BEA WebLogic Workshop™ Help

Version 8.1 SP4
December 2004

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Table of Contents

Securing Portal Applications

This topic provides a basic overview of WebLogic Portal security. The other portal topics, listed in Topics Included in this Section, provide implementation instructions.

WebLogic Portal uses the underlying WebLogic Server security architecture to let you create secure portal applications. The ultimate goal of portal security is to restrict access to portal resources and administrative functions to only those users who need access to those resources and functions.

Topics Included in This Section

Overview

Provides overview information on authentication, user and group management, authorization, and WebLogic Portal security management tools and resources.

Portal Security Scenario

Provides a scenario describing a fictitious company's portal application.

Implementing the Portal Security Scenario

Uses the portal scenario to identify the security touch points and link to implementation information.

Implementing Authentication

Provides details on the authentication examples contained in the Tutorial Portal.

Using Multiple Authentication Providers in Portal Development

Describes how to develop applications with users stored in external authentication providers.

How WebLogic Portal Uses the WebLogic Server Security Framework

Discusses WebLogic portal support and limitations for working with multiple authentication providers.

Overview

Note: Implementing security in a portal requires a basic understanding of standard security concepts, many of which are outside the scope of WebLogic documentation; for example, encryption, injection of SQL statements at login, and secure socket layers (SSL). The Related Topics section contains, among other things, links to information that will help give you a broader, more complete view of security and the issues surrounding it.

WebLogic Portal provides built−in functionality for authentication ("Who are you?") and authorization ("What can you access?").

Authentication

WebLogic Portal provides many authentication samples that you can incorporate into your portals. WebLogic Portal also provides many tools for user/group management.

Samples

Implementing Authentication contains details about the authentication examples contained in the Tutorial Portal.

WebLogic Portal also provides two sample login portlets you can reuse in your portals to authenticate WebLogic users:

- Login to Portal portlet − Provides basic login functionality.
- Login Director portlet − Provides login and shows the user the first desktop to which he is entitled.

You can also build other types of authentication supported by WebLogic Server.

User/Group Management

The WebLogic Administration Portal provides tools for managing users, groups, and setting user/group properties. For information on managing users and groups, see:

- Creating User Proflie Properties
- User/Group Management JSP Tags
- Using Portal Controls
- WebLogic Portal Javadoc
- The WebLogic Administration Portal help system

Authorization

There are three fundamental categories of things that can be secured in portals:

- The WebLogic Administration Portal
- Portal Resources
- Java 2 Enterprise Edition (J2EE) Resources

Using the WebLogic Server concept of security roles, WebLogic Portal lets you dynamically match users to roles at login. Different roles are, in turn, assigned to different portal resources, administrative tools, and J2EE resources so users can access only the resources and tools that their assigned roles allow.

WebLogic Administration Portal

The WebLogic Administration Portal provides the tools for managing users, portal delegated administration roles, visitor entitlement roles, interaction management rules, content management, and portal resources.

You can lock down the WebLogic administration portal with ***delegated administration***, which provides secure administrative access to the WebLogic Administration Portal tools. Delegated administration security is based on the delegated administration roles you create.

Portal Resources

The WebLogic Workshop Portal Extensions and the WebLogic Administration Portal provide tools for creating and managing portals, desktops, shells, books, pages, layouts, look & feels, and portlets. You can control access to portal resources for two types of users: administrators and visitors.

Administrators − You can control the portal resources that can be managed by portal administrators using ***delegated administration***.

Visitors − You can control visitor access to portals and portal resources with ***visitor entitlements***. Visitor entitlements are based on the visitor entitlement roles you create.

J2EE Resources

J2EE resources are the application framework and logic (Web applications, JSPs, EJBs, and so on) for which you can control visitor access. Security on J2EE resources is based on global security roles set up in WebLogic Server and applied to the individual J2EE resources. Security roles for J2EE resources are different than security roles that users can belong to, though both types of roles use the same roles architecture.

Default Users

The portal sample domain <BEA_HOME>\<WEBLOGIC_HOME>\samples\domains\portal and any portal domain you create with the Configuration Wizard include the following default users. You can add these usernames and passwords to your existing domains.

Username	***Password***	***Belongs to these groups***
weblogic ***Note***: This is the username for the portal sample domain. In a new portal domain created with the Configuration Wizard, this can be whatever was used for the primary system administrator.	weblogic ***Note***: This is the password for the portal sample domain. In a new portal domain created with the Configuration Wizard, this can be whatever was used for the primary system administrator.	Administrators PortalSystemAdministrators
portaladmin This is a default user for managing and setting up delegated administration on portals. If	portaladmin	Administrators PortalSystemAdministrators

your domain does not contain portals, you can safely delete this user.		

Related Topics

WebLogic Server Security

The Open Web Application Security Project (OWASP)

Setting up Unified User Profiles

Creating User Profile Properties

Using Portal Controls (for user/group management)

User/Group Management JSP Tags

For details on managing users and groups, see the WebLogic Administration portal online help system, also available on e−docs.

Portal Security Scenario

The following scenario describes the portal implementation needs of a fictitious company called Avitek, many of which involve security considerations. The topic that follows, Implementing the Portal Security Scenario, describes the security touch points in the scenario and provides links to implementation information.

Avitek needs two types of portal−based Web presence: an internal site for its employees and partners called "Inweb," and a public portal for its customers called "Outweb." It needs authentication for both sites. Inweb must live behind a firewall.

Outweb is set up on a server cluster for load balancing and failover.

For Inweb, Avitek needs to cater to three different types of users: managers, regular employees, and partners.

For the three types of users, Avitek wants to create only two portals: one for managers and employees and one for partners. Since there are five different partners, each partner must have a separate view of Inweb.

Some of the partners also perform contract work for Avitek, so they must also be able to access the employee portal desktop.

Avitek wants all Inweb users to authenticate before seeing any view of the portals.

For Outweb, Avitek provides information and services on a subscription basis, so it wants to provide a portal that lets all users see unsecured company information and log in to see secure information.

Avitek has a staff of two to administer all portals, and it wants to grant limited administrative access to certain partners to let them maintain their partner portal.

There are two JSP−based administration portlets that can never be seen by anyone other than Avitek's in−house administrators.

Avitek also wants to use its existing content management system for delivering content to its portals. The content management system vendor has created an interface to connect to BEA's Virtual Content Repository.

Avitek will use two user databases: The Intranet site will use an existing user database, and the public site will use the default WebLogic Server LDAP user database and is gradually adding users to it.

Related Topics

Implementing the Portal Security Scenario

Securing Portal Applications

Implementing the Portal Security Scenario

This topic walks you through the Portal Security Scenario, highlights the security touch points in the scenario, and points to the tools and information for implementing security.

Scenario	***Security Touch Points***
Avitek needs two types of portal−based Web presence: an internal site for its employees and partners called "Inweb," and a public portal for its customers called "Outweb." It needs authentication for both sites. Inweb must live behind a firewall. Outweb is set up on a server cluster for load balancing and failover.	Because one site must reside behind a firewall and the other outside the firewall, two servers are needed. • See Using WebLogic Server Clusters in the WebLogic Server documentation, especially the section on Security Options for Cluster Architectures.
For Inweb, Avitek needs to cater to three different types of users: managers, regular employees, and partners.	The three different audiences can be identified by their user profile settings. For example, you can create a user profile property called "user_type" that has three possible values: manager, employee, and partner. Each user is assigned an appropriate value. While user profile properties are not a direct security feature, you can use the properties to define the roles that determine secure access to resources. • See Creating User Profile Properties in this help system. • For information on creating visitor entitlement roles and mapping those roles to portal resources to secure the resources, see the WebLogic Administration help system. Users will use VPN to gain access through the firewall.

