tag. Thus, you can provide a specified field as key and content, site area, or site as value for the type as shown in the following example:

```
<wcm:contentComponent key="Summary" type="content" >
```

The tag shown in this example renders the Summary content component of the current content document into the JSP. The context has to be set before. In this example, Summary is a text component in the authoring template.

The libraryComponent tag

Like the contentComponent, the libraryComponent tag is very similar to the AprtixLibCmpnt tag. By using this tag, you are able to render different library components into the current JSP. The following example renders the MENU-Related Information library component for a previously specified context into the JSP:

```
<wcm:libraryComponent name="MENU-Related Information">
```

Note: You can find a complete description of the tag parameters for AprtixCmpnt and AprtixLibCmpnt in the IBM Workplace Web Content Management Information Center at:

<http://www-128.ibm.com/developerworks/workplace/documentation/webcontentmanagement/>

12.3.3 JSP tag library example

Now that we have reviewed the core tags for the library, the next step is to put a JSP together to test the functionality. We have a JSP directory within the Web application folder, so we can use that to store JSP files. Example 12-9 displays the Latest News menu component from the IBM Workplace Web Content Management server.

Example 12-9 Displaying the Latest News menu component

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
<%@ taglib uri="/WEB-INF/tld/wcm.tld" prefix="wcm" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" %>
<wcm:initworkspace username="Administrator" password="password">
  Could not get WCM workplace</wcm:initworkspace>
<wcm:setExplicitContext wcmWebAppPath=http://localhost:9080/content
wcmServletPath=
"/connect" path="/Intranet">Error in explicit path to WCM server
</wcm:setExplicitContext>

<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<TITLE>Latest News</TITLE>
The following is the latest news from the WCM server
</HEAD>
```

```

<BODY LEFTMARGIN=0 MARGINWIDTH=0 TOPMARGIN=0 MARGINHEIGHT=0

<!-- WCM component -->

<wcm:libraryComponent name="MENU-Site Area">
  Could not find Latest News Menu</wcm:libraryComponent>

<!-- End WCM component -->

</BODY>
</HTML>

```

When we save the JSP and run it from the Web browser, the JSP renders the Latest News menu from the IBM Workplace Web Content Management server as shown in Figure 12-5.



Figure 12-5 JSP preview

12.4 Setting up the development environment

This section describes how to install Rational® Application Developer 6 (RAD) and how to attach the development environment to a remote portal server. The remote portal server is already running the IBM Workplace Web Content Management task, as described in 3.2, “Installing IBM Workplace Web Content Management” on page 73.

Note: This section addresses assumes that you are not familiar with Rational Application Developer. If you have experience with portlet development and Rational Application Developer 6, you can skip to 12.4.2, “Installing the API Java files” on page 496.

12.4.1 Installing Rational Application Developer 6.0

You can install Rational Application Developer 6 (RAD) either from the installation CD-ROM or from a local or network directory. If you want to install from a directory make sure that you copy all files and directories from the first installation CD to a subdirectory called disk1. This subdirectory *must* be called disk1 and it must be in lower case. Create additional subdirectories to copy the contents of the other two CDs to subdirectories called disk2 and disk3. If you install from a network drive, map the drive locally mapped specifying a device name.

To install RAD, follow these steps:

1. Run **1launchpad.exe** from the root of the first installation CD to display the Rational Application Developer Installation Launchpad window shown in Figure 12-6.



Figure 12-6 RAD Launchpad window

2. From the launchpad, select **Install IBM Rational Application Developer V6.0**.

3. Wait for the installation wizard to open the Welcome window (as shown in Figure 12-7), and then click **Next** to continue the installation.

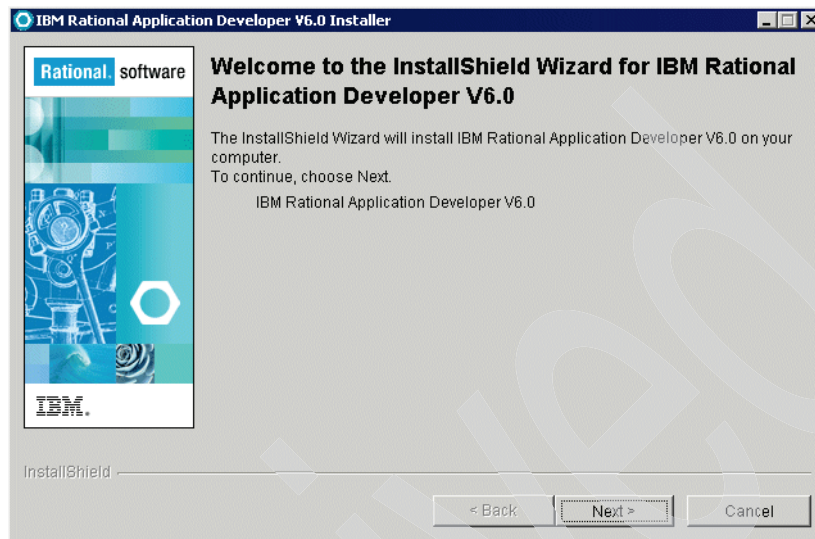


Figure 12-7 RAD Welcome window

4. Follow the instructions for reading and accepting the license agreement (Figure 12-8) and for specifying the target installation directory (Figure 12-9 on page 494).

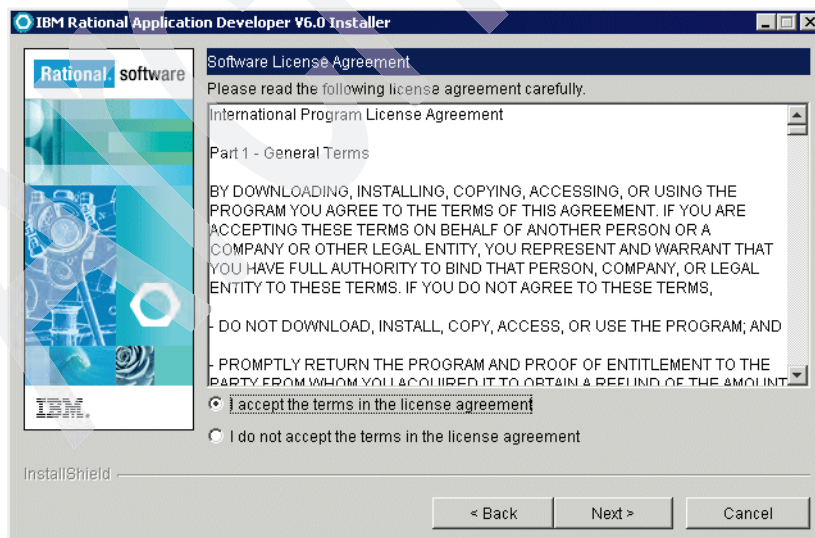


Figure 12-8 RAD License Agreement

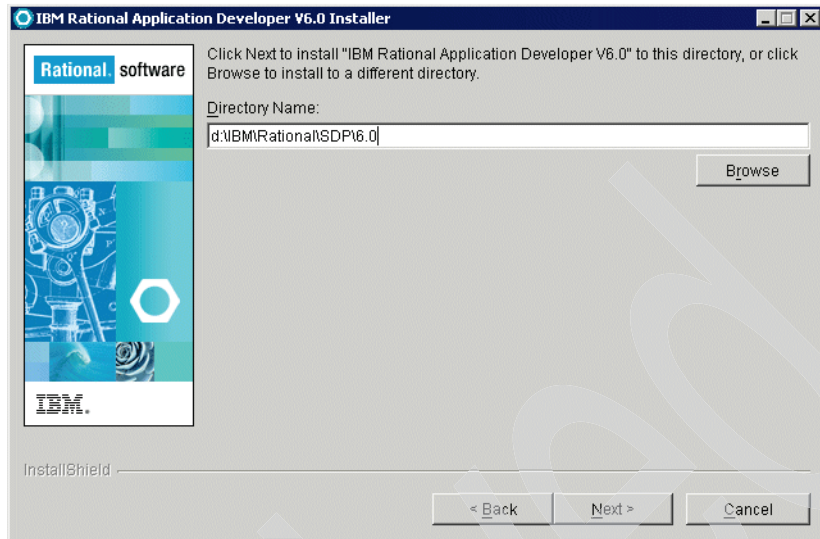


Figure 12-9 RAD target installation directory

5. In the Features window (Figure 12-10 on page 495), select the RAD features that you want to install. By default, the installation program installs the Integrated Development Environment and the optional WebSphere Application Server, Version 6.0 as the run-time environment for testing. If you want to install the user interface and documentation for supported national languages other than English, select to install the Language Pack.

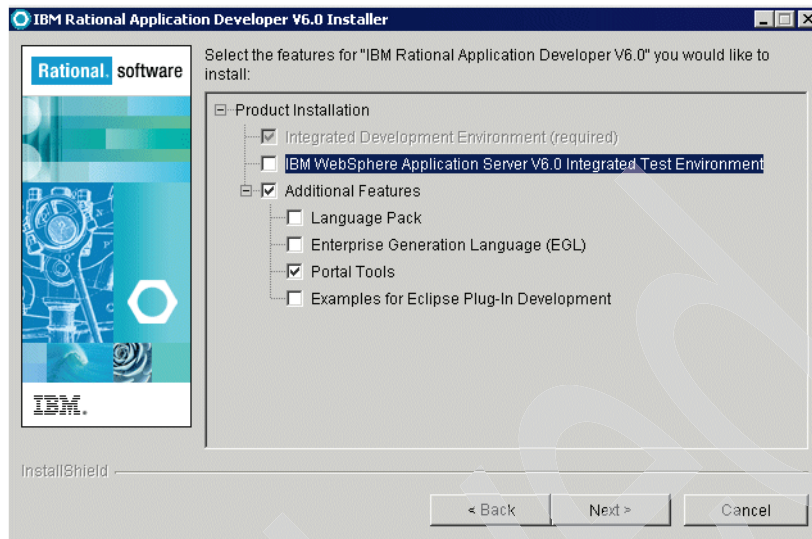


Figure 12-10 RAD - Options select

6. Figure 12-11 shows a summary of the selected options. Review the options and click **Next** to continue.

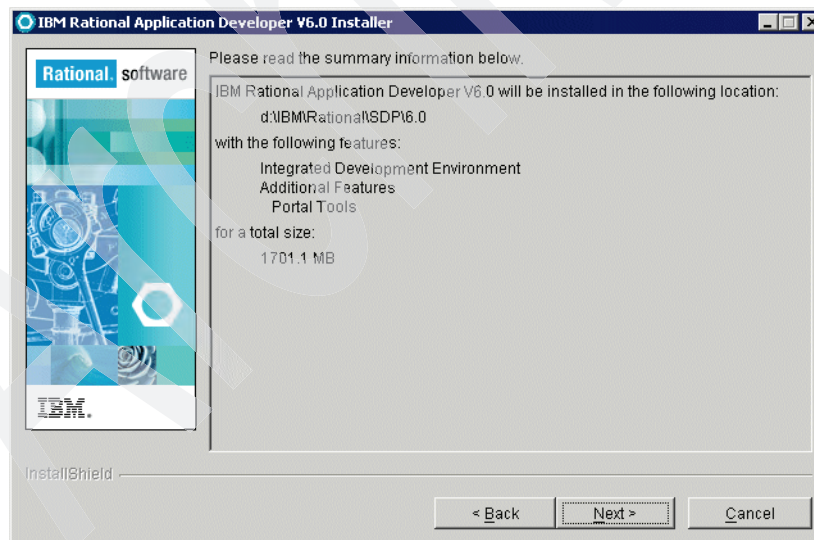


Figure 12-11 RAD summary

7. After you have installed the core product, you can install the Agent Controller (Figure 12-12). You can restart the installation launchpad at anytime if you want to install the Agent Controller later.

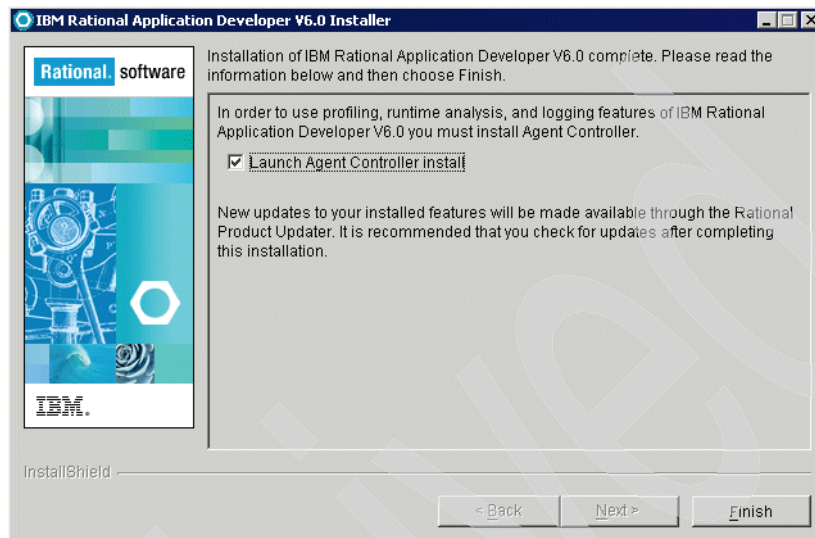


Figure 12-12 RAD, installing the Agent Controller

Tip: We recommend that you apply the available fixpacks to the RAD installation. In the examples in this book, we use V6.0.0.1 Interim Fix 002.

12.4.2 Installing the API Java files

When the installation is complete, RAD includes the necessary shared libraries for WebSphere Application Server and WebSphere Portal Server. The installation can also include optional installed test environments for these servers. If you install the WebSphere Portal 5.1 test environment, the shared libraries for IBM Workplace Web Content Management are *not* installed. You have to install these IBM Workplace Web Content Management API .jar files manually from an existing full installation (for example, your local development environment). We recommend that you copy all files to a new directory below the runtimes directory (for example, D:\IBM\Rational\SDP\6.0\runtimes\IWWCM).

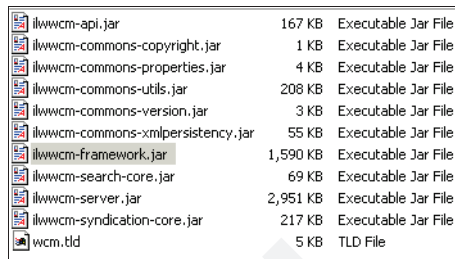
You need the .jar files that contain the IBM Workplace Web Content Management server and the JSP tag library definition. The jar files are stored in the following directory:

`<Drive>:\<Path>\WebSphere\PortalServer\wcm\shared\app`

The tag library definition is named wcm.tld and is located under the IWWCM.war directory as follows:

```
<Drive>:\<Path>\WebSphere\AppServer\installedApps\<Node>\wcm.ear\
ilwwcm.war\WEB-INF\tld
```

Copy all files to the new directory below the runtimes directory. If you copy all files to this directory, the directory should contain the files as illustrated in Figure 12-13.



ilwwcm-api.jar	167 KB	Executable Jar File
ilwwcm-commons-copyright.jar	1 KB	Executable Jar File
ilwwcm-commons-properties.jar	4 KB	Executable Jar File
ilwwcm-commons-utils.jar	208 KB	Executable Jar File
ilwwcm-commons-version.jar	3 KB	Executable Jar File
ilwwcm-commons-xmipersistence.jar	55 KB	Executable Jar File
ilwwcm-framework.jar	1,590 KB	Executable Jar File
ilwwcm-search-core.jar	69 KB	Executable Jar File
ilwwcm-server.jar	2,951 KB	Executable Jar File
ilwwcm-syndication-core.jar	217 KB	Executable Jar File
wcm.tld	5 KB	TLD File

Figure 12-13 Java files for the IBM Workplace Web Content Management API

12.4.3 Attaching to a remote server

You can install WebSphere Portal 5.1 as an integrated test environment into RAD. However, this WebSphere Portal 5.1 test environment installation does not install the necessary IBM Workplace Web Content Management functionality. Therefore, you cannot use an integrated test environment for the IBM Workplace Web Content Management API development. Instead, you can use a full-featured WebSphere Portal server that is attached as a remote server to the RAD.

The following example explains how to attach a remote Portal server. The section does not cover the necessary configuration of the WebSphere Application Server to use its remote debug possibilities. Refer to Chapter 16 of *IBM WebSphere Portal V5 - A Guide for Portlet Application Development*, SG24-6076, for a detailed description on how to set up remote debugging. To attach a remote Portal server, do the following:

1. If needed, start the RAD.
2. Switch to a perspective that includes the Server tab. (for example, the Web or J2EE perspective).
3. Right-click the background of the Server tab, and select **New Server** in the context menu (Figure 12-14).

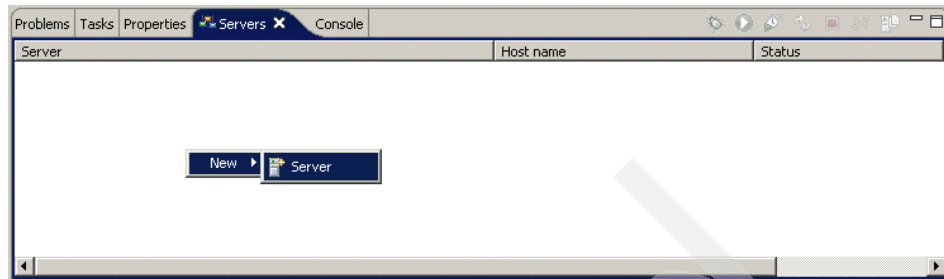


Figure 12-14 Context menu, New Server

4. A new window displays to allow you to define the new server configuration (Figure 12-15). Do the following:
 - a. Enter or select the full qualified host name of your remote WebSphere Portal server.
 - b. Select **WebSphere Portal 5.1 Server Attach** as the server type.
 - c. Click **Next** to continue.

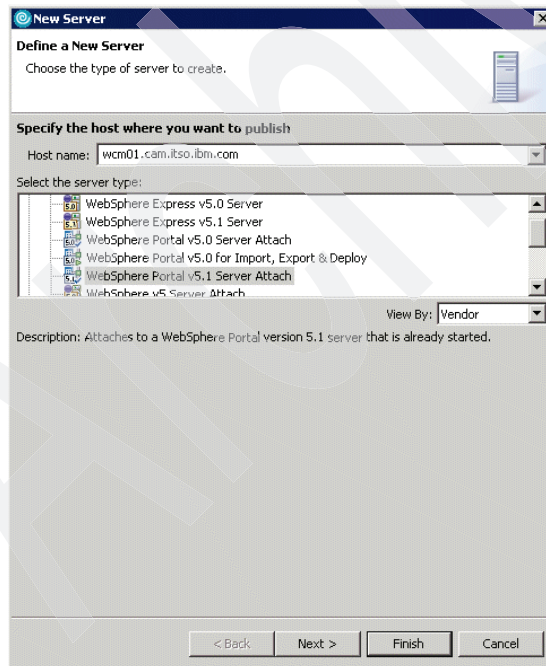


Figure 12-15 New server dialog

5. The Server Ports window displays (Figure 12-16). If you have not changed the ports configuration of your WebSphere Portal, keep the default settings by clicking **Next**.

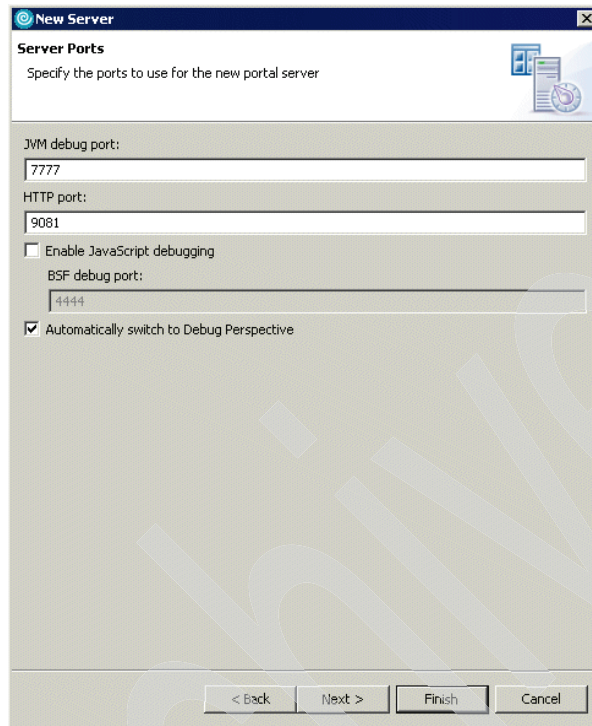
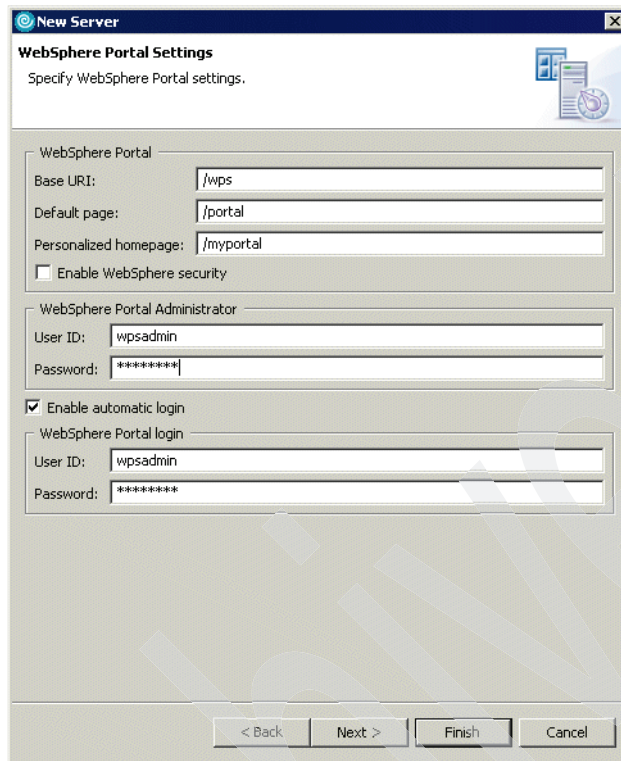


Figure 12-16 WebSphere Portal setting, server ports

6. A new dialog box displays to allow you to define your WebSphere Portal server settings (Figure 12-17 on page 500). Generally, you need to specify only the name and password of your Portal user.



New Server
WebSphere Portal Settings
Specify WebSphere Portal settings.

WebSphere Portal

Base URI: /wps
Default page: /portal
Personalized homepage: /myportal
☐ Enable WebSphere security

WebSphere Portal Administrator

User ID: wpsadmin
Password: *****

☒ Enable automatic login

WebSphere Portal login

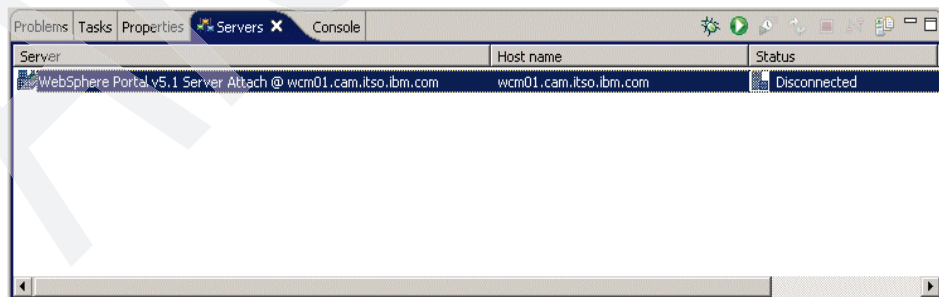
User ID: wpsadmin
Password: *****

< Back Next > Finish Cancel

Figure 12-17 WebSphere Portal setting, preview server

7. Click **Finish**.

The attached remote server is added to your server list, as shown in Figure 12-18.



Server	Host name	Status
WebSphere Portal v5.1 Server Attach @ wcm01.cam.itso.ibm.com	wcm01.cam.itso.ibm.com	Disconnected

Figure 12-18 Attached server

12.5 Examples of API development

This section contains two simple examples that illustrate development with the IBM Workplace Web Content Management API.

The first example provides good starting point for developers who are not familiar with the development of IBM Workplace Web Content Management portlets.

The second example integrates JSP component into an existing IBM Workplace Web Content Management Web site or portal. This example does not use the API itself.

Note: The IBM Workplace Web Content Management installation contains further API examples. You can find the code for these examples at:

