INTRODUCTION

Service-Oriented Architecture (SOA) is changing the way applications are architected and delivered. Monolithic applications are being transformed into reusable services that can be orchestrated, managed, and consumed by composite applications. The focus and value-add of SOA within an enterprise centers on agility: the ability to make organizations more nimble so they can better react to ever-changing market conditions. While agility is a key driver, it is not the only one. Equally, if not more important, is making every information worker more productive. So, not only do we have to think about how to build applications more effectively (with SOA), we also need build more effective applications. This is the primary objective of the Oracle WebCenter Suite: to provide the tools to deliver the next generation online work environment that dramatically improves efficiency and productivity by adapting to the way people work.

This white paper introduces Oracle WebCenter Suite, a new option in Oracle’s Fusion Middleware product line. Oracle WebCenter Suite is focused on providing tools and services for creating highly productive user work environments that take advantage of SOA to enable business users to bring complete context to their daily work tasks.

WHAT IS ORACLE WEBCENTER SUITE?

Oracle WebCenter Suite combines the standards-based, declarative development of Java Server Faces (JSF), the flexibility and power of portals, and a set of integrated Web 2.0 services to boost end-user productivity. Together, these services provide a unique ability to build applications that eliminate context shifts and maximize productivity. For example, using Oracle WebCenter Suite capabilities, you can build applications that allow the user to interact directly with instant messengers, voice over IP, and other collaborative services directly within the application. The user doesn’t merely pass through the application to get to these services; they are integrated into the very fabric of the application.

Oracle WebCenter Suite 10g Release 3 is targeted at the development community, delivering a rapid development environment comprising four major components: WebCenter Framework, WebCenter Services, WebCenter Anywhere, and the WebCenter Extension for JDeveloper.
The WebCenter Framework allows you to embed portlets, content, and customizable components in your application. Content is accessed via JSR170-based content integration services, which make it possible to expose data and applications residing in Oracle Content DB, third-party content management systems (via separately available content adapters), OracleAS Portal, and even the file system. All Framework pieces are integrated into JDeveloper, providing you unified access to these resources as you build your applications. Furthermore, the powerful Search framework allows you to seamlessly include enterprise-wide search capabilities into your application.

WebCenter Services incorporate Web 2.0 content, collaboration, and communication services that can be embedded directly into applications via portlets. In addition, APIs can be utilized in the Oracle Application Development Framework (ADF) to create custom declarative UIs and to integrate some of these services into business processes and application features. Oracle ADF is a J2EE framework that provides a visual and declarative development experience. For more information on ADF, visit the Oracle ADF home page.

WebCenter Services include:

- Oracle Content Database (Oracle Content DB), a database-centric content management solution
• Oracle Secure Enterprise Search, through which your users can locate data and documents that are stored anywhere within your enterprise

• A presence server, so users can see when others who have logged into the application are online*

• An instant messaging service, allowing users to communicate directly with other online users via text, voice, or video*

• WebCenter Voice Option, offering a rich set of telephony infrastructure components*

• A discussion forum application, enabling community discussions on a set of topics*

• A wiki/blog server, allowing people to collaborate simultaneously on documents and share ideas*

* Note: These features will be released shortly after general availability of Oracle WebCenter Suite 10g Release3.

**WebCenter Extension for JDeveloper**

Oracle JDeveloper is an integrated development environment (IDE) for building service-oriented applications using the latest industry standards for Java, XML, Web services, and SQL. Oracle JDeveloper supports the complete software development life cycle, with integrated features for modeling, coding, debugging, testing, profiling, tuning, and deploying applications. The WebCenter Framework capabilities and resources are seamlessly integrated and exposed in JDeveloper to maximize the productivity of enterprise application developers.

![Figure 2 - Drag and Drop WebCenter Components in the Component Palette](image)

All the development functionality you need for building WebCenter-enabled applications is available in the JDeveloper environment. JDeveloper's visual and
declarative approach and ADF work together to simplify application development and reduce mundane coding tasks. For example, code for many standard user interface widgets, such as buttons, lists of values, and navigation bars, is prepackaged for you.

WebCenter components are readily accessible from a catalog of resources available on the Components and Data Control palettes. Simply drag the resource you require and drop it on a JSF page, where it will be surfaced as a view component that is automatically bound to the ADF model.

WebCenter provides several wizards to help you with essential development tasks such as building a portlet, consuming an existing portlet, creating a data control to a content repository, and securing your application (to mention only a few). By significantly reducing the amount of coding you need to do, WebCenter dramatically increases your productivity as a developer.

For more information on JDeveloper development environment, visit the JDeveloper home page on OTN.

Application Templates

In JDeveloper, the easiest way to ensure that you properly define an application and its projects with the appropriate technology scope is to apply an application template. An application template automatically partitions the application into projects that reflect a logical separation of the overall work. The WebCenter Suite provides a template optimally configured for building WebCenter applications.