For the three types of users, Avitek wants to create only two portals: one for managers and employees and one for partners. Since there are five different partners, each partner must have a separate view of Inweb.	This requirement is easily solved by the flexibility of the WebLogic Portal architecture. When you create a portal file with the WebLogic Workshop Portal Extensions, you can add books, pages, and portlets to that file. The portal file serves as a template with which portal administrators can create multiple portal instances, or desktops. Each desktop can contain any combination of books, pages, and portlets provided by the template, and those portal resource instances can be assigned their own visitor entitlement and delegated administration roles for security. To meet this requirement, then, only one portal is required. That portal can be used as a template to create separate desktops for managers, employees, and for each of the five partners. When any user logs in, the system knows which type of user they are (based on user profile properties), and they are shown only the desktop(s) they can access (based on visitor entitlement and delegated administration roles that are tied to the user profile properties). There are two other options for meeting this requirement: creating two portal files within a single Web application (project), or creating two projects with a portal in each. A decision to create two portal files is simply an organizational decision. Both portal files serve as templates for portal administrators to construct portal desktops, and the same mechanisms for applying visitor entitlements and delegated administration apply. If Avitek created separate projects for the portals (Web applications), they could secure the J2EE resources (such as JSPs) in each separately, since Web applications have separate security scopes in WebLogic Server. However, since you can secure individual resources in a single Web application, you can secure J2EE resources for different audiences in that application using WebLogic Server security. Using separate Web applications for each portal means Avitek might have to implement single

	sign−on as a convenience for partners who can access the employee portal and employees who can access the partner portal. Users will be able to see all desktops they are entitled to in both Web applications. • See Assembling Portal Applications in this help system for instructions on creating portals. • See Securing WebLogic Resources in the WebLogic Server documentation, especially the sections dealing with securing URL (Web) resources such as JSPs and Enterprise Java Beans (EJBs). • The Portal Samples contain an example implementation of single sign−on.
Some of the partners also perform contract work for Avitek, so they must also be able to access the employee portal desktop.	Because you need to identify someone by certain characteristics, you could again use a user profile property to identify a user as both a partner and an employee. Instead of making the property a "choose one value" type (single, restricted), you could make the property a "choose multiple values" type (multiple, restricted). You can handle security access to portal desktops and other resources with visitor entitlements. If the employee and partner portals are located in separate Web applications, you can provide single sign−on for partners as a convenience, then handle security access to portal desktops and other resources with visitor entitlements. • See Creating User Profile Properties in this help system. • For information on creating visitor entitlement roles and mapping those roles to portal resources to secure the resources, see the WebLogic Administration help system. • The Portal Samples, especially the Tutorial Portal, contain example implementations of authentication, including single sign−on.
Avitek wants all Inweb users to authenticate before seeing any view of the portals.	You can set up the Intranet portal to use a front−end login JSP. After successful login, users are taken to the portal desktop to which they have access. • Designation of a login JSP occurs in the WebLogic Administration Portal. When

	creating a portal, enter the name of the login JSP file in the optional ***Portal URI*** field of the portal properties window. For information on creating portals, see the WebLogic Administration Portal help system. • You can also secure J2EE resources, including specific URL patterns, using your application's deployment descriptors. See Securing WebLogic Resources in the WebLogic Server documentation, especially the sections dealing with securing URL (Web) resources. • The Portal Samples, especially the Tutorial Portal, contain example implementations of authentication.
For Outweb, Avitek provides information and services on a subscription basis, so it wants to provide a portal that lets all users see unsecured company information and log in to see secure information.	Unlike the previous requirement in the scenario where a separate login JSP was required to access the portal, this requirement lets users access a portal without authenticating. The only time they must authenticate is when they want to view the protected information. Providing authentication based on whether or not users are subscribers is another instance where user properties are useful. For example, you could create a "subscriber" property and set it to "true" or "false." You could create a "subscriber" role that allows only subscribers view protected information at login. A best practices way of providing secure information is by putting secure portlets on a dedicated page. The page itself could even be secured (entitled). • See Creating User Profile Properties in this help system. • The Portal Samples contain a Login to Portal Portlet you can reuse in your portals. The samples also contain example implementations of authentication. • For information on creating pages and adding portlets to them, see Assembling Portal Applications in this help system and the Portal Management topics in the WebLogic Administration Portal help system.

Avitek has a staff of two to administer all portals, and it wants to grant limited administrative access to certain partners to let them maintain their partner portal.	Delegated administration for WebLogic Portal is set up in the WebLogic Administration Portal. A system administrator or super portal administrator can set up other administrators and delegate different levels of access to them. Delegated administration for tools and portal resources in the WebLogic Administration Portal can be defined by roles, users, and groups. In this part of the scenario you could create an administrator role based on user properties and define delegated administration accordingly. Or you could create two groups: "local administrators" and "partner administrators," add users to those groups, and set up delegated administration with those groups. With delegated administration roles set up, you can apply those roles to individual portal resources, giving the staff administrators full access and the partner administrators access to only their portal resources. • For information on creating users and groups and setting up delegated administration, see the WebLogic Administration Portal help system. • See Creating User Profile Properties in this help system.
There are two JSP−based administration portlets that can never be seen by anyone other than Avitek's in−house administrators.	Securing portlets is simple, straightforward, and powerful using visitor entitlements. In this requirement of the scenario, there may be a need for backup security assurance. This can be accomplished by securing the JSPs in question with WebLogic Server security policies. Security policies are server−level global roles that are applied to J2EE resources. • See Securing WebLogic Resources in the WebLogic Server documentation, especially the sections dealing with securing URL (Web) resources such as JSPs and Enterprise Java Beans (EJBs).
Avitek also wants to use its existing content management system for delivering content to its portals. The content management system vendor has created an interface to connect to BEA's Virtual	By adding compatible third−party content management systems into the Virtual Content Repository, content security in those third−party systems is maintained in the Virtual Content Repository.

Content Repository.	WebLogic Portal's Virtual Content Repository also provides limited content security beyond the security provided by compatible third−party content management systems. • See To control user access in the My Content Portlet topic. • See the Content Management topics in the WebLogic Administration Portal help system.
Avitek will use two user databases: The Intranet site will use an existing user database, and the public site will use the default WebLogic Server LDAP user database and is gradually adding users to it.	Inweb − Uses existing RDBMSRealm in the existing domain. This can remain the authentication provider/user database or Avitek can migrate the user database to the WebLogic Server LDAP user database. Outweb − Uses WebLogic Server's default LDAP user database. WebLogic Server's default LDAP user database provides all the power and functionality of the WebLogic Server security architecture. • See Managing WebLogic Security in the WebLogic Server documentation.

Summary

Following is a summary of the configuration Avitek will use for its Inweb and Outweb portal sites:

Inweb	***Outweb***
Inweb can experience minor down time, so it can be set up on a single server. Because there are many existing internal users Avitek wants to retain, it will use its existing RDBMSRealm for user storage and authentication. Inweb is set up behind a firewall, so users will use VPN to gain access from outside the firewall. Because of the flexibility of the portal framework, only one portal is needed. Multiple desktops can be created and entitled based on that single portal. Avitek will use BEA's Virtual Content	Outweb cannot experience down time, so it must be set up on a cluster. It will use the default WebLogic LDAP user database and authentication provider. Because of the flexibility of the portal framework, only one portal is needed. Multiple desktops can be created and entitled based on that single portal. Avitek will use BEA's Virtual Content Repository to connect to its BEA−compatible third−party content management system.

Repository to connect to its BEA−compatible third−party content management system.	

Samples

Portal Samples

Login to Portal Portlet

Related Topics

Securing Portal Applications

Portal Security Scenario

Implementing Authentication

WebLogic Portal provides many different ways to implement user login and authentication against any available authentication providers. The following sections provide nine authentication samples to help you better understand the choices you have in implementing authentication. These sections are taken directly from the samples in the Tutorial Portal.

Note: This topic describes how to implement authentication after authentication providers have been configured for use with WebLogic Server. For information on setting up authentication providers, see Using Multiple Authentication Providers In Portal Development.

In these samples, the portal Web project root is <WEBLOGIC_HOME>/samples/portal/portalApp/tutorial. Paths in the samples are relative to this root. All resources and configuration for these samples is included in the tutorial portal Web project which uses the <WEBLOGIC_HOME>/samples/domains/portal/config.xml server (called portalServer). To use these samples in your own domains and portal Web projects, import or mimic the files and configurations used in the samples.

This topic includes the following samples:

1. Form Based
2. Client Certificate
3. Backing File
4. Multi−Page User Registration Using Page Flow
5. Single Sign−on Within WebLogic with a Second Application
6. Auto Login
7. Basic Authentication
8. Portal Access that Requires Login
9. Perimeter Login
10. User Login Control

1. Form Based

Source Location: /portlets/login/formLogin/

The example /WEB−INF/web.xml specifies a CONFIDENTIAL transport−guarantee so the /portlets/login/formLogin/login_link.jsp must build a HTTPS URL to access the redirect.jsp. The redirect.jsp simply redirects back to the portal. This example leaves the protocol in HTTPS, but you could switch back to HTTP in redirect.jsp if you only wanted HTTPS to protect your username/password during login.