```
<Drive>:\<Path>\PortalServer\wcm\samples\API\doc\  
wcm-5-1-0-1-api-samples.htm
```

12.5.1 Developing a simple Portlet

This example creates a simple portlet that displays the content of a specified site area. The portlet is based on the basic portlet. It contains an edit mode and, of course, the view mode. In the edit mode, the user can select a site area. The selected site area is saved as portlet data. The view mode fetches the data from the portlet data and displays the content that is associated with the selected site area.

Figure 12-19 shows the sequence of actions in this example.

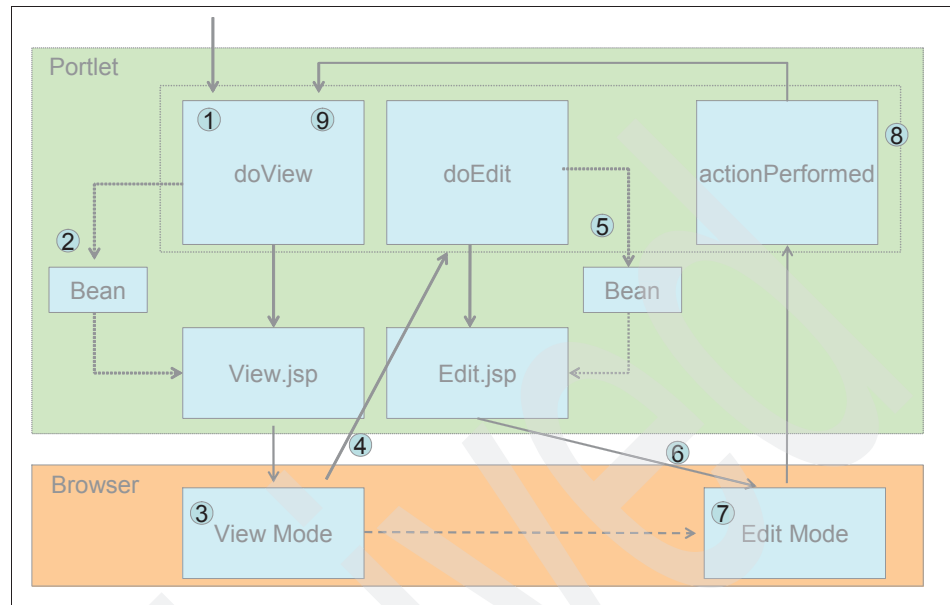


Figure 12-19 SimpleDisplay Portlet action sequence

The steps in this example are as follows:

1. The `doView` method is executed.
2. A JSP is called to render an initial screen.
A message is obtained as `JavaBean` from the request object. `JavaBeans™` are a special type of Java class that contain the business logic of the application. They are used to store and process data temporarily and to access backend resources such as databases.
3. The Portlet View mode screen is shown in the browser window.
4. The user clicks **Edit** to switch to Edit mode.
5. The `doEdit` method is executed. A message is obtained from the request object.
6. The Edit mode screen displays.
7. The user selects the desired site area and clicks **Submit**.
8. The `actionPerformed` method is executed to process the action. A message is stored in the `PortletData`.
9. The `doView` method is executed to complete the cycle.

Note: The IBM Workplace Web Content Management is integrated in WebSphere Portal since version 5.1. Therefore, we expect that some readers of this book who have already experience with IBM Workplace Web Content Management might not have extensive experience in the portlet development. For that reason, we start our IBM Workplace Web Content Management portlet development example from scratch.

Our goal is to provide a complete example that every developer can follow. If you have skills in portlet development, you can move through the first step quickly. We highly recommend that you read *IBM Rational Application Developer V6 - Portlet Application Development and Portal Tools*, SG24-6681, for a detailed introduction into portlet development.

Step 1: Creating a basic portlet

To create a basic portlet, do the following:

1. Start RAD if necessary.
2. Click **File** → **New** → **Portlet Project** to create a new portlet project.

The New Portlet Project window appears (Figure 12-20).

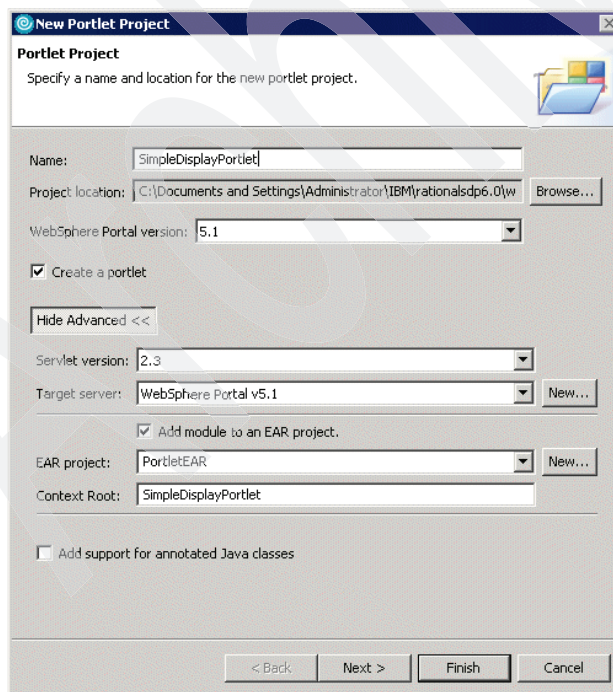


Figure 12-20 New Portlet Project window

3. Enter the following information:
 - a. Name: SimpleDisplayPortlet
 - b. WebSphere Portal version: 5.1
 - c. Servlet version: 2.3
 - d. Target server: WebSphere Portal V5.1
4. Select the following options:
 - a. Create a portlet
 - b. Add module to EAR project option
5. Select an EAR project or create a new project.
6. Click **Next**.

The Portlet Type windows displays (Figure 12-21).

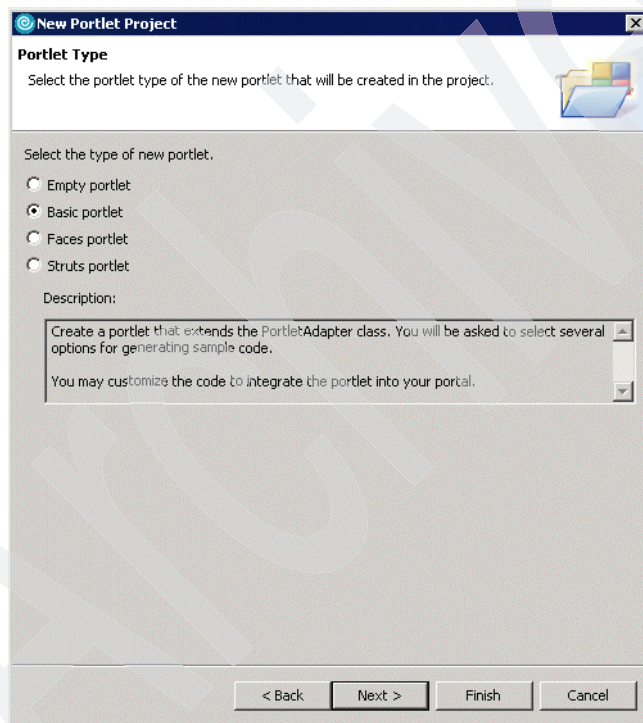


Figure 12-21 Portlet Type window

7. Select **Basic portlet** and click **Next**.

The Features window displays (Figure 12-22).

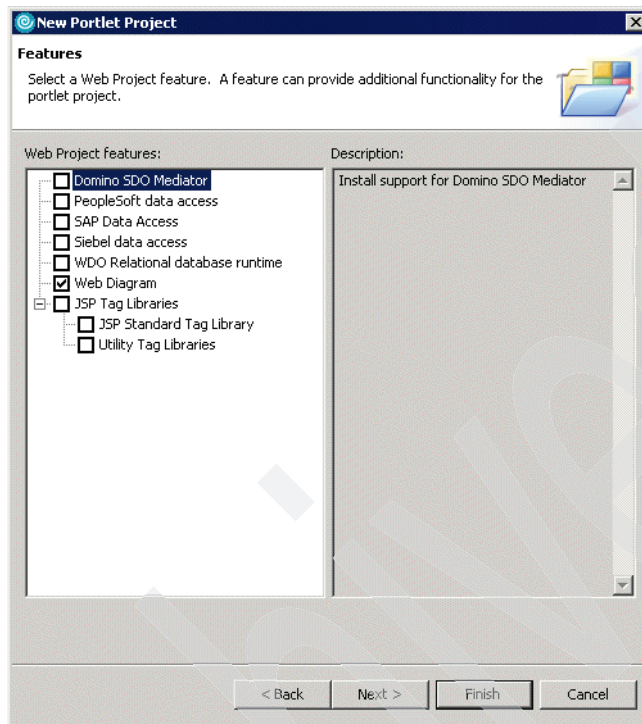


Figure 12-22 Features window

8. Keep the default settings and click **Next**.

The Portlet Settings window displays.

New Portlet Project

Portlet Settings
Define the general settings of the portlet.

General
Application name: SimpleDisplayPortlet application
Portlet name: SimpleDisplayPortlet portlet

Internationalization
Default locale: en English
Portlet title: SimpleDisplayPortlet portlet

Code generation options
☐ Change code generation options
Package prefix: simplifiedisplayportlet
Class prefix: SimpleDisplayPortletPortlet

< Back Next > Finish Cancel

Figure 12-23 Portlet Settings window

9. Keep the default settings again and click **Next**.

The next window (Figure 12-24) contains the event handling settings.

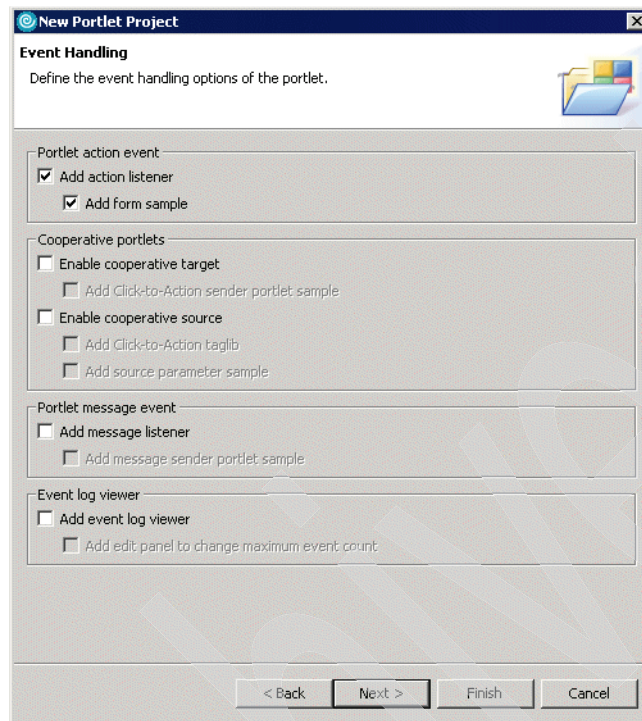


Figure 12-24 Event Handling window

10. Select **Add action listener** and **Add from sample**.
11. Click **Next**.

The Single Sign-On window appears.

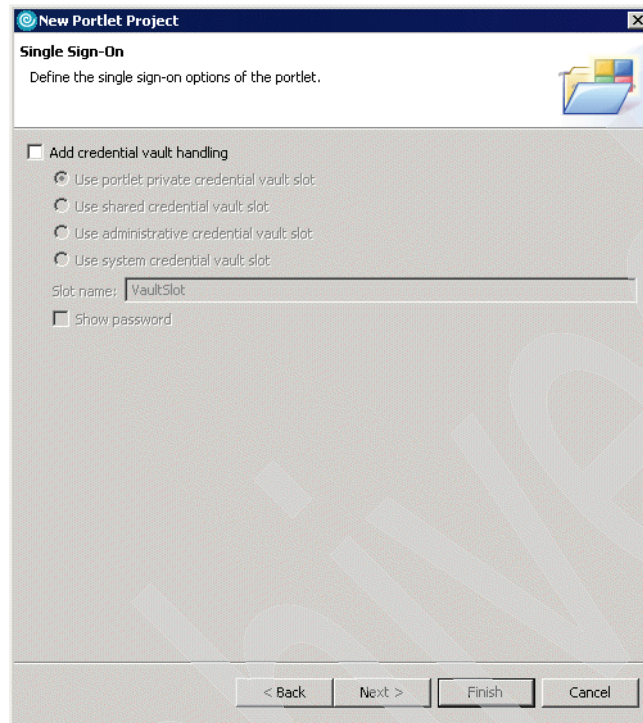


Figure 12-25 Single Sign-On window

12. Keep the default settings because you do not use any credentials in this example.
13. Click **Next**.

14. Select **Add edit mode** to add the sample code for the portlet edit mode to the new project (Figure 12-26).

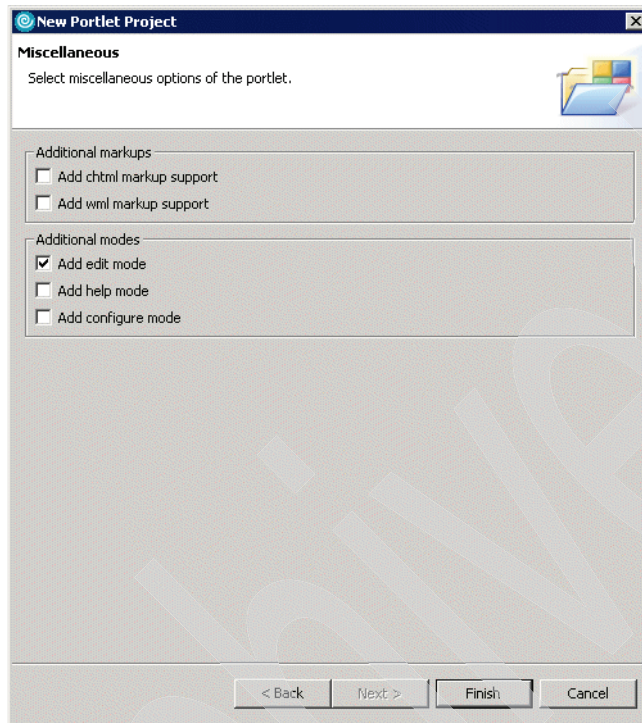


Figure 12-26 Miscellaneous windows

15. Click **Finish**.

The Portal-Tool wizard creates a new portlet project. You can find the project under Dynamic Web Projects in the Web perspective. Figure 12-27 illustrates the new project expanded in the Web perspective.

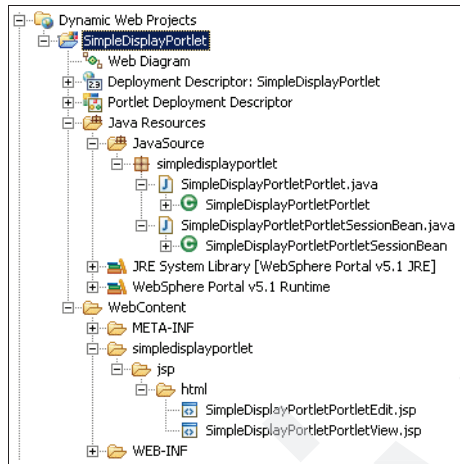


Figure 12-27 Web perspective, new portlet project.

16. Make sure that your development environment is set up correctly by selecting the new project. Right-click to open the context menu and select **Run** → **Run on server...** as illustrated in Figure 12-27.

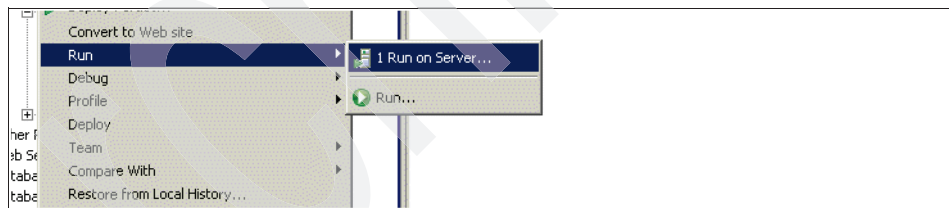


Figure 12-28 Run on Server...

The Server Selection window appears (Figure 12-29).

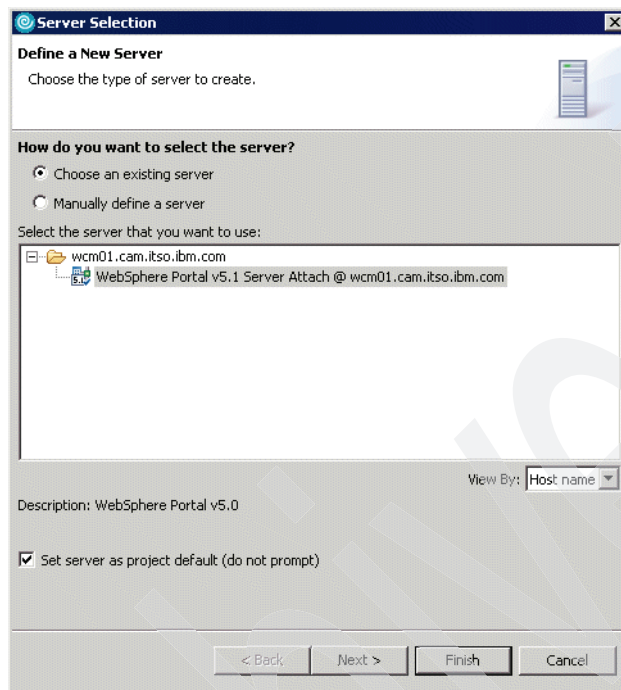


Figure 12-29 Select server

17. Select the WebSphere Portal Server V5.1 that you set up in 12.4, “Setting up the development environment” on page 491. Optionally, you can select **Set server as project default**.
18. Click **Finish**.

After the portlet deploys to the WebSphere Portal server, the Web browser shows the basic portlet within the portlet preview. If the portlet is displayed as illustrated in Figure 12-30 the environment is set up correctly.

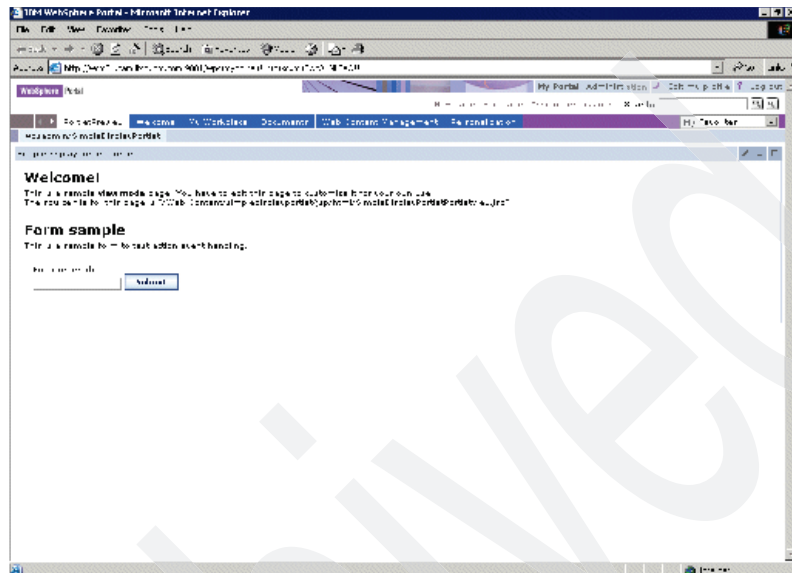


Figure 12-30 Portlet preview

Step 2: Extending the Java Build path

Before you can develop the IBM Workplace Web Content Management API portlet example, you have to extend the Java Build path and add the API libraries. To do this, follow these steps:

1. Expand the Dynamic Web Project Category of the Web perspective.
2. Right-click the SimpleDisplayPortlet project and select **Properties** from the context menu.

The Properties for SimpleDisplayPortlet appears (Figure 12-31).

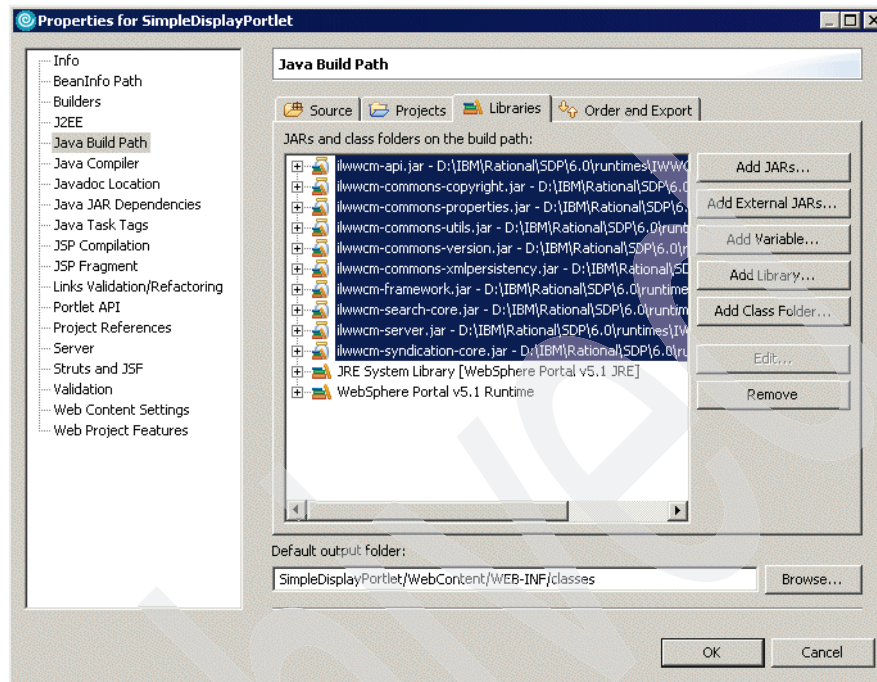


Figure 12-31 Java Build Path Settings

3. Select **Java Build Path** → **Libraries**.
4. Select **Add External JARs...**

5. Select all the IBM Workplace Web Content Management .jar files in your local runtimes directory of your RAD installation. 12.4.2, “Installing the API Java files” on page 496 provides details about these files.

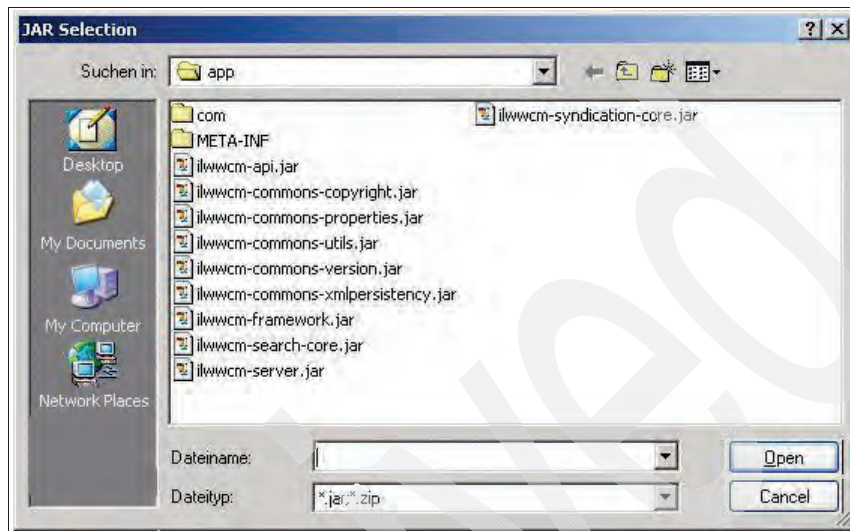


Figure 12-32 Select external JAR files

6. Click **OK**.
7. Extend the folders WebContent → WEB-INF and create a new folder named tld below the WEB-INF folder using the **File** → **New** → **Folder** menu. This new directory contains the JSP tag library definition file.

8. Select the new folder and import this definition file by selecting **File** → **Import**.
9. Select File system and click **Next**.
The Import windows appears (Figure 12-33).

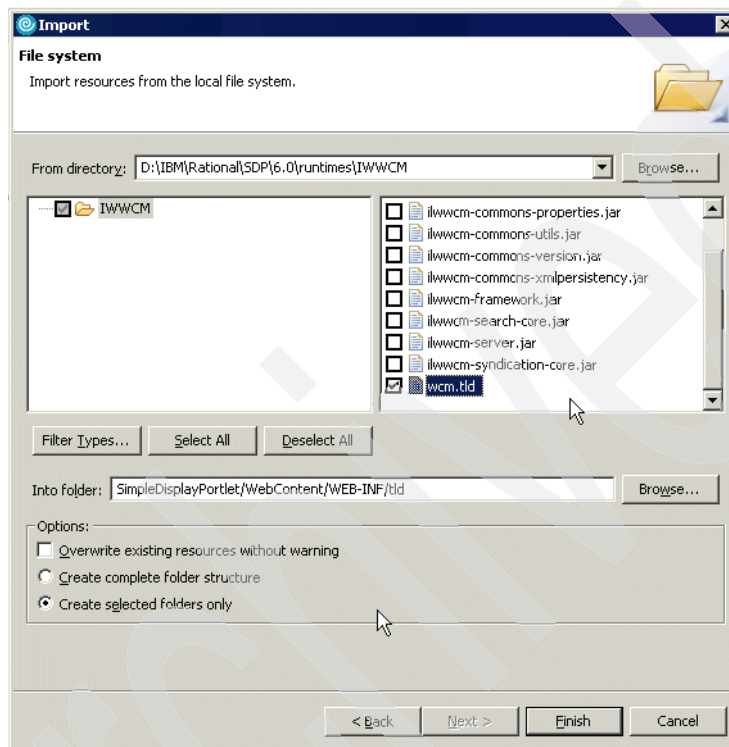


Figure 12-33 Importing wcm.tld

10. Browse to your runtime directory (as described in 12.4.2, “Installing the API Java files” on page 496) and select the wcm.tld file.
11. Click **Finish**.

Step 3: Developing the Portlet

After you have finished all the configuration steps that are necessary to create the portlet with IBM Workplace Web Content Management functionality, you can start writing the Java code. Because you used the basic portlet option to create a new portlet project, all required Java classes and JSP are already created by the portlet wizard. The class and JSP contain sample code that you need to replace with your code.

You need to develop the following code for this example:

- ▶ JSP for the edit mode of the portlet, `Edit.jsp`, which allows us to select a single site area
- ▶ JSP for the view mode of our portlet, `View.jsp`. This page uses the IBM Workplace Web Content Management JSP tag library to display the default content of the selected site area. The path to the content is retrieved as a string from the session.
- ▶ The `doEdit` method that iterates through all sites and gathers the names of all site areas. The `ArrayList` is added to the session and passed to the edit JSP.
- ▶ The `actionPerformed` method that saves the selected site area to the `PortalData`.
- ▶ The `doView` methods that search for the default content of the selected site area, that calculate the path to the content, and that add this path as string to session.
- ▶ A `JavaBean` that is added to the session in edit mode.
- ▶ A `JavaBean` that is stored in the `PortletData`.

Before you can display any content document, you need to select a site area and to store this area in the `PortletData`. Thus, you develop the Java code in the following order:

1. `SessionBean`
2. `doEdit` method
3. Edit mode JSP
4. `PortletData Bean`
5. `actionPerformed` method
6. `doView` methods
7. View mode JSP

SessionBean

The session bean `SimpleDisplayPortletPortletSessionBean` stores an array list of all site area document IDs and of all corresponding content paths. In addition to these two fields, the session bean contains corresponding get and set methods. Figure 12-10 illustrates the Java code of the session bean.

Example 12-10 SimpleDisplayPortletPortletSessionBean

```
package simpledisplayportlet;
import java.util.ArrayList;

public class SimpleDisplayPortletPortletSessionBean {

    private ArrayList contentPaths = null;
    private ArrayList siteAreaIds = null;

    public ArrayList getContentPaths() {
        return contentPaths;
    }

    public void setContentPaths(ArrayList contentPaths) {
        this.contentPaths = contentPaths;
    }

    public ArrayList getSiteAreaIds() {
        return siteAreaIds;
    }

    public void setSiteAreaIds(ArrayList siteAreaIds) {
        this.siteAreaIds = siteAreaIds;
    }
}
```

doEdit method

The first method that you develop is the `doEdit` method. This method gathers all document IDs and content paths of all site area's and stores both lists in the session bean. The lists are stored as array list. Thus, you need to import the array list class using the import statement that is shown in the first line of Example 12-11. The other two import statements in this example allow you to program using the IBM Workplace Web Content Management API.

Example 12-11 SimpleDisplayPortletPortlet class import

```
import java.util.ArrayList;
import com.ibm.workplace.wcm.api.*;
import com.ibm.workplace.wcm.api.exceptions.*;
```

Example 12-12 shows the entire source code of the `doEdit` method. The method does the following:

1. Before you can access any IBM Workplace Web Content Management functionality, you need to obtain a workspace for the current user.
2. Use the `Workspace.findByType` method to create a document ID iterator that contains the document IDs of all site documents within the site framework.
3. The `doEdit` method should gather all site areas of each site. Therefore, you use the `getAllChildren` method to create a second iterator. This second iterator contains the document IDs of all child documents below the current site. The children are either site area or content. You only continue for containers that match the document type `SiteArea`.
4. To show the hierarchal list of site areas in a user friendly way, you want to display the site area with the full path. You can build the full path by iterating through all parent site areas of the current site area. Each element in this path is concatenated with a forward slash.
5. At the end, you add the created array lists to the session bean.

Example 12-12 doEdit method

```
public void doEdit(PortletRequest request, PortletResponse response) throws PortletException,
IOException {
    SimpleDisplayPortletPortletSessionBean sessionBean = getSessionBean(request);
    //1. get the workspace
    Workspace ws = null;
    Repository repository = null;
    try {
        repository = WCM_API.getRepository();
        ws = repository.getWorkspace((java.security.Principal)request.getUser());
    } catch (ServiceNotAvailableException e) {
        e.printStackTrace();
    } catch (OperationFailedException e) {
        e.printStackTrace();
    }
    }

    ArrayList contentPaths = new ArrayList();
    ArrayList siteAreaIds = new ArrayList();

    //2. get all sites
    DocumentIdIterator siteIterator = ws.findByType(DocumentTypes.Site);
    Site siteDocument = null;
    while (siteIterator.hasNext()){
        try {
            siteDocument = (Site)ws.getById((DocumentId) siteIterator.next());
        } catch (DocumentRetrievalException e1) {
            e1.printStackTrace();
        } catch (AuthorizationException e1) {
```

```

        e1.printStackTrace();
    }

    //3. get all site areas for the current site
    DocumentIdIterator siteareaIterator = siteDocument.getAllChildren();
    DocumentId siteareaID = null;
    while (siteareaIterator.hasNext()){
        siteareaID = (DocumentId)siteareaIterator.next();
        if (siteareaID.isOfType(DocumentTypes.SiteArea)) {
            try {
                //4. Create site area path
                StringBuffer contentPath = new StringBuffer();
                SiteArea sitearea;
                sitearea = (SiteArea)ws.getById(siteareaID);
                contentPath.append("/"+sitearea.getName());
                DocumentId parentID = sitearea.getParent();

                while (parentID!=null) {
                    if (parentID.isOfType(DocumentTypes.SiteArea)) {
                        SiteArea siteareaParent = (SiteArea)ws.getById(parentID);
                        contentPath.insert(0,"/"+siteareaParent.getName());
                        parentID = siteareaParent.getParent();
                    } else {
                        contentPath.insert(0,"/"+siteDocument.getName());
                        parentID = null;
                    }
                }
                siteAreaIds.add(siteareaID.toString());
                contentPaths.add(contentPath.toString());
            } catch (DocumentRetrievalException e2) {
                e2.printStackTrace();
            } catch (AuthorizationException e2) {
                e2.printStackTrace();
            }
        }
    }
}

//5. set the session bean
repository.endWorkspace();
sessionBean.setContentPaths(contentPaths);
sessionBean.setSiteAreaIds(siteAreaIds);

getPortletConfig().getContext().include(EDIT_JSP+getJspExtension(request), request,
response);
// The Source will be with you, always.
}

```

Edit mode JSP

The edit mode JSP, Edit.jsp, retrieves the two army lists from the session bean. After that, you go through the lists and build an HTML SELECT field. Each site area creates one OPTION entry in the select box. The select box shows all the Web site paths and uses the string representation of the corresponding document ID as value, as shown in Example 12-13

Example 12-13 SimpleDisplayPortletPortletEdit.jsp

```
<%@ page session="false" contentType="text/html"
import="org.apache.jetspeed.portlet.*,simplifiedisplayportlet.*,java.util.*" %>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>

<% SimpleDisplayPortletPortletSessionBean sessionBean =
(SimpleDisplayPortletPortletSessionBean)portletRequest.getPortletSession().getAttribute(SimpleD
isplayPortletPortlet.SESSION_BEAN); %>
<H2>Please select a sitearea</H2>
<form method="POST" action="<portletAPI:createReturnURI><portletAPI:URIAction
name='<%=SimpleDisplayPortletPortlet.EDIT_ACTION%'/></portletAPI:createReturnURI>"
Site Area:
<SELECT name="siteArea">
<%
    if (sessionBean != null) {
        ArrayList paths = sessionBean.getContentPaths();
        ArrayList ids = sessionBean.getSiteAreaIds();
        String value = "";
        String name = "";
        if (paths != null && ids != null) {
            Iterator values = ids.iterator();
            Iterator names = paths.iterator();
            while (names.hasNext() && values.hasNext()) {
                value = (String) values.next();
                name = (String) names.next();
            }
        }
    }
%>
<OPTION value="<%=value%>"> <%=name%></OPTION>
<%
    }
}
%>
</SELECT><BR>
<INPUT type="submit" name="Submit" value="Submit">
</form>
```

PortletData Bean

The PortletData bean contains the document ID of the selected site area. The bean is used to store the selected data to the PortletData. The class that represents the Java bean needs to implement the Serializable interface. Instead of modifying existing code, you need to create a new class for this Java bean. As shown in Example 12-14, the class contains only a private string field and a get and set method.

Example 12-14 ContentBean class

```
package simpledisplayportlet;

import java.io.Serializable;

public class ContentBean implements Serializable {
    private String siteAreaId = null;

    public String getSiteAreaId() {
        return siteAreaId;
    }

    public void setSiteAreaId(String siteAreaId) {
        this.siteAreaId = siteAreaId;
    }
}
```

actionPerformed method

The final development step in the edit process of the portlet is the actionPerformed method. So far, you have developed the code that allows you to create a list of site areas and to pass this list to the edit JSP. You have created the JSP that displays in edit mode and the bean that stores the data. The missing step is the function that reads the data and stores it in the created bean. Example 12-15 shows the actionPerformed method.

The actionPerformed method is executed when the edit JSP is saved. You retrieve the value of the siteArea field from the request and store the value in the PortletData bean.

Example 12-15 actionPerformed method

```
public void actionPerformed(ActionEvent event) throws PortletException {
    if( getPortletLog().isDebugEnabled() )
        getPortletLog().debug("ActionListener - actionPerformed called");

    String actionString = event.getActionString();
    PortletRequest request = event.getRequest();
```

```

SimpleDisplayPortletPortletSessionBean sessionBean = getSessionBean(request);

if( EDIT_ACTION.equals(actionString) ) {
    if( request.getParameter("siteArea")!=null ) {
        PortletData data = request.getData();
        if( data!=null ) {
            ContentBean cb = new ContentBean();
            cb.setSiteAreaId(request.getParameter("siteArea"));
            data.setAttribute(NewPortletPortlet.WCM_CONTENT, cb);
            try {
                data.store();// save to data store
            }
            catch (IOException ioe) {
                if( getPortletLog().isErrorEnabled() )
                    getPortletLog().error("Error on PortletData.store(): "+ioe.getMessage());
            }
        }
    }
}
}

```

The PortletData bean can be accessed by using a defined key. The following example shows the definition of this key as a static string:

```
public static final String WCM_CONTENT = "contentPath";
```

The doView method

You now need to implement the part that shows the content in the portlet, as shown in the code example in Example 12-16 on page 523. The important steps within this function are the following:

1. You need to access the workplace for the current user again.
2. You retrieve the content bean from the PortletData. If the content bean cannot be retrieved, an error message is generated, and the doView method is left.
3. The SiteArea document is searched by the document ID that is stored in the content bean.
4. The view JSP can access content documents only by their path. Therefore, you go through all parent site area documents of this content document and create the path.
5. You add the generated path as session bean to the request.
6. You invoke the JSP to render.

Example 12-16 The doView method

```
public void doView(PortletRequest request, PortletResponse response) throws PortletException,
IOException {
    SimpleDisplayPortletPortletSessionBean sessionBean = getSessionBean(request);
    if( sessionBean==null ) {
        response.getWriter().println("<b>NO PORTLET SESSION YET</b>");
        return;
    }
    //1. get the workspace
    Workspace ws = null;
    Repository repository = null;
    try {
        repository = WCM_API.getRepository();
        ws = repository.getWorkspace((java.security.Principal)request.getUser());
    } catch (ServiceNotAvailableException e) {
        e.printStackTrace();
    } catch (OperationFailedException e) {
        e.printStackTrace();
    }
    //2. get the site area from the content bean
    ContentBean cb =
(ContentBean)request.getData().getAttribute(SimpleDisplayPortletPortlet.WCM_CONTENT);
    if( cb==null ) {
        response.getWriter().println("<b>NO CONTENTBEAN YET</b>");
        return;
    }
    try {
        //get the defaultcontent for the site area
        SiteArea sitearea = (SiteArea)ws.getById(ws.createDocumentId(cb.getSiteAreaId()));
        Site site = (Site)ws.getById(sitearea.getParent());
        Content content = (Content)ws.getById(sitearea.getDefaultContent());

        //create the path
        StringBuffer contentPath = new StringBuffer();
        contentPath.append("/"+sitearea.getName());
        contentPath.append("/"+content.getName());

        DocumentId parentID = sitearea.getParent();

        while (parentID!=null) {
            if (parentID.isOfType(DocumentTypes.SiteArea)) {
                SiteArea siteareaParent = (SiteArea)ws.getById(parentID);
                contentPath.insert(0,"/"+siteareaParent.getName());
                parentID = siteareaParent.getParent();
            } else {
                contentPath.insert(0,"/"+site.getName());
                parentID = null;
            }
        }
    }
}
```

```

    }

    //5. add the pathstring to the request
    request.setAttribute(SimpleDisplayPortletPortlet.WCM_CONTENT, contentPath.toString());
} catch (DocumentRetrievalException e1) {
    e1.printStackTrace();
} catch (AuthorizationException e1) {
    e1.printStackTrace();
} catch (DocumentIdCreationException e1) {
    e1.printStackTrace();
} catch (PropertyRetrievalException e) {
    e.printStackTrace();
}

//6. Invoke the JSP to render
getPortletConfig().getContext().include(VIEW_JSP+getJspExtension(request), request,
response);
}

```

View mode JSP

The View mode JSP retrieves the `SessionBean` from the request (Example 12-17). The IBM Workplace Web Content Management workspace is set for the current user, and the context is set to the path that was stored in the `SessionBean`. Finally, the content is rendered.

Example 12-17 SimpleDisplayPortletPortletView.jsp

```

<%@ page session="true" contentType="text/html" import="java.util.*, simpledisplayportlet.*,
java.security.Principal"%>

<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<%@ taglib uri="/WEB-INF/tld/wcm.tld" prefix="wcm" %>
<portletAPI:init/>

<% SimpleDisplayPortletPortletSessionBean sessionBean =
(SimpleDisplayPortletPortletSessionBean)portletRequest.getPortletSession().getAttribute(SimpleD
isplayPortletPortlet.SESSION_BEAN); %>

<% String content =
(String)portletRequest.getAttribute(SimpleDisplayPortletPortlet.WCM_CONTENT); %>
<% String appPath =
"http://" + portletRequest.getServerName() + ":" + portletRequest.getServerPort() + "/wps/wcm"; %>