The WebCenter Application template consists of a project for the data model (Model); a project for creating portlets (Portlets); and a project for consuming portlets, components, and data controls (ViewController). For simplicity, the WebCenter Application template folds these three projects into one application, but you are welcome to arrange your applications and projects in the way that best suits your requirements.

WebCenter Anywhere

The need for productivity and efficiency extends beyond the office. In today’s fast-paced world with ever more powerful mobile devices, it has become critical to work effectively wherever you are. WebCenter Suite leverages Oracle’s robust wireless platform to bring the benefits of a unified work environment to all types of mobile technologies, including connected devices such as PDAs and Smart phones, Mobile Voice, Mobile Messaging and even Telnet-based devices for industrial users. In addition, users might want to access their WebCenter application directly from their Windows Desktop and Office tools.

WebCenter Anywhere is focused on bringing the productivity and efficiency of context-rich WebCenter applications to users wherever they happen to be.
ADF Mobile

As WebCenter developers begin to reap the benefits of increased productivity from JSF technology, Oracle seeks to bring the same benefits to mobile application development with ADF Mobile.

Starting with Oracle Application Server 10g Release 3 (10.1.3.2.0), ADF Mobile and JDeveloper leverage the JSF standard to facilitate the rapid development of applications such as:

- Mobile browser applications for PDAs (Microsoft PocketPC, Palm Treo, and so on)

  Developing mobile browser applications for PDAs with ADF Mobile leverages the same methodologies involved in developing JSF applications for the desktop, but with a few mobile-specific extensions. With support for over 60 of the standard ADF Faces components, developers can build applications with the rich component set, each rendered appropriately for small-screen mobile devices. In this way, WebCenter developers can reuse their desktop browser application’s Model and Controller layers and simply assemble a new View layer for PDAs, using a similar set of ADF Faces components.

Oracle Drive

Oracle Drive is the desktop client for Oracle Web Distributed Authoring and Versioning (WebDAV) servers, such as Oracle Content DB and Oracle Portal. Oracle Drive allows you to access Oracle WebDAV server files as a mapped drive in Windows Explorer, exposing the native Explorer functionality. Additional content store-specific options are accessible from right mouse menus. When you are disconnected from the network, you can use the offline and synchronization capabilities of Oracle Drive to manage files. You can also use Oracle Drive to back up files from your hard disk to a server.

BUILDING A WEBCENTER APPLICATION

Using the WebCenter Framework, you can easily build customizable applications by leveraging the customizable components. WebCenter Framework provides the foundation for including Web 2.0 services in applications, allowing you to add enterprise mashup capabilities that enable end users to get the content they require to get their job done. You can also integrate JSR 168, WSRP, and Oracle portlets into WebCenter applications and include content from disparate content repositories. Using WebCenter Framework, you can link components together so that they operate synchronously, resulting in a cohesive, easy-to-understand, easy-to-use application.

Making Applications Customizable

Customizable components provide the ability to customize the behavior of the application at runtime. These components enable the site administrator to make changes to the users’ view of the application content. For example, the
administrator may decide to hide a certain piece of content or move a component from the bottom of the page to the top. These changes and settings are then persisted for end users.

Customization applied to the application is saved to a metadata repository, Oracle’s Metadata Services (MDS). All metadata, including base application definitions and runtime customizations, is stored in the central metadata store. This centralized metadata strategy allows WebCenter to bring together work done by information workers at runtime and IT developers at design-time in a single, complementary development lifecycle. This means that new versions of applications can be developed and deployed without losing any of the customization work done by information workers over the life of the product.

WebCenter Services provides two customizable components:

- **showDetailFrame**
- **panelCustomizable**

**The showDetailFrame Component**

A showDetailFrame surrounds one or more JSF and/or ADF components and can provide a border and a header with a menu for actions such as minimizing the content. A showDetailFrame component provides the following:

- Minimize, maximize, and restore the display of the contained component(s)
- Header and border around the contained component(s)
- Actions menu to perform specific tasks on the child component(s)

**The panelCustomizable Component**

A panelCustomizable component offers horizontal and vertical layout capabilities to a group of ADF components. Similar to a showDetailFrame component, a panelCustomizable component can display a header to provide menu actions. A panelCustomizable component enables you to do the following:

- Hide and show child components
- Rearrange child components within the panelCustomizable component

You can include JSF and ADF Faces components as items in panelCustomizable and showDetailFrame components on your application pages.

**Building Portlets**

The WebCenter Framework enables you to build and deploy portlets using the WebCenter Framework Extension for JDeveloper. The WebCenter Framework
supports all the production portlet standards, including JSR 168 and WSRP 1.0, as well as a preliminary version of WSRP 2.0.

Oracle JDeveloper’s portlet creation wizards help you build WSRP/JSR 168 and PDK-Java (Portlet Development Framework for Oracle Portal 10g) portlets quickly and easily. Another set of wizards guide you through portlet producer deployment. For testing purposes, you can deploy your portlets to the WebCenter Preconfigured OC4J, which you can start and stop from the JDeveloper toolbar. This means that you never have to leave the JDeveloper environment while developing and testing your portlets.