Note: The <form−login−page> URLs specified in web.xml will be different based on whether you are running the portal from a .portal file in WebLogic Workshop or assembled from the database when you create a desktop in the WebLogic Administration Portal. For example, when running a file based portal such as sample.portal (in development) the <form−login−page> element might be specified with /samplel.portal?_nfpb=true&_pageLabel=login. When running an assembled portal (in production) the <form−login−page> element might be specified with /appmanager/samplePortal/sampleDesktop?_nfpb=true&_pageLabel=login.

2. Client Certificate

Source Location: /portlets/login/clientCert/

These are the steps for using client certificate authorization:

1. Comment out the FORM or BASIC <login−config> in /WEB−INF/web.xml and uncomment the CLIENT−CERT <login−config>. This is necessary because the Web application can have only a single login−config.
2. Next you can do one of 2 things:
   - ♦ Import the democlient−cert.p12 client certificate into your browser, located in /portlets/login/clientCert/.

     or
   - ♦ Generate your own certificate using openssl.

   Note: democlient−cert.p12 was created for demonstration purposes and is not meant for production use. If you choose to generate your own certificate using openssl, they have instructions at their Web site at www.openssl.org.
3. If you import democlient−cert.p12, the following is for importing into IE version 6:
   a. Double−click the democlient−cert.p12 file.
   b. Click ***Next*** when the Certificate Import Wizard appears.
   c. democlient−cert.p12 should be displayed in the filename textbox. Click ***Next***.
   d. Do not type a password for the private key. Click ***Next***.
   e. You can select a certificate store or not. Click ***Next***.
   f. Click ***Finish***.

The following steps are for configuring WebLogic Server to use SSL and the democlient−cert correctly.

4. With a running portalServer, open the WebLogic Administration Console (http://<server>:<port>/console).
5. Using the tree view pane, navigate to ***Security*** > ***Realms*** > ***realmName*** > ***Providers*** > ***Authentication*** > ***DefaultIdentityAsserter***.
6. In the User Name Mapper Class Name textbox, enter examples.login.ExampleUserNameMapper.
7. Move the X.509 certificate type to the Chosen box and click ***Apply***.
8. ***Navigate to Security*** > ***Realms*** > ***realmName*** > ***Users*** and create a new user called support with password of password.
9. Navigate to ***Servers*** > ***portalServer*** and click the ***Keystores and SSL*** tab.
10. Click the ***Show*** link for the Advanced Options at the bottom of the page.
11. Select ***Client Certs Requested But Not Enforced*** from the Two Way Client Cert Behavior drop down (or enforced depending on the desired behavior) and click ***Apply***.
12. Add the examples.login.ExampleUserNameMapper.class to the system classpath. This can be accomplished by adding the class to netuix_system.jar. The ExampleUserNameMapper extracts the username from the e−mail of the Subject DN in the X.509 certificate. For example, the democlient−cert.p12 has a Subject DN with an e−mail of support@bea.com and the resulting username is "support". This is why the support user was added to the realm in the previous steps.

Because the form−based login example uses SSL, the one−way SSL was already configured for the server. If you need to enable client−certificate authentication for any server, it is a prerequisite to configure one−way SSL (see "Configuring SSL" in "Managing WebLogic Security" at http://e−docs.bea.com/wls/docs81/secmanage/ssl.html).

After you take these steps, you can access the portal and /portlets/login/formLogin/login_link.jsp to use your client certificate to get logged in to the portal Web application. See the form−based login example for an explanation of the use of the login link to access a protected resource.

3. Backing file

Source Location: /portlets/login/backingFileLogin/

This example uses portal personalization code and a backing file to log in (/WEB−INF/src/portlet/login/LoginBacking.java). The backing file also redirects back to the portal so that database state is not clobbered by control state.

4. Multi−Page User Registration Using Page Flow

Source Location: /portlets/login/pageflowLogin/

This example uses Java Page Flow to show how a multi−page user registration portlet can be accomplished. This example has 4 pages:

1. The first displays a simple user registration page.
2. The second page gathers more user information that could be stored in user properties (personalization code).
3. The third page will optionally authenticate the user or show a summary page (2 links).
4. The fourth page is either the logged in status of the user or the summary page.

5. Single Sign−on Within WebLogic with a Second Application

Source Location: /portlets/login/ssoLogin/

This is an example of single sign−on between two Web applications. For single sign−on to work, both Web applications must have matching cookie name entries in web.xml. Since by default WebLogic sets the cookie names to be identical if unspecified for a Web application, this behavior should work by default.

6. Auto Login

Source Location: /portlets/login/autologin/

Note: This example uses cookies, which is an insecure authentication method.

This example shows how to use cookies and encoding for autologin. When you login and select the "autologin" checkbox, it will encode your username and password. The username and password will be added as a cookie to the response for a life of 1 day. After this point, if you leave the portal and come back, you will be logged in automatically, including when you exit the browser. If you log out, the cookies will be deleted and you will no longer be automatically logged in when you revisit the portal.

This example uses a backing file (/WEB−INF/src/portlet/login/AutoLoginBacking.java).

7. Basic Authentication

Source Location: /portlets/login/basicLogin/

This example uses the same principles as the form−based login. To use basic authentication, simply uncomment the FORM or CLIENT−CERT based authentication methods in web.xml, and use the basic method of authentication. You can use one of the default users in the realm such as visitor1/password to log in.

8. Portal Access that Requires Login

Source Location: /portlets/login/loginRequiredPortal/

This example shows a portal that is only accessible after user authentication. To enable this, simply add a security constraint entry in web.xml for all URL resources. For Example:

```
<security−constraint>
    <web−resource−collection>
        <web−resource−name>login</web−resource−name>
        <description>Security constraint for the whole portal</description>
        <url−pattern>/*</url−pattern>
    </web−resource−collection>
    <auth−constraint>
        <description>all users</description>
        <role−name>AnonymousRole</role−name>
    </auth−constraint>
</security−constraint>
```

9. Perimeter Login

Source Location: /portlets/login/perimeterLogin/

See the following WebLogic Server documentation topics:

- "The Authentication Process" in "WebLogic Security Service Architecture" at http://e−docs.bea.com/wls/docs81/secintro/archtect.html.
- "Configuring Security Providers" at http://edocs/wls/docs81/secmanage/providers.html, especially the following sections:
    - ♦ Configuring a WebLogic Authentication Provider
    - ♦ Configuring a WebLogic Identity Assertion Provider

Related Topics

Tutorial Sample

How WebLogic Portal Uses the WebLogic Server Security Framework

How WebLogic Portal Uses the WebLogic Server Security Framework

When you build portal applications, you secure them using authentication (Who are you?) and authorization (What can you access?).