<wcm:initworkspace user="<%= (java.security.Principal)portletRequest.getUser() %>" />
<wcm:setExplicitContext wcmServletPath="/connect" wcmWebAppPath="<%=appPath%>"
path="<%=content%>">
<%= error %>
</wcm:setExplicitContext>
<wcm:content pageDesign="PortalPageLayout" >Content error - <%= error %></wcm:content>

```

Note: We assume that you use this example with the River Bend example Web site and use the PortalPageLayout as the page design during the content rendering. If you use this example in a different context, you use a different page design. You also can leave this JSP parameter empty and use the default presentation template of the content.

Step 4: Previewing the Portlet

Finally, you can preview the portlet by following these steps:

1. Right-click the SimpleDisplayPortlet project and chose **Run** → **Run on Server...** from the context menu.
2. Select the previously configured WebSphere Portal Server (as described in 12.4.3, “Attaching to a remote server” on page 497).

The new portlet is deployed to WebSphere Portal. During the deployment, a new portal preview page is created, and the portlet is added to this page.

After these steps the browser shows and displays the new portlet. Because you have not selected a site area yet, the NO CONTENTBEAN YET error displays.

3. Click the small wrench symbol to switch to edit mode.
4. As shown in Figure 12-34, the Edit.jsp with the **option select box appears. Select a site area from the select box and click Submit.**

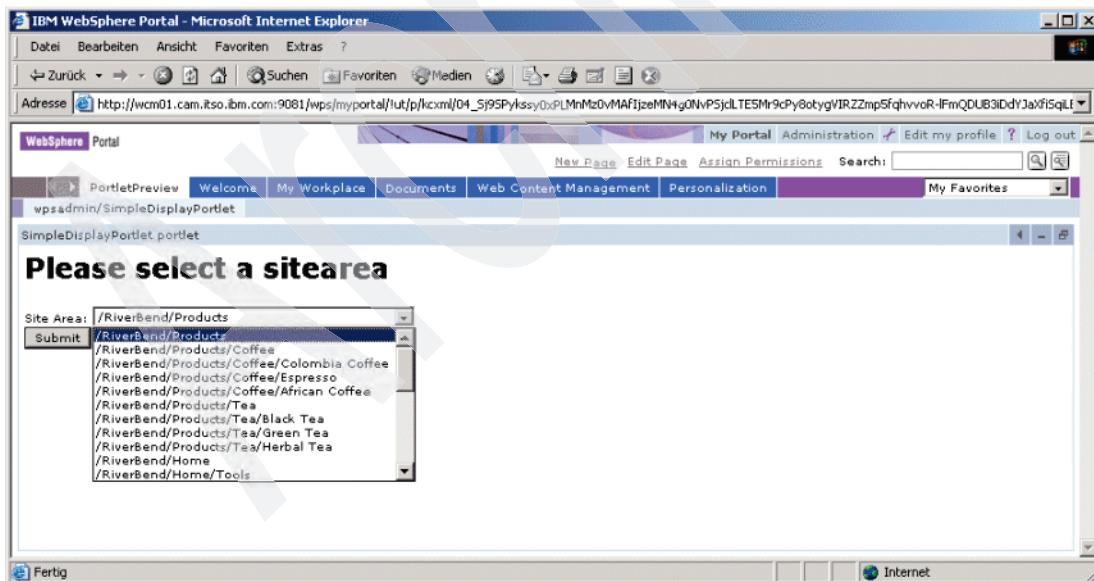


Figure 12-34 SimpleDisplayPortlet, edit mode

After you submit the selection in the Edit.jsp, the `actionPerformed` method executes, and the content bean is saved. Then, the `doView` method executes. Finally, the content is displayed in the portlet, as shown in Figure 12-35.

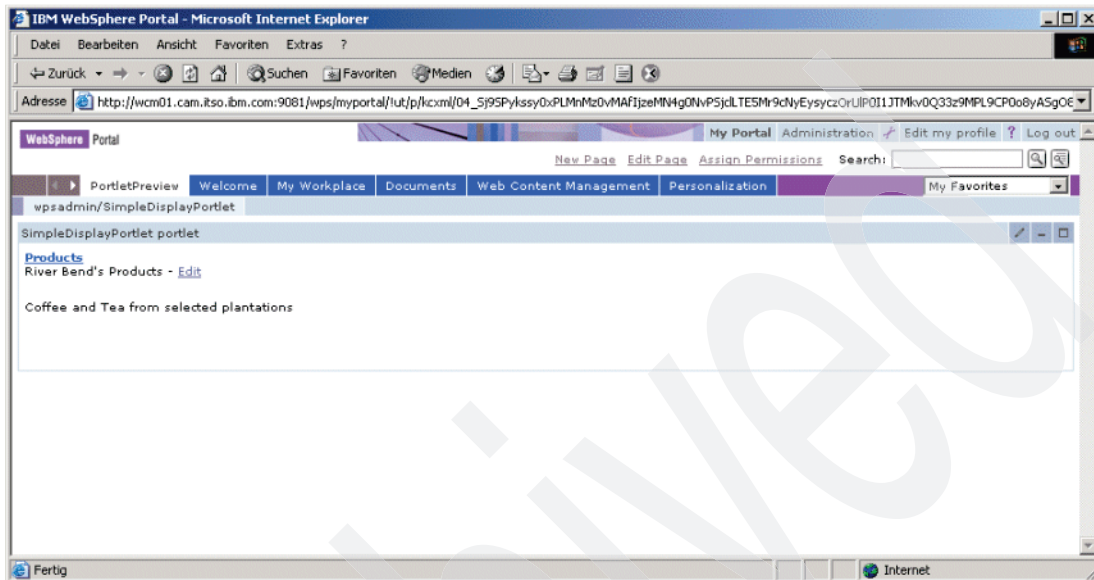


Figure 12-35 SimpleDisplayPortlet, view mode

12.5.2 Working with JSP components

This example shows the deployment and use of the JSP component feature. (You can find general information about the JSP component in 12.3.1, “JSP components” on page 486.) Within this example, we extend the content presentation of the River Bend example to demonstrate how you can use JSP components to add dynamic functionality to Web content. In addition, we discuss the necessary steps to deploy JSP components.

To keep the example as simple as possible, we add a basic time and date function to our River Bend Web site. In addition to this functionality, we create a JSP that use the `java.util.Date` object to print the current data and time. Figure 12-18 shows the code for this JSP.

Example 12-18 JSP component

```
<p>The current date is:
<%=new java.util.Date().toString()%></p>
```

To begin, create a simple text file named `time.jsp` and store this file in the JSP directory below the Web content management .war file:

```
<Drive>:\<Path>\WebSphere\AppServer\installedApps\<Node>\wcm.ear\
ilwwcm.war\jsp
```

Then, integrate the new JSP into the content presentation as follows:

1. Log in to IBM Workplace Web Content Management and open the Web Content Authoring portlet if necessary.
2. Expand Design and Development.
3. Click **New** and select **JSP Component**.
4. Click **OK**.
5. Enter the information about the JSP component as shown in Figure 12-36.

The screenshot shows the 'Web Content Authoring' window. At the top, there's a toolbar with 'Close', 'Save', and 'Save...' buttons. Below the toolbar, the 'Untitled' document is open. The 'Required Fields' section is expanded, showing the 'Identification' tab. The 'Name' field is filled with 'Time', and the 'Description' field is filled with 'Date and Time'. The 'Type' is set to 'JSP Component'. The 'Authors' field is filled with 'wpsadmin' and has a 'Select Authors' button next to it. The 'Owners' field is also filled with 'wpsadmin' and has a 'Select Owners' button next to it. The 'JSP Component' tab is expanded, showing the 'Path' field filled with '/jsp/time.jsp'. Below the 'Path' field, there is an 'Errors' section with a text area containing the message: 'JSP Component time.jsp could not be rendered.'

Figure 12-36 JSP component

Enter the following values:

- a. Name: Time
- b. Description: Date and Time
- c. Path: /jsp/time.jsp
- d. Errors: JSP component time.jsp could not be rendered.

The path *must* begin with a forward slash. The path represents the directory below the iwwcm.war directory. This field is used to enter an error message to display if the JSP is unavailable.

Note: The name of the JSP file is case sensitive.

Important: The JSP referenced within a JSP component must not include a reference, directly or indirectly, to the same JSP component. This includes references within IBM Workplace Web Content Management tags or the API. If it does, a loop is created and your Server will crash.

- 6. Click **Save** and **Close**.
- 7. A new JSP component category appears in Design and Development → component Library.
- 8. Open the Presentation Templates view and open the Two Column Table presentation.
- 9. Click **Edit**.

10. Add a reference to the JSP component in the presentation template HTML code. In this, example we add the `<AptribLibCmpnt name="Time"/>` tag below the HTML table, as shown in Figure 12-37.

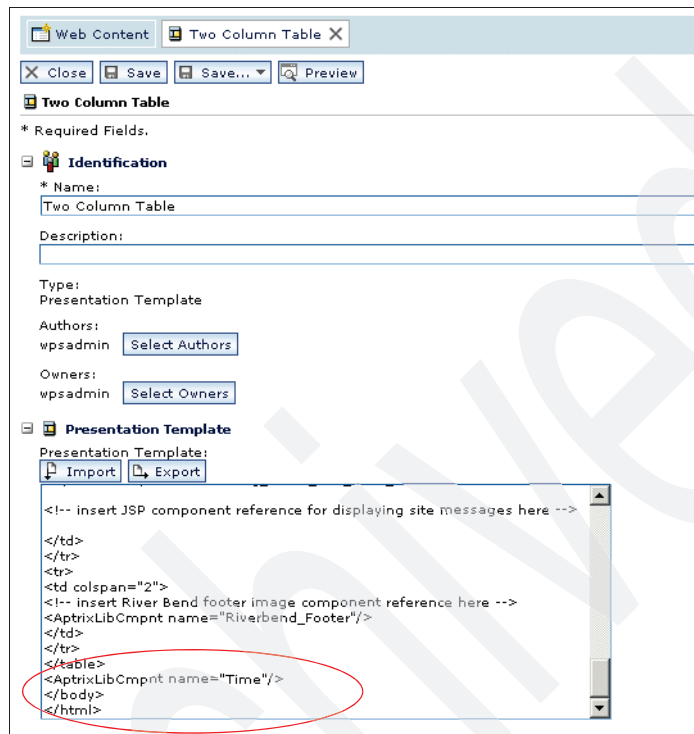


Figure 12-37 JSP Component example - Presentation Template

11. Save and Close the document.
12. Repeat steps 8 through 11 for the other both presentation templates.

13. Preview the Web site. Figure 12-38 illustrates the integrated JSP component within the River Bend Web site.

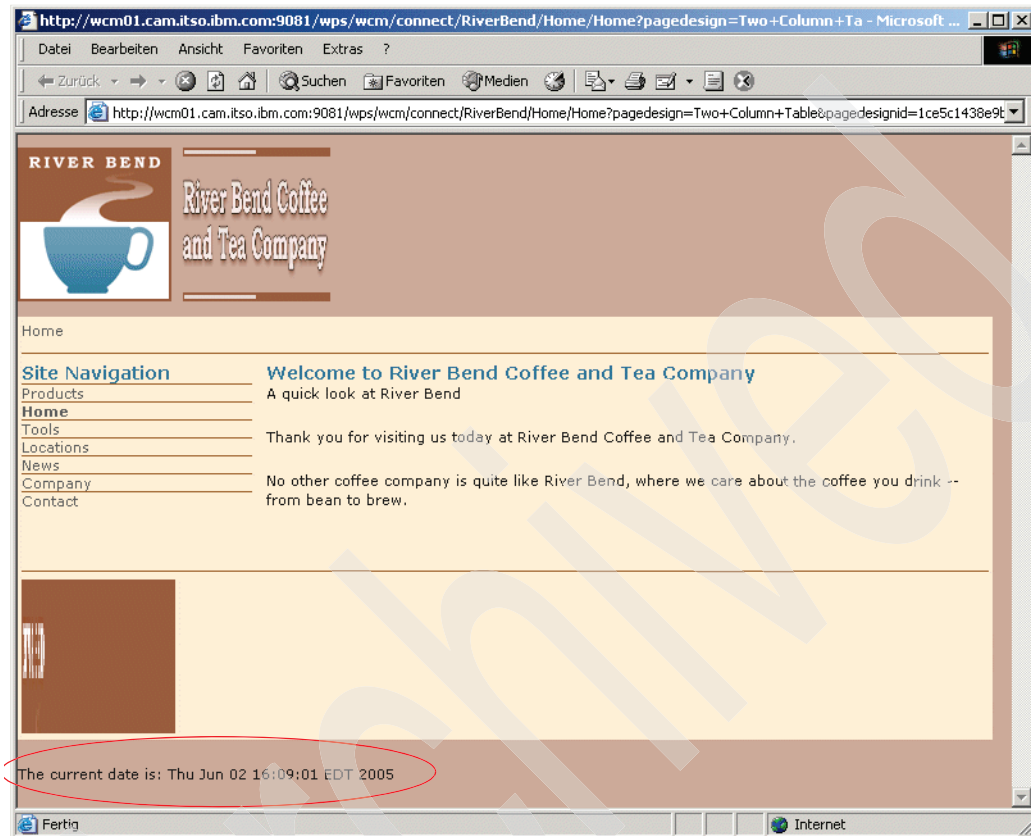


Figure 12-38 JSP Component, preview with a Web browser

12.6 Deployment considerations

Before you deploy your changes, consider the following:

- Installation

Until version 2.0, the IBM Workplace Web Content Management API was not part of the core product installation. It was necessary to create a shared library and to add the library to the class loader of the application server. Since version 5.1, the API is installed as part of the IBM Workplace Web Content Management configuration utility. You do not need to install an API library to use the IBM Workplace Web Content Management capability.

Tip: Refer to the developerWorks article “Setting up a debug environment for IBM Workplace Web Content Management” if you are planning to use the API in a previous version. You can find this article at:

<http://www.ibm.com/developerworks/lotus/library/debug-wwcm/>

► API authentication

A user name and password from WebSphere Member Manager is required to access secured IBM Workplace Web Content Management resources. The workspace cannot be retrieved using an LTPA token.

► Deployment

Java resources created for API access, such as JSPs, class files, jars, and so forth, are not syndicated between nodes of the WebSphere Application Server. Therefore, you need to deploy all used resources to each IBM Workplace Web Content Management node within the environment.

The API does not support native remote access to the IBM Workplace Web Content Management objects. The Java code that uses the API must reside on the same server as IBM Workplace Web Content Management. Each node requires a full instance of IBM Workplace Web Content Management for access.

If you want to access the IBM Workplace Web Content Management objects remotely, we recommend an approach that explores the necessary IBM Workplace Web Content Management functions as a Web service.

Archived

Advanced topics and examples with the API

This chapter provides advanced examples how to use the IBM Workplace Web Content Management API to create custom portlets. The chapter contains the following sections:

- ▶ Multi-lingual Content Display Portlet
- ▶ Using the Customizable Template Portlet
- ▶ Extensions to the Customizable Template Portlet

13.1 Multi-lingual Content Display Portlet

In this section, we extend our Simple Display Portlet with the multi-language capabilities that are described in Chapter 9, “Advanced integration with WebSphere Portal” on page 403. (More specifically, refer to 9.5, “Implementing a custom display Portlet” on page 418.) So far, our portlet displays the default content of the selected site area. The default content is examined in the `doView` method. To implement the discussed multi-language approach, you have to change this point to do the following:

- ▶ Get the user’s default language code through the request from the browser settings.
- ▶ Search for a content which ends with the language code within the desired site area.

To implement these changes, follow these steps:

1. Add a method to the portlet that retrieves the language code from the request. The result is returned as string. Example 13-1 shows the code of this method.

Example 13-1 `getLanguageFromHeader`

```
private String getLanguageFromHeader(PortletRequest request) {
    String headerLang = request.getHeader("Accept-Language");
    int dash = headerLang.indexOf("-");
    if (dash > -1) {
        return headerLang.substring(0, dash);
    }
    return headerLang;
}
```

2. Change the part of the `doView` method that searches for the default content. Example 13-2 shows a source code fragment of the current `doView` method that retrieves the default content.

Example 13-2 Initial `doView` code fragment

```
...
if( cb==null ) {
    response.getWriter().println("<b>NO CONTENTBEAN YET</b>");
    return;
}
try {
    SiteArea sitearea =
        (SiteArea)ws.getById(ws.createDocumentId(cb.getSiteAreaId()));
    Site site = (Site)ws.getById(sitearea.getParent());
    //search the default content
    Content content = (Content)ws.getById(sitearea.getDefaultContent());
```

```

StringBuffer contentPath = new StringBuffer();
contentPath.append("/"+sitearea.getName());
contentPath.append("/"+content.getName());
...

```

3. Add code that retrieves the content in the user's default language behind the single line that searches for the default content. Then, the content is replaced by the content in the user's default language, if it exists. Thus, the portlet displays the default content, if content in the user's default language is not found.

Example 13-3 illustrates the necessary code:

Example 13-3 Final doView code fragment.

```

...
Site site = (Site)ws.getById(sitearea.getParent());

Content content = (Content)ws.getById(sitearea.getDefaultContent());
// begin multi-lingual extension
String userLang = getLanguageFromHeader(request);

DocumentIdIterator siteAreaChildren = sitearea.getChildren();
DocumentId lastSiteAreaChildId = null;
while (siteAreaChildren.hasNext()) {
    lastSiteAreaChildId = (DocumentId)siteAreaChildren.next();
    if (lastSiteAreaChildId.isOfType(DocumentTypes.Content)) {
        if (lastSiteAreaChildId.getName().endsWith("_" + userLang)) {
            content = (Content)ws.getById(lastSiteAreaChildId);
            break;
        }
    }
}
// end multi-lingual extension
StringBuffer contentPath = new StringBuffer();
...

```

13.2 Using the Customizable Template Portlet

The standard IBM Workplace Web Content Management authoring interface for creating content is quite robust and makes all options available to content authors. For some content, however, you might not use or want all the features that are available. You might instead want to streamline the authoring process so that only elements and actions that are needed to create, preview, and publish your Web content are displayed.

This section provides the step-by-step detailed information about how to deploy and configure the Customizable Template Portlet — named '*Content Custom UI*' — for IBM Workplace Web Content Management 2.5 and 5.1.0.1. This Portlet, in conjunction with the sample workflow portlet named *Workplace Web Content Management Sample Workflow*, creates an easy three-click environment for Web content creation.

This section extends the River Bend Web site that you created in Chapter 7, “Building the River Bend Web site with IBM Workplace Web Content Management” on page 253. We assume that you also might want to build this example within any other context. Thus, you create a new authoring template and menu component. You can integrate this example into the River Bend Web site or any other Web site that is created with IBM Workplace Web Content Management.

The scenario

The River Bend Web site contains a Bulletin Board that can be accessed through the public Web site and through the intranet portal. The portal can be accessed by Bulletin Board authors and employees with read access to the board, as shown in Figure 13-1 on page 537.

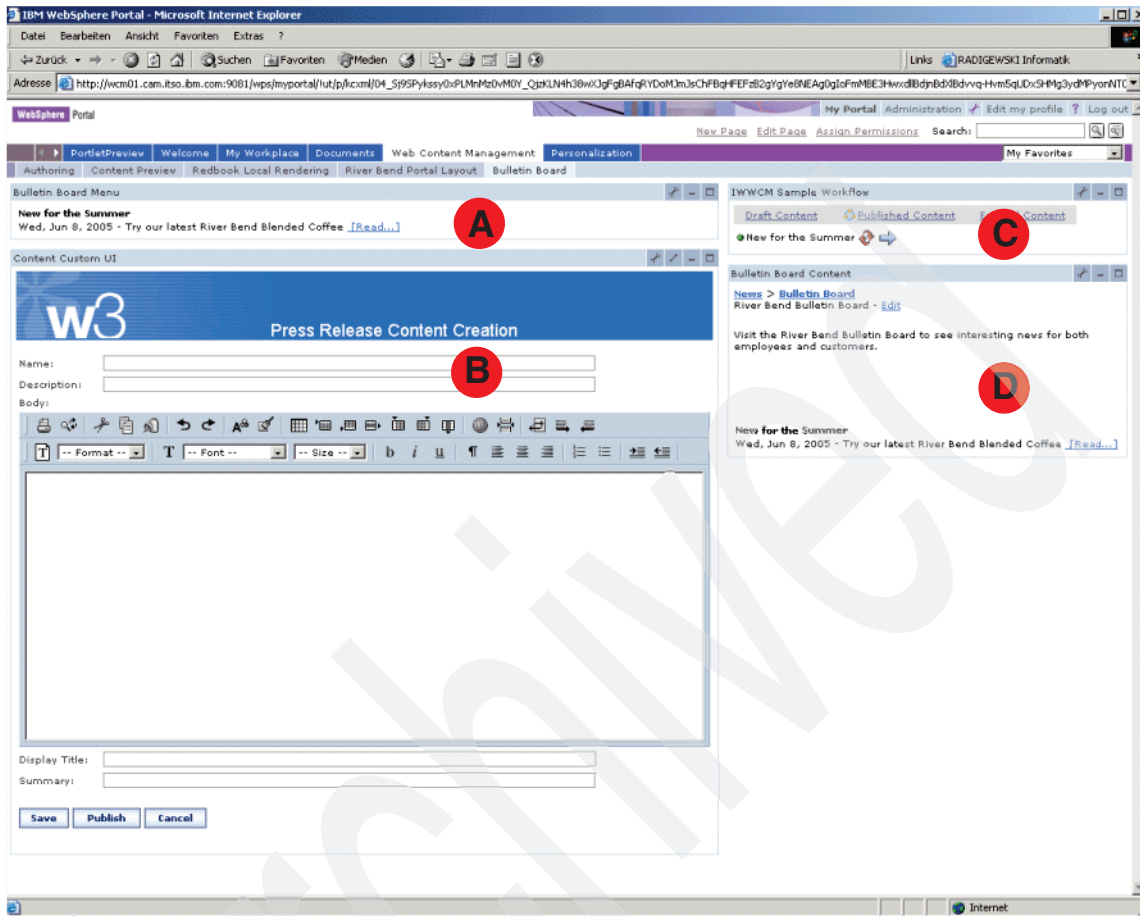


Figure 13-1 Streamlined authoring portal page

Non-authors navigate to the Bulletin Board page and only see the published Bulletin Board (A) and the selected Bulletin Board Content (D) portlets.

When authors log in to WebSphere Portal and navigates to the same Bulletin Board page, they find, in addition to A and B, the Content Custom UI (B) and Workplace Web Content Management Sample Workflow (C) portlets. Authors can add a new Bulletin Board item using these portlets in the following steps:

1. Fill Content Custom UI (B) with article information and submit the information.
2. Preview the item as it would appear in the Bulletin Board Content portlet (D).
3. Publish the item using the check mark (approve) icon in the IBM Workplace Web Content Management Sample Workflow portlet (C).

The new content is available immediately to everyone in the company.

You first create the following components:

- ▶ Authoring Template:
 - Name: Bulletin Board
 - Components
 - Display Title (Text Component)
 - Summary (Text Component)
 - Body (Rich Text Component)
- ▶ Presentation Template:
 - Name: Bulletin Board
 - Template

```
<h3><AptribCmpnt context="current" key="Display Title"
type="content"/></h3>
<AptribCmpnt context="current" key="Body" type="content"/>
```

Then, open the site area /RiverBend/News/Bulletin Board and create a template mapping for the new templates, as shown in Figure 13-2.

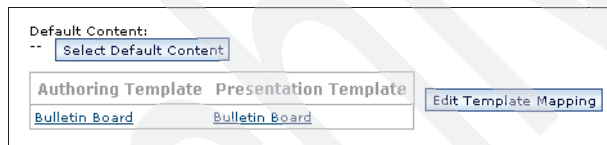


Figure 13-2 Template mapping

Note: The portlet is written for IBM Workplace Web Content Management 2.5 and WebSphere Portal 5.1.0.1. The Content Custom UI and Workplace Web Content Management Sample Workflow portlets have been tested with both of these platforms. These portlets are code samples using the IBM Workplace Web Content Management API. Thus, support is handled through normal IBM Workplace Web Content Management support of the API.

13.2.1 Installing the portlets

In preparation to install the portlets, download the following .war files to a location on your local system:

- ▶ iwwcm-custom-template-portlet_25.war (for Workplace Web Content management 2.5 or WebSphere Portal 5.1)
- ▶ iwwcm_sample_workflow_list.war

Then, follow these steps to install the portlets:

1. Navigate to the WebSphere Portal Administration → Portlet Management → Web Modules page and select **Install**.
2. Browse and select the first WAR file:
ilwwcm-custom-template-portlet_25.war
3. Click **Next**. You see the confirmation message that is shown in Figure 13-3.

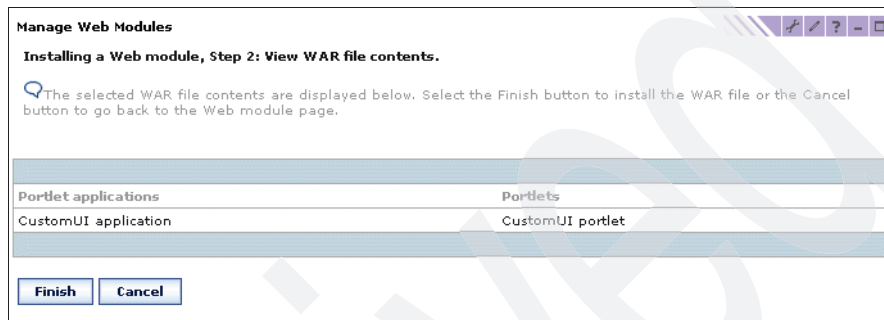


Figure 13-3 Portal Administration Install Web Modules

4. Click **Finish**.
5. Repeat these steps for the second WAR file:
ilwwcm_sample_workflow_list.war
6. Click **Next**.

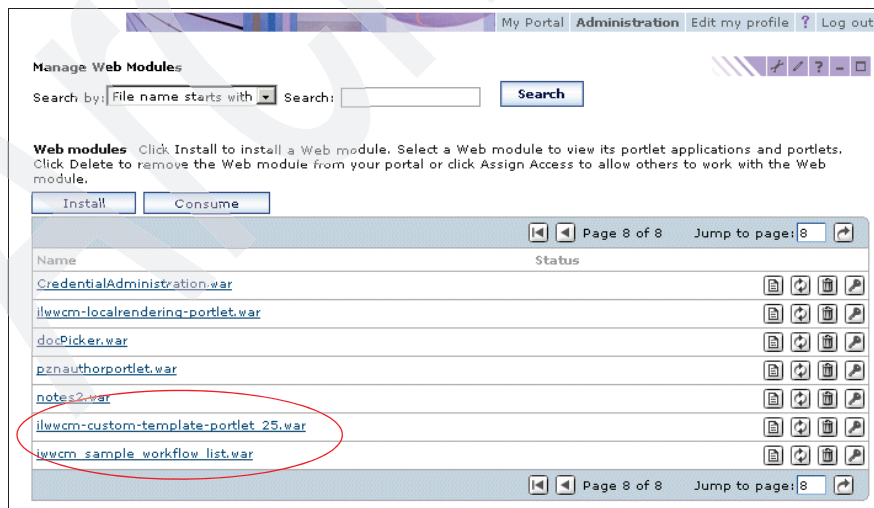


Figure 13-4 Portal Administration Install Web Modules

13.2.2 Creating a Bulletin Board page with portlets

The new portlets are added to a new Bulletin Board page, and then are configured to manage the Bulletin Board template content that is displayed on the page. To create a Bulletin Board page with portlets, follow these steps:

1. Navigate to the WebSphere Portal Administration → Portlet Management → Portlet.
2. Search for the Web Content Viewer Portlet and create a copy called Bulletin Board Menu.
3. Create a second copy and name it *Bulletin Board Content*.

Note: For more information about the Web Content Viewer portlets, refer to the IBM Workplace Web Content Management Information Center at:

<http://www-128.ibm.com/developerworks/workplace/documentation/webcontentmanagement/>

4. Navigate to the WebSphere Portal Administration → Portal User Interface → Manage Pages.
5. Create an new Portlet Page called *Bulletin Board*. You can locate the page at any desired place within your Portal. If you are adding the page to the River Bend Web site, we suggest that you create the new Page below *Web Content Management*.

The new page has four portlets (two on the left and two on the right), as follows:

- ▶ Left Portlets:
 - Copy of the Web Content Viewer portlet that is configured to show a list of published Bulletin Board articles.
 - The Content Custom UI that is used for creating and editing Bulletin Board content items.
- ▶ Right Portlets
 - The Workplace Web Content Management Sample Workflow portlet that is configured to show the DeptNews template content.
 - A copy of the Web Content Viewer portlet that is configured to show document details.

6. To add the portlets to the page, select **Page Layout**.

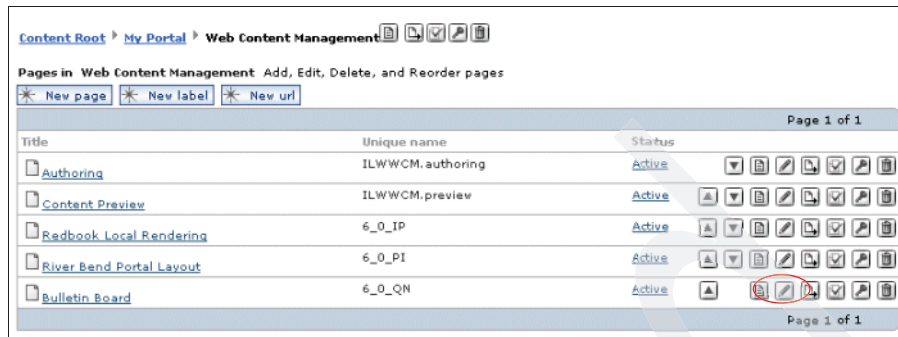


Figure 13-5 Edit page layout

7. Add the Portlets to the page. Arrange the portlets as shown in Figure 13-6 using the arrow buttons. Select **Done**.

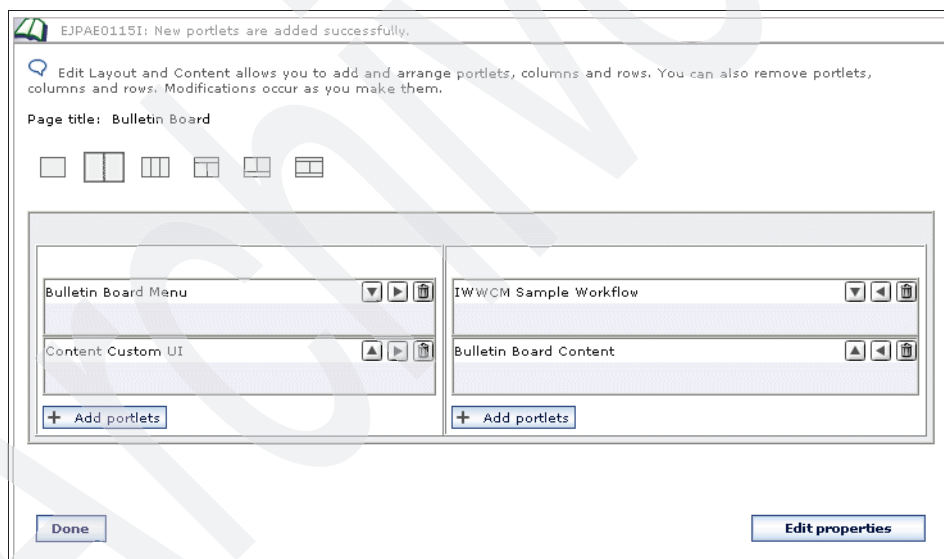


Figure 13-6 Arrange Portlets on the Bulletin Board page

The page now has all the necessary portlets to edit quickly new IBM Workplace Web Content Management content.

Note: You can control the access to these portlets using the WebSphere Portal → Administration → Access → Resource Permissions → Portlets menu options. You need to select each of the new portlets, assign access to the user, and add the author group to that user. This action allows you to control the access to this author group only. Feel free to create an author group and assign access to these portlets for this group only.

13.2.3 Configure the Portlets

Next, you need to configure the portlets on the page. Depending on the portlet, you have to use either the edit or the configure mode.

Bulletin Board menu configuration

To use the edit mode to configure the Portlets, follow these steps:

1. Click **Configuration** on the Bulletin Board menu portlet.

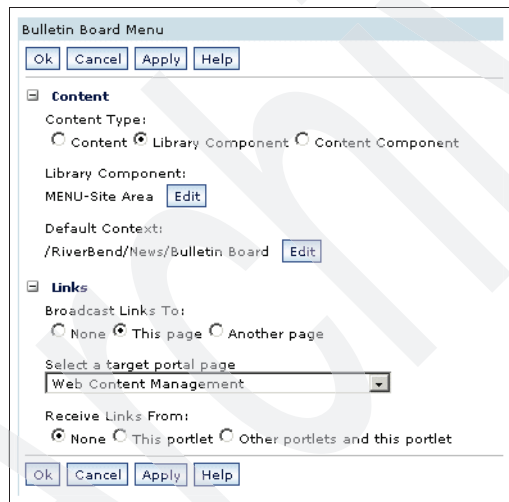
The image shows a 'Bulletin Board Menu' configuration dialog box. It has a title bar and four buttons: 'Ok', 'Cancel', 'Apply', and 'Help'. The dialog is divided into two main sections: 'Content' and 'Links'. In the 'Content' section, 'Content Type' has three radio buttons: 'Content', 'Library Component' (which is selected), and 'Content Component'. Below this, 'Library Component' has a text field 'MENU-Site Area' with an 'Edit' button. 'Default Context' has a text field '/RiverBend/News/Bulletin Board' with an 'Edit' button. In the 'Links' section, 'Broadcast Links To' has three radio buttons: 'None', 'This page' (which is selected), and 'Another page'. Below this is a dropdown menu labeled 'Select a target portal page' with 'Web Content Management' selected. 'Receive Links From' has three radio buttons: 'None' (which is selected), 'This portlet', and 'Other portlets and this portlet'. At the bottom are 'Ok', 'Cancel', 'Apply', and 'Help' buttons.

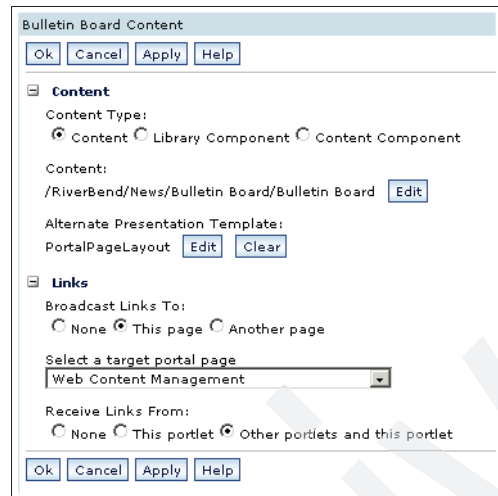
Figure 13-7 Configure the Bulletin Board menu Portlet

2. Under Content Type, select **Library Component**.
3. Chose /RiverBend/News/Bulletin Board as default content.
4. Select to broadcast links to this page.
5. Under Receive Links From, select **None**.
6. Click **OK**.

Bulletin Board content configuration

To use the configuration mode, follow these steps:

1. Click **Configuration** on the Bulletin Board menu portlet.



The screenshot shows the 'Bulletin Board Content' configuration dialog box. It has a title bar with 'Bulletin Board Content' and buttons for 'Ok', 'Cancel', 'Apply', and 'Help'. The dialog is divided into two main sections: 'Content' and 'Links'. In the 'Content' section, 'Content Type' has three radio buttons: 'Content' (selected), 'Library Component', and 'Content Component'. Below this, 'Content' is set to '/RiverBend/News/Bulletin Board/Bulletin Board' with an 'Edit' button. 'Alternate Presentation Template' has 'PortalPageLayout' selected with 'Edit' and 'Clear' buttons. The 'Links' section has 'Broadcast Links To' with radio buttons for 'None' (selected), 'This page', and 'Another page'. Below this is a dropdown menu for 'Select a target portal page' currently showing 'Web Content Management'. 'Receive Links From' has radio buttons for 'None', 'This portlet', and 'Other portlets and this portlet' (selected). At the bottom are 'Ok', 'Cancel', 'Apply', and 'Help' buttons.

Figure 13-8 Configure the Bulletin Board content Portlet

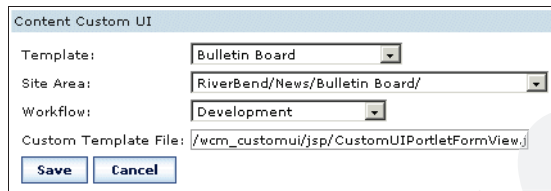
2. Select **Content** as the Content Type.
3. Chose /RiverBend/News/Bulletin Board as default content.
4. Select Broadcast Links to **This Page**.
5. Select Receive Links From: Other Portlets and this portlet.
6. Click **OK**.

Content Custom UI Portlet configuration

For the Content Custom UI Portlet, both configure and edit modes are available. The settings in the configure mode are set for the Portlet in general and are carried with the Portlet as an instance is placed on a portal page. The configure settings are generally set by a systems administrator. The edit settings are unique to each instance of the Portlet on a page and are normally set by the content administrator. Authorization to configure and edit can be set for groups using Portal Access Control. Where settings are the same in both modes, edit overrides the configure settings.

To set the configuration setting, follow these steps:

1. On the Bulletin Board page, select the configure icon of the Content Custom UI portlet.
2. Select the Bulletin Board template, the site area /RiverBend/News/Bulletin Board/, the Development Workflow, and leave the default JSP custom template file, as shown in Figure 13-9.
3. Select **Save**.



Content Custom UI

Template:

Site Area:

Workflow:

Custom Template File:

Figure 13-9 Configuring the Custom Content UI Portlet

The default JSP file `CustomUIPortletFormView.jsp` uses the authoring template to construct an entry field for each template attribute. A text field is used for text fields, a Rich Text editor is used for Rich Text fields as well as HTML fields, and text with browse button is used for image fields. If you need to change the entry field or placement, you can supply a custom JSP file to adjust as needed. For more information about creating a custom JSP, see “Customizing a JSP format” on page 557.

Figure 13-10 shows the Custom Content UI after configuration.

The screenshot shows a web browser window titled "Content Custom UI". At the top, a message box states "Template configuration successful." Below this is a blue header bar with the "w3" logo on the left and the text "Press Release Content Creation" on the right. The main form contains the following fields and elements:

- Name:** A text input field.
- Description:** A text input field.
- Body:** A large text area with a rich text editor toolbar above it. The toolbar includes icons for undo, redo, bold, italic, underline, link, unlink, bulleted list, numbered list, indent, outdent, and other formatting options.
- Display Title:** A text input field.
- Summary:** A text input field.
- Buttons:** "Save", "Publish", and "Cancel" buttons at the bottom.

Figure 13-10 Custom Content UI after configuration

4. Review the Content Custom UI portlet. It shows the ID information and the fields of the Bulletin Board template. This information is the only information a user needs to complete the form. Pressing Cancel on the Content Custom UI refreshes the portlet and removes the message at the top.

IBM Workplace Web Content Management sample Workflow Portlet configuration

The Content Custom UI and the Workplace Web Content Management Sample Workflow Portlets work together to create the Quick Edit authoring environment. So, you need to coordinate the Workflow Portlet settings with the Custom Content UI settings (Figure 13-11 on page 546).

The screenshot shows a web form titled "IWWCM Sample Workflow". Below the title is a label "pageMapping:". The form contains two main sections. The first section is labeled "Template:" and features a dropdown menu with "Bulletin Board" selected. The second section is labeled "Target Page Mapping for Preview (Optional):" and includes a text input field followed by the text "(eg, myCTP)". At the bottom left of the form is a blue "Submit" button.

Figure 13-11 Workflow Portlet settings

To coordinate the settings, follow these steps:

1. Go to Configuration of the Workplace Web Content Management Sample Workflow Portlet.
2. Select the Bulletin Board authoring template to show a list of items that are in draft, published, or expired status for that template. The Target Page Mapping for Preview field allows you to set a page for preview if it is other than the current page. Because the preview is on the same page as the other Portlets, you can keep the Target Page Mapping for Preview page empty.
3. Select **Submit**.

The configuration is now complete and the new Quick Edit authoring page Department News is ready for use (Figure 13-12 on page 547).

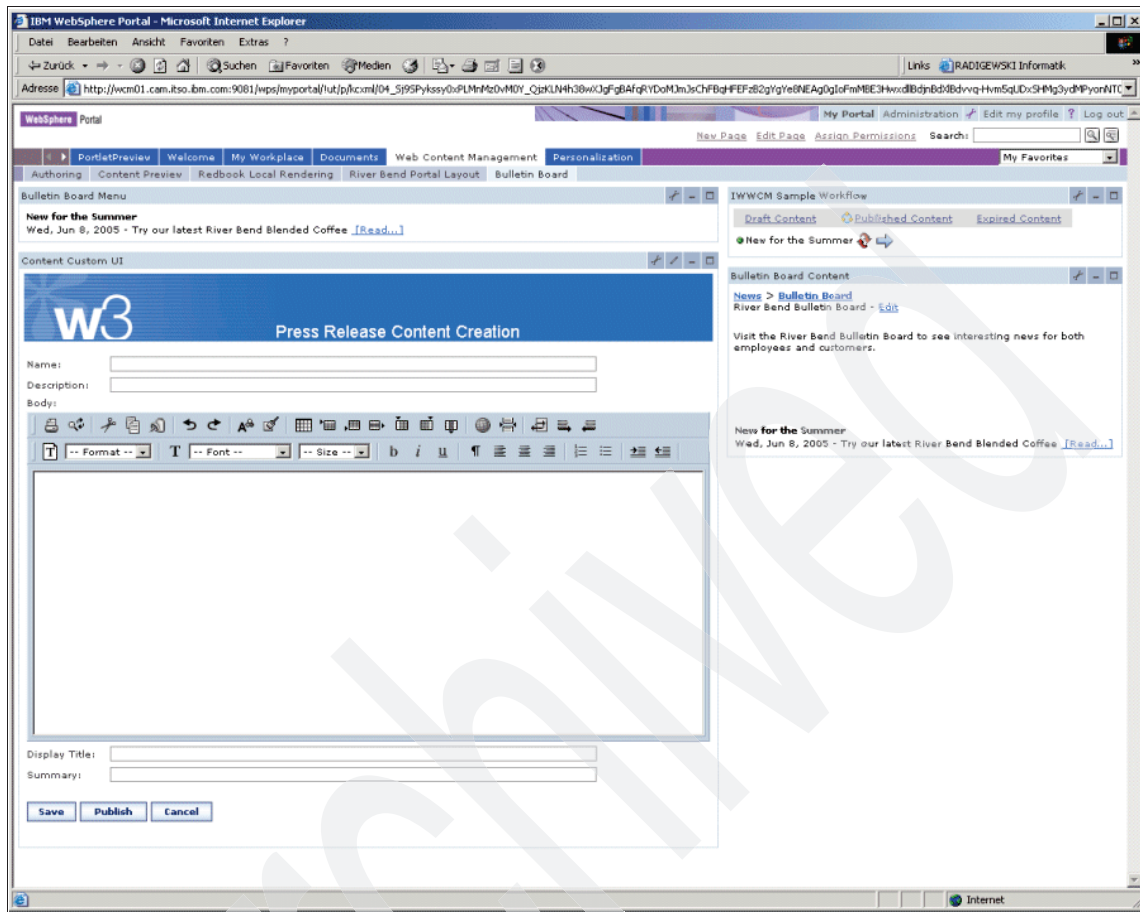


Figure 13-12 Configured Bulletin Board contribution page

13.2.4 Demonstration of Quick Edit

To demonstrate how easily you can create content, follow these steps:

1. Enter content attributes into form.

In the Content Custom UI Portlet, complete the fields for the new Bulletin Board article (Figure 13-13). Be sure to include name and title fields.

Content Custom UI

Quick Content Creation

Version 2.5.1 (06032005-0920)

Name:

Description:

Summary:

Display Title:

Body:

Publish Date (optional): Month Day Year

Expiry Date (optional): Month Day Year

Figure 13-13 Complete content for Bulletin content

Note that you can set the effective and expiry dates. These dates can be left blank to take the default, which is the publish date (today) or the effective date, with no planned expiration.

The new news item now appears as a draft item in the Workplace Web Content Management Sample Workflow Portlet (Figure 13-14).

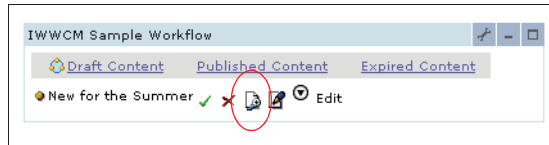


Figure 13-14 Preview the content

2. Preview content as it would appear on Web site.

If you select Preview (identified with hover help) in the Workflow list, you should see the item in the Department News Detail portlet below it (Figure 13-15).

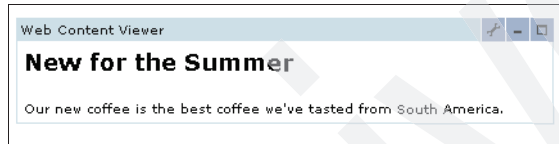


Figure 13-15 Preview the content

If you need to make any changes, you can select the Edit icon or Click-to-Action function to populate the Custom Content UI Portlet with the News Article values. You can make changes easily and then submit the content. You can repeat this action until the content is ready to be published.

3. Publish the content to the Web site.

When you are satisfied with the entry, select Publish (green check mark) in the Workplace Web Content Management Sample Workflow Portlet. The Department News Detail Portlet shows the item as published on the live site.

The Workplace Web Content Management Sample Workflow Portlet moves the content to the next stage when Approve is selected. Because the Quick Edit Workflow next stage is Publish, the content is published. If there are more stages before Publish, all draft stages need to be approved before the content is published to the Web site.

13.2.5 Customizing the Content Custom UI Portlet

This section examines the default JSPs as well as two custom JSPs that show how the Content Custom UI Portlet can produce a customized authoring environment.

The JSP files that correspond to this Portlet are stored in the following directory:

```
<Drive>:\<Path>\WebSphere\PortalServer\installedApps\  
CustomUI_PA_1_0_P1.ear\PA_1_0_P1.war\wcm_customui\jsp\html
```

Additional JSP examples from the Sampleset.zip are stored in the additional subdirectory, *sampleset1* under *html*.

The presentation JSP file CustomUIPortletFormView.jsp is the default JSP. You can change this default in the Configure and Edit modes of the Portlet, or you can modify the default to your own format. For example, you can change the image to your company logo.

It is recommended that before you change any JSP files or updating or reinstalling the Portlet, you copy JSP files and any other image or JavaScript to a back-up directory outside of the Portal.

Also, note that the Workplace Web Content Management Sample Workflow JSP files are found in the following directory:

```
<Drive>:\<Path>\WebSphere\PortalServer\installedApps\  
iwwcm_samp_kflow_list_PA_1_0_P2.ear\PA_1_0_P2.war\  
IWWCMWorkflow\jsp\htm
```

Default JSP

The default JSP file CustomUIPortletFormView.jsp for the Content Custom UI Portlet has three sections: header, body, and footer. The header and footer are separated in specific JSP files to allow them to be shared easily in the different main JSP files.

The header, inserted from CustomUIPortletViewHeader.JSP, includes all the needed tag libraries and initializes the session bean and the workspace (Example 13-4 and Example 13-5 on page 551).

Example 13-4 CustomUIPortletViewHeader.JSP - Abstract

```
<%@ page session="false" contentType="text/html" import="java.util.*,  
com.ibm.workplace.wcm.customui.portlet.*,  
com.ibm.workplace.wcm.customui.portlet.util.*, com.ibm.workplace.wcm.api.*"%>  
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>  
<!-- Load Portal tag library and initialize constants -->  
<%@ taglib uri="/WEB-INF/tld/engine.tld" prefix="wps" %>
```

```
<%-- Load the Rich Text Editor Tag Library --%>
<%@ taglib uri="/WEB-INF/tld/DocEditor.tld" prefix="rte" %>
```

Example 13-5 CustomUIPortletViewHeader.JSP - Abstract

```
<%
    CustomUIPortletViewBean viewBean =
    (CustomUIPortletViewBean)portletRequest.getAttribute(CustomUIPortlet.VIEW_BEAN)
    ;
    CustomUIPortletSessionBean sessionBean =
    (CustomUIPortletSessionBean)portletRequest.getPortletSession().getAttribute(Cus
    tomUIPortlet.SESSION_BEAN);
    Vector rteFields = new Vector();
    Workspace workspace = sessionBean.getWorkspace();
    // This helps the wcm tag env use the same workspace
    pageContext.setAttribute(Workspace.WCM_WORKSPACE_KEY, workspace);
    %>
<wcm:initworkspace user="<%= (java.security.Principal)portletRequest.getUser()
%>" />
```

The body of the JSP file is contained in the default JSP itself and contains an iterator that cycles through the template fields and handles the display of the fields according to the field type (Example 13-6).

Example 13-6 Template fields iterator

```
Map templateFields = sessionBean.getTemplateFields();
for (Iterator e = templateFields.keySet().iterator(); e.hasNext();)
{
    String fieldName = (String) e.next();
```

Each field gets the attribute name as the label (Example 13-7).

Example 13-7 Template labels

```
<LABEL class="wpsLabelText"><%=fieldName%>:</LABEL>
<%String fieldClass = (String) templateFields.get(fieldName);
```

For text fields, a string entry field is created (Example 13-8). Note that the name of the input field is created using the Namespace appended with the Field name. The value of the input field is retrieved from the session bean referenced by the field name.

Example 13-8 Template fields

```
if (fieldClass.equals("com.ibm.workplace.wcm.api.WCM_TextComponent"))
{
%>
<td> <INPUT class="wpsEditField"
name="<portletAPI:encodeNamespace value='<%=fieldName%>' />"
type="text" size="100%"
value="<%=sessionBean.getContentValues(fieldName)%>" /></td>
```

For Rich Text fields, an IBM Rich Text editor is used (Example 13-9).

Example 13-9 Rich Text Editor

```
if (fieldClass.equals("com.ibm.workplace.wcm.api.WCM_RichTextComponent") ||
fieldClass.equals("com.ibm.workplace.wcm.api.WCM_HTMLComponent"))
{String fieldNameWithNamespace=portletResponse.encodeNamespace(fieldName);
rteFields.addElement(fieldNameWithNamespace);
String editorName = fieldNameWithNamespace + "Editor";
editorName = editorName.replace(' ', '_');
String listener = "IBM_RTE_setEditorHTML('" + editorName + "', '" +
sessionBean.getContentValues(fieldName)+ "')";
```

Image fields get a file selection field (Example 13-10).

Example 13-10 Image fields

```
if (fieldClass.equals("com.ibm.workplace.wcm.api.WCM_ImageComponent") ||
fieldClass.equals("com.ibm.workplace.wcm.api.WCM_FileComponent"))
{
%>
<td><INPUT dir="<%=bidiFileLTR%>" class="wpsEditField"
name="<portletAPI:encodeNamespace value='<%=fieldName%>' />" size="100%"
type="file" /></td>
```

Other fields types are not given entry fields, just labels. The footer performs the fields validation and handles the submit, publish, and cancel actions.

Example 13-11 JavaScript actions

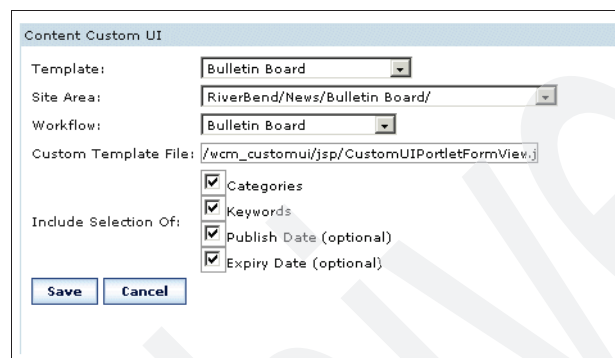
```
function <portletAPI:encodeNamespace value='CustomUI' />setPublish()
function <portletAPI:encodeNamespace value='CustomUI' />setCancel()
function <portletAPI:encodeNamespace value='CustomUI' />doSubmit()
function <portletAPI:encodeNamespace value='CustomUI' />validateFields()
```

Attribute formatting

This section explains the formatting for attributes.

Using categories and keywords

When in the edit mode of the Customizable Template Portlet, you can select whether categories and publish and expiry dates are displayed (Figure 13-16). When the settings are selected, the default JSP renders multi-selection boxes for categories, shows text areas for keywords, and displays the date selection values (Figure 13-17). By default, the setting of these dates is optional.

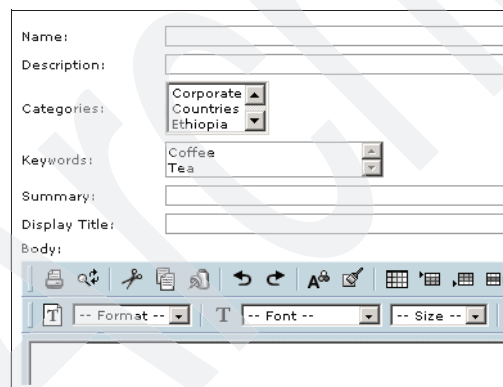


The 'Content Custom UI' dialog box contains the following settings:

- Template: Bulletin Board
- Site Area: RiverBend/News/Bulletin Board/
- Workflow: Bulletin Board
- Custom Template File: /wcm_customui/jsp/CustomUIPortletFormView.jsp
- Include Selection Of:
 - ☒ Categories
 - ☒ Keywords
 - ☒ Publish Date (optional)
 - ☒ Expiry Date (optional)

Buttons: Save, Cancel

Figure 13-16 Edit mode settings



The form displays the following fields and controls:

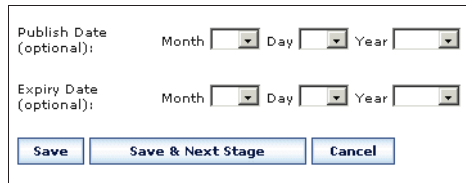
- Name: Text input field
- Description: Text input field
- Categories: Multi-select dropdown menu with options: Corporate, Countries, Ethiopia
- Keywords: Text input field with a multi-select dropdown menu showing Coffee and Tea
- Summary: Text input field
- Display Title: Text input field
- Body: Rich text editor with a toolbar containing icons for undo, redo, bold, italic, link, unlink, list, and table, along with text formatting options like font face and size.

Figure 13-17 Category and keyword selection fields

Setting dates

You can set the effective and expiry dates explicitly (Figure 13-18 on page 554). If these settings are left blank, the default is that the publish date (today) is equal to the effective date, and there is no planned expiration. Sample JavaScript is provided (in the CustomUIPortletViewFooter.jsp) to perform some validation on

the effective and expiry dates. Each date contains three input fields (month, day, and year). If one of these input fields is set, all three must be set. If invalid dates are selected, a message alerts the user that the date is not set correctly.



The form contains two sections. The first section is labeled 'Publish Date (optional):' and contains three dropdown menus for 'Month', 'Day', and 'Year'. The second section is labeled 'Expiry Date (optional):' and also contains three dropdown menus for 'Month', 'Day', and 'Year'. Below these sections are three buttons: 'Save', 'Save & Next Stage', and 'Cancel'.

Figure 13-18 Publish and expiry date fields

Sample Edits

JavaScript is provided to validate the name that the user has entered for a content. The code checks to ensure that the content name does not contain any invalid characters (/, \, % and ?). If the content name contains an invalid character, a message alerts the user that the content name is invalid.

Sample Rich Text editor extensions

When creating a new content item using the Bulletin Board template, an author might want to create a link to another content item such as a previously published article or an image (or other Library component). IBM Workplace Web Content Management allows AptrixCmpnt tags to be inserted into a Rich Text field that can be formatted to provide this capability. Two new icons are added to the Rich Text editor to create these links — one for link referencing an existing content item and the other for a library object (Figure 13-19). Hover help indicates each icon's function.

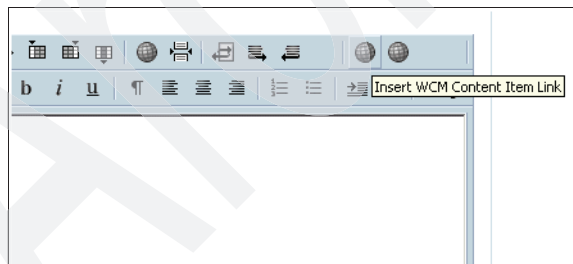


Figure 13-19 Sample button on Rich Text editor to insert a link to a content item

Adding Links to other Content

If you click the Insert Content Item icon, a window displays that filters the content items and allows you to select a content item for the reference link (Figure 13-20 on page 555).

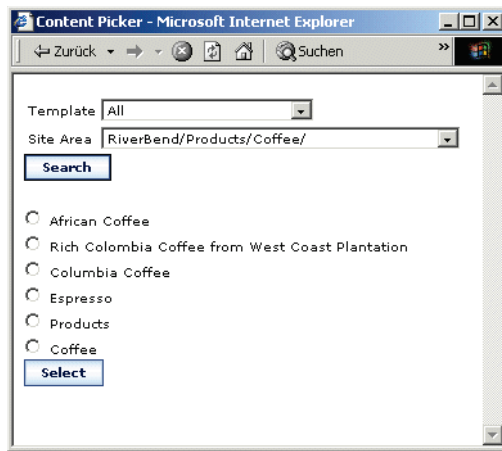


Figure 13-20 Sample content item selection

When selected, the correct Aprix Component tag is inserted into the Rich Text editor field.

Important: There is no visual indication for the link until the Rich Text editor field is used in a presentation template. If you select the scroll in the upper-right of the Rich Text editor, the HTML is displayed. When the item is saved and previewed, the link is rendered.

Adding Link to Library items

You can add a link to a Library item in a similar manner. Do the following:

1. Select the adjacent icon (Insert Library Component Link)
2. Select a library component. The resulting tag is generated (Figure 13-21).

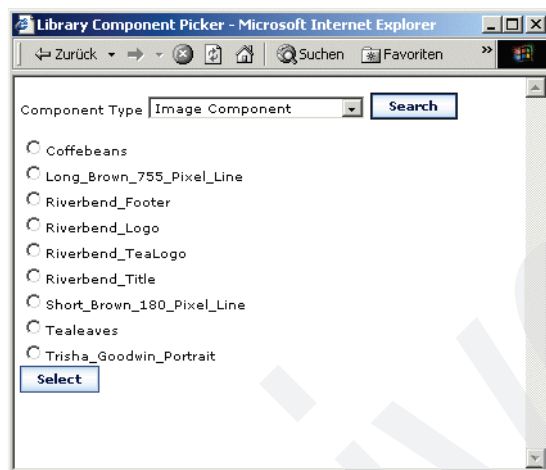


Figure 13-21 Sample selection of an Image library component

As was the case when inserting a content item, you can see the generated HTML for your link (Figure 13-22) and see the embedded library component itself when you preview the HTML (Figure 13-23 on page 557).

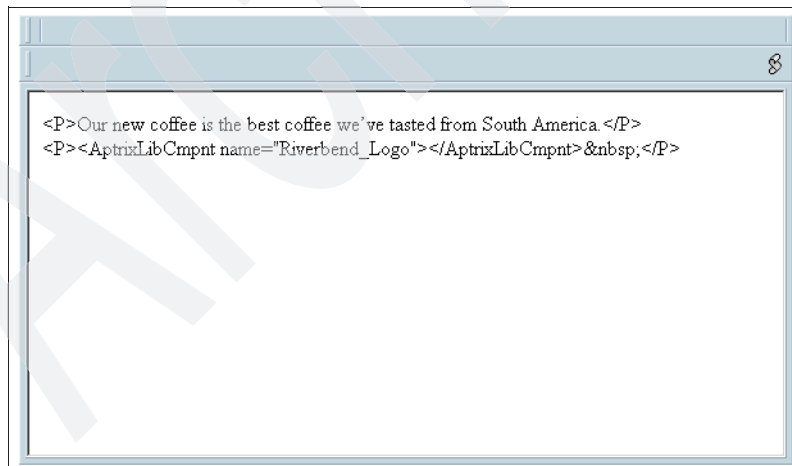


Figure 13-22 Generated HTML for link



Figure 13-23 Image preview

Customizing a JSP format

To create a custom JSP to use with the Customizable Template Portlet rather than the default, simply make a copy of the default and add your custom JSP code.

As an example, use the `CustomUIPortletFormViewNewsButton.jsp` that is provided with this book. Unzip the `CustomJSP.zip` file into the following directory:

```
<Drive>:\<Path>\WebSphere\PortalServer\installedApps\  
CustomUI_PA_1_0_1BP.ear\PA_1_0_1BP.war\wcm_customui\jsp\html
```

This custom JSP requires proper placement of the following additional files:

- ▶ Place the `welcome_5.gif` file in the images directory under the html directory.
- ▶ Unzip the `utils.zip` at the same level as the `wcm_customui\jsp` directory.

In this JSP, a new icon takes the place of the default. Each field of the template is aligned in an HTML table to allow all fields to fit on the browser window without the user having to scroll. Use the Edit icon to change the JSP name in the Custom Template File field.

Table 13-1 lists the JSPs that the CustomUIPortletFormViewNewsButton.jsp file includes.

Table 13-1 JSPs included with CustomUIPortletFormViewNewsButton.jsp

Included JSP	Functions
CustomUIPortletViewHeader.jsp	Taglibs and session object
utils/chooser.jsp	Sample pop-up selection to populate Title field
CustomUIPortletViewImageTable.jsp	Image reference
CustomUIPortletViewRequiredFields.jsp	Name and Description fields
CustomUIPortletViewCategories.jsp	Category and Keyword entry fields
CustomUIPortletViewDates.jsp	Publish and expiry date entry fields
CustomUIPortletViewButtons.jsp	Action buttons (Save, Save & Next Stage, Cancel)
CustomUIPortletViewFooter.jsp	Field validation and submit

Field Names

You must code the name and description fields to add a content item to the repository (Example 13-12).

Example 13-12 Field names

```
<%-- The Name and Description field MUST exist --%>
<tr> <td> <LABEL class="wpsLabelText">
<portletAPI:text
bundle="com.ibm.workplace.wcm.customui.portlet.customPortletUI"
key="NAME" />:</LABEL></td>
<td> <INPUT class="wpsEditField" name="<portletAPI:encodeNamespace
value='<%=UIConstants.NAME%>' />" type="text" size="100%"
value="<%=sessionBean.getContentValues(UIConstants.NAME)%>" tabindex="1"/></td>
</tr>

<tr> <td> <LABEL class="wpsLabelText">
<portletAPI:text
bundle="com.ibm.workplace.wcm.customui.portlet.customPortletUI"
key="DESCRIPTION" />:</LABEL></td>
<td> <INPUT class="wpsEditField" name="<portletAPI:encodeNamespace
value='<%=UIConstants.DESCRPTION%>' />" type="text" size="100%"
value="<%=sessionBean.getContentValues(UIConstants.DESCRPTION)%>"
tabindex="2"/></td>
</tr>
<%-- End required fields --%>
```

Then, for each template field, use the name from the authoring template component name that is appended to the Namespace as the field name. Get the value of the field from the session as shown in Example 13-13. If you use these conventions, you will be able to render with any HTML input field.

Example 13-13 Getting value for the template field

```
<td class="ArticlePagesBodycopyBold" width="151">News Title:</td>
<td width="1176"><TEXTAREA rows="2" cols="100" class="wpsEditField"
name="<portletAPI:encodeNamespace
value='NewsTitle' />"><%=sessionBean.getContentValues("NewsTitle")%></TEXTAREA>
```

Adding a button

You can add functionality to the JSP as with any other JSP Portlet application. In Example 13-14, a button is added to show a selection list to fill in the title. The JSPs referenced are included in the utils.zip file and should be unzipped into the following directory:

```
<Drive>:\<Path>\WebSphere\PortalServer\installedApps\
CustomUI_PA_1_0_S5.ear\PA_1_0_S5.war\wcm_customui
```

Example 13-14 Adding a button to show a selection list to fill in the title

```
<input type=button value="Search for Title"
onClick="javascript:chooseItem('<%=response.encodeURL("/wcm_customui/Utils/News
Title.jsp")%>', '<portletAPI:encodeNamespace
value='CustomTemplate' />', '<portletAPI:encodeNamespace
value='NewsTitle' />')"></td>
</tr>
```

When the button is selected the NewsTitle.jsp displays a selection of titles. When a selection is made, the title input field is populated (Figure 13-24). The list of titles is hard coded but could be supplied from any query results.

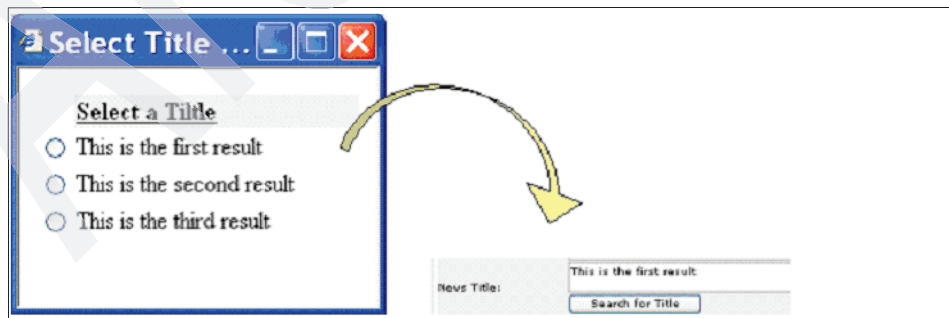


Figure 13-24 Title selection and field population

Using multiple templates

Multiple templates can also use the same Content Portlet. This action is allowed by overriding the template, site area, and workflow settings. See the CustomUIPortletFormView_t1.jsp in the Sampleset.zip file which is deployed in the following directory:

```
<Drive>:\<Path>\WebSphere\PortalServer\installedApps\  
CustomUI_PA_1_0_S5.ear\PA_1_0_S5.war\wcm_customui\jsp\html\  
sampleset1
```

To use this JSP file in your Content Portlet, configure it with sampleset1\CustomUIPortletFormView_t1.jsp in the Custom Template File field.

The JSP has two divisions — one for each template format. Both templates contain fields 1 through 4. In Example 13-15, the MyTemplate1 requires fields 2 and 4 and MyTemplate2 requires fields 1 and 3.

Example 13-15 JSP field data for each template format

```
<div id="<portletAPI:encodeNamespace value='MyTemplate1' />" style="display: <%=  
currentTemplate.equals("MyTemplate1")?"block":"none"%>">  
<table> <tr>  
<td> <LABEL class="wpsLabelText"><%=t_field2%>:</LABEL></td>  
<td> <INPUT class="wpsEditField" name="<portletAPI:encodeNamespace  
value='<%=t_field2%>' />" type="text" size="50"  
value="<%=sessionBean.getContentValues(t_field2)%>" /></td>  
</tr><tr><td> <LABEL class="wpsLabelText"><%=t_field4%>:</LABEL></td>  
<td> <INPUT class="wpsEditField" name="<portletAPI:encodeNamespace  
value='<%=t_field4%>' />" type="text" size="50"  
value="<%=sessionBean.getContentValues(t_field4)%>" /></td>  
</tr></table>  
</div></td></tr>  
  
...  
  