![JSR 168 Java Portlet Wizard - Step 1 of 8: General Portlet Properties](image)

**Figure 3– Building a Portlet Using the JSR 168 Java Portlet Wizard**

**Portletizing JSF Applications**

Portletizing an application means making an application available as a portlet. A portletized application can be plugged into a portal, where end users can interact with the application in the same way they would with the stand-alone application. Portletizing provides a convenient mechanism for integrating disparate applications and publishing them in one convenient location for the end users. Business users can use portletized applications to create enterprise mashups.

The JSF-Portlet Bridge in Oracle WebCenter Suite allows developers to portletize a JSF application as a JSR 168 portlet. The portletized application can then be consumed by any portal framework that supports the portlet standards, such as an Oracle WebCenter application or OracleAS Portal.

To publish a JSF application as a portlet, you must ensure that your JSF pages emit markup that conforms to JSR 168 portlet markup fragment rules. Most of the ADF Faces components render portlet-compatible markup.
When you portletize an ADF application, there are a number of guidelines and best practices to keep in mind. These are described in the Oracle WebCenter Framework Developers Guide.

**Integrating Portlets**

Your WebCenter applications can consume portlets. The WebCenter Framework portlet integration supports all the production portlet standards including JSR 168 and WSRP 1.0, as well as a preliminary version of WSRP 2.0.

![Figure 4– A WebCenter Application That Integrates Various Portlets](image)

Before using the portlets in your application, you must register the portlet producer with the application. Registration informs the application how to access the producer. Wizards are available to guide you through the registration process. After successful registration, the portlets are automatically added to JDeveloper’s Component Palette. You can drag individual portlets from the Component Palette and drop them onto your application pages as you would any other component.

You can also use add Oracle prebuilt portlets and third-party portlets to your WebCenter applications. Packaged applications often come with their own sets of portlets that enable you to access particular data or functions used by the application. Provided these portlets were built with compatible technology (WSRP, JSR 168, or PDK-Java), you can include them in your WebCenter application as well.

You can link portlets by passing parameters between portlets and ADF Faces components, and between portlets and the page. In this fashion, you can create a context-sensitive application, where the data displayed by the portlets changes depending upon the page context.
Business User Web 2.0 Enterprise Mashup Tools

In Web 2.0 terminology, a *mashup* is the result of integrating complementary elements from two or more sources. Without WebCenter Framework, creating mashups on the Web is a difficult and challenging task; using the tools provided by WebCenter Framework, however, business developers can create their own mashups in minutes. A set of declarative wizards guides users through the process of combining data and services to suit their individual requirements. WebCenter’s versatile publishing tools include:

- **Rich Text/Blog Portlet**: A portlet that offers browser-based, rich text editing at runtime. It allows users to author and publish formatted HTML on a JSF page. The Rich Text/Blog portlet is a useful tool for blogging or for posting enterprise announcements and news items. Once the portlet is added to a page, users with appropriate permissions can edit text items directly in the page. All the necessary editing tools are presented in place. Depending on the security privileges set for the page, the Rich Text portlet can be used to broadcast information (such as a simple blog) to a wide audience or to a more narrowly defined group.

![Rich Text Portlet](image)

**Frank Nimphius' Blogbuster**

October 19, 2006

ADF Faces: Create Printable Pages

As certain at the sun raising each morning, there is a need for printing in web applications. With printing I don’t mean printing in terms of reporting, but printing in terms of printing web pages without buttons, links and images. I did blog about printing a while back, explaining how to print from a print page launched in a dialog. Though this blog post is worth reading, I was looking for something that doesn’t require changes in the application.

Now I am back with a different approach based on Stylishjets, CSS. Below is an image that shows the EditMain page from our IDE demo reference demo. This page contains a table listing the customer interactions that we want to print.

![Figure 5 – A Simple Blog Using the Rich Text/Blog Portlet](image)

- **Web Clipping Portlet**: A browser-based declarative tool that empowers business users to create their own feeds with no programming required. In addition, users can use the WebClipping Portlet to clip some information to determine whether they need to ask IT to provide a supporting feed. Using the Web Clipping portlet, users can create Web 2.0 mashups from nearly any source.

- **OmniPortlet**: A tool that allows business developers to quickly leverage Web services, RSS, and XML feeds as data sources for their enterprise mashups. Using a wizard-based approach, business users can quickly produce new mashups from all the standard feeds that developers produce. In addition, developers can easily build AJAX-based user interfaces and add them into the wizard for users to select. Figure 6 shows
how quickly users can define new mashups from existing Web services and XML or RSS feeds.

Figure 6 – Building a Mashup with OmniPortlet Wizard

Portlet Interaction with ADF Faces Components

ADF Faces components and portlets are tightly integrated. They can reside on the same JSF page and can be contextually wired together in a declarative manner, without your writing a single line of code. User interaction in a portlet can trigger partial refresh on the page, making other components on the page rerender. Similarly, when the state of an ADF view component changes on the page, portlets can refresh their content, ensuring that they remain in context.