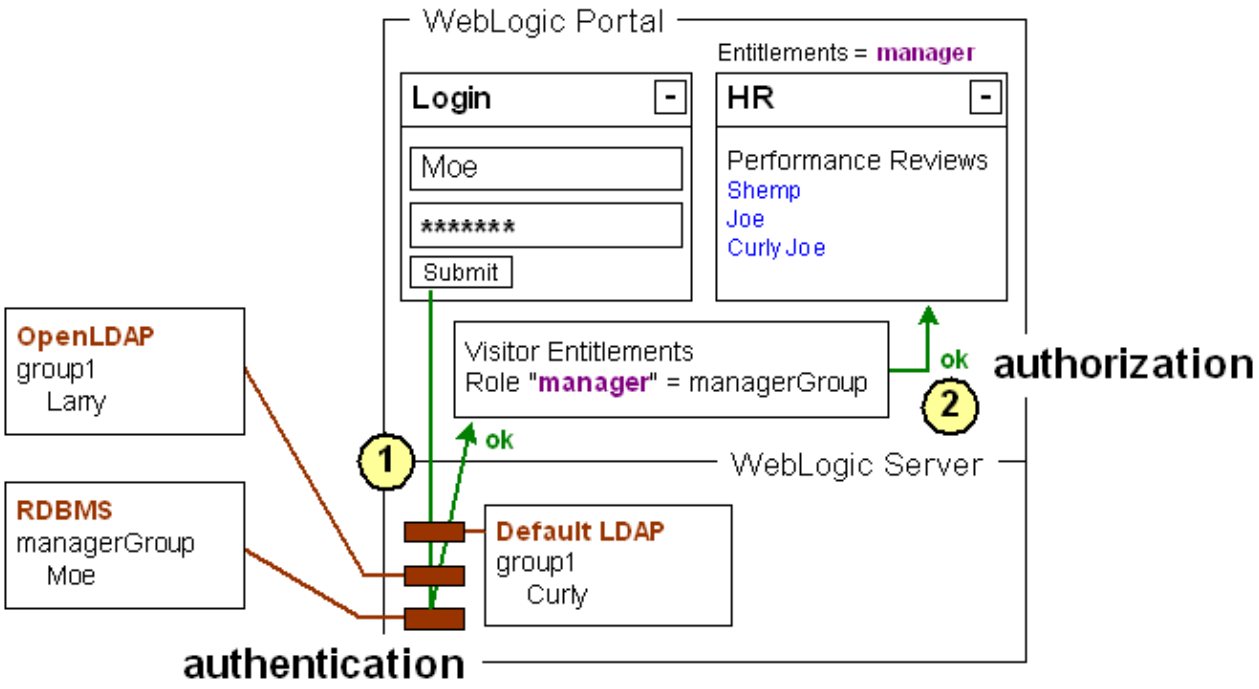
Authentication − WebLogic Portal uses WebLogic Server for login authentication, whether you are using only WebLogic Server's default LDAP user store, one or more external authentication providers configured for use with WebLogic Server, or both. The WebLogic Portal user and group management framework communicates with multiple authentication providers through WebLogic Server for basic user and group operations.

See: Using Multiple Authentication Providers in Portal Development | Implementing Authentication

Authorization − WebLogic Portal has its own authorization framework that lets you define roles for who can access which portal resources. You can define delegated administration roles for your portal administrators, and you can define entitlement roles for visitors to your portals. When a portal administrator logs in to the WebLogic Administration portal, the administrator sees only the areas he can administer. When a visitor logs in to a portal, the visitor sees only the books, pages, and portlets to which he is entitled.

See: Delegated Administration Overview | Overview of Visitor Entitlements

The following illustration shows authentication and authorization in more detail.



In this example, three authentication providers are being used by WebLogic Server: an OpenLDAP provider, an RDBMS (relational database) provider, and WebLogic Server's default LDAP provider. The two external authentication provider servers are running, and they have been added to WebLogic Server as authentication providers in the WebLogic Server Administration Console.

1	The user logging in is authenticated against all available authentication providers. Moe belongs to the RDBMS authentication provider and can log in successfully (unless authentication is set to "REQUIRED" on more than one provider).
2	On successful login to a portal, WebLogic Portal uses its DefaultRoleMapper to see if the user belongs to any delegated administration and visitor entitlement roles, and the user is granted access to only the resources he is allowed to access. The role called "manager" is defined so that anyone belonging to the group called "managerGroup" is part of that role. The HR portlet is set up so that only members of the "manager" role can view it.

If you are using more than the LDAP authentication provider supplied by WebLogic Server, see Using Multiple Authentication Providers in Portal Development.

Related Topics

Implementing Authentication

Using Multiple Authentication Providers in Portal Development

WebLogic Portal supports the use of multiple authentication providers in a Portal domain, which means that users in external providers can log in to your portal applications. It also means that in your code you have access to potentially multiple user stores.

In Portal Controls, JSP tags, and classes in the Portal API that deal with user and group management, you can specify the authentication provider you want. (Use the WebLogic Administration Portal to see a list of available authentication providers.)

For user and group management, the roles you specify in the WebLogic Administration Portal Authentication Security Provider Service determine who can perform certain management tasks in your portal applications.

Note: It is possible (but not recommended) to store an identical username or group name in more than one authentication provider. For example, user "foo" can reside in the default WebLogic Server LDAP provider and in an external RDBMS provider. In that case, WebLogic Portal uses only one user profile for user "foo."

If you are using an RDBMS authentication provider, be aware of case sensitivity when entering names for users and groups. For example, user "Bob" is different than user "bob."

Setting up Multiple Authentication Providers

For information on setting up and configuring multiple authentication providers, see Using Multiple Authentication Providers with WebLogic Portal in the WebLogic Administration Portal help system.

Related Topics

How WebLogic Portal Uses the WebLogic Server Security Framework

Unified User Profiles Overview

If you have an existing store of users, groups, and additional properties (such as address, e−mail address, phone number, and so on), unified user profiles are a necessary part of bringing those user properties into the WebLogic Portal environment, where they can be used for retrieving and editing property values and setting up personalization, delegated administration, and visitor entitlements.

This topic describes the unified user profile, when to use it, and when not to use it.

Note: This topic contains the terms "user store" and "data store." A user store can contain users and groups, as well as additional properties. A data store implies that the store does not have to contain users and groups. It can simply contain properties.

What is a Unified User Profile?

Here is an example that explains what a unified user profile is and does:

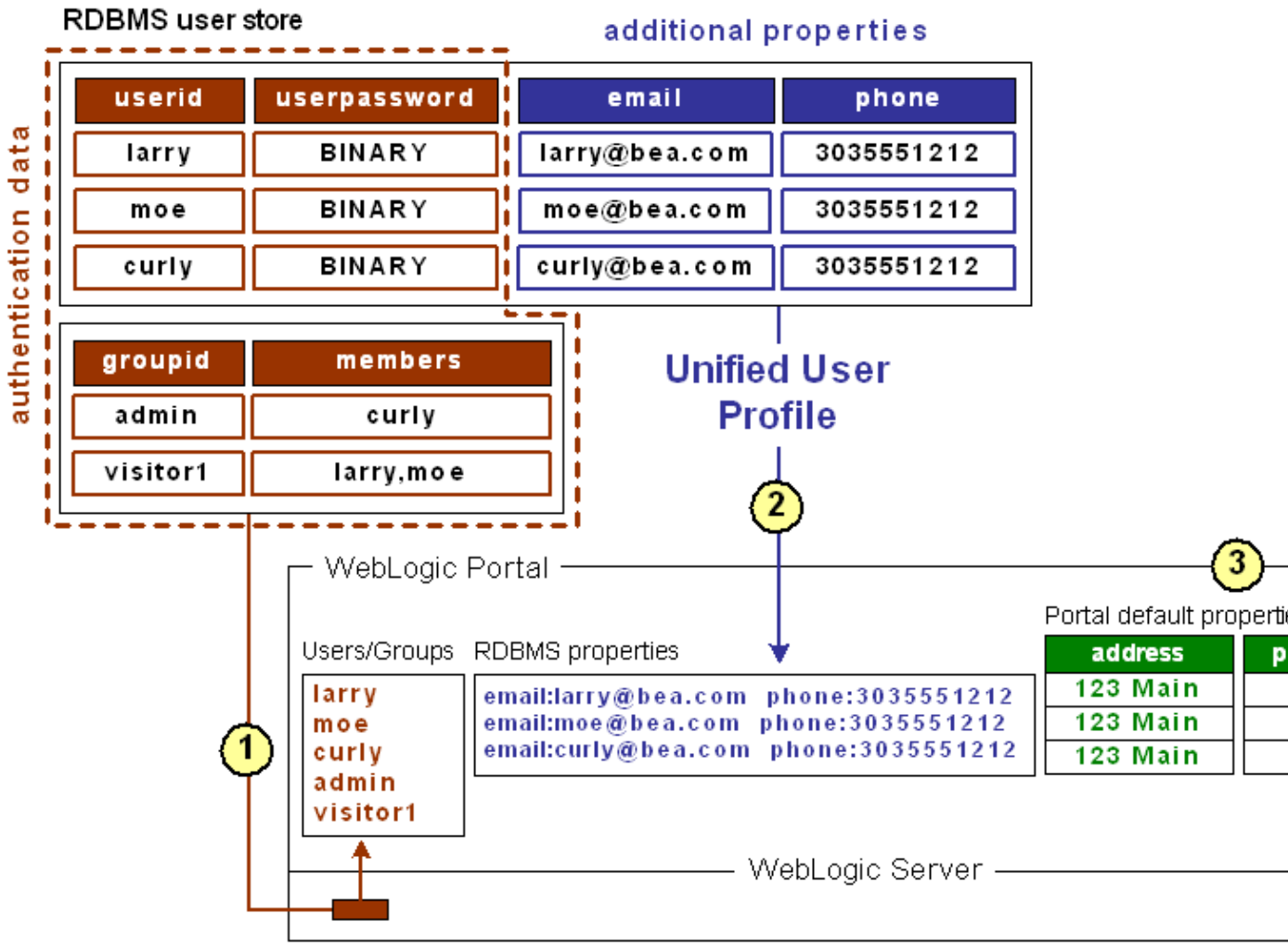
Let's say you're creating a new portal application that you want users to be able to log in to. Let's also say your users are stored in an RDBMS user store outside of the WebLogic environment. You could connect WebLogic Server (your portal application's domain server instance) to your RDBMS system, and your users could log in to your portal application as if their usernames and passwords were stored in WebLogic Server. If authentication was all you wanted to provide through your RDBMS user store, you could stop here without needing a unified user profile.