<div id="<portletAPI:encodeNamespace value='MyTemplate2' />" style="display: <%=  
currentTemplate.equals("MyTemplate2")?"block":"none"%>">  
<table> <tr>  
<td nowrap> <LABEL class="wpsLabelText"><%=t_field1%>:</LABEL>  
<INPUT class="wpsEditField" name="<portletAPI:encodeNamespace  
value='<%=t_field1%>' />" type="text" size="50"  
value="<%=sessionBean.getContentValues(t_field1)%>" /></td>  
<td nowrap> <LABEL class="wpsLabelText"><%=t_field3%>:</LABEL>  
<INPUT class="wpsEditField" name="<portletAPI:encodeNamespace  
value='<%=t_field3%>' />" type="text" size="50"  
value="<%=sessionBean.getContentValues(t_field3)%>" /></td>  
</tr></table></div>
```

The content contributor can use the same Portlet for either template.

13.2.6 Customizing the Workplace Web Content Management Sample Workflow Portlet

The Workplace Web Content Management Sample Workflow JSP file has three divisions that are used to render draft, published, and expired content. This Portlet uses inter-portlet communications to interact with the Customizable Template Portlet and Preview Portlet. The workflow JSP file `IWWCMWorkflowPortletView.jsp` sends messages in the following formats to the CTP:

```
"action = Action"
"Attribute = Value"
```

So, to send the message to the Customizable Template Portlet for editing the content, you would need to send two messages, as follows:

```
action=EDIT
docId=selectedDocId
```

The following code generates this message:

```
<a href='<portletAPI:createURI><portletAPI:URIAction
name='<%=IWWCMWorkflowPortlet.WORKFLOW_ACTION%>' /><portletAPI:URIParame
ter name="<%=IWWCMWorkflowPortlet.DOCID%>"
value="<%=docId%>" /><portletAPI:URIParame
ter name="<%=IWWCMWorkflowPortlet.ACTION%>"
value="<%=IWWCMWorkflowPortlet.EDIT%>" /></portletAPI:createURI>'></a>
```

The Preview Portlet receives messages in the following format:

```
WCM_GLOBAL_CONTEXT=[site area]/[content name]?id=[value]
```

The following code formats this message:

```
<a href="<%=currentURI %>
?WCM_GLOBAL_CONTEXT=<%=IWWCMWorkflowPortlet.getContentUrl(docId,portletRequ
est,sessionBean)%>">
</a>
<%} %>
```

13.3 Extensions to the Customizable Template Portlet

This section describes the source code that implements the Link Management Extensions that are discussed in 9.5, “Implementing a custom display Portlet” on page 418. Refer to this section for a detailed explanation of the concept behind these extensions.

For the user interaction the Customizable Template Portlet implements an `ActionListener` (`actionPerformed` method). This method is also responsible for the save event. It delegates the write to IBM Workplace Web Content Management backend operation to a method called `writeToWCM()`. To implement the link management, you have to extend this method.

After the content document is saved [`workspace.save(content)`], you add some code that creates the link-list. As outlined in Chapter 9, “Advanced integration with WebSphere Portal” on page 403, we have to parse the rendered content for all occurrences of the `link2content` JavaScript function.

Example 13-16 renders the content document and then passes the resulting HTML to the `parseContent` method.

Example 13-16 Rendering and passing HTML to the `parseContent` method

```
try {
    RenderingContext context = workspace.createRenderingContext(p_request,
null, map);
    context.setRenderedContent("/WP Experts/Home/" + content.getName());
    String rendered = workspace.render(context);
    System.err.println("content rendered: " + rendered);
    parseContent(rendered, content);
    p_sessionBean.setContentSaveResults(workspace.save(content));
    p_sessionBean.setContentSaveResults(workspace.save(siteArea));
} catch (Exception ge) {
    ge.printStackTrace();
}
```

The `parseContent` method then uses a regular expression to build the link list, as shown in Example 13-17.

Example 13-17 `parseContent` method to build a link list

```
try {
    Pattern Regex =
Pattern.compile("(?:link2content\\(')(.*)'(?:',')(.*)'(?:'\\)\"",
Pattern.CANON_EQ);
    Matcher RegexMatcher = Regex.matcher(content);
    while (RegexMatcher.find()) {
        for (int i = 1; i <= RegexMatcher.groupCount(); i++) {
```

```

        sb.append(RegexMatcher.group(i));
        sb.append(";");
    }
}
sb.deleteCharAt(sb.length() - 1);
} catch (PatternSyntaxException ex) {
    // Syntax error in the regular expression
}

```

After this the result is stored in the links field, as shown in Example 13-18.

Example 13-18 Result is stored in the links field

```

if (doc.hasComponent("links")){
    TextComponent bodyComponent;
    try {
        System.err.println("setting component");
        bodyComponent = (TextComponent) doc.getComponent("links");
        bodyComponent.setText(sb.toString());
        doc.setComponent("links", bodyComponent);
    } catch (ComponentNotFoundException e) {
        e.printStackTrace();
    } catch (IllegalTypeChangeException e) {
        e.printStackTrace();
    }
}
}

```

In the next step, you use the built link list in the rendering phase of the Custom Workplace Web Content Management Portlet. You do this in the `doView` method of the Portlet. As outlined in Chapter 9, “Advanced integration with WebSphere Portal” on page 403, you have to create the final links in this step. 9.5, “Implementing a custom display Portlet” on page 418 explains the concepts behind this.

The code builds a JavaScript array for us, as shown in Example 13-19. The result is JavaScript code that you use in a later step. It uses a `StringTokenizer` to cut the link list into pieces and then calls `getPortalLink` to create the links. The resulting JavaScript array uses the combination of `pagename+docId` as its key.

Example 13-19 JavaScript which uses a StringTokenizer to cut the link list into pieces

```

TextComponent bodyComponent = (TextComponent) doc.getComponent("links");
StringTokenizer stok = new StringTokenizer(bodyComponent.getText(), ";");
linkBuf.append("var links = new Array();\n");
while (stok.hasMoreElements()) {
    String pageName = stok.nextToken();
    if (stok.hasMoreElements()) {
        String docId = stok.nextToken();

```

```

        linkBuf.append("links[\"");
        linkBuf.append(pageName);
        linkBuf.append(docId);
        linkBuf.append("\"] = \");
        linkBuf.append(getPortalLink(request, pageName, docId));
        linkBuf.append("\";\n");
    }
}

```

The `getPortalLink` method makes use of the `CreateURLCommand` class to create links to a Portal page, as shown in Example 13-20. It also allows you to add a URL parameter.

Example 13-20 The `getPortalLink` method makes use of the `CreateURLCommand`

```

try {
    CreateUrlCommand iCreateUrlCommand = new CreateUrlCommand();
    iCreateUrlCommand.setRunData(runData);
    iCreateUrlCommand.setContentNode(iComposition);
    iCreateUrlCommand.setPacCheck(null);
    iCreateUrlCommand.setRequestID(true);
    iCreateUrlCommand.addQueryData(DOCUMENT_ID, param);
    iCreateUrlCommand.execute();
    URL iURL = iCreateUrlCommand.getURL();
    if (iURL != null) {
        uri = iURL.toString();
    }
} catch (CommandException e) {
    System.err.println(e.getMessage());
}

```

Now, you can make use of the JavaScript array. Example 13-21 embeds the link list array on the `PortletView.jsp` and also implements the `link2content` function. The function takes the unique name of the Portal page and the document ID as input. It then does a lookup in the JavaScript array and resolves the link. Finally, the link then is executed.

Example 13-21 JavaScript array embedding the link list array

```

<script language="JavaScript" type="text/javascript">
<!--
<%= portletRequest.getAttribute(WCMCustomPortlet.JS_LINK_ARRAY) %>
function link2content(node, id) {
    window.location.href=links[node+id];
}
//-->
</script>

```

Migrating to IBM Workplace Web Content Management 5.1

This chapter describes how to migrate a Web site from a previous version of IBM Workplace Web Content Management into IBM Workplace Web Content Management 5.1. It includes the following sections:

- ▶ Overview of migrating data from a previous version
- ▶ Transferring file system data
- ▶ Verifying and troubleshooting file system data transfer
- ▶ Migrating users and groups
- ▶ Verifying and troubleshooting user and group migration
- ▶ Creating the unrestricted resources
- ▶ Migrating data from a previous version
- ▶ Enabling user migration
- ▶ Migrating users directly to WebSphere Member Manager
- ▶ Migrating users to WebSphere Member Manager using a user migration mapping file
- ▶ Disabling user migration

14.1 Overview of migrating data from a previous version

In earlier versions of Lotus Workplace Web Content Management, you could store data and users in the file system as well as in the supported JDBC databases and IBM Content Manager. In IBM Workplace Web Content Management 2.5 and 5.1, the file system is no longer a supported data persistency.

If you upgrade from an older version of Lotus Workplace Web Content Management and your data was stored on the file system, you need to follow the correct data transfer and migration procedures. If your data was stored in a JDBC database or IBM Content Manager, you do not need to transfer the data, but you do need to configure IBM Workplace Web Content Management to access the data. For all data storage types, you need to migrate the Lotus Workplace Web Content Management users and groups to IBM WebSphere Member Manager and configure the data so that it uses the WebSphere Member Manager users and groups.

Note: You can upgrade from any previous version of Lotus Workplace Web Content Management. To upgrade from a version of Aptrix, first upgrade to one of the versions of Lotus Workplace Web Content Management and then upgrade to IBM Workplace Web Content Management 2.5 or 5.1. The steps for upgrading Aptrix to Lotus Workplace Web Content Management are out of the scope of this book. Refer to the Lotus Workplace Web Content Management Information Center for the steps for upgrading Aptrix at:

<http://www-128.ibm.com/developerworks/workplace/documentation/webcontentmanagement/>

Note: IBM Workplace Web Content Management 2.5 requires WebSphere Portal 5.0.2. If you upgrade to Portal 5.1, you need to upgrade to IBM Workplace Web Content Management 5.1. When upgrading from IBM Workplace Web Content Management 2.5 to IBM Workplace Web Content Management 5.1, you need to uninstall 2.5 and install 5.1 on Portal 5.1. There is no need to migrate the IBM Workplace Web Content Management data, because version 2.5 stores data in a supported repository and uses a supported user repository such as WebSphere Member Manager or LDAP. This book describes the process for upgrading from a previous version of Lotus Workplace Web Content Management to IBM Workplace Web Content Management 2.5 or 5.1.

14.1.1 Differentiating upgrading, transferring, and migrating

The terms *upgrading*, *transferring*, and *migrating* are used when describing the process of moving from a previous version of Lotus Workplace Web Content Management to a newer version.

Upgrading refers to the process of installing a newer version of IBM Workplace Web Content Management. This process includes:

- ▶ Transferring, which refers to the process of moving existing data from one repository to another.
- ▶ Migrating, which refers to the process of recreating existing Lotus Workplace Web Content Management users and groups in WebSphere Member Manager and changing the Workplace Web Content Management data so that it points to the new WebSphere Member Manager users and groups.

14.1.2 Upgrading IBM Workplace Web Content Management

Table 14-1 provides an overview of the steps for upgrading IBM Workplace Web Content Management. Figure 14-1 on page 568 illustrates these steps.

Table 14-1 Steps for upgrading Workplace Web Content Management

Step	Action
1	Back up your existing data repository and users folder prior to upgrading as follows: ▶ Back up your existing data using the backup solution recommended for the data repository. ▶ Back up your existing users by copying the [ILWWCM_HOME]\connect\users folder. This folder does not contain groups, which are saved in the data folder. The users are also saved in the data folder. Note: IBM Workplace Web Content Management does not have a built-in backup feature.
2	Install the new version of IBM Workplace Web Content Management.
3	Configure the new data repository. Note: If you intend to use IBM Cloudscape database, there is no need to perform this step, because at installation IBM Workplace Web Content Management configures Cloudscape as the default data repository.
4	Edit the wpconfig.properties file to point to the file system location. Note: This step is only required if the existing data is stored in the file system.

Step	Action
5	Transfer existing data into the new data repository. Note: This step is only required if the existing data is stored in the file system.
6	Migrate the existing Lotus Workplace Web Content Management users and groups to WebSphere Member Manager and configure the data to use the new WebSphere Member Manager users and groups.
7	Run the resource checker utility to recreate the resources in the <code>WEB_APP_HOME</code> folder so that the unrestricted resources appear on the Web site.
8	Test your upgraded data and correct any errors using the troubleshooting tips.

Important: Test the upgrade in a test environment before upgrading the production environment.

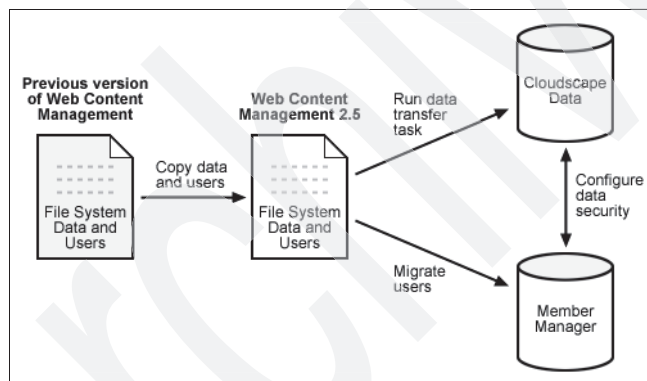


Figure 14-1 Transferring file system data and migrating users

14.2 Transferring file system data

IBM Workplace Web Content Management 2.5 and 5.1 can use data from an older version of Lotus Workplace Web Content Management. The only changes that might be required when upgrading are as follows:

- If the older version stored data in the file system, then the data needs to be transferred to a Cloudscape database and then, if required, to one of the supported data repositories, which include a JDBC database or Content Manager.

- If the older version stored data in a supported JDBC database or Content Manager, then you do not need to do anything to the data unless you plan to use a different JDBC database.

14.2.1 Transferring file system data

Table 14-2 provides an overview of the steps for transferring file system data from an older version of Lotus Workplace Web Content Management.

Table 14-2 Transferring file system data

Step	Action
1	Copy the existing Lotus Workplace Web Content Management data folder to the <code>ILWWCM_HOME\ilwwcm</code> folder. Notes: ► The file system data folder of the previous version is located in the previous version's <code>ILWWCM_HOME\ilwwcm</code> folder. ► The destination folder can be different, provided that the path specified in the <code>wpconfig.properties</code> file is correct.
2	Point to the copied file system data by entering the full path to the data folder in the <code>wpconfig.properties</code> file. The following is an example of how the line of code might appear: <pre>WcmFileSystemDataLocation=C:/WebSphere/PortalServer/wcm/ilwwcm/data</pre>
3	Run the <code>transfer-wcm-filesystem-to-cloudscape</code> task to transfer the file system data from the <code>ILWWCM_HOME\ilwwcm</code> folder to a Cloudscape database.
4	If required, transfer the data to another database type. Note: If you intend to store data in another database type, you must first transfer the file system data to a Cloudscape database and then transfer it to the required database type.

Note: IBM Workplace Web Content Management must be installed before you can transfer the file system data to a Cloudscape database.

14.2.2 Using the wpconfig.properties file during data transfer

The system uses the following parameter to locate the Lotus Workplace Web Content Management data on the file system:

`WcmFileSystemDataLocation`

The wpconfig.properties file is located in the `WPS_HOME\config` folder.

14.2.3 Running the data transfer task

Run the task from the `WPS_HOME\config` directory using a command prompt as follows:

- ▶ In Windows, at the command prompt type the following:
`wpsconfig.bat transfer-wcm-filesystem-to-cloudscape`
- ▶ In UNIX, at the command prompt type the following:
`wpsconfig.bat/.sh transfer-wcm-filesystem-to-cloudscape`

14.2.4 Ensuring valid data format

If you plan to use an IBM DB2 database on z/OS®, make sure the data that was stored in the file system has a valid format. If the format is invalid, problems occur when you transfer the data from the Cloudscape database to the DB2 database on z/OS.

An XML file format is considered valid only if it has one of the following end of line styles:

- ▶ Carriage return and line feed (CRLF), which is the Windows end of line style
- ▶ Line feed (LF), which is the UNIX end of line style

An XML file that contains *both* CRLF and LF is considered *invalid* because it has inconsistent end of line styles.

14.3 Verifying and troubleshooting file system data transfer

This section presents information that guides you in verifying that the data transfer was a success and in identifying the cause of errors.

14.3.1 Verifying the data transfer

To verify that data transfer from the file system to the Cloudscape database was successful, the following messages should appear:

- ▶ Build Successful should appear in the command prompt.
- ▶ Utility Completed should appear in the *WPS_HOME\log\wcm_migration.log* file.

14.3.2 Examining the error messages

If errors occur during file system data transfer, the error messages appear in the command prompt and in the *wcm_migration.log* file. Three types of errors might occur:

- ▶ `NullPointerException`
- ▶ Connection Error: Failed to start database
- ▶ No schema exists for the user

NullPointerException

For file system data transfer, a `NullPointerException` indicates that data folders are missing. Do the following to resolve the errors:

- ▶ In the *wpconfig.properties* file, make sure the value of the `WcmFileSystemDataLocation` parameter is correct. This is the path to the copied file system data.
- ▶ In the *wpconfig.properties* file, make sure that the IBM Workplace Web Content Management parameters have values and that the values are correct, particularly the password for the `WcmDbPassword` parameter.

- Make sure the following data folders exist, particularly the folders indicated in ***italic bold*** font. Create the missing folders, even if they did not exist prior to data transfer. The folders can be empty.
 - [ILWWCM_HOME]\ilwwcm\cmpnts
 - [ILWWCM_HOME]\ilwwcm\contents
 - ***[ILWWCM_HOME]\ilwwcm\directoryProfile***
 - [ILWWCM_HOME]\ilwwcm\itemGatherers
 - [ILWWCM_HOME]\ilwwcm\profileMappers
 - [ILWWCM_HOME]\ilwwcm\resources
 - [ILWWCM_HOME]\ilwwcm\security
 - [ILWWCM_HOME]\ilwwcm\sites
 - [ILWWCM_HOME]\ilwwcm\sites\sitearea
 - [ILWWCM_HOME]\ilwwcm\styles
 - [ILWWCM_HOME]\ilwwcm\subscribers
 - [ILWWCM_HOME]\ilwwcm\syndicators
 - [ILWWCM_HOME]\ilwwcm\taxonomy
 - [ILWWCM_HOME]\ilwwcm\taxonomy\categories
 - [ILWWCM_HOME]\ilwwcm\templates
 - [ILWWCM_HOME]\ilwwcm\workflows
 - [ILWWCM_HOME]\ilwwcm\workflows\actions
 - [ILWWCM_HOME]\ilwwcm\workflows\stages

Connection Error has occurred: Failed to start database

If the build failed and the wcm_migration.log reports the error Connection Error has occurred: SQL Exception: Failed to start database, it indicates that the WebSphere Portal server is started. The data transfer can only occur while the WebSphere Portal server is stopped. Stop the WebSphere Portal server by using the **WebSphere_Portal** command before running the file system data transfer task.

No Schema Exists

If the build failed and the wcm_migration.log reports an error that no schema exists for the user, it indicates that no database schema has been defined and the database administrator user name values in the wpconfig.properties and aptrixjpe.properties files are different. In the aptrixjpe.properties file, if you do not define a database schema using the jdbc.tableSchema=[JDBC_SCHEMA] parameter, then the schema defaults to the user name that is defined by the jdbc.username=APP parameter. If you do not define a schema, make sure that the user name values are the same in both files, as follows:

- In wpconfig.properties the default value is WcmDbUser=APP.
- In aptrixjpe.properties the default value is jdbc.username=APP.

14.3.3 Known limitations

Table 14-3 describes the known limitations of the IBM Workplace Web Content Management data transfer process.

Table 14-3 Known limitations

Limitation	Description
No directory checking	The data transfer process does not validate that all folders exist. If a folder is missing, a <code>NullPointerException</code> error message appears.
No property validation	The data transfer process does not validate the IBM Workplace Web Content Management parameters in the <code>wpconfig.properties</code> file. If a parameter value is missing or is incorrect, a <code>NullPointerException</code> error message appears.
No tracing	Because there is no tracing, you cannot debug the errors easily.

14.3.4 Files that are needed by the IBM Support Desk

If you encounter problems that you cannot solve while transferring file system data to a Cloudscape database and if you have a support contract with the IBM Support Desk, send the following files or folders to the support desk:

- ▶ `wpconfig.properties` located in `WPS_HOME\config`
- ▶ `wcm_migration.log` located in `WPS_HOME\log`
- ▶ data folder located in `ILWWCM_HOME\ilwwcm`

14.4 Migrating users and groups

WebSphere Member Manager manages the users and groups that are granted access to IBM Workplace Web Content Management items. In previous versions, Lotus Workplace Web Content Management itself created and managed users and groups.

After you transfer the data to the new data repository, you need to migrate the Lotus Workplace Web Content Management users from the previous version to WebSphere Member Manager. When you migrate a Lotus Workplace Web Content Management user or group to WebSphere Member Manager, the migration process changes the references within the IBM Workplace Web Content Management items from the Lotus Workplace Web Content Management user or group to the WebSphere Member Manager user or group.

automatically. The site developer might need to edit selected items whose references were not updated automatically.

Important: When user migration is complete, you need to grant access to the authoring portlet for each user or group.

14.4.1 Migrating users

Table 14-4 provides an overview of the steps for migrating users from an older version of Lotus Workplace Web Content Management.

Table 14-4 Migrating users

Task	Procedure
1	Before uninstalling the previous version of Lotus Workplace Web Content Management, run the User Checker utility with <code>create=true</code>. Notes: ▶ The utility ensures that all users in the data repository have a corresponding Lotus Workplace Web Content Management Framework user. ▶ The Lotus Workplace Web Content Management Framework was the code that created users in previous versions of Lotus Workplace Web Content Management. The Lotus Workplace Web Content Management Framework no longer exists in IBM Workplace Web Content Management 2.5 and 5.1 because its function is performed by WebSphere Member Manager.
2	Back up the existing Lotus Workplace Web Content Management users folder. Note: The users folder is located in <code>ILWWCM_HOME\connect</code>.
3	Copy the existing Lotus Workplace Web Content Management users folder to the new version's <code>ILWWCM_HOME\wcm\connect</code> folder.
4	If you require that WebSphere Portal security be enabled, make sure it is enabled prior to running user migration. By default, WebSphere Portal security is not enabled. Note: You can run user migration when WebSphere Portal security is not enabled. If you enable WebSphere Portal security after running user migration, you need to run user migration again.
5	Enable migration by opening the WebSphere Application Server Administrative Console and adding the IBM Workplace Web Content Management migration .jar file (<code>ilwwcm-migration.jar</code>) to the Web Content Management Shared Library (<code>WcmLib</code>).

Task	Procedure
6	Open the Portal migration page and migrate the users and groups using one of the following methods: ► Migrate the users and groups directly to WebSphere Member Manager. ► Generate, review, and then process a user migration mapping file and migrate the users and groups to WebSphere Member Manager using the user migration mapping file. Note: The following URL accesses the migration interface: http://[HOST]:[PORT]/wps/wcm/jsp/migration/migration.jsp
7	If you use a user migration mapping file, run the XMLaccess command to create new WebSphere Member Manager users and groups for existing Lotus Workplace Web Content Management users and groups. Note: The user migration process specified in Task 6 does not create new users and groups in WebSphere Member Manager. It only compares and verifies existing users and groups.
8	Rebuild the indexes by deleting the system folder that is located in the <i>ILWWCM_HOME\ilwwcm</i> folder and start the WebSphere Portal server. Note: The system folder contains the indexes.
9	Disable migration by opening the WebSphere Application Server Administrative Console and removing the IBM Workplace Web Content Management migration .jar file (ilwwcm-migration.jar) from the Web Content Management Shared Library (wcmLib).

Important: Do not delete the *ILWWCM_HOME\connect\users* folder because you need it to repeat the user migration process. Repeat the user migration process if you enable or disable WebSphere Portal security or if you change your user repository.

Tip: You can run the user migration process when the data is in the Cloudscape database, or later if you transfer the data to another database type. You do not need to run the user migration process every time you transfer data to another database type.

14.4.2 Creating migrated users and groups with WebSphere Member Manager

When you migrate a user to WebSphere Member Manager, it does not recreate a new user if it contains an existing user that is an exact match. It *does* create a new user if the Lotus Workplace Web Content Manager user does not exist in WebSphere Member Manager. WebSphere Member Manager is case-sensitive and does not recognize spaces. Therefore, WebSphere Member Manager creates a new user if the case does not match and creates a new user with underscores (_) in place of spaces if the Lotus Workplace Web Content Management name contains spaces. This behavior also applies to the recreation and migration of groups.

The migration process might result in a user having two user names in WebSphere Member Manager. For example, the user *John Smith* might have a Lotus Workplace Web Content Management user name of *John Smith* and a WebSphere Member Manager user name of *John A Smith*, resulting in the creation of a new WebSphere Member Manager user name to correspond to the existing Lotus Workplace Web Content Management user name.

The migration process might result in a Lotus Workplace Web Content Management user not being created in WebSphere Member Manager. For example, the user *Mary Jones* might refer to a Lotus Workplace Web Content Management user who is not the same *Mary Jones* who is an existing WebSphere Member Manager user. The migration process would cause the IBM Workplace Web Content Management items to reference the existing Mary Jones user in WebSphere Member Manager, resulting in the Lotus Workplace Web Content Management Mary Jones losing access to the Workplace Web Content Management items.

14.4.3 Migrating users and groups

There are two ways that you can migrate Lotus Workplace Web Content Management users and groups to WebSphere Member Manager:

- ▶ Migrating directly to WebSphere Member Manager
- ▶ Migrating to WebSphere Member Manager using a user migration mapping file

Migrate Lotus Workplace Web Content Management users and groups directly to WebSphere Member Manager if you are sure that existing Lotus Workplace Web Content Management users and groups *exactly* match the existing WebSphere Member Manager users and groups. Direct migration is suitable if you are installing WebSphere Portal for the first time and do not have existing WebSphere Member Manager users or groups.

Migrate Lotus Workplace Web Content Management users and groups using a user migration mapping file if the match is *not* exact between existing Lotus Workplace Web Content Management users and groups and existing WebSphere Member Manager users and groups. A user migration mapping file contains the original Lotus Workplace Web Content Management user and groups and the new WebSphere Member Manager user and groups names. When you generate a user migration mapping file, you can edit the file and map the existing Lotus Workplace Web Content Management users and groups to the correct WebSphere Member Manager users and groups. This method is recommended if you were using WebSphere Portal and WebSphere Member Manager prior to using IBM Workplace Web Content Management.

Important: Direct migration might result in an error message that the required `sn` attribute for each user does not contain a value. The reason for the empty attribute is that users and groups in earlier versions of Lotus Workplace Web Content Management do not provide a `lastname` attribute. If you encounter this error, you need to migrate users and groups using a user migration mapping file and run the `XMLaccess` command.

14.4.4 Migrating to WebSphere Member Manager using a user migration mapping file

The `migrationUsers.properties` file is the user migration mapping file. It contains the mapping of the original Lotus Workplace Web Content Management users and groups to the new WebSphere Member Manager users and groups. You need to review and modify this file before processing it. The original Lotus Workplace Web Content Management users and groups are listed on the left, and the WebSphere Member Manager users and groups are listed on the right. Modify the file as follows:

- ▶ Change only the proposed WebSphere Member Manager names.
- ▶ Change the property values of the WebSphere Member Manager name to use the distinguished name, for example:
`uid=username,o=default organization`
`cn=group name,o=default organization`
- ▶ Do not use an ampersand (&) in user or group names because it causes errors.

The following line of code is an example of how to change the group name `All Staff` to `All_WP_Staff` using the distinguished name:

```
All\ Staff=cn=All_WP_Staff,ou=groups,dc=ibm,dc=com
```

The migrationUsers.properties file is located in the *ILWWCM_HOME*migration folder.

14.4.5 Adding new users and groups to WebSphere Member Manager using the migrationUsers.xml file

The user migration mapping file compares only existing WebSphere Member Manager user and group names with existing Lotus Workplace Web Content Management user and group names. When you use the user migration mapping file, WebSphere Member Manager does not create new user or group names if the names are not already in WebSphere Member Manager. In order to create new WebSphere Member Manager users and groups for existing Lotus Workplace Web Content Management users and groups, run the **XMLaccess** command.

The **XMLaccess** command uses the migrationUsers.xml file for user and group definition. The migrationUsers.xml file contains the required default attributes to define a user or group. If your LDAP server requires different attributes for user or group definition, change the file to contain the appropriate attributes for the LDAP server. Ask your LDAP administrator for the required attributes for your LDAP server.

Important: Users and groups in earlier versions of Lotus Workplace Web Content Management do not provide a `lastname` attribute. You must type a value for the `lastname` attribute in the migrationUsers.xml file. Otherwise, you will not be able to run the **XMLaccess** command. The value for the `lastname` attribute can be any value. The definition and use of the migrationUsers.xml file requires WebSphere Portal administration skills.

14.4.6 Migrating the Administrators group

In previous versions of Lotus Workplace Web Content Management, the Administrators group was granted full access to the Lotus Workplace Web Content Management user interface, including every item and the application backend files. Any user who was a member of this group was granted super user access.

When you migrate users and groups, you have the option to migrate the previous Lotus Workplace Web Content Management Administrators group. If you migrate the Administrators group, the migration program does not create a new Administrators group in WebSphere Member Manager because the `wcmadmins` group already exists. Instead, it copies the members of the Administrators group into the `wcmadmins` group.

To migrate the Administrators group, select **Import Administrators Group** in the WebSphere Portal migration page.

14.4.7 Migrating LDAP users and groups

If you used an LDAP server to manage users and groups in Lotus Workplace Web Content Management 2.0, you must still run user migration. However, you must perform one of the following steps prior to migrating users or groups.

- ▶ Configure WebSphere Member Manager to use your existing LDAP server.
- ▶ Copy your existing LDAP users or groups into the LDAP server being used by WebSphere Member Manager.

Important: You cannot migrate custom LDAP attributes that were used to map Lotus Workplace Web Content Management profiling values to LDAP users and groups.

14.4.8 UI Access Group

The *UI Access Group* is a predefined group in the previous version of Lotus Workplace Web Content Management. In earlier versions the group was called *AprixUIAccessGroup*. Members of this group gain access to the user interface of earlier versions. This group no longer exists in IBM Workplace Web Content Management 2.5 and 5.1.

When you migrate users, users who were members of the UI Access Group or *AprixUIAccessGroup* become members of the virtual Portal group *all authenticated portal users*. This group has access to the Web Content Management label and the pages on which the authoring and rendering portlets are located.

When you migrate groups, make sure the UI Access Group or *AprixUIAccessGroup* is not a member of a group.

14.4.9 ALL USERS group

The *ALL USERS* group is predefined group in previous versions of Lotus Workplace Web Content Management. This group no longer exists in IBM Workplace Web Content Management 2.5 and 5.1.

When you migrate users, users who were members of the ALL USERS group become members of the virtual Portal group *all users*. Members of the *all users* group are granted access to items that do not require restricted access.

14.4.10 Anonymous user

The *Anonymous* user is a predefined user in previous versions of Lotus Workplace Web Content Management. This group no longer exists in IBM Workplace Web Content Management 2.5 and 5.1.

When you migrate users, the Anonymous user maps directly to the virtual Portal anonymous user. Items that granted access to the Lotus Workplace Web Content Management Anonymous user now grant access to the Portal anonymous user.

14.4.11 Member Fixer utility

The *Member Fixer* utility checks the security settings in all IBM Workplace Web Content Management items. When run in fix mode, it removes references to users and groups who do not exist in WebSphere Member Manager and updates user and group names that have been changed in WebSphere Member Manager.

If nonexistent users and groups are referenced in IBM Workplace Web Content Management items, performance might be slowed while they are checked. Performance might be improved if nonexistent users and groups are removed.

Attention: Running the Member Fixer utility in fix mode can cause serious security problems. For example, if a user or group was not migrated correctly or if IBM Workplace Web Content Management is not properly connected to WebSphere Member Manager, valid users and groups might be removed from the security sections in IBM Workplace Web Content Management items.

View a report of users and groups using the following URL:

`http://[HOST]:[PORT]/wps/wcm/connect/?MOD=MemberFixer`

Remove references to nonexistent users and groups, and update changes to users and groups in IBM Workplace Web Content Management items using the following URL:

`http://[HOST]:[PORT]/wps/wcm/connect/?MOD=MemberFixer&fix=true`

14.5 Verifying and troubleshooting user and group migration

This section guides you in verifying the success of user and group migration and in identifying the cause of errors.

14.5.1 Verifying user migration

To verify that the user migration is successful, take a random sample of users and groups and do the following:

- ▶ Open WebSphere Portal administration to verify the creation of users and groups.
- ▶ Open the IBM Workplace Web Content Management authoring portlet, and open a content item. Verify that expected categories and keywords appear in the Profile section.
- ▶ Log on to the authoring portlet as different users to verify that security and access controls are working.
- ▶ Log on to a rendered site as different site visitors to verify access to rendered content.
- ▶ Run the User Reference tool in *non-update* mode to make sure that IBM Workplace Web Content Management and WebSphere Member Manager data matches, as in the following example:

```
http://[HOST]:[PORT]/wps/wcm/connect?MOD=AJPReferenceChecker
```

14.5.2 Using the log files

Table 14-5 lists the log files and the types of errors that they report.

Table 14-5 Log files containing user migration errors

Log file	Contents
SystemErr.log	Contains exceptions
Wps_<date>.log	Contains JSP compilation exceptions
SystemOut.log	Contains startup errors

The log files are located in the *WPS_HOME*\log folder.

14.5.3 Examining common errors and resolutions

Table 14-6 lists the errors that might appear in the browser and possible ways to resolve the problem.

Table 14-6 Common errors and resolutions

Error	Problem and resolution
A compilation error appears when you access the migration interface.	Problem: The ilwwcm-migration.jar was not added to the WcmLib classpath field. Resolution: Add the ilwwcm-migration.jar to the WcmLib classpath field.
The migration process fails.	Problem: User and group names contain illegal characters (for example, [,] and *). Resolution: Change the user and group names in the previous version of Lotus Workplace Web Content Management to contain only acceptable characters.
A file not found exception error appears when you start the migration process in the migration interface.	Problem: The users in the <i>ILWWCM_HOME</i>\connect folder are not found. Resolution: ▶ Make sure that you specified the correct <i>ILWWCM_HOME</i>\connect folder in the Input Connect User Path field in the migration interface. ▶ Make sure that all users are in the <i>ILWWCM_HOME</i>\connect folder. ▶ Run the UserChecker utility with create=true in the previous version of Lotus Workplace Web Content Management and then copy the users to IBM Workplace Web Content Management 2.5 or 5.1.
The sn attribute for each user does not contain a value.	Problem: Users and groups in earlier versions of Lotus Workplace Web Content Management do not provide a lastname attribute, which results in an empty sn attribute. This occurs only when using direct migration. Resolution: Migrate users and groups using a user migration mapping file and run the XMLaccess command. You must type a value for the lastname attribute in the migrationUsers.xml file. Otherwise, you will not be able to run the XMLaccess command. The value for the lastname attribute can be any value.

Error	Problem and resolution
You cannot see the data in the authoring portlet.	Problem: You have not run the user migration process. Resolution: Run the user migration process using the migration interface.
You cannot see data in the authoring portlet, but the header indicates that there are several pages of data.	Problem: The data has not been configured to use the WebSphere Member Manager users and groups. Resolution: Repeat the user migration process using the migration interface.
You cannot access content even though you have the proper security access levels.	Problem: The names provided in the user migration mapping file do not match the correct LDAP or WebSphere Member Manager users. This occurs only when migrating users using the user migration mapping file. Resolution: Make sure that you are using the fully qualified distinguished name in the migrationUsers.properties file. If WebSphere Portal security is disabled, the distinguished name is one of the following: <pre>uid=<user>,o=Default Organization cn=<group>,o=Default Organization</pre> Update the file and select Use User Migration Mapping File Option in the migration interface.
Users do not have the proper categories or keywords.	Problem: The categories and keywords have not been added. This occurs only when migrating users using the user migration mapping file. Resolution: Execute XMLaccess using the <code>ILWWCM_HOME\migration\migrationUsers.xml</code> input file.

Error	Problem and resolution
When editing and saving content, the security mappings disappear.	Problem: The names provided in the user migration mapping file do not match the correct LDAP or WebSphere Member Manager users. This occurs only when migrating users using the user migration mapping file. Resolution: Make sure that you are using the fully qualified distinguished name in migrationUsers.properties. If Portal security is disabled, the distinguished name is uid=<user>,o=Default Organization cn=<group>,o=Default Organization Update the file and select Use User Migration Mapping File Option in the migration interface.
You cannot see the data in the authoring portlet, but the migration was successful.	Problem: The indexes have not been updated with the newly configured data. Resolution: Stop the WebSphere Portal server, delete the indexes, and start the WebSphere Portal server to rebuild the indexes.

14.5.4 Files needed by the IBM Support Desk

If you encounter problems that you cannot solve and if you have a support contract with the IBM Support Desk, send the following files or folders to the support desk:

- ▶ Data folder located in *ILWWCM_HOME\ilwwcm*
- ▶ Users folder located in *ILWWCM_HOME\connect*
- ▶ Wps_<date>Log located in *WPS_HOME\log*
- ▶ SystemOut.log located in *WPS_HOME\log*
- ▶ SystemErr.log located in *WPS_HOME\log*
- ▶ Connect.log with trace level 3 located in *ILWWCM_HOME\connect\log*

14.6 Creating the unrestricted resources

Resources, which comprise Web page elements such as images, are part of the data transfer from one repository to another. Resources that require restricted access appear on a Web page only after a site visitor logs on. Resources that do not require restricted access appear when anonymous and logged on site visitors view the Web page. When you transfer the resources to IBM Workplace Web Content Management 2.5 or 5.1, unrestricted resources require additional

step before they will appear on a Web page, because they are stored and accessed differently than restricted resources.

14.6.1 Where resources are stored

IBM Workplace Web Content Management stores resources that do not require restricted access in the following two locations:

- ▶ In the specified resources repository, defined by the parameter `resource.persistance` in the `aptrixjpe.properties` file.
- ▶ In the `WEB_APP_HOME\resources` folder.

When file system data is transferred to a Cloudscape database, the `WEB_APP_HOME\resources` folder is not created automatically. You need to create it using the Resource Checker utility.

IBM Workplace Web Content Management stores resources that require restricted access in the specified resources repository only.

14.6.2 Accessing unrestricted resources

When a Web page opens, it accesses unrestricted resources from the `WEB_APP_HOME\resources` folder. The Web page accesses restricted resources from the repository, but it cannot access unrestricted resources from the resources repository even though they are also stored there. When you test if the data transfer and user migration was successful, you will preview a Web page. You are prompted to log on, which provides access to the restricted resources. However, the unrestricted resources do not appear on the Web page.

Figure 14-2 on page 586 shows a previewed Web page with an image placeholder in place of the image, because the image does not require restricted access and the `WEB_APP_HOME\resources` folder does not exist after the data transfer.



Figure 14-2 Image place holder in place of an image

14.6.3 Resource Checker utility

The Resource Checker, when run in fix mode, creates the `WEB_APP_HOME/resources` folder and copies the unrestricted resources in the repository and recreates them in this folder. Recreate the resources using the following URL:

`http://[HOST]:[PORT]/wps/wcm/connect?MOD=AJPEResourceChecker&fix=true`

14.7 Migrating data from a previous version

This section explains the process for migrating data from a previous version of IBM Workplace Web Content Management.

14.7.1 Transferring file system data to a Cloudscape database

To transfer file system data to a Cloudscape database, do the following:

1. Copy the data folder from the *ILWWCM_HOME\ilwwcm* folder of the previous version of Lotus Workplace Web Content Management to the *ILWWCM_HOME\ilwwcm* folder of IBM Workplace Web Content Management 2.5 or 5.1, as illustrated in Figure 14-3.

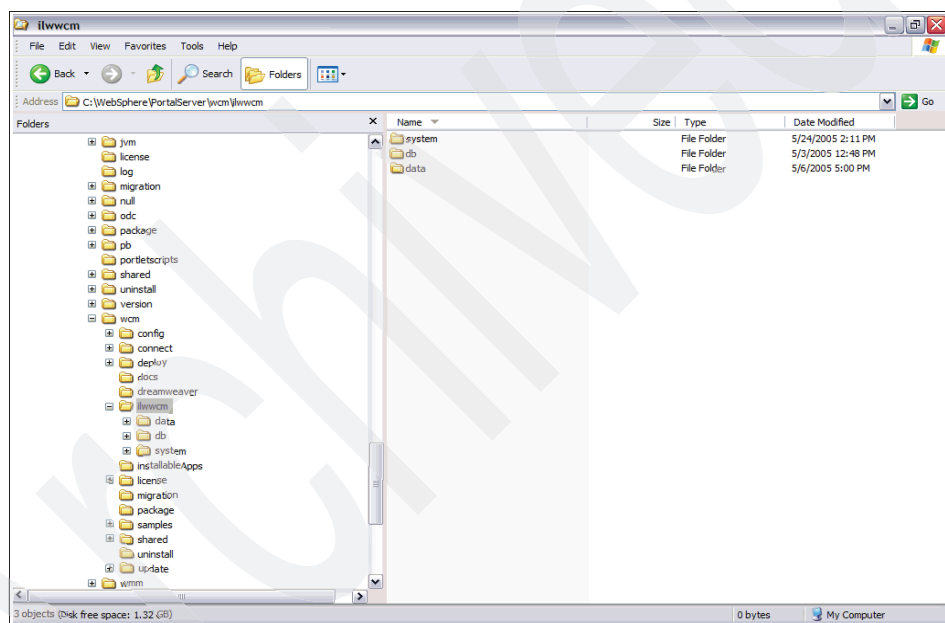


Figure 14-3 Copy the previous version's data folder to ILWWCM_HOME\ilwwcm

2. Open the wpconfig.properties file in a text editor, as shown in Figure 14-4.

Note: The wpconfig.properties file is located in *WPS_HOME\config*.

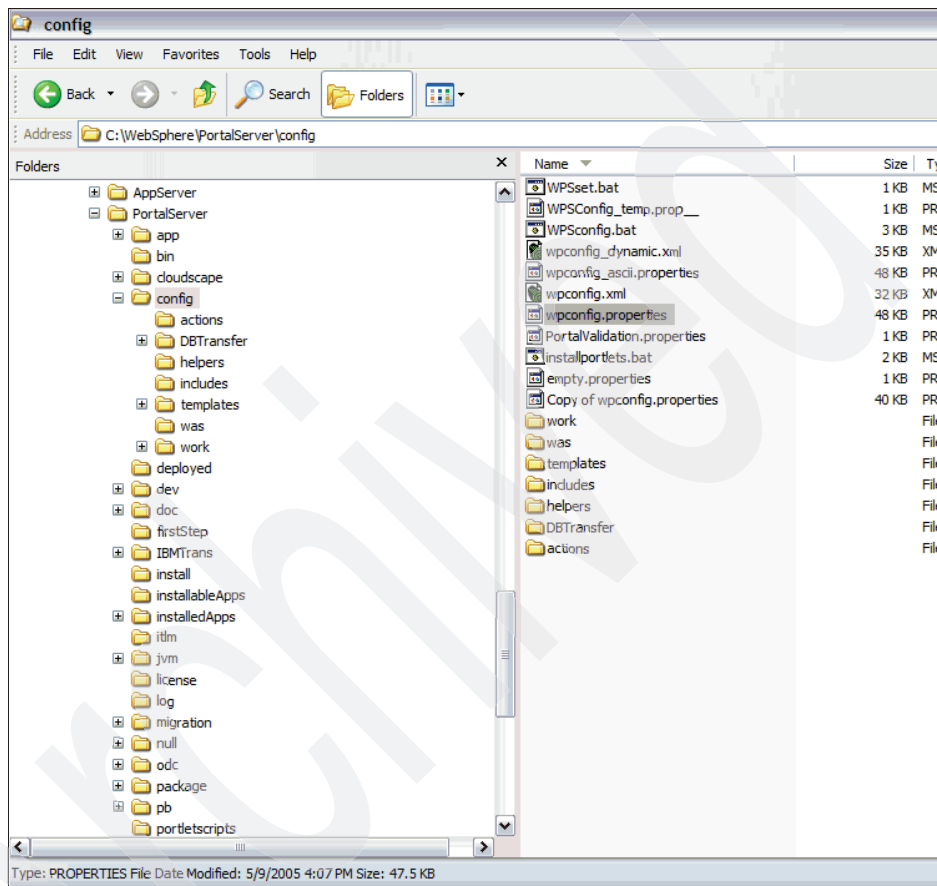


Figure 14-4 Navigate to wpconfig.properties file

3. Find the following line of code:

```
WcmFileSystemDataLocation=
```

4. Enter the path to the file system data files that you copied in the previous procedure (Figure 14-5). The following line of code is an example of how the code might appear:

```
WcmFileSystemDataLocation=C:/WebSphere/PortalServer/wcm/ilwcm/data
```

```
ReplaceWithYourPassword

pIdShort: The WCM admin group ID
dShort=wcmadmins

e WCM database URL
Database element of this value should match the value of WcmDbName
5, the database element of this value should match the location
f the database to be used when establishing connections with
2 datasource used by WCM
udscape for Windows, the path in the WcmDbUrl must not include spaces.
he short path for Windows.
{ jdbc:db2j:C:/path/to/WCMDB }
{ jdbc:db2j:WCMDB }
{ jdbc:db2j:*LOCAL/QWCMDB }
(remote): { jdbc:as400:<hostName>/QWCMDB }
{ jdbc:db2j:<location> }
ote db): { jdbc:db2j://<server address>:<port>/<location> }
{ jdbc:oracle:thin:@<YourDatabaseServer>:1521:WCMDB }
{ jdbc:microsoft:sqlserver://hostname:1433;User=wcmuser;Password=password;SelectMethod=curs
db2j:C:/WEBSPH~1/PORTAL~1/wcm/ilwcm/db/WCMDB

DataLocation: The location of the ILWCM 2.5 filesystem data directory to transfer
only used when migrating from 2.5 filesystem.
ataLocation=C:/WebSphere/PortalServer/wcm/ilwcm/data

The name of class SqlProcessor will use to import SQL files, also known as "JDBC provider"
{ com.ibm.db2j.jdbc.DB2jDriver }
{ COM.ibm.db2.jdbc.app.DB2Driver }
{ com.ibm.db2.jdbc.app.DB2Driver }
(remote db): { com.ibm.as400.access.AS400JDBCdriver }
{ com.ibm.db2.jcc.DB2Driver }
ote db): { com.ibm.db2.jcc.DB2Driver }
{ oracle.jdbc.driver.OracleDriver }
{ com.microsoft.jdbc.sqlserver.SQLServerDriver }
m.ibm.db2j.jdbc.DB2jDriver

s: The name of class SqlProcessor will use to import SQL files via data source
{ com.ibm.db2.jcc.DB2ConnectionPoolDataSource }
ote db): { com.ibm.db2.jcc.DB2ConnectionPoolDataSource }
{com.ibm.db2.jdbc.app.UDBXADataSource}
com.ibm.db2j.jcc.DB2ConnecitonPoolDataSource
```

Figure 14-5 The path to the file system data files

5. Make sure that all the IBM Workplace Web Content Management parameters have values and that the values are correct, particularly the password for the WcmDbPassword parameter.

Note: The four IBM Workplace Web Content Management parameters for Content Manager do not require values.

6. Save and close the wpconfig.properties file.
7. Stop the WebSphere_Portal application server.
8. Start the WebSphere_Portal application server.
9. Open a command prompt.
10. Change the directory to the WPS_HOME\config directory.

Note: Make sure you change to the correct folder on your system. For example, Portal server might be installed on the D drive.

11. At the *WPS_HOME*\config prompt, enter the following line of code for the Windows platform:

```
wpsconfig.bat transfer-wcm-filesystem-to-cloudscape
```

The file system data is transferred to the Cloudscape database and a Build Successful message appears in the command prompt window.

Note: For the UNIX platform, enter the following:

```
wpsconfig.bat/.sh transfer-wcm-filesystem-to-cloudscape
```

12. Close the command prompt window.
13. Open the wcm_migration.log file from the *WPS_HOME*\log folder in a text editor.

Note: The log file should contain the text Utility Completed to confirm that the data transfer was successful.

14. Close the wcm_migration.log file.

14.8 Enabling user migration

This section explains how to install the migration .jar to enable user migration.

Important: In IBM Workplace Web Content Management 5.1, you might need to add the ilwwcm-migration.jar file to your installation before performing this procedure. Read the product release notes for the latest information.

To install the migration .jar file, do the following:

1. Start server1 of the WebSphere Application Server, if it is not already started.
2. Open the WebSphere Application Server Administrative Console.

Note: Use port 9090 when accessing the WebSphere Application Server Administrative Console. Otherwise, you will not have access to the Shared Libraries.

3. Click the Environment navigator.
4. Click **Shared Libraries**.
5. Click **WCMLib**.
6. In the Classpath field, enter the full path to the ilwwcm-migration.jar. In a standard installation the code appears as follows:

```
${WPS_HOME}/wcm/migration/ilwwcm-migration.jar
```
7. Click **Apply**.
8. Click **Save**.
9. Click **Save** to save the master configuration.
10. Stop the WebSphere_Portal application server.
11. Start the WebSphere_Portal application server.

14.9 Migrating users directly to WebSphere Member Manager

To migrate users directly to WebSphere Member Manager, do the following:

1. Copy existing users from the previous version of Lotus Workplace Web Content Management to IBM Workplace Web Content Management 2.5 or 5.1 by copying the users folder from the *ILWWCM_HOME*\connect folder of the previous version of Lotus Workplace Web Content Management to the *ILWWCM_HOME*\connect folder of IBM Workplace Web Content Management 2.5 or 5.1.

Note: The connect folder contains the users folder, which is needed for data migration.

2. Migrate existing users by first starting the WebSphere Portal server, if it is not already started.
3. In the browser Address field, enter the following URL to access the IBM WebSphere Portal migration user interface:

```
http://[HOST]:[PORT]/wps/wcm/jsp/migration/migration.jsp
```
4. In the Input WebSphere Portal Administrator Login Name field, enter the Portal administrator name.
5. In the Input Connect User Path field, enter the path to the copied *ILWWCM_HOME*\connect\users folder.

Note: You copied these users from the previous version of IBM Workplace Web Content Management.

6. Select **Import Administrators Group** if you need to migrate the Administrators group.

Note: The Administrators group is mapped to the Administrators group that is defined in the connect.cfg file. The default value for this group is wcmadmins.

7. Make sure that **Update WMM?** is selected.

Note: The Update WMM? option is selected by default.

8. Click **Migrate**.

The users and groups are created in WebSphere Member Manager and the transferred IBM Workplace Web Content Management data points to the new users and groups in WebSphere Member Manager.

9. Stop the WebSphere Portal server.
10. Delete the system folder located in the *ILWWCM_HOME\ilwwcm* folder.

Note: The system folder contains the indexes.

11. Start the WebSphere Portal server.

The indexes are rebuilt.

12. To verify that the migration was successful, open the IBM Workplace interface and log on.
13. Select the Web Content Management tab.
14. Expand the Content Library view.
15. Click **Content by Title**.
16. Click a content document.

The content document opens in the authoring portlet.

17. Expand the Security section of the content document.

Make sure that the user and group names appear in the Security section. If the names appear, the migration was successful.

14.10 Migrating users to WebSphere Member Manager using a user migration mapping file

This section explains how to:

- ▶ Generate a user migration mapping file.
- ▶ Review and edit, if required, the user migration mapping file.
- ▶ Process the user migration mapping file.
- ▶ Verify that user migration was successful.

14.10.1 Generating a user migration mapping file

Tip: Follow these steps to migrate Lotus Workplace Web Content Management users and groups using a user migration mapping file if the match is not exact between existing Lotus Workplace Web Content Management users and groups and existing WebSphere Member Manager users and groups.

To generate a user migration mapping file, do the following:

1. In the browser type the following URL in the Address field to access the IBM WebSphere Portal migration user interface:

`http://[HOST]:[PORT]/wps/wcm/jsp/migration/migration.jsp`

2. In the Input WebSphere Portal Administrator Login Name field, enter the Portal administrator name.
3. In the Input Connect User Path field, enter the path to the copied `ILWWCM_HOME\connect\users` folder.

Note: These are the users that you copied from the previous version of IBM Workplace Web Content Management.

4. Select **Import Administrators Group** to migrate the Administrators group. Members of the Administrators group are added to the `wcmadmins` group.
5. Click **Generate User Migration Mapping File**.

This option allows you modify the user and group names, as shown in Figure 14-6 on page 594.

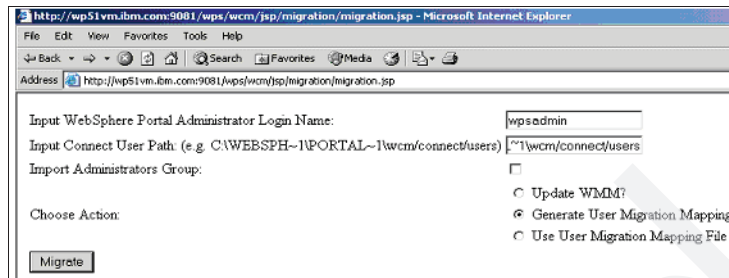


Figure 14-6 Option to generate a User Migration Mapping File

6. Click **Migrate**.

The user migration mapping file is generated. The file is called migrationUsers.properties and is located in the *WPS_HOME\wcm\migration* folder.

Note: Do not close the browser window.

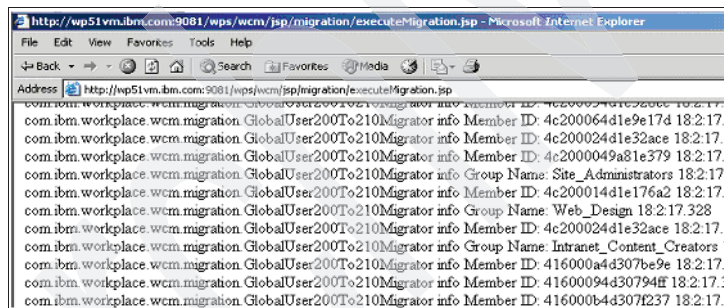


Figure 14-7 Status of the migration

14.10.2 Reviewing and editing the user migration mapping file

To review and edit the user migration mapping file, do the following:

1. From the *WPS_HOME\wcm\migration* folder, open the migrationUsers.properties file in a text editor.
2. If required, modify the user names in the migrationUsers.properties file as follows:
 - The original Lotus Workplace Web Content Management users and groups are listed on the left, and the WebSphere Member Manager users and groups are listed on the right.

- Change only the proposed WebSphere Member Manager names.
- Change the property values of the WebSphere Member Manager name to use the distinguished name. For example:

```
uid=username,o=default organization
cn=group name,o=default organization
```

- Do not use an ampersand (&) in user or group names because it causes errors.

The following line of code is an example of how to change the group name *All Staff* to *All_WP_Staff*, using the distinguished name:

```
All\ Staff=cn=All_WP_Staff,ou=groups,dc=ibm,dc=com
```

Note: Make sure you modify this file before processing it.

3. Save and close the file.

14.10.3 Processing the user migration mapping file

To process the user migration mapping file, do the following:

1. In the open browser window, click **Back**.
2. Click **Use User Migration Mapping File**.
3. Click **Migrate**.

The browser window shows that the data has been updated with the new user and group names.

4. Stop the WebSphere Portal application server.
5. Delete the system folder that is located in the *WPS_HOME\wcm\ilwwcm* folder.

Note: The system folder contains the indexes.

6. Start the WebSphere Portal application server.
The indexes are rebuilt.

14.10.4 Verifying that the migration was successful

To verify that the migration was successful, do the following:

1. Open the Workplace interface and log on.
2. Click **Web Content Management**.

3. Expand the Content Library view.
4. Click **Content by Title**.
5. Click a content document.
The content document opens in the authoring portlet.
6. Expand the Security section of the content document.

Note: Make sure that the user and group names appear in the Security section. If the names appear, the migration was successful.

14.10.5 Adding new users or groups to WebSphere Member Manager

Important: The user migration mapping file updates only existing Lotus Workplace Web Content Management users and groups. If existing Lotus Workplace Web Content Management users and groups do not exist as WebSphere Member Manager users and groups, you must run the **XMLaccess** command.

Users and groups in earlier versions of Lotus Workplace Web Content Management do not provide a `lastname` attribute. You must type a value for the `lastname` attribute in the `migrationUsers.xml` file. Otherwise, you will not be able to run the **XMLaccess** command. The value for the `lastname` attribute can be any value.

To add new users or groups to WebSphere Member Manager, do the following:

1. Open a command prompt.
2. Go to `/PortalServer/bin`.
3. Run the following command:

```
xmlaccess.bat -in /path/PortalServer/wcm/migration/migrationUsers.xml -user portalAdmin -pwd password -url http://host:port/wps/config
```

Note: Ensure that you replace `path`, `portalAdmin`, `password`, `host`, and `port` with the actual setting of your IBM Workplace Web Content Management installation.

This command is only usable in a Windows platform. For the UNIX platform, use the following command:

```
xmlaccess.sh -in /path/PortalServer/wcm/migration/migrationUsers.xml -user portalAdmin -pwd password -url http://host:port/wps/config
```

14.11 Disabling user migration

To uninstall the migration .jar file to disable user migration, do the following:

1. Start server1 of the WebSphere Application Server, if it is not already started.
2. Open the WebSphere Application Server Administrative Console.

Note: Use port 9090 when accessing the WebSphere Application Server Administrative Console. Otherwise, you will not have access to the Shared Libraries.

3. Click the **Environment** navigator.
4. Click **Shared Libraries**.
5. Click **WCMLib**.
6. In the Classpath field, delete the full path to the ilwcm-migration.jar. In a standard installation, the code appears as follows:
`${WPS_HOME}/wcm/migration/ilwcm-migration.jar`
7. Click **Apply**.
8. Click **Save**.
9. Click **Save** to save the master configuration.
10. Stop the WebSphere Portal application server.
11. Start the WebSphere Portal application server.

14.11.1 Recreating the resources

To recreate the resource folder and the resources, enter the following URL in the browser Address field:

`http://[HOST]:[PORT]/wps/wcm/connect?MOD=AJPResourceChecker&fix=true`

The Resource Checker runs. When completed, the log appears in the browser window.

Archived

Migrating content into IBM Workplace Web Content Management

This chapter describes how to migrate a Web site from another Web content management tool into IBM Workplace Web Content Management 5.1. It includes the following sections:

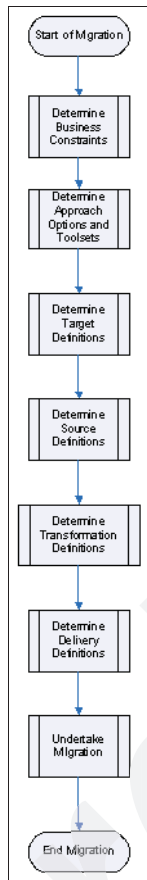
- ▶ 15.1, “A roadmap to migrating Web content” on page 600
- ▶ 15.2, “Overview of the Vamosa Content Migrator” on page 609

Note: For the purposes of this chapter, we highlight Vamosa’s Content Migrator tool set. For additional information, you can refer to the following:

http://www.vamosa.com/index/what_we_do/technology_and_products/vamosa_content_migrator.htm

Note that Vamosa’s technology represents one of the strategic options for migrating content into IBM Workplace Web Content Management. However, other technical approaches are available. Regardless of which technology that you use to migrate content, the content in 15.1.1, “Determining the business case for the migration” on page 601 through 15.1.6, “Determining quality assurance measures” on page 607 includes key fundamental business issues that you need to address in your migration.

15.1 A roadmap to migrating Web content



Undertaking a migration of Web pages or documents into a content management system (CMS) is often approached as a *last minute* exercise. Normally, assurances from CMS vendors imply that there are no issues that are associated with this part of implementing a successful CMS strategy. However, as with many other aspects of IT implementations, it is the correctness and validity of the data that is entered into a system that determines its success or failure.

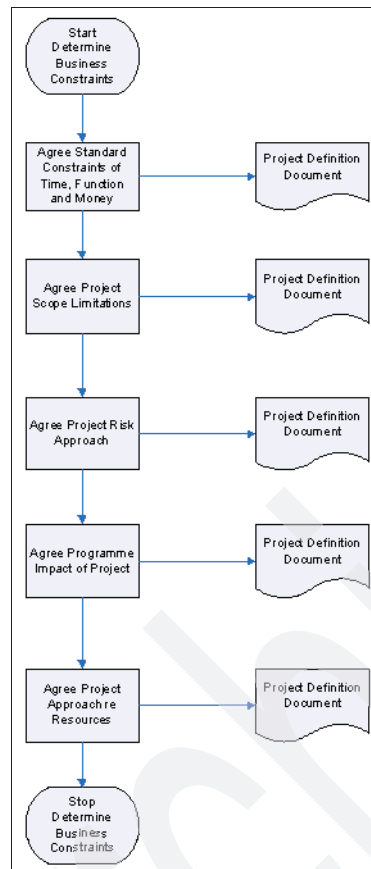
Thus, IBM has teamed with Vamosa to recommend the use of Vamosa's unique toolset to migrate all of an organization's existing content into IBM Workplace Web Content Management. As CMS has matured as a sub-set of IT systems, the provision of a successful and proven migration methodology has become more important in determining the suitability of one CMS product over another.

Conceptually, it is not difficult to understand how to migrate existing data, in that you can visualize the end result easily. However, when trying to understand the process steps that are required to move data from the existing environment to the target CMS system, difficulties arise. In these cases, a roadmap that shows the route for the migration and that contains some simple rules can help you achieve a successful migration. This section demonstrates a roadmap and provides the rules that are required.

The roadmap requires a number of distinct steps. If you follow these basic steps, the migration will be successful.

Each step in this roadmap has an associated flowchart that outlines the processes that is required within that step. The remainder of this section describes each step.

15.1.1 Determining the business case for the migration



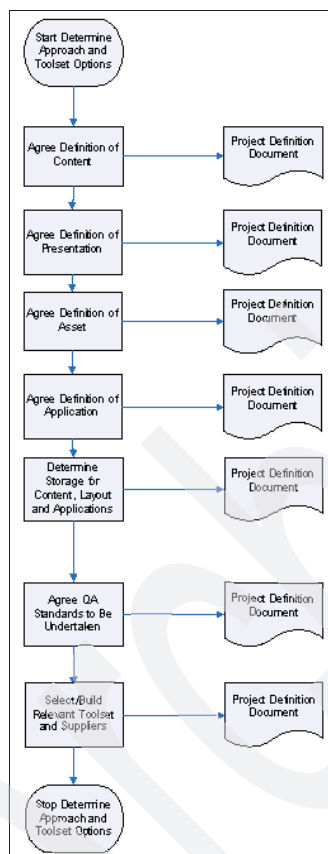
As with all projects, you should begin your migration project by outlining the business case for the migration. This business case is constrained by time, cost, and function. These constraints should be embedded within the philosophy of the migration, and they should be reflected within the project operation.

The measure of success for the migration requires that you set the business case clearly at this stage of the process. The migration requirement does not impinge into the area of business change. Instead, the business change project should actively set the boundaries for the migration project.

Vamosa Implementation Architecture (VIA) is the best practice content migration method. It is designed to leverage the maximum automation benefit from Vamosa Content Migrator. VIA defines the means to extract, document, and implement the valid business constraints for the migration project.

15.1.2 Determining the migration approach and toolset

Based upon the constraints placed upon the project it is necessary to identify and ratify the migration approach and required toolset necessary to satisfy the objectives. If the volume of data that you need to migrate is significant, then an automated approach is the best solution. However, if the volume is smaller — less than 5000 pages — then a manual approach is sufficient.



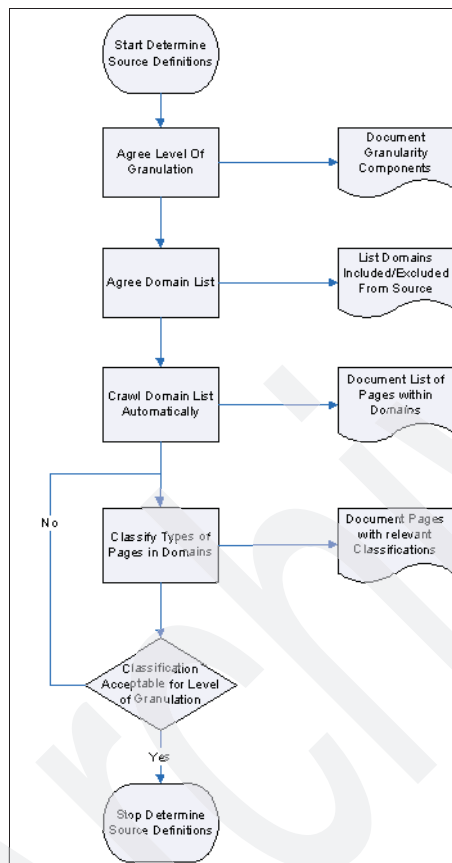
Migrations throw up a number of options that can be exploited. This is even more prevalent when you use an automated approach. These options fall into four distinct types, with each type being a super-set of the type before it, as follows:

- *As-is migration.* In this case, the source pages and assets are migrated with no change to their nature and no additional information is added to their definition.
- *Enhanced migration.* In this case, the source pages have additional metadata that is created with regards to their properties and also to the requirements of IBM Workplace Web Content Management.
- *Standardized migration.* In this case, the input pages are broken down into smaller levels of granularity, and these smaller levels are migrated and re-assembled in IBM Workplace Web Content Management. This case is often referred to as *applying standard templates*.
- *Restructured migration.* In this case, as the pages are migrated, then the navigation elements are updated to reflect a new information architecture.

At this stage, it is important to ensure that the original constraints are *built into* these decisions. All migrations as they progress tend to attract a growing requirements list. This scope creep occurs if the original constraints — and the business requirements upon which they are founded — are ignored. It is important to tie back the choices for which type of migration is undertaken based upon those original constraints and the success criteria that is designed for them.

15.1.3 Determining source classifications for the migration

Often within migrations, the quality of the input data is ignored until later on in the project. A successful migration strategy determines the source classifications to be assured that the current data can fit into the target formats.



The following are the aspects that you need to define for the source information:

- ▶ Domains
- ▶ Information architecture
- ▶ List of pages and assets
- ▶ Number of page classifications
- ▶ Available granularity by classification

By defining these elements, you can determine the potential complexity of the migration. Separation of pages and assets is crucial at this stage. All images and documents that are linked from pages are categorized as assets and need to be recorded as such.

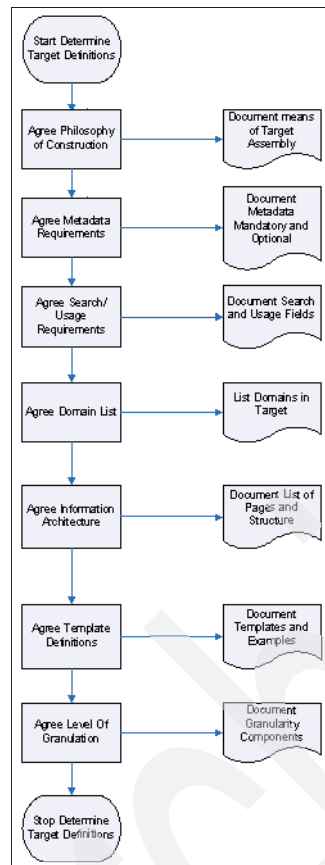
Classification of the input pages and assets allows the migration strategy to take account of all input parameters, which is vital to achieving the necessary granularity.

Based upon our experience when using Vamosa Content Migrator, in any single Web site migration, there are typically a limited number of source classifications because the actual pages conform to a

subtle presentation format that is acceptable to the users who interact with them. If the formats are too numerous or are not related, then the current system is not usable and an automated content migration is unfeasible.

We have found that, depending on the size of the source volume, there is a maximum of 20 to 30 classification types. You can investigate these classification types manually. For example, you can investigate a sample page from each type to best determine the granularity that can be achieved during the retrieval process.

15.1.4 Determining the target definition for the migration



As part of any migration, the definition of the target system determines the transformation requirements. At this stage, organizations can fall easily into designing a complicated solution that cannot be implemented. Consequently, there is a close relationship between the emergent information regarding the source classification types and the potential for populating the designed target system successfully.

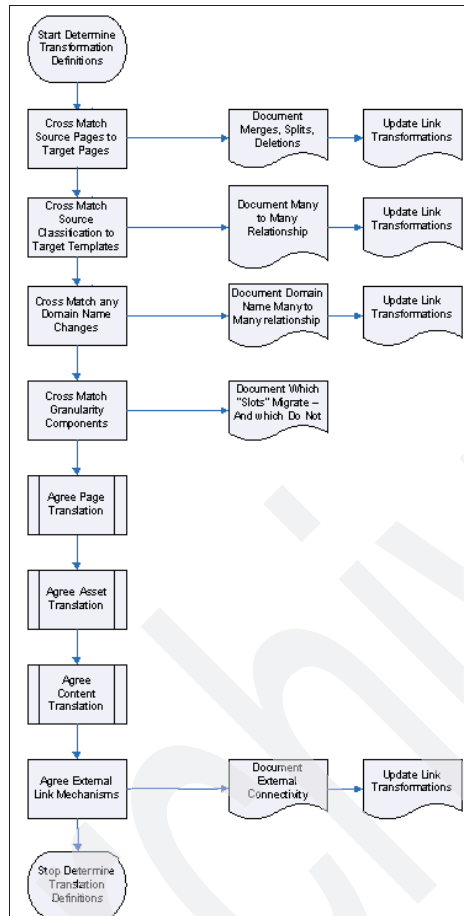
You need to match the definitions in the source environment with those in the target environment. However, you can assume safely that the requirements of the target system are greater than those enabled currently in the source system.

These requirements include the following:

- ▶ Domain hierarchy
- ▶ Information architecture
- ▶ Template definitions
- ▶ Granularity specifications
- ▶ Metadata requirements

You should remember that as part of this process you need to incorporate the requirements of the IBM Workplace Web Content Management implementation, including aspects such as naming conventions, ownership, security, expiration dates, classifications, or taxonomies and so forth.

15.1.5 Determining the transformation requirements for the migration



Now that you have defined both the source and target systems, you can determine the transformation rules for the migration. These rules need to address all aspects of the target system as the primary recipient of the existing data. Subsequently, when determining the rules, you need to *work backwards* from the target system to the source system. The overall approach is to make the source data *fit* the target data and not to change the target to suit the source.

The definition and the design of the rules, therefore, are driven from the target system requirements. Where the source system cannot supply relevant data to populate the target system, then you need to apply effective auto-generation logic within the rule definitions. In addition, where the target data cannot be populated, either due to inappropriate source data or auto-generation that cannot be completed, then you need to define an exception handling format.

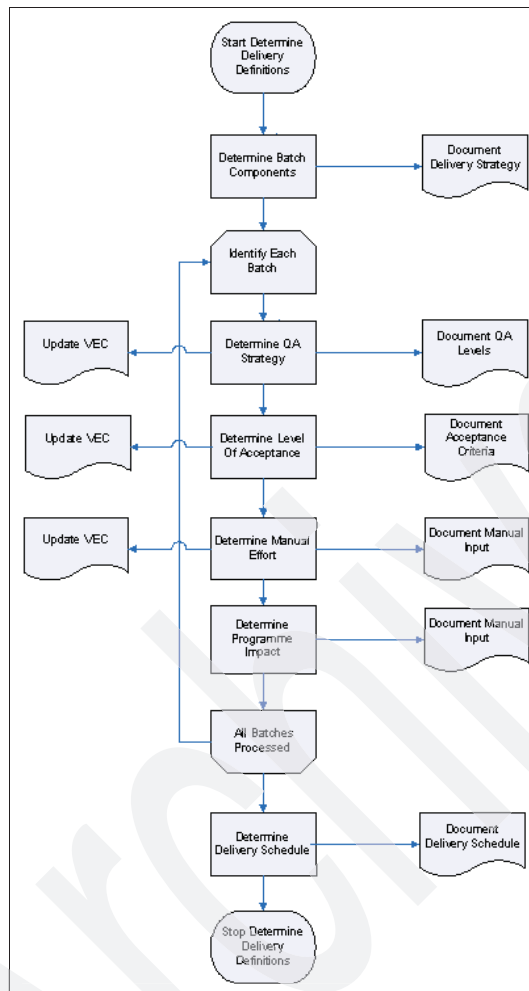
For example, if the target system requires a Title field to be less than 50 characters long but the source system has a title that is 65 characters long, you should document this exception for action. The exception could state that the source is truncated to 50 characters or that the data element is flagged so that it is not migrated until the correct rule handling is defined. The transformation rules can be broken down into the following key areas:

- ▶ Domain name
- ▶ Information architecture
- ▶ Classification match to template
- ▶ Granularity cross-matching
- ▶ Page changes
- ▶ Asset requirements
- ▶ Content enhancement

You should design and test these rules prior to actual coding. Using desk-checking of the actual pages that are identified as part of the classification is invaluable in ensuring that the rules work when they are actioned.

You should also associated a verification, exception, and correction document (a VEC document) to each rule. This document determines the test of verification — what exception is identified — and the corrective action to take. In the previous example, the verification checks the title field to see if it conforms to requirements. The exception is that the length is greater than 50 characters. The correction is to truncate the title to the first 50 characters and to dispose of the rest. This action is then incorporated into the conversion rule action.

15.1.6 Determining quality assurance measures



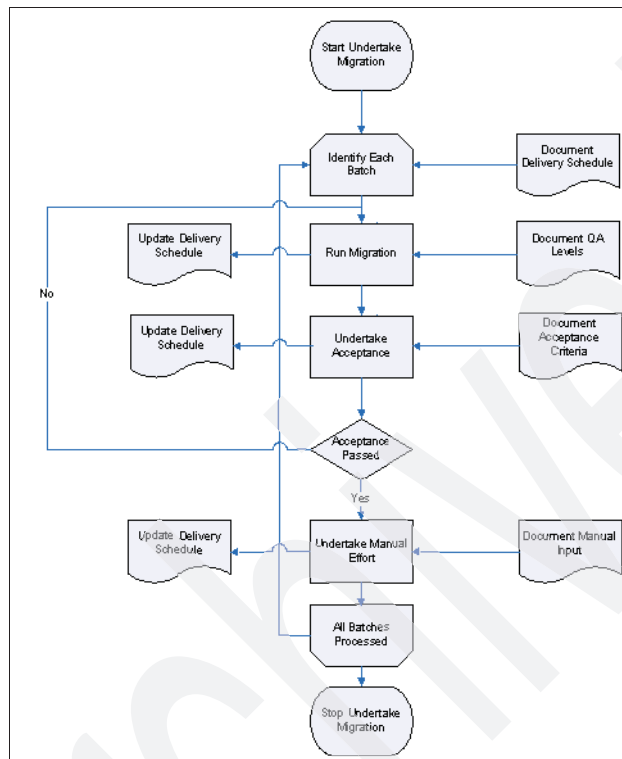
All migrations have a requirement that measures them against success criteria. Where volumes are small, this measurement is not an onerous task, but as volumes increase, then the need for a number of smaller batches arises. Undertaking a migration of one million pages can indeed be achieved in one large scale batch operation. However, the quality assurance function can prove to be difficult when it comes to checking all of these pages.

When you determined the approach and toolset for the migration, you also identified the quality assurance requirements. Now, it is necessary to design an approach that allows you to undertake these quality assurance aspects. Typically, the design involves breaking down the migration into smaller, more manageable sub-projects that are migrated in either a parallel or a sequential manner. If you take this approach, you need to ensure that the relationship between link items is maintained across the sub-projects.

Vamosa Content Migrator is unique in the fact that the toolset supports this dynamic link handling and ensures that the information architecture is maintained.

15.1.7 Undertaking the migration

At this point, the actual undertaking of the migration should be a relatively straightforward task.



The definitions have set out a framework of operation that reflects both source information and the required target system. By defining both the requirements that exist currently and the requirements that are to be, the transformation rules and the exception handling, and the quality assurance that is necessary for the migration, all the technical elements are in place to proceed with the migration.

If during the initial migrations additional facts come to light (that is, any unforeseen data sources), then you should reflect this information

further back in the source definition and the transformation requirements. Matching the final migration to the success criteria is extremely important at this stage. It should not be possible to simply reject migrations without identifying those migrations as failing the success criteria based upon the transformation rules that you defined previously.

15.2 Overview of the Vamosa Content Migrator

This section provides an overview of the Vamosa Content Migrator software product and illustrates step-by-step the process of migrating data using this product.

Vamosa Content Migrator is a unique software product that performs all of the tasks that are involved in the complex process of extracting content from any source, transforming that content to make it compliant at both the technical level and the business level, and then loading that content into IBM Workplace Web Content Management.

15.2.1 How Vamosa Content Migrator works

Vamosa Content Migrator uses a multi-step process to ensure that all content is processed adequately to suit the requirements of IBM Workplace Web Content Management. When migrating a Web site into IBM Workplace Web Content Management, Vamosa Content Migrator uses four stages to achieve the desired results, as described in the following sections.

Step 1: Identifying the content to be migrated

Vamosa Content Migrator can selectively identify the content which is the candidate for the migration project. The VIA methodology describes the best way to break down the project into sub-components, as illustrated in Figure 15-1 on page 610. Each sub-project can then be defined by the URLs that define the content. Vamosa Content Migrator then crawls across this content to allow the Vamosa Business Content Process Management repository to be defined for this specific sub-project.

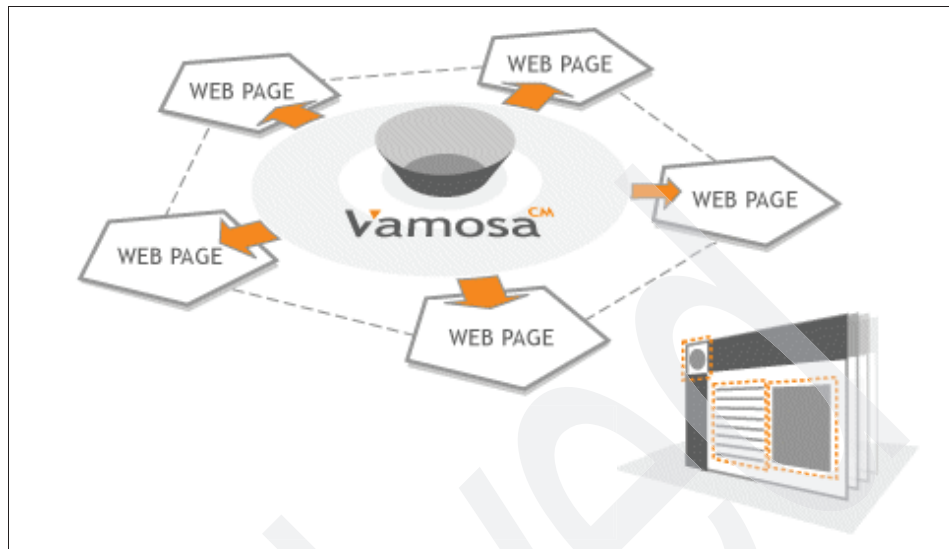


Figure 15-1 Vamosa Content Migrator scans the entire site to break down into parts

This initial phase also uses the Vamosa Content Migrator Source Classifier to categorize the existing content so that the automated migration can take place. The Vamosa Content Migrator Source Classifier generates *source templates* that break down an HTML page into its logical parts, such as navigation, content, header, and footer text. The range of templates generated are then matched to each page during the extract process where the *fragments of HTML code* that make up a Web page are broken down into their constituent parts and held separately within the Vamosa repository. When in the repository, these fragments of HTML code are further processed during the enhancement stage by applying specific business rules to the various fragment types.

As each page passes through the Vamosa Content Migrator Source Classifier, rules help the Vamosa Content Migrator Source Classifier to determine whether or not a particular table cell contains a high degree of links or a high degree of text. The HTML code is then annotated with the classification results, which in turn are read by the extract process to break down the Web page into its constituent parts. This classification is also used to *map* source page types to target application templates.

Step 2: Extracting the content to be migrated

After you have identified the migration candidate content, the next task is to actually take that content and place it in the Vamosa Business Content Process Management repository for further processing, as illustrated in Figure 15-2. The level of granularity that the content is broken down to is a *project parameter*. Vamosa Content Migrator can break content into chunks ranging from pages down to single characters. This flexibility allows you to map the old structure that is identified through the Vamosa Content Migrator Source Classifier into the templates or slots as required by the target system.

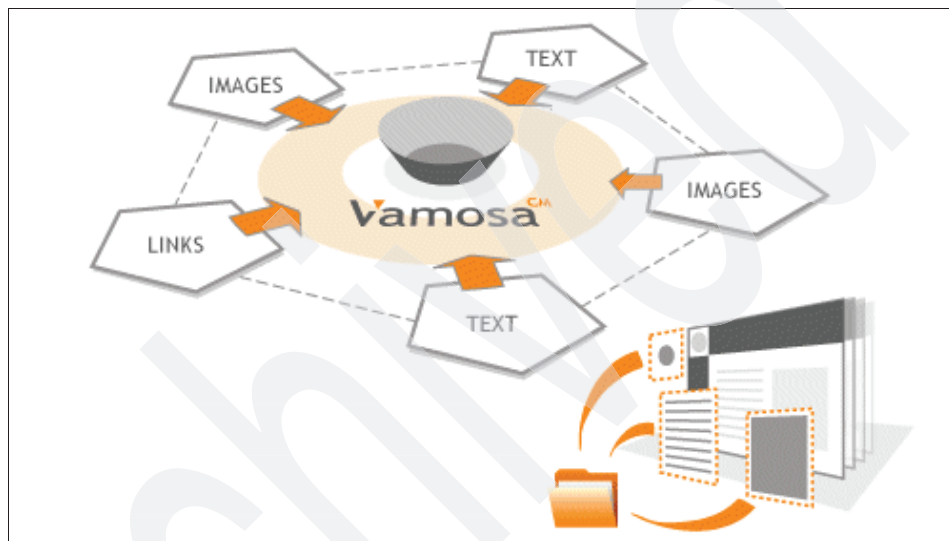


Figure 15-2 Vamosa Content Migrator extracting and storing content

The following sets of data discuss how Vamosa Content Migrator can break content into more manageable portions of data:

- ▶ Set 1 captures the relevant structure and context of the existing content. Vamosa captures and stores existing metadata within the repository. Vamosa also creates new metadata for the content that is extracted. This allows Vamosa Content Migrator to process all data based upon the actual value of that data and its source location and relationship to other content. This type of data is referred to as *placeholder data*.
- ▶ Set 2 captures all files that are used by the source system. These files include PDFs, presentations, spreadsheets, and word processing files as well as all images, sound, and movie elements. This type of data is referred to as *asset data*.
- ▶ Set 3 captures all HTML data from the source system. This type of data is referred to as *content data*.

This data is placed into the Vamosa Business Content Process Management repository for further processing.

Note: Vamosa Content Migrator does *not* alter or destroy the source system. The source system remains completely intact until the target system is ready to move to production.

Step 3: Re-engineering the extracted content to make it fit for its purpose

This step allows for full programmatic access to the stored components of placeholders, assets, and content, as illustrated in Figure 15-3.

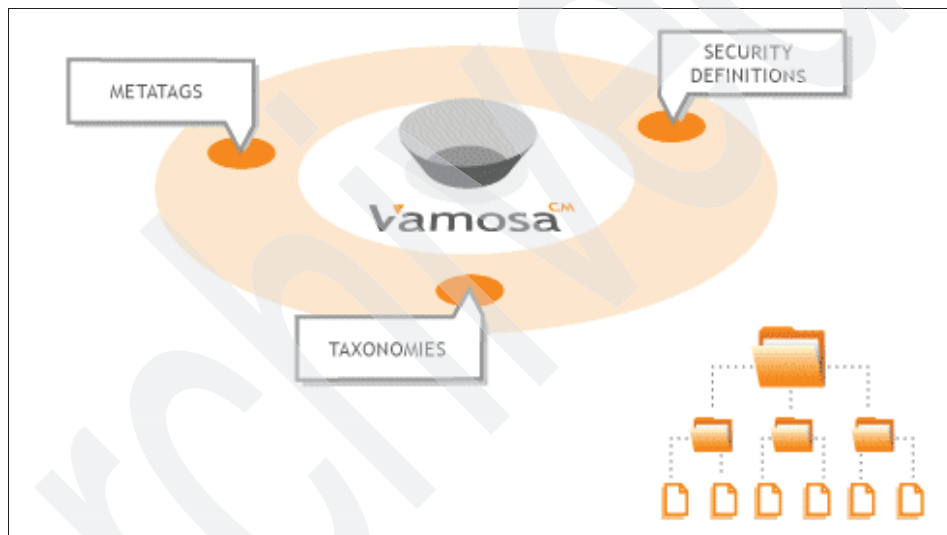


Figure 15-3 Vamosa Content Migrator categorizes Web content into the new structure

To allow customization at this stage, a scripting API is available. Default scripts are provided with Vamosa with the intent of providing a basic enhancement of data to allow loading the content into IBM Workplace Web Content Management. These default scripts are available automatically on the creation of a task library within the Vamosa Content Migrator tool and, generally, can be used as is.

For more complex requirements, however, it is necessary to write (or modify) scripts on a per project basis. You can write scripts using the rule editor that is built into the Vamosa tool or using an external editor and then pasting in the scripts when they are complete.

There are two scripting languages used:

- ▶ Jython, a JAVA-enabled Python engine that is available at:
<http://www.jython.org>
- ▶ XSL, a stylesheet translation language which defines templates to modify the XML content.

Typically, scripts are written to achieve some (or all) of the following:

- ▶ HTML pre-processing
- ▶ Navigate from initial URL and change fixed links to relative links
- ▶ Links process
- ▶ Page names
- ▶ Page locations
- ▶ Page metadata
- ▶ Asset names
- ▶ Asset locations
- ▶ Asset metadata
- ▶ Content extract
- ▶ Content merging
- ▶ Content deletion
- ▶ Tag extract
- ▶ HTML changes
- ▶ Font standardization
- ▶ Menu standardization
- ▶ Spelling correction
- ▶ Text replacement
- ▶ De-duplication
- ▶ XML post-processing

This list is not exhaustive. The ability for Vamosa to enhance content is limited only by the time that is available within the project to complete these tasks.

Step 4: Loading content into IBM Workplace Web Content Management

When the content is in the state required by the project, it is *application ready*. The final part of the Vamosa Content Migrator automated process is to load this application ready data into IBM Workplace Web Content Management. Vamosa Content Migrator can achieve this load in a number of ways, but each method achieves the same end results as illustrated in Figure 15-4.

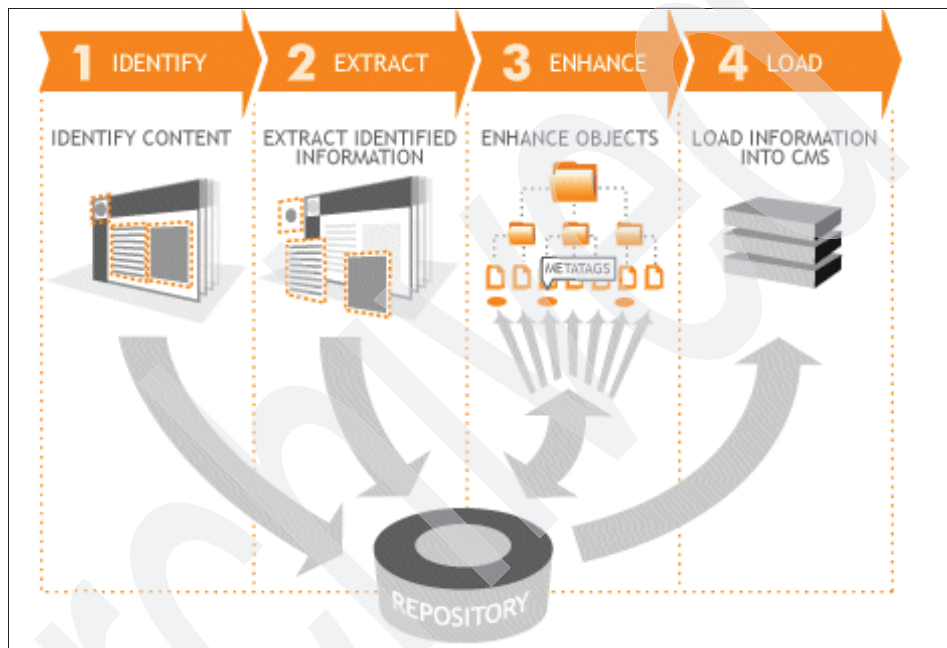


Figure 15-4 Overview of the steps that are involved in loading data

15.2.2 Reasons for using a specialized method

These four stages are the overall process by which data is migrated. As the data is migrated, it needs to be updated and enhanced in order to allow the correct load into IBM Workplace Web Content Management. The overall aspect of VIA is directly focused to this aspect of the migration. Indeed the requirement to specify data enhancement is referred to deliberately as *business* rules. Therefore, VIA has a supporting management infrastructure built in that is designed to control and manage the overall method process steps.

Although this method outlines the steps that are associated with data migration, it does not represent a form of project definition that would work successfully.

Essentially, Figure 15-4 on page 614 demonstrates how data is migrated, but it does not define the process to undertake that migration.

For example, if the target content management system is an IBM Workplace Web Content Management system, it is not permissible to load data that contains basic HTML components, because the HTML components are present in the source system. These HTML components must be *enhanced* to become IBM Workplace Web Content Management components. By following the method, you can answer and build these questions into the migration steps and incorporated them easily into the Vamosa Business Content Process Management engine.

To ensure that specialist knowledge is not required and that service personnel can undertake successful migrations, this method has specific steps .

Figure 15-5 illustrates the VIA method components.

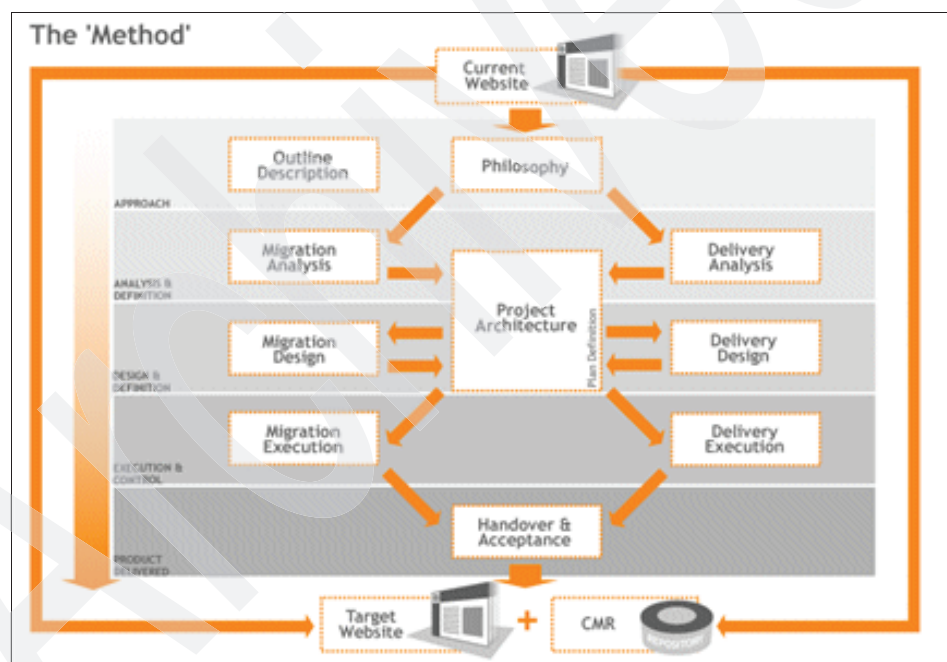


Figure 15-5 VIA method components

15.2.3 The law of diminishing returns

Although the theory of achieving 100% data migration is the goal of any migration, it is usual in most circumstances that data is migrated within the context of iterative migration sets that work towards a 100% success rate.

Typically the *business rules* that are designed and defined exploit a large sub-set of the migrated data. However, after the first set of runs, there is a minor sub-set of data that has not been addressed correctly. Additional business rules are defined, and the four migration steps are run again. This process is repeated until a high enough success rate has been achieved.

This process, however, succumbs to the *law of diminishing returns*. Each iteration achieves a smaller return on the additional successful data that is migrated. As part of the migration process, it is important to define at the initiation of the project what the success levels for the project should be, bearing in mind the law of diminishing return.

15.2.4 Quality assurance and project management

VIA allows for the effective quality control and project management of a content migration project. VIA identifies the processes that are required to generate the necessary checkpoints when running the projects to ensure that all data is processed and accounted for. This identification obviates the requirement to perform manual counts of content processed. Vamosa Content Migrator is *VIA ready* and allows for the easy access of the relevant quality assurance and project or process management data through its intuitive user interface.

The VIA documentation also includes full exemplars of project plans to allow the fast path commencement of a VIA controlled migration project using the Vamosa Content Migrator toolset.

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described in this appendix.

Locating the Web material

The Web material that is associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG246792>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select **Additional materials** and open the directory that corresponds with the redbook form number, SG246792.

Using the Web material

The additional Web material that accompanies this redbook includes the files shown in the following tables.

Table 15-1 Installation product documentation

File name	Subfile	Description
WebContentManagement-2-5-0-InstallationGuide.pdf		Product documentation covering installation for IBM Workplace Web Content Management 2.5 and 5.1

Table 15-2 Resources for building the River Bend site

File name	Subfile	Description
Resources_for_RiverBendWebsite.zip		Top level wrapper for all files and code samples

Table 15-3 Customization examples

File name	Subfile	Description
Material_for_customization.zip		Top level wrapper for all files and code samples
	SimpleDisplayPortlet.zip	RAD Project Interchange file of the sample built in the Chapter 12, "Working with the application programming interface" on page 469. (See specifically, 12.5.1, "Developing a simple Portlet" on page 501.)
	ilwwcm-custom-template-portlet.war	Customizable Template Portlet
	CustomJSP.ZIP	Additional files that are used in the Customizable Template Portlet example

File name	Subfile	Description
	utils.zip	Additional files that are used in the Customizable Template Portlet example

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zipped file into this folder.

Archived

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 622. Note that some of the documents referenced here might be available in softcopy only.

- ▶ *IBM WebSphere Portal for Multiplatforms V5.1 Handbook*, SG24-6689

Online resources

These Web sites and URLs are also relevant as further information sources:

- ▶ Download full Product documentation (Information Center):
<http://www.lotus.com/ldd/notesua.nsf>
- ▶ Workplace Web Content Management 2.5 Information Center.
<http://www-10.lotus.com/ldd/notesua.nsf/find/wcm25>
- ▶ Workplace Web Content Management 5.1 Information Center.
<http://www-10.lotus.com/ldd/notesua.nsf/find/wcm51>
- ▶ WebSphere Portal 5.1 (Information Center):
<http://publib.boulder.ibm.com/pvc/wp/510/ent/en/InfoCenter/index.html>
- ▶ IBM Workplace Web Content Management ibm.com Home page:
<http://www.ibm.com/software/workplace/webcontentmanagement>
- ▶ WWCM Support Gateway:
<http://www.ibm.com/software/lotus/support/workplace-webcontent-mgmt/support.html>
- ▶ Workplace Web Content Management Forum on developerWorks:
http://www.ibm.com/developerworks/forums/dw_wpforums.jsp

- ▶ IBM Workplace Solutions Catalog (for the Customizable Template Portlet and Editor Integration API)

<http://catalog.lotus.com>

- ▶ There are many articles on Workplace Web Content Management in developerWorks:

<http://www.ibm.com/developerworks/>

- ▶ Education Offerings from IBM:

<http://www.lotus.com/services/education.nsf/wdocs/educationhomepage>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications, and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Numerics

2-phase linkings 421

A

access rights within security model 158
Access Web Content Management 257
accessing unrestricted resources 585
action, scheduled move 272
actionPerformed method 502, 521, 562
adding
 button 559
 Content Viewer Portlets 337
 groups to WebSphere Member Manager 578, 596
 left navigator 305
 users to WebSphere Member Manager 578, 596
administrator password 83
administrator user ID 83
Administrators group, migrating 578
advanced cache 349, 359, 374
advantages, design 54, 56, 58, 61
alerts
 e-mail 112
ALL USERS group 579
alternate rendering concepts 46
anonymous user 580
any organization 206
AO Foundation 206
 Advanced search 245
 example 220
 intuitive step 220
 navigation 235
 Surgery Module 214
API 12, 37, 39, 46, 167, 430, 469, 531
application program interface. *See* API.
applying standard templates when migrating 602
approach for migration 602
approval Workflow 271
AprixCmpnt context 176, 278, 370, 538
AprixLibCmpnt name 175, 287, 369, 455, 529
aprixsearch.properties file 328
architectural items and components 154

as-is migration 602
assigning
 e-mail action to Reject stage 274
 expire action to the Expired stage 272
 scheduled move and publish actions to the Published stage 273
assigning content item as default page 291
attaching to a remote server 497
attribute
 lastname 577
attribute formatting 553
authoring 48, 65
 IBM Workplace Web Content Management 2.5 137
Authoring Portlet 15, 36, 41, 49, 59, 74–75, 127, 136, 147–148, 152, 258, 323, 482, 574, 581
 configuration changes 49, 63
 content item 323
 current content object 323
 development section 155
 displayed document 147
 future release 148
Authoring Template 15, 154, 161, 177, 197, 227, 233, 268, 274, 276, 283, 341, 343–345, 483, 536
 content item 345
 image components fields 190
 pairing with presentation template 283
 related field 185
 text component 490
Authoring to Staging 67
autoLoad 116, 327, 361
automated rendering 119

B

base example of a good information architecture 201
base server installation 73
basic API development sample 501
basic caching 349, 359
basic mail routing configuration 112
beginning the implementation 414
benefits of using IBM Workplace Web Content Management 12

- breadcrumb 170
- breadcrumb navigator 310–311
- broken links 170
- building
 - category 266
 - Site Framework 259
 - taxonomy 266
 - Web site 256
 - Workflow 268
- bulk resource import 470
- Bulletin Board 264, 536, 540
 - configuration 543
 - menu configuration 542
 - Portlet 543
- business case for migration 601
- business goals 217
- business stakeholders 249
- business-critical information 7

C

- cache 350
- cache events timeline 378
- cache recommendations for components types 373
- cache size 358
- cache strategy, documenting 376
- cache type 358, 377
- cache, default 357
- cache 115
- cache module 116, 364
- cache options 117
- caching 114, 349, 359, 368
 - choosing between strategies 351
 - distributed map 385
 - implementing 353
 - options 380
 - paradigms 350
 - performance impacts 371
 - servlet 383
 - strategy 349, 351–352, 371, 379
- carriage return and line feed (CRLF) 570
- category 164–165, 228, 266–267, 483
- category management 164
- category menu 319–320
- changing the connect.cfg file 446
- chapters in this book, overview 26
- choosing a caching strategy 351
- Cloudscape 42, 51, 73–74, 129, 567, 587
- Column Table 276, 528

- breadcrumb navigator 311
- HTML Quick Search form 334
- image library references 288
- Left Navigator 308
- Tools Navigator 326
- commands
 - XMLAccess 417, 575
- common errors and resolutions 582
- communications 250
- component architecture 34
- component catalog 376
- component library 152, 155, 175, 177, 285, 293, 478
 - image components 191
- component sub-interfaces 482
- component types 149
- componentization 149, 151
- config mode 415
- configuration
 - post installation 112
- configure
 - cache options 117
 - custom UI Portlet 543, 550
 - DB2 Connect client on WebSphere Portal and IBM Workplace Web Content Management server 79
 - default cache 357
 - IBM Workplace Web Content Management 77