A User Interaction Example:

In the figure below, the drop-down list called “Volume for last” is a JSF component that determines the contents of the pie chart generated by OmniPortlet. When the user selects a value from the drop-down list, the value is passed to OmniPortlet. OmniPortlet reads the value and performs a partial page refresh, without causing the entire page to be refreshed.
Styling a WebCenter Application

Achieving a consistent and attractive look and feel is an important part of application design and development. Skins are the best way to globally style an application. Based on the CSS 3.0 syntax, a skin is a style sheet that is specified in one place for an entire application. Instead of styling each component or inserting a style sheet on each page, you can create one skin for the entire application. Every component automatically uses the styles as described by the skin. If an application is constructed to use a skin, no design-time code changes are required.

Web Center applications use ADF Faces for their visual components. ADF Faces components delegate the display of the component to a renderer. Renderers determine the different ways a component can be displayed on a client, or how to display the component on different clients. Included with ADF Faces are HTML render kits for display on both desktop screens and PDAs. Leveraging these renderers in combination with skins, you can customize how components are displayed; for example, you can change the orientation or location of tabs. By default, applications created using ADF Faces components use the Oracle skin, but you can create your own skin to change the colors, fonts, and even the location of portions of ADF Faces components, by setting styles for components in one CSS file.

Oracle ADF Faces provides three skins for use in your applications:

- **Oracle**: The Oracle skin, which is the default skin, conforms to Oracle's user interface standards for applications (known as Oracle Browser Look and Feel, or Oracle BLAF).
- **Minimal**: The Minimal skin is based Simple skin but includes some icons.
- **Simple**: The Simple skin contains almost no formatting and is the starting point for all custom skins.
All of these skins are included in the ADF Faces 10.1.3 component library.

In addition to the default skins, you can create your own custom skin with your company's preferred look-and-feel. Custom skins can extend or override the style definitions provided through the Simple skin. When you apply your own CSS, (that is, your custom skin), everything that you did not include in your CSS is inherited from the Simple skin. You are unlikely to customize the look and feel of every component available in the Oracle ADF Faces component library. By reviewing your application using the Simple skin, for example, you can determine the components to customize.

INTEGRATING CONTENT USING THE JAVA CONTENT REPOSITORY STANDARD (JCR)

A vital part of today's businesses is managing document content, which is increasingly being managed by enterprise content management systems. Integrating content from these systems usually requires custom coding against proprietary and complex APIs, which makes creating new applications that leverage these content systems costly and hard to maintain or upgrade.

With the ratification of JSR 170, which specifies the Content Repository for Java API and the Java Content Repository (JCR), access to content becomes standardized and creating new applications that integrate content is made much easier.

Oracle WebCenter Suite leverages the standard and provides an easy alternative to pure coding against JCR APIs for content integration. Using data controls (as specified by JSR 227), WebCenter hides the complexity of the JCR standard behind this generic framework, making integration simply a matter of dragging and dropping the relevant data controls into your application. Adapters written to the JCR API standard provide access to the underlying content system.

A data control is a container for all the data objects, collections, methods, and operations used to create UI components within your application. The data control abstracts the complexity of JCR and surfaces basic operations. As part of WebCenter Suite you have access to four repositories:

- OracleAS Portal
- Oracle Content DB
- Sharepoint
- Documentum

For development and testing purposes, you can also use the WebCenter Suite’s JCR adapter for file system access.

Since JCR is a standard that is currently being adopted by a wide variety of vendors, the range of available adapters is steadily growing. In a case where you have content stored in a proprietary content system (for example, digital media assets), you can create your own JCR adapter and use it together with the WebCenter Framework.
Using the Content Data Control Wizard in JDeveloper, you can create and configure a data control to connect to a specific JCR repository. The data control can be used in your application, regardless of the adapter underneath it.

**Figure 8 – Creating a Content DB Data Control Using the Data Control Wizard**

**Data Control Attributes**

Every content management system stores many attributes for the objects that it manages. These attributes build a set of structured metadata around what is often unstructured information. Structured metadata is essential for categorizing content and for searches; hence, it is important to expose those attributes through an application that leverages the content in any way. The Content Data Control exposes all attributes: the ones that are common across all repositories, repository-specific attributes, and custom attributes.

**Functionality Provided by the Data Control**

To keep things simple, the data control surfaces real-world operations—rather than the atomic API calls—to allow easy application development. The functionality provided by the data control includes Search and Advanced Search, as well as getting the files and folders of a repository, the attributes of these files and folders, and the URL for a file's content.