However, let's say you also stored e−mail and phone number information (properties) for users in your RDBMS user store, and you wanted to be able to access those properties in your portal applications. In this case, you need to create a unified user profile for your RDBMS user store that lets you access those additional properties from your code.

Technically speaking, a unified user profile is a stateless session bean you create (with associated classes) that lets WebLogic Portal read property values stored in external data stores, such as LDAP servers and databases. Once connected to an external data store with a unified user profile, you can use portal JSP tags, controls, and the WebLogic Portal API to retrieve user property values from that store. You can also take the extra step of surfacing these external properties in the WebLogic Administration portal, where the properties can be used to define rules for personalization, visitor entitlements, and delegated administration.

Whether or not you have additional properties stored in your external user store, the external users and groups you connect to WebLogic Server are automatically assigned the default user property values you have set up in WebLogic Portal, without the use of a unified user profile. With the WebLogic Administration Portal, you can change the default WebLogic Portal property values for those users. These values are stored in WebLogic Portal's RDBMS data store using the Portal schema.

The following figure shows where a unified user profile fits between an external user store and the WebLogic environment.



1	This external RDBMS user store, which supports authentication, contains users (principals) and passwords in one database table and groups (principals) in another. Giving a user store authentication capabilities (as an authentication provider or identity asserter) involves configuration steps not associated with the unified user profile configuration process. (See Developing Security Providers for WebLogic Server.) Unified user profile configuration is not dependent on the authentication provider configuration and vice versa. Once the RDBMS authentication provider is connected to WebLogic Server, WebLogic Server (and WebLogic Portal) can see those users and groups. Those users can log in to your portal applications, and you can include those users and groups in your rules for personalization, delegated administration, and visitor entitlements. Also, WebLogic Portal's ProfileWrapper maps the principals to properties kept in the Portal schema, thereby establishing the user profile.
2	***Unified User Profile*** − The same external table that contains users and passwords also contains additional properties (email and phone) for each user. These additional properties are not part of authentication; but they are

	part of each user's profile. If you want to access these properties in your portal applications (with the WebLogic Portal JSP tags, controls, or API), you must create a unified user profile for the RDBMS user store. When you create the unified user profile, the ProfileWrapper includes the external properties in the user profile. The unified user profile consists of a stateless session bean and associated classes that you create. If you want to surface any of these properties in the WebLogic Administration Portal to be used in defining rules for personalization, delegated administration, or visitor entitlements, create a user profile property set for the external user store in addition to implementing your unified user profile session bean. The property set provides metadata about your external properties so that WebLogic Workshop and the WebLogic Administration Portal know how to display them. Properties from an external data store are typically read only in the WebLogic Administration Portal.
3	WebLogic Portal lets you create user/group properties and set default values for those properties. Any user or group in WebLogic Server, whether created in the default LDAP store or brought in through a connection to an external user store, is automatically assigned those default property values; and you can change the default values for each user or group, programmatically or in the WebLogic Administration Portal. This does not involve unified user profiles, because the properties to be retrieved are local, not stored in an external data store. In the illustration, after the authentication provider or identity asserter provides the principals, the ProfileWrapper combines the principals with the external properties of email and phone (retrieved by the unified user profile) and the default WebLogic Portal properties of address and postal code, all of which make up the full user profile.

What a Unified User Profile is Not

A user profile is not a security realm, and it does not provide authentication. It is not even the external user store itself. It is the connection (stateless session bean with associated classes) that lets you read properties in the external user store.

When Should You Create a Unified User Profile?

Create a unified user profile for an external data store if you want to do any of the following:

- Use WebLogic Portal's JSP tags, controls, or API to retrieve property values from that external store.
- Surface external properties in the WebLogic Administration Portal for use in defining rules for personalization, delegated administration, or visitor entitlements. Users and groups are not considered properties.

When Don't You Need a Unified User Profile?

You do not need to create a unified user profile for an external data store if you only want to:

- Provide authentication for users in the external user store.
- Define rules for personalization, delegated administration, or visitor entitlements based only on users or groups in an external user store, not on user properties.
- Define rules for personalization, delegated administration, or visitor entitlements based on the WebLogic Portal user profile properties you create in WebLogic Workshop, which are kept in the Portal schema.

Setting up a Unified User Profile

See Setting up Unified User Profiles.

Related Topics

Using Multiple Authentication Providers in Portal Development (external user stores)

Setting up Unified User Profiles

This topic provides guidelines and instructions on creating a unified user profile to access user/group properties from an external user store. (See Unified User Profiles Overview for overview information.)

Best Practices: When possible, use WebLogic Portal's user profile functionality (default UserProfileManager) to assign properties to users and groups. Given the choice between creating and storing additional properties in an external user store (which requires write access to that external store, which must be implemented) and creating and storing them in WebLogic Portal, doing so in WebLogic Portal can greatly improve performance on accessing property values. If you are storing users and groups in an external store, the ideal configuration is storing only users, groups, and passwords in the external store and creating and setting additional properties in WebLogic Portal. With that configuration, performance is optimal and you do not have to create a unified user profile.

To create a UUP to retrieve user data from external sources, complete the following tasks:

Create an EntityPropertyManager EJB to Represent External Data

Deploy a ProfileManager That Can Use the New EntityPropertyManager

If you have an LDAP server for which you want to create a unified user profile, WebLogic Portal provides a default unified user profile you can modify. See Retrieving User Profile Data from LDAP.

Create an EntityPropertyManager EJB to Represent External Data

To incorporate data from an external source, you must first create a stateless session bean that implements the methods of the com.bea.p13n.property.EntityPropertyManager remote interface. EntityPropertyManager is the remote interface for a session bean that handles the persistence of property data and the creation and deletion of profile records. By default, EntityPropertyManager provides read−only access to external properties.

In addition, the stateless session bean should include a home interface and an implementation class. For example:

```
MyEntityPropertyManager
extends com.bea.p13n.property.EntityPropertyManager

MyEntityPropertyManagerHome
extends javax.ejb.EJBHome
```

Your implementation class can extend the EntityPropertyManagerImpl class. However the only requirement is that your implementation class is a valid implementation of the MyEntityPropertyManager remote interface. For example:

```
MyEntityPropertyManagerImpl extends
com.bea.p13n.property.internal.EntityPropertyManagerImpl
```

or

```
MyEntityPropertyManagerImpl extends
javax.ejb.SessionBean
```

Recommended EJB Guidelines

We recommend the following guidelines for your new EJB:

- Your custom EntityPropertyManager is not a default EntityPropertyManager. A default EntityPropertyManager is used to get/set/remove properties in the Portal schema. Your custom EntityPropertyManager does not have to support the following methods. It can throw java.lang.UsupportedOperationException instead:
    - getDynamicProperties()
    - getEntityNames()
    - getHomeName()
    - getPropertyLocator()
    - getUniqueId()
- If you want to be able to use the portal framework and tools to create and remove users in your external data store, you must support the createUniqueId() and removeEntity() methods. However, your custom EntityPropertyManager is not the default EntityPropertyManager so your createUniqueId() method does not have to return a unique number. It must create the user entity in your external data store and then it can return any number, such as −1.
- The following recommendations apply to the EntityPropertyManager() methods that you must support:
    - getProperty() − Use caching. You should support the getProperties() method to retrieve all properties for a user at once, caching them at the same time. Your getProperty() method should use getProperties().
    - setProperty() − Use caching.
    - removeProperties(), removeProperty() − After these methods are called, a call to getProperty() should return null for the property. Remove properties from the cache, too.
- Your implementations of the getProperty(), setProperty(), removeProperty(), and removeProperties() methods must include any logic necessary to connect to the external system.
- If you want to cache property data, the methods must be able to cache profile data appropriately for that system. (See the com.bea.p13n.cache package in the WebLogic Portal Javadoc.)
- If the external system contains read−only data, any methods that modify profile data must throw a java.lang.UnsupportedOperationException. Additionally, if the external data source contains users that are created and deleted by something other than your WebLogic Portal createUniqueId() and removeEntity() methods can simply throw an UnsupportedOperationException.
- To avoid class loader dependency issues, make sure that your EJB resides in its own package.
- For ease of maintenance, place the compiled classes of your custom EntityPropertyManager bean in your own JAR file (instead of modifying an existing WebLogic Portal JAR file).