 - IBM Workplace Web Content Management 2.5 137
 - IBM Workplace Web Content Management
 - cache policy 391
 - Java Edition
 - e-mail routing 112
 - fatal e-mail alerts 112
 - library 341
 - Local Rendering Portlet 103
 - parameters 573
 - Personalized List Portlet 451
 - Portal Document Manager
 - documents 341
 - folder 341
 - library 341
 - Portlet 109, 545
 - Portlet in Personalization demo 464
 - portlets 338
 - WebSphere Portal 5.1.01 446
- configuring
 - dynamic cache 385

- confirming configuration 86
- connect tags 368
- connect.cfg file 91, 115, 327, 358, 446, 592
 - advanced cache configuration 360
 - basic cache configuration 359
 - default module property 361
 - ModuleConfig section 116
- connection error 572
- considerations for reusability 151
- content 6, 143, 162, 169, 448, 479, 489
 - editing considerations 146
 - rendering 43, 45
- content author 8, 12, 14, 28, 42, 48, 63, 72, 143, 146, 154, 160, 222, 231, 249, 273, 408
- content component 104, 150, 370, 380, 478, 489
- content component container 478
- content component interface 481
- content creation 3, 8–10, 63, 154, 161, 470
- content custom UI 536, 543, 550
- content document 144–145, 214, 233, 408, 436, 479, 522, 562, 592
- content item 152–154, 177, 228, 285, 292, 430, 554, 581
- content item, assigning as the default content page 291
- content layer 150
- content Library 152, 289
- content management
 - importance 4
- content management within the context of applications 5
- content model 433
- content owner 12, 17–18, 154, 217, 219–220
- content page 151, 172, 203, 223, 288, 414, 425, 430, 478
 - multiple thousands 203
 - related information 314
 - search queries 333
 - third column 321
- content viewer
 - Portlet 464
- Content Viewer Portlets 337
 - creating and copying 336
- content, migrating 599
- content, testing 289
- content-centric items 154
- context 5
- copying
 - Content Viewer Portlets 336
 - Local Rendering Portlet 96
 - River Bend Authoring Template 276
- crawling
 - custom approach 436
 - regular Web sites mode 436
 - WebSphere Portal site mode 433
- creating
 - Actions 272
 - approval Workflow 271
 - Authoring Template 274, 343
 - breadcrumb navigator 310
 - Bulletin Board page with portlets 540
 - categories for the site taxonomy 267
 - content 288–289, 293, 461
 - content item 291
 - Content Viewer Portlets 336
 - content with the River Bend Portal Document Manager Authoring Template 345
 - database alias 80
 - Development Express Workflow 270
 - documents 342
 - Edit Document Navigator 323
 - e-mail action 273
 - folder 342
 - HTML component to format the site area menu 317
 - HTML components to specify bold and plain text formatting 308
 - HTML Quick Search Form 333
 - HTML Search Form 331
 - IBM Workplace Web Content Management content using Profile 461
 - IBM Workplace Web Content Management database on DB2 server 78
 - image components 285
 - image library components 285
 - intranet Workflow stages 268
 - Left Navigator 305
 - library 341
 - menu using categories 319
 - menus 314
 - Personalization Rule 449
 - Portal Page 337
 - Portal Page Layout presentation template 334
 - presentation templates 276–277
 - Profile Rule 457
 - Publish stage 269
 - Related Information Menu 321
 - scheduled move action 272

- search results template 330
- site 259
- site area menu component 314
- site areas 260
- Site Map Navigator 312
- site map navigator 312
- site taxonomy 266
- Tools Navigator 325
- two-column table presentation template 279
- two-column table search presentation template 281
- Unrestricted Resources 584
- creating content 3, 8–10, 63, 154, 161, 470
- custom caching with connect tags 368
- custom JSP 544, 557
- custom Portlet 411, 418
- custom site 471
- Customizable Template Portlet 30, 147, 535, 553, 562, 618
- customization 561

D

- data caching 362
- data format 570
- data modeling, importance 14
- data transfer 570
 - verifying 571
- data transfer task 570
- database
 - administrator password 83
 - administrator user ID 83
 - URL 84
- database alias 80
- database platform 35, 77
- DB2 Server 86
- DB2 software 13
- default cache 356
- default content 103, 288, 428, 480, 516, 534
 - activated site area 181
- default content pages 316
- default JSP 550
- defining the information architecture 200
- delivery definitions for migration 607
- demilitarized zone (DMZ) 66
- deploying 531
 - considerations 530
 - Portlet 95, 107
- Description field 289, 558

- design advantages 54, 56, 58, 61
- design approach 249
- design components 174
- design disadvantages 55–56, 59, 62
- designing
 - Portlets 486
- determining
 - components of the Web site 229
- developing a caching strategy 371
- development 47, 65
- development components 174
- Development Express Workflow 270
- direct access 54, 56, 201
 - IBM Workplace Web Content Management server 54, 56
- direct feature links 241
- disabling User Migration 597
- disadvantages, design 55–56, 59, 62
- display
 - site map navigator in the site index 313
- Display Title 152, 275, 370, 538
- distinguished name 583
- distributed map 385
- document 476
 - parent site area 522
 - Security section 592
- Document ID 477
- document ID 425, 477, 520–521, 564
- Document Manager 192
- document type 177, 201, 228, 477
- documenting
 - cache strategy 376
- documents 342
- doEdit method 502, 517
- doView method 502, 522, 534
- dynamic cache 348
 - configuring 385
 - distributed map 385
 - installing 385
 - servlet caching 383
- dynamic cache monitor
 - installing 386
- dynamically generated links 170

E

- e.printStackTrace file 476, 480, 563
- edge component 384, 400
- Edge Side Include 395

- Edit Document Navigator 323
- edit document navigator component 324
- edit mode 415
- edit mode JSP 520
- editable component sub-interfaces 478
- editing
 - aptrixsearch.properties file 328
 - challenges 146
 - connect.cfg file 327
 - Customizable Template Portlet 147
 - separate content and presentation 146
 - User Migration Mapping file 594
 - wpconfig.properties file 588
- editing considerations 146
- e-mail 112–113
- e-mail action 273
- e-mail configuration for fatal mail alerts 112
- e-mail, fatal alerts 112
- embedded hyperlinks 243
- enablement 250
- enabling
 - cache 115
 - edge component 400
 - IBM Workplace Web Content Management Cloudscape 74
 - IBM Workplace Web Content Management Search Module 327
 - Portal Personalization 446
 - servlet caching on portal server 386
 - the correct caching module 116
 - the correct caching subsystem 115
 - user migration 590
- enhanced migration 602
- ensuring valid data format 570
- Enterprise Content Management 6
- environments 63
- error
 - connection error 572
 - examining 571, 582
 - failed to start database 572
 - No schema exists 572
 - resolution 571, 582
 - user migration 581
- ESI. *See* Edge Side Include
- example infrastructure 63
- exception handling format 605
- expiration parameters 356
- expiry date 272, 553, 604
- explicit mapping approach 414

- exporting
 - site to HTML or XML 471
- extending Web site with JSP support 485
- external products 440
- external workflow applications 471

F

- failed to start database error 572
- feature links 241
- field names 558
- file
 - aptrixsearch.properties 328
 - connect.cfg 91, 115, 327, 358, 446, 592
 - connect.cfg file 327
 - e.printStackTrace 476, 480, 563
 - linkBuf.append 563
 - migrationUsers.properties 83, 577, 583–584, 594
 - migrationUsers.xml 578
 - time.jsp 528
 - utils.zip 559
 - wpconfig.properties 570, 588
- File Resource component 188
- file system data 568–569, 587–588
- files needed by IBM Support Desk 573, 584
- fix
 - PK02213 92
- fixes 87, 127
- fixpack
 - PQ98531 93
- fixpacks 127
- formatting HTML components 308
- formatting the Navigator 308
- fully qualified distinguished name 583

G

- generating
 - User Migration Mapping file 593
- goals 217

H

- high performance (caching) options 380
- home page
 - initial layout 199
- href 175, 304, 561
- HTML 45, 47, 170, 175, 181, 189, 228, 251, 305, 308, 314, 325, 373, 430, 470–471, 544, 610, 615

- HTML component 317
- HTML components 308
- HTML formatting components 309
- HTML Quick Search Form 333–334
- HTML Search Form 331–332

I

- IBM developerWorks 203
- IBM HTTP Server 45, 54, 56, 115, 351, 440
 - friendly message page 66
 - mod_rewrite module 440
 - recent version 120
 - required resources 122
 - single set 61
 - static files 115
- IBM Redbook team
 - image component 292
- IBM Redbook Web site 622
- IBM Support Desk 573, 584
- IBM Workplace Collaboration Services 24
- IBM Workplace Web Content Management 8–10
 - activity 54–55
 - administration 56, 62
 - administrator 154, 156
 - advanced caching 349
 - API 46, 376, 432, 436, 470–471, 473
 - application 34–35, 72, 122, 129
 - authoring portlet 36
 - backend 425
 - benefits of using 12
 - cache policy 391
 - cache 115
 - caching 349, 381
 - caching performance impact 371
 - capability 530
 - changes from older version 41
 - changes to installation 71
 - code 23
 - component 156, 362
 - component library 175
 - configuration 86, 348, 361
 - configuration change 87
 - configuration file 72
 - configuration, changing 82
 - configuration, confirming 86
 - configure DB2 Connect client 79
 - configure with DB2 Universal Database 77
 - connect.cfg file 91, 118
 - content repository 35
 - content server 35
 - CPU entitlement 23
 - creating content 461
 - creating database on DB2 server 78
 - data 77, 566, 569
 - database 73, 81
 - administrator password 83
 - administrator user ID 83
 - URL 84
 - definitions 152
 - document 408, 436–437
 - document ID 427
 - document title 436
 - EAR file 72
 - enabling with Cloudscape 74
 - engine 114, 368
 - environment 36–37, 47, 70, 230
 - feature 5, 22
 - fields 152
 - file system data 588
 - fix 93
 - fix PK02213 92
 - FixPack PQ98531 93
 - framework 348
 - function 155
 - functional differences between versions 126
 - functionality 470
 - functions 152
 - implementation 376
 - improving performance 348
 - Information Center 85, 118, 127, 361, 363
 - infrastructure 53
 - installation changes 71
 - installing 590
 - API Java file 496
 - instance 349
 - integrating with WebSphere Portal 57
 - integration 446
 - introduction to 9
 - item 576, 578
 - item navigator 152
 - items 152, 154
 - JavaServer Pages tags 40
 - known limitations 573
 - license 23, 25, 41
 - Local Rendering Portlet 37
 - Local Rendering Server 37
 - location 231

- log 113
- logging 114
- logs 87
- migrating content 587, 599–605, 607–608
- navigation 439
- navigator 436
- new features 14
- node 531
- object 349, 531
- ODC editors 127
- overview 10
- page 149
- parameter 570, 589
- parameters 573
- performance 348, 375
- personalization rule 447
- planning, site area 223
- Portal Personalization 128
- Portal search considerations 435
- Portlet 57, 501
- Portlet application 503
- positioning 11
- pre-rendered URLs 120
- pre-rendering 28, 44, 351–352, 364, 380
- previous version 30, 565
 - file system data folder 569
 - group names 582
 - Lotus Workplace Web Content Management users 573
 - migrating data 566
 - predefined group 579
 - predefined user 580
 - Web site 30
- process 46
- processor 61
- product 3, 10
- project 16–17
- public API 39, 46
- Remote Rendering Portlet 38
- removing 122
- render 43, 45
- rendering Portlet 37, 438
- Rendering Portlets 95
- repository 72, 362, 408
- request 44
- resources 585
- sample 538
- sample Workflow Portlet configuration 545
- sample Workflow Portlet customization 561
- search 228
- search collection 435
- Search Module 327, 430
- separating 60
- server 35, 46, 71, 157, 348, 431, 441, 473, 491
- server administrator 156
- server components 71
- server crash 67
- server tier 63
- ship 114
- simple 37, 46
- site 172, 349, 351
- site framework 223
- software 44
- stand-alone environment 43
- stand-alone license 22
- Standard Edition 9, 24
- stores resource 585
- system 10, 14
- table 87
- tag 351
- tier 60
- upgrading 567
- user 72
- using 21
- Version 2.5 11, 24, 126, 129, 566, 574
 - authoring 137
 - configuring 137
 - differences 126
 - installing 128–130
 - license 24
 - overview 126
 - running 126
- Version 5.1 10, 27, 35, 70, 87, 126, 197, 374, 503, 566
 - component architecture 34
 - differences 126
 - installing 27, 87, 92
 - key components 35
 - known issues with pre-rendering 119
- Version 5.1.0.1 374
- WebSphere Portal 23, 37
- Workflow 50, 152
- identifies the properties 361
- identifying performance impacts of caching 371
- iFixes 127
- image component 163, 175, 190, 275, 285, 292
 - width fields 287
- image components, creating 285

- image components, referencing 285
- image fields 177
- image library components 285
- implementation
 - beginning 414
- implementing
 - caching strategy 353
 - custom display Portlet 418
 - link management 422
 - links, 2-phase linking 421
 - pre-rendering strategy 353
- importing
 - from HTML 470
 - from third-party 471
- improved configuration technique 72
- improving performance 348
- Including rule in JSP 467
- including rule on presentation template 453
- information architecture 3, 14, 17, 170, 199, 206, 215, 372
 - additional features 204
 - addresses key principles 28
 - critical component 232
 - elegant solution 214
 - exceptional example 214
 - good example 205
- information architecture layer 150
- information architecture, defining 200
- Information Center 85, 118, 127
- information design 232
- information within the context of Web content 4
- initworkspace tag 488
- installation changes 71
- installing
 - dynamic cache 385
 - dynamic cache monitor 386
 - Edge Side Include 395
 - IBM Workplace Web Content Management 73, 530, 590
 - IBM Workplace Web Content Management 2.5 128–130
 - IBM Workplace Web Content Management 5.1 27, 87, 92
 - IBM Workplace Web Content Management API
 - Java files 496
 - overview 70
 - post installation configuration 112
 - PQ98531 fix 93
 - Rational Application Developer 6.0 492

- Remote Rendering Portlet 107
- Rendering Portlets 95
- WebSphere Portal 5.1.01 446
- integrating
 - IBM Workplace Web Content Management 446
 - WebSphere Portal and IBM Workplace Web Content Management 57
- intranet 268
- intranet portal 216, 536
- item
 - labeled version 153
- item and component security 153, 158
- iterator 479, 485, 551

J

- J2EE application 40
- Java 158
- Java code 23, 40, 42, 474
- Java Content Repository 445–446, 448
- JavaScript array 423, 563
- JCR. *See* Java Content Repository.
- JSP 40, 192, 278, 280, 354, 364, 407, 423, 467, 470, 485–487, 490, 526, 528, 544, 550
- JSP components 526
- JSP support 485
- JSP tag library 487, 490

K

- key words 228

L

- labeled version, content item 153
- lastname attribute 577
- law of diminishing returns 616
- layout
 - approach 249
 - portal page 407
- LDAP 42, 70, 72, 92, 129, 246, 445, 447, 583
- LDAP users and groups, migrating 579
- left navigator 307
- left navigator component 309
- Library component 104, 158, 286, 368, 376, 470, 481, 490, 554
- library, creating and configuring 341
- license 22–23, 25, 41
 - IBM Workplace Web Content Management 2.5 24

- line feed (LF) 570
- link integrity 427
- link management 418–419, 425
 - implementing 422
- linkBuf.append file 563
- linking process 424
- links
 - 2-phase linking 421
 - broken 170
 - dynamically generated 170
- Links section 106
- local rendering
 - Portlet 15, 37, 41, 74, 95–96, 103, 336, 364–365, 408, 414–415, 437
 - server 37
- log 113
- log files, using 581
- logging levels 113
- logs 87
- Lotus Workplace 37, 41, 126
- Lotus Workplace Web Content Management 573

M

- mail routing configuration 112
- making the implementation more functional 416
- Management API 432
- managing content 5
- manual rendering 119
- manual versus automated rendering 119
- manually running the pre-rendering engine 119
- mapping file 577, 593
- master configuration 90, 389, 591
- Member Fixer utility 580
- menu component 170, 182, 315, 321, 370, 481, 536
- menu form 317
- menus, creating 314
 - creating
 - menus 238
- MENU-Site Area 176, 315, 491
- metadata 161–162, 164, 223, 225–226, 228, 266, 371, 376–377, 449, 602, 604, 611, 613
- method
 - actionPerformed 502, 521, 562
- migrating 567
 - Administrators group 578
 - applying standard templates 602
 - approach and toolset 602

- as-is migration 602
- business case 601
- content 599–605, 607–608
- data 587
- data from previous version 566
- delivery definitions 607
- determining source classifications 603
- determining target definition 604
- enhanced migration 602
- exception handling format 605
- groups 573
- law of diminishing returns 616
- LDAP users and groups 579
- quality assurance requirements 607
- restructured migration 602
- roadmap 600–605, 607–608
- rules 600
- standardized migration 602
- transformation requirements 605
- undertaking 608
- user and group 581
- User Migration Mapping file 577
- users 573–574, 591
- users and groups 576
- using a user migration mapping file 593
- verifying 581
- migration
 - user and group 581
 - verifying 581, 595
- migrationUsers.properties file 577, 583–584, 594
- migrationUsers.properties file, modifying 83
- migrationUsers.xml file 578
- Model SPI 417
- modifying
 - presentation template for the Search page 329
 - wpconfig.properties file 83
- modular approach to using components 151
- multilingual 425
- multilingual and personalized content display 425
- multilingual content display Portlet 534
- multiple Portal pages 439
- multiple rendering Portlets 41

N

- Name field 152, 259
- navigational tools 170
- Navigator 177, 308
- no schema exists error 572

NullPointerException 571

O

Objectives and Key Success Factors 218

ODC (Rich Text) editors 127

overview of chapters 26

P

page design 15, 141, 149, 228, 478

parameters 573

parent site area documents 522

performance 348, 375

improving 348

personalization demo 457

personalization essentials 444

Personalization Rule 449

personalization rule 374, 443, 447

personalized content 425

placeholder tag 181, 307, 317, 370

Planning the site framework and site areas 223

populating the JCR with existing content information 448

Portal Document Manager 127, 192, 196, 343–345

Authoring Template 343

Authoring Template, creating 343

configure 341

integration 127, 196, 341

integration for Version 5.1 196, 341

pairing the Authoring Template and the presentation template 344

path reference 193

Portal page 43, 58, 61, 106, 334, 375, 404, 407, 415, 418–419, 433, 450–451, 543, 564

Portal Page Layout 334

Portal Personalization 128, 446

Portal search

considerations 435

Portlet 15, 23, 36–38, 45, 57, 73, 77, 147, 229, 258, 334, 337–338, 404, 433, 437, 445, 450, 464, 488, 491, 501, 503, 533, 538, 542

content viewer 464

placing copy on a page 99

rendering 41

Portlet configuration 545

Portlet customization 561

Portlet Data Bean 521

Postbody field 175, 294

image component coffeebeans 294

pre-rendered URLs 120

pre-rendering 28, 351, 353, 363, 380

pre-rendering engine, running manually 119

pre-rendering via HTTP Server 351, 365

pre-rendering via IBM Workplace Web Content Management 352, 364

presentation 143

presentation template 15, 46, 152, 155, 174, 232–233, 276–277, 279, 281, 283–284, 303–304, 307, 310–311, 322, 324, 326, 329, 334, 344, 371, 419, 432, 453, 483, 525, 538

breadcrumb navigator component 310

caching events timeline 376

consistent design element 191

document navigator component 324

entry 377

entry ID 377

HTML Quick Search Form 334

image components 285

left navigator 307

left navigator component 307

new Component 344

new component 344

Portal Document Manager component 344

referencing images 287

same cache 379

section 279

stylesheet 303–304

Tools Navigator Component 326

presentational layer 150

previous version 566

file system data folder 569

group names 582

Lotus Workplace Web Content Management users 573

migrating data 566

predefined group 579

predefined user 580

Web site 30

Private (Trusted) Network 66

processing

User Migration Mapping file 595

production 49, 66

Profile 161, 461

Profile Rule 457

property name 358

Publish stage 269

PUMA SPI 426

Q

quality assurance for migration 607
Quick Edit demo 548
Quick Search Form 333–334

R

Rational Application Developer 487
Rational Application Developer 6.0
installing 492
Read-only mode 158
Really Simple Syndication 8, 46
rearrange the site area order 265
re-creating the resources 597
Redbooks Web site
Contact us xxii
referencing
breadcrumb navigator component in the presentation template 311
category menu 319–320
edit document navigator component in the presentation template 324
HTML formatting components in the left navigator component 309
HTML Quick Search Form in the presentation template 334
HTML Search Form in the Search content item 332
image component in a content item 292
image components 285
images in a presentation template 287
left navigator in the presentation template 307
menu format in the site area menu's component design 317
new component in the presentation template 344
related information menu in the Three Column Table presentation template 322
site area menu in specified default content pages 316
stylesheet in the presentation template 304
Tools Navigator component in the presentation template 326
Related Information Menu 321
relative paths 119
remote DB2
database 73
server 70
Remote Rendering Portlet 15, 36, 38, 41, 95, 107

configuration settings 117
configure mode 109
Credential section 110
credentials 110
remote server, attaching to 497
removing
IBM Workplace Web Content Management 122
RendererUser 118, 364
rendering 119
content 43, 45, 478
rendering IBM Workplace Web Content Management content 43, 45
rendering options, overview 43
rendering Portlet 37, 438
Repository interface 474
representative skills 17
request.getHeader 426, 534
Resource Checker utility 586
Resource Description Framework 226
resources 121, 188
re-creating 597
resources folder 121, 585
responsibilities 19
restructured migration 602
results of poor information architecture 215
return on investment (ROI) 441
re-usability 151
reusability 149, 151
reviewing
User Migration Mapping file 594
Rich Text component 188
Rich Text editor 554
River Bend 28, 63, 66, 105, 141, 151–152, 165, 178, 202, 223, 230, 253, 255, 259, 265, 267, 276, 345, 412, 525, 530
authoring 36
Authoring Template 276
building the Web site, prerequisites 257
Coffee 28, 65, 141, 202, 253, 255
Colombia Coffee 295
creating content 293
default content page 292
development 163, 274
External News 276
External News Authoring Template 274
footer 280
footer image component reference 278
Home page 149
infrastructure 64–65

- logo 277
 - MENU-Site Area 184
 - navigation 202
 - news brewing 291
 - open the Web site 303
 - page overview 229
 - Portal Document Manager 343–344
 - Portlets 337
 - River Bend Portlets 338
 - site 28, 64, 67, 105, 141, 253–254
 - site area 184
 - site area menu 316
 - site example 146
 - site framework 170
 - Tea 155
 - Web site Categories taxonomy 289
 - Web site Category 266
 - Web site display 177
 - Web site displays content 184
 - Web site text component 188
 - River Bend Coffee and Tea Company. *See* River Bend.
 - roadmap to migration 600–605, 607–608
 - roles 17, 19
 - routing, e-mail 112
 - running
 - data transfer task 570
 - IBM Workplace Web Content Management 126
 - wpsconfig script 85
- S**
- same server
 - published content 15
 - sample physical architectures 53
 - sample Rich Text editor extensions 554
 - sample Workflow 545, 561
 - saving the stylesheet as a file resource component 304
 - scheduled move action 272
 - Search content item 332
 - search criterion 165, 170, 225, 306, 315, 449
 - Search Form 331
 - Search Module 327, 430–431
 - Search Module Index 431
 - search product 440
 - search result 244, 276, 283, 391, 430
 - search results template 330
 - search rules 194
 - security 153, 158
 - access rights 158
 - delete access 159
 - edit access 158
 - live access 158
 - read access 158
 - Search Module 431
 - security model
 - Java edition 158
 - Security section 592
 - security settings
 - refining Workflow security through values in ap-
trixjpe.properties file 160
 - select e-mail 273
 - selected site area
 - document ID 521
 - separating
 - IBM Workplace Web Content Management and
WebSphere Portal 60
 - separating content and presentation 143
 - session bean 517, 520, 550
 - setContext 489
 - setExplicitContext 488
 - setting proper expectations about working with this
product 16
 - setting up Caching 356
 - setting up the Development Environment 491
 - settings section 111
 - site 172, 259
 - site acceptance 249
 - site area 118, 154, 165, 172, 199, 202, 223,
260–262, 291, 317, 364, 426, 453, 478, 521, 534
 - AO Foundation navigation 235
 - appropriate name 262
 - Authoring Template 233
 - children site areas 307
 - content paths 517
 - creating menu component 314
 - default content 118, 181, 307
 - default content page 288
 - default content pages 316
 - format content 276
 - hierarchal list 518
 - ID 475
 - individual designs 172
 - link order 261
 - main navigation 236
 - menu 316
 - menu format 314

- menu specific 184
- new content 316
- new pairs 173
- parent site areas 310, 518
- rearranging order 265
- sequential placement 262
- site area 292
- template mappings 285
- top levels 202
- site design 27–28, 151, 199
- site framework 3, 154, 170, 177, 199, 223, 259, 380, 479–480
 - appropriate Authoring Template 177
 - children site area nodes 265
 - container 480
 - only root 173
 - same hierarchy 263
 - site areas 173
- site index 313
- site map
 - navigator 312–313
 - navigator component 312
- Site Map Navigator 312
- site map navigator 312–313
- site navigators 234
- site taxonomy 266–267
- site visitor 29, 37, 42, 72, 169, 331, 333, 349–350, 581, 584
- software bundles 24
- source classifications for migration 603
- stage
 - Publish 269
- staging environment 48–49
 - functional elements 49, 64
 - performance testing 49
- Staging to Production 67
- stakeholders 249
- stand-alone license 22
- standardized migration 602
- static pre-rendering 44, 114
- String headerLang 534
- string representation 520
- stylesheet 106, 155, 276–277, 303–304, 348
- supported browsers 126
- supported JVMs 127
- syndication 50, 67, 153, 157, 231
- system programming interface 417
- system-defined security level 159

T

- target definition for migration 604
- taxonomy 164, 193, 266–267, 484
- technical background to seed-list indexing method 433
- Technote
 - 1201178 87
- template catalog 376
- template, search results 330
- testing
 - database catalog 81
- testing your site 289
- text component 188, 275, 373, 482, 538
- text field 162, 544
- third-party products 440
- Three Column Table 322
- time to implement 441
- time.jsp file 528
- tool for managing large amounts of content 21
- Tools Navigator 325
- Tools Navigator Component 326
- Tools Navigator component 326
- toolset for migration 602
- transferring 567
 - file system data 568, 587
- transferring file system data 569
- transformation requirements for migration 605
- troubleshooting 571, 581
 - data transfer 571
 - user and group migration 581
- two-column table presentation template 279
- two-column table search presentation template 281
- typical component types 149
- typical responsibilities within the different roles 19

U

- UI access group 579
- uninstalling IBM Workplace Web Content Management 122
- Unrestricted Resources 584
- unrestricted resources, assessing 585
- upgrading 567
- use case scenarios 222, 470
- user goals 217
- user interface 7, 10, 36, 158, 494, 578
- user migration 574
 - error 581
 - mapping file 575

- process 575
- User Migration Mapping file 577, 593–595
- User Migration, disabling 597
- user profile 485
- User Repository 42
- user-centric design 219
- user-defined security level 159
- user-friendly URLs 421
- username component 194
- utils.zip file 559

V

- valid data format 570
- Vamosa 600–601, 603, 607, 609–613, 615–616
- Vamosa Content Migrator. *See* Vamosa.
- Vamosa Implementation Architecture (VIA) 601
- verification, exception, and correction document (VEC) 606
- verifying
 - data transfer 571
 - migration 581, 595
- version management 153, 155
- view content in a Portlet 334
- view mode JSP 524
- views navigator 259
 - content library 291
 - image component 286
 - right side 290
 - site framework view 291
 - site management 292

W

- WCM_API class 474
- wcm_migration 571
- wcmadmins group 73, 578
 - Administrators group 578
- WcmDbLibrary 85
- WcmDbName 84
- WcmDbUrl 84
- WcmDbUser / WcmDbPassword 84
- Web content 3–4, 33, 46, 69, 97, 254, 258, 429, 489, 526, 533
- Web content management 69
 - key concepts 3, 8, 19, 254
 - overview 6
 - process 12
- Web content management process 12
- Web page 478

- fast rendering 49
- left side 305
- search query 333
- Web server 9, 45, 122, 348
 - alias entry 367
 - cache settings 375
 - static HTML 45
- Web site
 - appropriate places 21
 - basic level 223
 - customized design 155
 - design elements 158
 - determining components 229
 - different elements 14
 - different sections 225
 - extending 485
 - hierarchical structure 154
 - information architecture 216
 - left-hand 305
 - logical arrangement 234
 - overview of building 256
 - share information 218
 - site map 223
 - specific objectives 222
 - voice 391
- WebSphere Application Server 40, 87, 132, 348, 386, 486, 574
 - administrative user 133
 - enterprise application 380
 - installation directory 93
 - Login screen 88
 - necessary configuration 497
 - necessary shared libraries 496
 - root directory 40
 - web-app directory 486
- WebSphere Dynamic Cache Service 380
- WebSphere Member Manager 15, 35, 44, 72, 257, 566–567, 576, 591
 - adding new users or groups 578, 596
 - correct access 87
 - existing Mary Jones user 576
 - migrating 577
 - migrating users 591
 - migrating using a mapping file 593
 - migrationUsers.xml 578
 - new Administrators group 578
 - user names 576
- WebSphere Portal 42
- WebSphere Portal 10, 45, 348, 433

- administrative user 134
- base installation 131
- changes 91
- configure DB2 Connect client 79
- different physical server 70
- entitlement 23
- extensive knowledge 20
- IBM Workplace Web Content Management 229
- IBM Workplace Web Content Management solution 37
- improved integration 10
- installing and configuring 446
- integrating with IBM Workplace Web Content Management 57
- integration 404
- limited IBM Workplace Web Content Management entitlement 23
- logs 87
- navigation 405
- search 433
- separating from IBM Workplace Web Content Management 60
- Server 572
- WebSphere Member Manager capabilities 42
- Windows 2000 Advanced Server 129
- wireframe demonstration 250
- work products 251
- Workflow 50, 160, 166, 230, 271, 484, 545, 561
 - actions 168
 - compared with syndication 49
 - security 160
 - stage 484
 - Stages 167
 - stages 268
- working with JSP components 526
- wpconfig.properties 72, 82, 567
 - data folder 569
 - database administrator user name values 572
 - IBM Workplace Web Content Management parameters 573
- wpconfig.properties file 570, 588
- wpsconfig script, running 85
- ws.getByld 480, 534
- XML version 75, 391
- XMLAccess 417, 575

X

- XML 471
- XML configuration interface 417
- XML structure 433

Archived



Redbooks

Web Content Management for Portal 5.1 and IBM Workplace Web Content Management 2.5

(1.0" spine)
0.875" <-> 1.498"
460 <-> 788 pages



Redbooks

IBM Workplace Web Content Management

for Portal 5.1 and IBM Workplace Web Content Management 2.5

Installation, configuration, and deployment best practices

Building and managing a site

Advanced topics

IBM Workplace Web Content Management is a powerful and sophisticated Web content management product that is designed to accelerate development and delivery of critical business information. It enables end-to-end collaboration for content creation, approvals, management, retention, and publishing across Internet, intranet, extranet, and portal assets. Despite its power, it is remarkably easy to use.

This IBM Redbook provides a comprehensive approach to understanding and using IBM Workplace Web Content Management. It addresses the specific technical aspects of IBM Workplace Web Content Management and discusses its underlying architecture and features. It also recommends deployment scenarios and provides detailed discussions on how to best use the product to take advantage of built-in features and its extensible API.

To address the needs of a broad range of users, this book provides a basic tutorial and then builds upon this foundation to explore advanced topics and opportunities for customization. This book discusses key concepts and best practices for creating a accurate Information Architecture and site framework. It also addresses how to upgrade data from earlier versions of the product and how to migrate data from external Web content management systems.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6792-00

ISBN 0738494259