To use data control functionality in your application, simply drag the data control method or attribute from the Data Control Palette and drop it onto the application page as a view component. All the ADF bindings are automatically created for you.
Securing the Content Data Control

The main issue behind Content Management systems, besides organizing content, is controlling access to it; therefore, another major aspect of the Content Data Control is managing access or, rather, honoring access restrictions of the back-end repository. The WebCenter Framework provides full flexibility in terms of which user context is used by the data control: a Public user, a predefined user, or the user that is authenticated to the application and set in the JAAS security context.

ACCELERATING YOUR APPLICATION WITH WEBCENTER SERVICES

As part of WebCenter Suite, a wide range of Web 2.0 services and tools is available for you to extend the functionality of your applications and enrich the end-user experience. These Web 2.0 services provide enterprises with a means to tap into the critical thinking of every employee to build better processes and make enterprise information (applications and content) more relevant to the entire organization. Earlier in this paper, we introduced some Web 2.0 services, specifically OmniPortlet, Web Clipping, and the Rich Text/Blog Portlet. The following section describes all the other Web 2.0 services that are integrated within WebCenter Suite.

Secure Enterprise Search (SES)

Oracle Secure Enterprise Search 10g enables a secure, high quality, easy-to-use search across all enterprise information assets. You can integrate WebCenter applications (ADF applications) into this unified search experience at two levels:

- **Crawler-Based Approach:** SES can crawl and index all content within all pages of the application. The crawler supports a number of built-in source types, as well as a published plug-in architecture for adding new types.
• **Custom Search Approach:** You can build a custom search UI in your WebCenter application by using the Web Services API that SES provides. Thanks to JDeveloper's Web Services data control, this is very easy to achieve and requires little or no coding effort.

![Figure 10 – Integrating SES into WebCenter Using the Web Services Data Control](image)

Key features of SES include:

• **Security:** The ability to search password-protected sources securely. Oracle’s search technology provides single-sign-on (SSO) based security where available, and it can also employ application-specific security where SSO is not available.

• **Comprehensiveness:** The ability to search across all your sources—Web pages, files-in-file servers or desktop drives, databases, applications, mail servers, and more.

• **Excellent User Experience:** An intuitive, out-of-the-box Web user interface for both search and administration which has both the clean look-and-feel and ease of use that users have come to enjoy for Internet searches.

• **Relevant Results:** Brings to the intranet a high level of relevance that users associate with Internet searches. This high search quality is enabled by number of patented improvements, including a new relevance model geared to the intranet.

• **Sub-second Performance:** New internal index design techniques provide high-performance, high-throughput queries over millions of documents.

For more information, visit the [Oracle Secure Enterprise Search](https://otn.oracle.com) page on OTN.
**Oracle Communication and Mobility Server (OCMS)**

Oracle Web Center provides a scalable telephony infrastructure that you can use to embed telephony components such as call control and presence into applications built with Web Center. Out of the box, Web Center provides a Presence Server and an Oracle Communicator softphone, which can be used to extend a SIP-based instant messaging and voice/video network. Also available are Parlay X presence Web services APIs, which provide presence information to any WebCenter-based application. Examples of presence-enabled applications include package tracking, location-based services, and browser-based instant messaging. Finally, the Web Center Voice Option enhances the WebCenter offering by bundling a rich set of telephony infrastructure components based on JSR 116 (SIP Servlets). The SIP Servlet API and its associated Service Creation Environment enables development and deployment of custom telephony services such as click-to-call, call routing, call barring, conferencing, and voice mail.

For more information, visit the [Oracle Communication and Mobility Server](https://otn.oracle.com) page on OTN.

**Note:** This feature will be released shortly after general availability of Oracle WebCenter Suite 10g Release 3.

**Oracle Content Database**

Oracle Content Database (Oracle Content DB), the default content repository for WebCenter Suite, is a full-fledged content management system that allows you to professionally manage your content through the Web or from your desktop applications. A rich library of ready-to-use Web services is provided to content-enable your enterprise in a service-oriented environment.

Oracle Content DB bridges the gap between limited capability file servers and the specialized, expensive, and complex content management applications that are so widely available.

The details on the integration points between Oracle Content DB and WebCenter are discussed above in the section *Integrating Content Using the Java Content Repository Standard (JCR)*.

**Web Services**

Oracle Content DB provides complete access to application functionality programmatically via J2EE- and Microsoft .NET-compatible Web services APIs. These APIs are being used to integrate Oracle Content DB with other Oracle products and can also be leveraged to automate and extend custom application functionalities.

**Enterprise File and Document Management**

Oracle Content DB provides a number of significant file and document management capabilities in three areas: rich user interfaces, flexible access control, and user management and policy-based management behaviors.
Oracle Content DB delivers a rich experience for both Web and Windows desktop users. Its highly interactive Web interface allows content to be accessed easily anytime, anywhere from any Web browser. The interface provides a rich user experience offering a dynamic tree view, right-click pull-down menu, and drag-and-drop capabilities.

Advanced integration with Microsoft Windows through the Oracle Drive Windows plug-in permits users to seamlessly access Oracle Content DB content and functionality directly from Windows Explorer. Finally, Oracle Drive features offline content management capabilities, allowing users to access and manage content easily when they are disconnected from the network.