Before you deploy your JAR file, follow the steps in the next section.

Deploy a ProfileManager That Can Use the New EntityPropertyManager

A "user type" is a mapping of a ProfileType name to a particular ProfileManager. This mapping is done in the UserManager EJB deployment descriptor.

To access the data in your new EntityPropertyManager EJB, you must do ***one*** of the following:

- Modifying the Existing ProfileManager Deployment Configuration − In most cases you will be able to use the default deployment of ProfileManager, the UserProfileManager. You will modify the UserProfileManager's deployment descriptor to map a property set and/or properties to your custom EntityPropertyManager. If you support the createUniqueId() and removeEntity() methods in your

custom EntityPropertyManager, you can use WebLogic Administration Portal to create a user of type "User" with a profile that can get/set properties using your custom EntityPropertyManager.

- Configuring and Deploying a New ProfileManager − In some cases you may want to deploy a newly configured ProfileManager that will be used instead of the UserProfileManager. This new ProfileManager is mapped to a ProfileType in the deployment descriptor for the UserManager. If you support the createUniqueId() and removeEntity() methods in your custom EntityPropertyManager, you can use the WebLogic Administration Portal (or API) to create a user of type "MyUser" (or anything else you name it) that can get/set properties using the customized deployment of the ProfileManager that is, in turn, configured to use your custom EntityPropertyManager.

ProfileManager is a stateless session bean that manages access to the profile values that the EntityPropertyManager EJB retrieves. It relies on a set of mapping statements in its deployment descriptor to find data. For example, the ProfileManager receives a request for the value of the "DateOfBirth" property, which is located in the "PersonalData" property set. ProfileManager uses the mapping statements in its deployment descriptor to determine which EntityPropertyManager EJB contains the data.

Modifying the Existing ProfileManager Deployment Configuration

If you use the existing UserProfileManager deployment to manage your user profiles, perform the following steps to modify the deployment configuration.

Under most circumstances, this is the method you should use to deploy your UUP. An example of this method is the deployment of the custom EntityPropertyManager for LDAP property retrieval, the LdapPropertyManager. The classes for the LdapPropertyManager are packaged in p13n_ejb.jar. The deployment descriptor for the UserProfileManager EJB is configured to map the "ldap" property set to the LdapPropertyManager. The UserProfileManager is deployed in p13n_ejb.jar.

1. Back up the p13n_ejb.jar file in your enterprise application root directory.
2. From p13n_ejb.jar, extract META−INF/ejb−jar.xml and open it for editing.
3. In ejb−jar.xml, find the <env−entry> element, as shown in the following example:
   ```
   <!−− map all properties in property set ldap to ldap server −−>
   <env−entry>
       <env−entry−name>PropertyMapping/ldap</env−entry−name>
       <env−entry−type>java.lang.String</env−entry−type>
       <env−entry−value>LdapPropertyManager</env−entry−value>
   </env−entry>
   ```

   and add an <env−entry> element after this to map a property set to your custom EntityPropertyManager, a shown in the following example:
   ```
   <!−− map all properties in UUPExample property set to MyEntityPropertyManager −−>
   <env−entry>
       <env−entry−name>PropertyMapping/UUPExample</env−entry−name>
       <env−entry−type>java.lang.String</env−entry−type>
       <env−entry−value>MyEntityPropertyManager</env−entry−value>
   </env−entry>
   ```
4. In ejb−jar.xml, find the <ejb−ref> element shown in the following example:
   ```
   <!−− an ldap property manager −−>
   <ejb−ref>
       <ejb−ref−name>ejb/LdapPropertyManager</ejb−ref−name>
       <ejb−ref−type>Session</ejb−ref−type>
       <home>com.bea.p13n.property.EntityPropertyManagerHome</home>
       <remote>com.bea.p13n.property.EntityPropertyManager</remote>
   </ejb−ref>
   ```

and add an <ejb−ref> element after this to map a reference to an EJB that matches the name from the previous step with ejb/ prepended as shown in the following example:

```
<!−− an example property manager −−>
<ejb−ref>
    <ejb−ref−name>ejb/MyEntityPropertyManager</ejb−ref−name>
    <ejb−ref−type>Session</ejb−ref−type>
    <home>examples.usermgmt.MyEntityPropertyManagerHome</home>
    <remote>examples.usermgmt.MyEntityPropertyManager</remote>
</ejb−ref>
```

The home and remote class names match the classes from your EJB JAR file for your custom EntityPropertyManager.

5. If your EntityPropertyManager implementation handles creating and removing profile records, you must also add Creator and Remover entries. For example:

```
<env−entry>
    <env−entry−name>Creator/Creator1</env−entry−name>
    <env−entry−type>java.lang.String</env−entry−type>
    <env−entry−value>MyEntityPropertyManager</env−entry−value>
</env−entry>

<env−entry>
    <env−entry−name>Remover/Remover1</env−entry−name>
    <env−entry−type>java.lang.String</env−entry−type>
    <env−entry−value>MyEntityPropertyManager</env−entry−value>
</env−entry>
```

This instructs the UserProfileManager to call your custom EntityPropertyManager when creating or deleting user profile records. The names "Creator1" and "Remover1" are arbitrary. All Creators and Removers will be iterated through when the UserProfileManager creates or removes a user profile. The value for the Creator and Remover matches the ejb−ref−name for your custom EntityPropertyManager without the ejb/ prefix.

6. From p13n_ejb.jar, extract META−INF/weblogic−ejb−jar.xml and open it for editing.
7. In weblogic−ejb−jar.xml, find the elements shown in the following example:

```
<weblogic−enterprise−bean>
    <ejb−name>UserProfileManager</ejb−name>
    <reference−descriptor>
        <ejb−reference−description>
            <ejb−ref−name>ejb/EntityPropertyManager</ejb−ref−name>
            <jndi−name>${APPNAME}.BEA_personalization. EntityPropertyManager</jndi−name>
        </ejb−reference−description>
```

and add an ejb−reference−description to map the ejb−ref for your custom EntityPropertyManager to the JNDI name. This JNDI name must match the name you assigned in weblogic−ejb−jar.xml in the JAR file for your customer EntityPropertyManager. It should look like the following example:

```
<weblogic−enterprise−bean>
    <ejb−name>UserProfileManager</ejb−name>
    <reference−descriptor>
        <ejb−reference−description>
            <ejb−ref−name>ejb/EntityPropertyManager</ejb−ref−name>
            <jndi−name>${APPNAME}.BEA_personalization. EntityPropertyManager</jndi−name>
        </ejb−reference−description>
        <ejb−reference−description>
            <ejb−ref−name>ejb/MyEntityPropertyManager</ejb−ref−name>
            <jndi−name>${APPNAME}.BEA_personalization. MyEntityPropertyManager</jndi−nam
        </ejb−reference−description>
```

Note the ${APPNAME} string substitution variable. The WebLogic EJB container automatically substitutes the enterprise application name to scope the JNDI name to the application.