The security model within Oracle Content DB is extraordinarily flexible, with fine-grained permissions and access specified at the folder and document level. Role-based access control includes a rich set of out-of-the-box roles and also permits customer-defined roles to be built up from the basic permissions. Finally, access management is available through groups as well as individual users.

Behaviors controlling management of the information lifecycle such as versioning, attribution, and records management can be specified through policies established on a folder-by-folder basis. Versioning can be applied to content automatically based on a specific event such as check-in, applied manually by users, or disabled. The maximum number of versions to retain before archiving can also be specified.

Category attributes can similarly be applied manually or automatically. Individual Category attributes can be given default values and mandatory user entry of values can be specified. Records management is managed through policies on a folder-by-folder basis. While Content DB policies will normally be managed by a small subset of users with specific administrative permissions, the tools for setting up policies are straightforward and consistent, allowing use of a delegated policy management model as dictated by the needs of the business.

For more information, visit the [Oracle Content Database](http://otn.oracle.com) page on OTN.

**Discussions**

Another WebCenter Web 2.0 service is discussion forums, which promote a sense of community and allow members to share and discuss information and opinions. Discussion forums are ideal for broadcasting thoughts to a wide audience or in situations where not all participants are logged on simultaneously. WebCenter Suite discussion forums are powered by the award-winning Jive Software Forums technology. WebCenter Discussions is an open-architecture, standards-based discussion forum solution that provides several powerful features to boost community participation, facilitate question resolution, and promote knowledge reuse.
Developing Applications Embedding WebCenter Discussions

Oracle WebCenter Discussions has a rich set of APIs that allow developers to integrate Jive Forums directly into their JSF applications. The following is a broad overview of the major components in the system:

- **Skin**: The skin is the front end to the system, usually written in JSP or JSF. The skin is responsible for calling the APIs and displaying the results to users in HTML (or some other format). The default skin includes many features that can be configured through the administration tool, so most developers find that making small changes to the default skin will suffice. If more comprehensive UI or feature changes are required, a totally new skin can be created.

- **Filters**: Filters dynamically reformat the contents of messages to provide additional functionality. For example, the URL filter turns any URL that a user types in a message into a clickable link. Another filter syntax highlights Java source code. Filters are applied dynamically, and any number of custom filters can be created and installed.

- **Core API**: The core API provides all the business logic for creating, manipulating, and displaying discussion content. The core API is divided into a few different sections:
  - **Security**: Security through permissions checking is implemented at the method level via methods that throw an UnauthorizedException. In this case, users that call the method without proper permissions will generate the exception.
  - **Extended Properties**: Extended properties are used to store additional data about objects. Almost all major objects can have extended properties, including categories, forums, threads, messages, attachments, users, and groups. The core APIs include a set of methods to add, set, get, delete, and iterate these extended properties.
  - **Internationalization**: Internationalization is the process of changing the format of dates and numbers to obey rules unique to the user's region and translating user-interface text to their language. This includes character encoding, handling international characters, translating the user interface, and testing the resource bundles that modified.
  - **Authentication**: By default, Web Center Discussions calls a module that uses LDAP for authentication and accessing user data. You can configure the system to perform authentication and load user data from any other external system by implementing a few of the forums interfaces.

- **Back End**: A database is used to store all data. By default, it uses an Oracle database. Authentication and user and group storage is pluggable so that alternative back ends can be used as well.
Key Features

Key features of WebCenter Discussion’s architecture include:

- **Open Standards:** As a pure J2EE application, WebCenter Discussions is deployable on a wide variety of application servers including Oracle Application Server. The open database schema is available for many different databases, including MySQL and Oracle. Out-of-the-box support for LDAP or Active Directory authentication and user data integration is included.

- **Easily Customizable:** WebCenter Discussions are designed to be easily customized for your own deployment. Plug in your own authentication system, modify the user interface, use RSS or XML to access the data, or add your own processing logic using the full API.

- **Scalable and Fault Tolerant:** The server-side Java environment and extensive caching make WebCenter Discussions extremely fast. Leveraging clustering support in Oracle Application Server means the application can be deployed on multiple servers for scalability and fault tolerance.

- **Secure:** WebCenter Discussion applications have been proven secure by thousands of customer installations. The administration console includes role-based access rights and permission checking done at the Java method level to guarantee that users have access only to the content they should.

Key features of WebCenter Discussion’s functionality include:

- **Simple and Powerful Interface:** The interface is easy to use with a consistent look and feel and one-click access to commonly used tasks.

- **Search:** WebCenter Discussion forums can be crawled by SES, and search submission and results can be surfaced in WebCenter applications.

- **Expert Search:** Community members can locate other members by searching through basic information such as name or email address, or perform more complex searches through extended profile fields.

- **Threaded, Flat, and Tree Mode:** Messages can be viewed in several different modes: flat mode lists messages in chronological order, threaded messages are grouped by a specific subject, and tree mode shows one message per page with a tree structure for other messages in that topic.

- **Email Notifications:** Users are notified by email or the Web when information is updated.

- **Web-Based Administration Tool:** Administrators can easily delete, edit, and move messages; manage permission-levels; and organize the hierarchy of categories and forums.

- **Permissions-Based Security:** You can restrict users or groups from reading forums, creating new topics, or posting messages.
• **Moderation Support:** Moderators can screen and edit messages before they are posted.

• **Role-Based Administration:** Use role-based administration to designate multiple users or groups as system administrators, forum administrators, or moderators.

**Note:** This feature will be released shortly after general availability of Oracle WebCenter Suite 10g Release3.

**Wiki**

A *wiki* is a small piece of server software that lets teams freely create and edit Web content using any Web browser. With wikis, users can access the information from anywhere and can edit it from anywhere. The standards-based, open-source wiki server that is included with Oracle WebCenter brings wiki capability to WebCenter applications. WebCenter Wiki makes it easy for your team to collaborate and share knowledge.

Basic features of WebCenter Wiki include:

• **Page Versioning**: As contributing users make edits to a wiki page, the wiki server keeps track of the previous changes. This allows users to audit the changes and track how their pages evolve.

• **Viewing and Restoring Versions**: After a change has been made, a user may need to revert back to a previous version. WebCenter’s wiki server allows for users to preview the previous version of the page and then restore it, if required.

• **Search Integration**: All wiki pages can be searched via SES and the crawler-based approach. In addition, WebCenter’s wiki server exposes a set of open APIs to give developers the means to integrate any search engine to search the pages.

• **Page Locking**: When a user is editing a page, other users are able to view, but not edit, the same page. When the first user has finished with the edits, another user is allowed to edit the page. This prevents users from overwriting each other’s updates.

• **Page Reference/Breadcrumbs**: As more and more users create wiki pages, WebCenter Wiki provides an easy-to-navigate way of traversing pages and uses a breadcrumb model to indicate which pages have been visited.

• **List of Most Recent Updates**: As wiki pages are added and updated, users can go to one place to see a listing of the most recently changed pages.

**Note:** This feature will be released shortly after general availability of Oracle WebCenter Suite 10g Release3.
SECURING THE WEBCENTER APPLICATION

Traditionally, J2EE applications are secured using container-based security constraints. These effectively map the path to a Web resource (a page) to a static role. Given that the mapping of these roles is defined within the deployment descriptor of the application, standard container security is extremely limited and inflexible. By their very nature, WebCenter applications are dynamic, runtime driven, and often involve input from the users with different levels of access. Therefore, the use of traditional J2EE container-based security is too static and limited in scope. ADF security addresses these limitations by introducing a declarative, dynamic security model based on the Java Authentication and Authorization Service (JAAS).

Oracle ADF implements JAAS through integration with JAZN (Oracle's JAAS provider). Although JAAS was designed to support both authentication and authorization, it has, to date, required custom code to implement the authorization component of this service. ADF Security overcomes the hurdles to the uptake of JAAS-based authorization model by allowing for a fully declarative, dynamic JAAS implementation.

With ADF Security, the access policy is defined by the granting of a specific activity (such as view or customize) rather than a static association of role membership and Web resource (security constraint). While the required actions are deployed with the application, the users and roles granted these actions are mapped dynamically at runtime. Hence, changes in the security profile information (such as the post-deployment addition of a new role) are automatically applied without a need to redeploy the application. Furthermore, given that ADF Security is not constrained by the URL to a Web resource, you can declaratively define more granular permissions within your application.

Authentication

ADF Security extends the default authentication model of the J2EE container by supporting both an implicit and explicit authentication process. That is, as with a standard security constraint, accessing a secured resource automatically results in an authentication challenge if the user is not currently authenticated. In addition, ADF Security introduces a known authentication “end point”, allowing for the definition of standard login and logout links. This, along with the concept of public pages (described below), allows an end user to access public pages in a secured application without being forced to log in. A previously authenticated user can explicitly log out and still remain in the application (a concept not found in J2EE container security).

Implicit Authentication

When a user first accesses an application secured by ADF Security, the security processes interrogate the subject (the container that represents the user) and determine the user’s associated principals (user principal and role principals). If there is no subject—because the user is completely new—ADF Security creates a
subject containing the anonymous user principal and the anyone role principal. The addition of these anonymous principals allows the unauthenticated user to access Web resources that are to be considered “public.”