8. Update p13n_ejb.jar for your new deployment descriptors. You can use the jar uf command to update the modified META−INF/ deployment descriptors.
9. Edit your application's META−INF/application.xml to add an entry for your custom EntityPropertyManager EJB module as shown in the following example:
```
<module>
    <ejb>UUPExample.jar</ejb>
</module>
```
10. If you are using an application−wide cache, you can manage it from the WebLogic Administration Console if you add a <Cache> tag for your cache to the META−INF/application−config.xml deployment descriptor for your enterprise application like this:
```
<Cache Name="UUPExampleCache" TimeToLive="60000"/>
```
11. Verify the modified p13n_ejb.jar and your custom EntityPropertyManager EJB JAR archive are in the root directory of your enterprise application and start WebLogic Server.
12. Use the WebLogic Server Administration Console to verify your EJB module is deployed to the enterprise application and then use the console to add your server as a target for the EJB module. You need to select a target to have your domain's config.xml file updated to deploy your EJB module to the server.
13. Use the WebLogic Workshop Property Set Designer to create a User Profile (property set) that matches the name of the property set that you mapped to your custom EntityPropertyManager in ejb−jar.xml for the UserProfileManager (in p13n_ejb.jar). You could also map specific property names in a property set to your custom EntityPropertyManager, which would allow you to surface the properties and their values in the WebLogic Administration Portal for use in creating rules for personalization, delegated administration, and visitor entitlements.

Your new Unified User Profile type is ready to use. You can use the WebLogic Administration Portal to create a user, and it will use your UUP implementation when the "UUPExample" property set is being modified. When you call createUser("bob", "password") or createUser("bob", "password", null) on the UserManager, several things will happen:

- A user named "bob" is created in the security realm.
- A WebLogic Portal Server profile record is created for "bob" in the user store.
- If you set up the Creator mapping, the UserManager will call the default ProfileManager deployment (UserProfileManager) which will call your custom EntityPropertyManager to create a record for Bob in your data source.
- Retrieving Bob's profile will use the default ProfileManager deployment (UserProfileManager), and when you request a property belonging to the "UUPExample" property set, the request will be routed to your custom EntityPropertyManager implementation.

Configuring and Deploying a New ProfileManager

If you are going to deploy a newly configured ProfileManager instead of using the default ProfileManager (UserProfileManager) to manage your user profiles, perform the following steps to modify the deployment configuration. In most cases, you will not have to use this method of deployment. Use this method only if you need to support multiple types of users that require different ProfileManager deployments deployments that allow a property set to be mapped to different custom EntityPropertyManagers based on ProfileType.

An example of this method is the deployment of the custom CustomerProfileManager in customer.jar. The CustomerProfileManager is configured to use the custom EntityPropertyManager (CustomerPropertyManager) for properties in the "CustomerProperties" property set. The UserManager EJB

in p13n_ejb.jar is configured to map the "WLCS_Customer" ProfileType to the custom deployment of the ProfileManager, CustomerProfileManager.

To configure and deploy a new ProfileManager, use this procedure.

1. Back up the p13n_ejb.jar file in your enterprise application root directory.
2. From p13n_ejb.jar, extract META−INF/ejb−jar.xml, and open it for editing.
3. In ejb−jar.xml, copy the entire <session> tag for the UserProfileManager, and configure it to use your custom implementation class for your new deployment of ProfileManager.
   In addition, you could extend the UserProfileManager home and remote interfaces with your own interfaces if you want to repackage them to correspond to your packaging (for example., examples.usermgmt.MyProfileManagerHome, examples.usermgmt.MyProfileManager).
   However, it is sufficient to replace the bean implementation class:
   You must create an <env−entry> element to map a property set to your custom EntityPropertyManager. You must also create a <ejb−ref> element to map a reference to an EJB that matches the name from the PropertyMapping with ejb/ prepended. The home and remote class names for your custom EntityPropertyManager match the classes from your EJB JAR file for your custom EntityPropertyManager.

   Also, if your EntityPropertyManager implementation handles creating and removing profile records, you must also add Creator and Remover entries. This instructs your new ProfileManager to call your custom EntityPropertyManager when creating or deleting user profile records.

   Note: The name suffixes for the Creator and Remover, "Creator1" and "Remover1", are arbitrary. All Creators and Removers will be iterated through when your ProfileManager creates or removes a user profile. The value for the Creator and Remover matches the <ejb−ref−name> for your custom EntityPropertyManager without the ejb/ prefix.
4. In ejb−jar.xml, you must add an <ejb−ref> to the UserManager EJB section to map your ProfileType to your new deployment of the ProfileManager, as shown in the following example:
   ```
   <ejb−ref>
       <ejb−ref−name>ejb/ProfileType/UUPExampleUser</ejb−ref−name>
       <ejb−ref−type>Session</ejb−ref−type>
       <home>com.bea.p13n.usermgmt.profile.ProfileManagerHome</home>
       <remote>com.bea.p13n.usermgmt.profile.ProfileManager</remote>
   </ejb−ref>
   ```

   The <ejb−ref−name> must start with ejb/ProfileType/ and must end with the name that you want to use as the profile type as an argument in the createUser() method of UserManager.
5. From p13n_ejb.jar, extract META−INF/weblogic−ejb−jar.xml and open it for editing.
6. In weblogic−ejb−jar.xml, copy the <weblogic−enterprise−bean> tag, shown in the following example, for the UserProfileManager and configure it for your new ProfileManager deployment:
   ```
   <weblogic−enterprise−bean>
       <ejb−name>MyProfileManager</ejb−name>
       <reference−descriptor>
           <ejb−reference−description>
               <ejb−ref−name>ejb/EntityPropertyManager</ejb−ref−name>
               <jndi−name>${APPNAME}.BEA_personalization. EntityPropertyManager</jndi−name>
           </ejb−reference−description>
           <ejb−reference−description>
               <ejb−ref−name>ejb/PropertySetManager</ejb−ref−name>
               <jndi−name>${APPNAME}.BEA_personalization. PropertySetManager</jndi−name>
           </ejb−reference−description>
           <ejb−reference−description>
               <ejb−ref−name>ejb/MyEntityPropertyManager</ejb−ref−name>
   ```

```
            <jndi-name>${APPNAME}.BEA_personalization. MyEnitityPropertyManager</jndi-nam
        </ejb-reference-description>
    </reference-descriptor>
    <jndi-name>${APPNAME}.BEA_personalization. MyProfileManager</jndi-name>
</weblogic-enterprise-bean>
```

You must create an <ejb−reference−description> to map the <ejb−ref> for your custom EntityPropertyManager to the JNDI name. This JNDI name must match the name you assigned in weblogic−ejb−jar.xml in the JAR file for your custom EntityPropertyManager.
Note the ${APPNAME} string substitution variable. The WebLogic Server EJB container automatically substitutes the enterprise application name to scope the JNDI name to the application.

7. In weblogic−ejb−jar.xml, copy the <transaction−isolation> tag for the UserProfileManager, shown in the following example, and configure it for your new ProfileManager deployment:
```
<transaction-isolation>
    <isolation-level>TRANSACTION_READ_COMMITTED</isolation-level>
    <method>
        <ejb-name>MyProfileManager</ejb-name>
        <method-name>*</method-name>
    </method>
</transaction-isolation>
```
8. Create a temporary p13n_ejb.jar for your new deployment descriptors and your new ProfileManager bean implementation class. This temporary EJB JAR archive should not have any container classes in it. Run ejbc to generate new container classes.
9. Edit your application's META−INF/application.xml to add an entry for your custom EntityPropertyManager EJB module, as shown in the following example:
```
<module>
    <ejb>UUPExample.jar</ejb>
</module>
```
10. If you are using an application−wide cache, you can manage it from the WebLogic Server Administration Console if you add a <Cache> tag for your cache to the META−INF/application−config.xml deployment descriptor for your enterprise application as shown in the following example:
```
<Cache Name="UUPExampleCache" TimeToLive="60000"/>
```

    Verify the modified p13n_ejb.jar and your custom EntityPropertyManager EJB JAR archive are in the root directory of your enterprise application and start your server.
11. Use the WebLogic Server Administration Console to verify your EJB module is deployed to the enterprise application and add your server as a target for the EJB module. You must select a target to have your domain's config.xml file updated to deploy your EJB module to the server.
12. Use the WebLogic Workshop Property Set Designer to create a User Profile (property set) that matches the name of the property set that you mapped to your custom EntityPropertyManager in ejb−jar.xml for the UserProfileManager (in p13n_ejb.jar). You could also map specific property names in a property set to your custom EntityPropertyManager, which would allow you to surface the properties and their values in the WebLogic Administration Portal for use in creating rules for personalization, delegated administration, and visitor entitlements.

Your new Unified User Profile type is ready to use. You can use the WebLogic Administration Portal to create a user, and it will use your UUP implementation when the "UUPExample" property set is being modified. That is because you mapped the ProfileType using an <ejb−ref> in your UserManager deployment descriptor, ejb/ProfileType/UUPExampleUser.