**Figure 11 - ADF Security Implicit Authentication**

**Explicit Authentication**

In an explicit authentication scenario, an unauthenticated user (that is, a user with only the anonymous user principal and anyone role) actively clicks a Login link located on a public page. The Login link contains a direct request to the ADF Authentication Servlet, which is a component of the ADF security infrastructure. This servlet acts as a known end point for a standard container-secured Web resource; that is, the Authentication Servlet is secured via a J2EE security constraint. Hence, an unauthenticated user attempting to access this servlet is immediately directed to the container’s internal authentication logic. Given the container will only redirect back to the application secured by the constraint (in this case, the servlet, not the original page being accessed by the user), the current “public” page is passed as a parameter on the request to the adfAuthentication Servlet. This allows it subsequently to redirect back to the page on which the Login link was located. Assuming a successful authentication, the user subject now
contains its own user principal and role principals as well as the original anyone role principal.

Figure 12 - ADF Security Explicit Authentication

Authorization

The Authorization Editor exposes enterprise roles that are defined in the policy store and displays the actions (such as View, Invoke, and Edit) that may be applied to a specific component type (a page, a method, an iterator, or an attribute). To implement an authorization policy, the desired action is checked against the associated role. This is subsequently written to the policy store defined for the application, which is either a file-based repository or an LDAP directory. The policy itself defines a permission type, the resource that is secured, the actions that can be performed against that resource, and the user or role to whom that policy is being assigned or granted.
At runtime, when the user attempts to access a secured page (such as mypage.jspx in figure 12) the Oracle ADF security filters intercept the request to determine whether the current user has appropriate access to that page.

In the unauthenticated user case, the anonymous user principal and the anyone role principal within the subject are used to define the user. Hence, if view privilege on the requested page had been granted to the anyone role, (such as the public.jsp page in figure 12) the user would be forwarded to that page.

Since mypage.jspx requires that the view privilege be assigned to a role other than anyone, the user is challenged to authenticate. After authentication, the user has a specific principal, which can be used to evaluate the security policy (that is, determine which role is required to view mypage.jspx and whether the requesting user is a member of that role). In this example, if the user were a member of the Staff role, then they would be allowed to navigate to mypage.jspx.

Similarly, for users trying to access a page on which they do not have the view privilege (secpage.jsp in Figure 12), access is simply denied.
DEPLOYING THE WEBCENTER APPLICATION

In terms of deployment, WebCenter applications behave exactly like traditional J2EE applications except for the fact that, in addition to the deployed application code, they include any customizations applied to the application at runtime. Therefore, simply creating and deploying a packed EAR file to the runtime environment is not enough, as this would not carry over the customizations (both for the page and any portlets within the page). To address this additional complexity, the WebCenter Framework comes with the Lifecycle Tool to deploy the associated customizations along with the application itself. The Lifecycle Tool effectively automates the process of handling customizations, freeing up developers and administrators to focus on the overall deployment.
While the steps required to deploy the application to production will differ based on the desired topology (development to test to production, develop to production, and so on) and the point in the lifecycle where customization is applied, the general lifecycle of a WebCenter Application is as follows:

1. The application is developed using the WebCenter template, which specifies the packaging of the application code, runtime customizations, and configuration and connection details for any WebCenter services used by the application into a generic EAR file.
2. After the generic EAR file is transferred to the deployment server, the Lifecycle Tool produces a targeted EAR file, which is specific for the server on which the application is going to run.
3. The targeted EAR file is then deployed to a stand-alone OC4J or a full Oracle Application Server environment, using either Enterprise Manager or the command-line interface.

The Lifecycle Tool

In order to account for the various methods of deployment used by J2EE developers, the lifecycle tool is exposed in three separate ways:

- **The Embedded Lifecycle Tool**, which is embedded within the developer framework to allow one-click deployments from JDeveloper. Because the configuration information of the deployed application is assumed to match that of the developer’s machine, the developer is not asked for new configuration information.

- **Command Line Lifecycle Tool**, which is a stand-alone version of the tool intended for use on the production server. This interactive tool walks the administrator through the connections and rewire the application as required. It also supports the recording of the administrator’s responses for subsequent deployments.

- **ANT Tasks**, which incorporates the Lifecycle Tool into ANT scripts, exposing WebCenter Lifecycle Tool’s functionality through a number of ANT Tasks. These correspond to the various functions seen in the command-line version of the tool.

Migrating Security and Application Roles

The WebCenter Lifecycle tool does not migrate the security policies used by the application. Instead, this function is performed by the OracleAS JAAS Provider migration tool (also known as JAZN Migration tool), which migrates the policy data from the development environment’s XML-based provider (system-jazn-data.xml) to the policy store in the runtime environment (either an XML-based provider or an LDAP-based provider, specifically, OID). These providers function as repositories for realm, policy, and login module information.
CONCLUSION

Oracle WebCenter Suite is an exciting new offering from Oracle. It brings together the power of standards-based development and portals to provide context-rich applications that dramatically improve efficiency. It provides the natural user interaction environment for your SOA applications and allows you to utilize a wide range of services including structured and unstructured content, business intelligence, business processes, communication, and collaboration to create better, more effective user experiences. And, because Oracle WebCenter Suite is a core component of Oracle Fusion Applications, the applications you build with the suite will seamlessly blend with both the current and future applications from Oracle.