Now, when you call createUser("bob", "password", "UUPExampleUser") on the UserManager, several things will happen:

- A user named "bob" is created in the security realm.
- A WebLogic Portal Server profile record is created for "bob" in the WebLogic Portal RDBMS repository.
- If you set up the Creator mapping, the UserManager will call your new ProfileManager deployment, which will call your custom EntityPropertyManager to create a record for Bob in your data source.
- Retrieving Bob's profile will use your new ProfileManager deployment, and when you request a property belonging to the "UUPExample" property set, the request will be routed to your custom EntityPropertyManager implementation.

Retrieving User Profile Data from LDAP

WebLogic Portal provides a default unified user profile for retrieving properties from an LDAP server. Use this procedure to implement the LDAP unified user profile for retrieving properties from your LDAP server.

The LdapRealm security realm and the LdapPropertyManager unified user profile (UUP) for retrieving user properties from LDAP are independent of each other. They do not share configuration information and there is no requirement to use either one in conjunction with the other. A security realm has nothing to do with a user profile. A security realm provides user/password data, user/group associations, and group/group associations. A user profile provides user and group properties. A password is not a property.

In order to successfully retrieve the user profile from the LDAP server, ensure that you've done the following:

1. If you have already deployed the application on a WebLogic Portal instance, stop the server.
2. Extract p13n_ejb.jar from your application root to a temporary directory.
3. In the temporary directory, open META−INF/ejb−jar.xml, which contains a commented block called "Ldap Property Manager." Uncomment and reconfigure this section using the following steps:
    a. Remove the closing comment mark (−−>) from the end of the "Ldap Property Manager" block, just before the "Property Set Web Service EJB" block, and add it to the end of the first paragraph of the Ldap Property Manager block, like this:

```
<!−− Ldap Property Manager
     To use this, uncomment it here as well as in weblogic−ejb−jar.xml.
     Configure the LDAP connection and settings using the env−entry values
     Do not forget to uncomment the ejb−link and method−permission tags for
     An easy way to ensure you don't miss anything is to search for "ldap"
     weblogic−ejb−jar.xml. Search from the beginning to the end of the file
−−>
```

    b. In the "Ldap Property Manager" block, look for the following default settings and replace them with your own:

ldap://server.company.com:389	Change this to the value of your LDAP server URL.
uid=admin, ou=Administrators, ou=TopologyManagement, o=NetscapeRoot	Change this to the value of your LDAP server's principal.
<env−entry−value>weblogic</env−entry−value>	Change "weblogic" to your LDAP server's principalCredential.

ou=People,o=company.com	Change this to your LDAP server's UserDN.
ou=Groups,o=company.com	Change this to your LDAP server's GroupDN.
<env−entry−value>uid</env−entry−value>	Change "uid" to your LDAP server's usernameAttribute setting.
<env−entry−value>cn</env−entry−value>	Change "cn" to your LDAP server's groupnameAttribute setting.

c. In the "User Profile Manager" and "Group Profile Manager" sections, find the following lines:

```
<!−− <ejb−link>LdapPropertyManager</ejb−link> −−>
<ejb−link>EntityPropertyManager</ejb−link>
```

Uncomment the LdapPropertyManager line and delete the EntityPropertyManager line in both sections.

d. In the <method−permission> and <container−transaction> sections, find and uncomment the following:

```
<!−−
<method>
    <ejb−name>LdapPropertyManager</ejb−name>
    <method−name>*</method−name>
</method>
−−>
```

e. Check to see that you have uncommented all Ldap configurations by doing a search for "Ldap" in the file.

f. Save and close the file.

4. In the temporary directory, open META−INF/weblogic−ejb−jar.xml and perform the following modifications:

a. Uncomment the "LdapPropertyManager" block:

```
LdapPropertyManager
<weblogic−enterprise−bean>
    <ejb−name>LdapPropertyManager</ejb−name>
    <enable−call−by−reference>True</enable−call−by−reference>
    <jndi−name>${APPNAME}.BEA_personalization.LdapPropertyManager</jndi−nam
</weblogic−enterprise−bean>
```

b. In the "Security configuration" section of the file, uncomment the LdapPropertyManager method:

```
<method>
    <ejb−name>LdapPropertyManager</ejb−name>
    <method−name>*</method−name>
</method>
```

c. Check to see that you have uncommented all Ldap configurations by doing a search for "Ldap" in the file.
d. Save and close the file.

5. Replace the original p13n_ejb.jar with the modified version.

a. Rename the original p13n_ejb.jar to use it as a backup. For example, rename it to p13n_ejb.jar.backup.
b. JAR the temporary version of p13n_ejb.jar to which you made changes. Name it p13n_ejb.jar.
c. Copy the new JAR to your application's root directory.

6. Start the server and re−deploy the application.
7. The properties from your LDAP server are now accessible through the WebLogic Portal API, JSP tags, and controls.

   If you want to surface the properties from your LDAP server in the WebLogic Administration Portal (for use in defining rules for personalization, delegated administration, and visitor entitlements), create a user profile property set called ldap.usr, and create properties in the property set that exactly match the names of the LDAP properties you want to surface.

Enabling SUBTREE_SCOPE Searches for Users and Groups

The LdapPropertyManager EJB in p13n_ejb.jar allows for the inspection of the LDAP schema to determine multi−valued versus single−value LDAP attributes, to allow for multiple userDN/groupDN, and to allow for SUBTREE_SCOPE searches for users and groups in the LDAP server. Following are more detailed explanations:

The determination of multi−value versus single−value LDAP attributes allows a developer to configure the ejb−jar.xml deployment descriptor for the LdapPropertyManager EJB to specify that the LDAP schema be used to determine if a property is single− or multi−value.

To enable SUBTREE−SCOPE for users and groups:

1. Stop the server.
2. Extract p13n_ejb.jar from your application root directory to a temporary directory and edit the temporary META−INF/ejb−jar.xml by setting the following env−entries.

```
<!−− Flag to specify if LDAP attributes will be determined to be single value
or multi−value via the schema obtained from the attribute. If false,
then the attribute is stored as multi−valued (a Collection) only if it has
more than one value. Leave false unless you intend to use multi−valued LDAP
attributes that may have only one value. Using true adds overhead to check
the LDAP schema. Also, if you use true beware that most LDAP attributes are
multi−value. For example, iPlanet Directory Server 5.x uses multi−value for
givenName, which you may not expect unless you are familiar with LDAP schemas.
This flag will apply to property searches for all userDNs and all groupDNs. −−>

<env−entry>
    <env−entry−name>config/detectSingleValueFromSchema</env−entry−name>
    <env−entry−type>java.lang.Boolean</env−entry−type>
    <env−entry−value>true</env−entry−value>
</env−entry>

<!−− Value of the name of the attribute in the LDAP schema that is used
to determine single value or multi−value (RFC2252 uses SINGLE−VALUE).
This attribute in the schema should be true for single value and false
```

```
or absent from the schema otherwise. The value only matters if
config/detectSingleValueFromSchema is true. -->

<env-entry>
    <env-entry-name>config/singleValueSchemaAttribute</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>SINGLE-VALUE</env-entry-value>
</env-entry>
```

It is not recommended that true be used for config/detectSingleValueFromSchema unless you are going to write rules that use multi−valued LDAP attributes that have a single value. Using config/detectSingleValueFromSchema = true adds the overhead of checking the LDAP schema for each attribute instead of the default behavior (config/detectSingleValueFromSchema = false), which only stores an attribute as multi−valued (in a Collection) if it has more than one value.

This feature also implements changes that allow you to use SUBTREE_SCOPE searches for users and groups. It also allows multiple base userDN and groupDN to be specified. The multiple base DN can be used with SUBTREE_SCOPE searches enabled or disabled.

A SUBTREE_SCOPE search begins at a base userDN (or groupDN) and works down the branches of that base DN until the first user (or group) is found that matches the username (or group name).

To enable SUBTREE_SCOPE searches you must set the Boolean config/objectPropertySubtreeScope env−entry in the ejb−jar.xml for p13n_ejb.jar.jar to true and then you must set the config/userDN and config/groupDN env−entry values to be equal to the base DNs from which you want your SUBTREE_SCOPE searches to begin.

For example, if you have users in ou=PeopleA,ou=People,dc=mycompany,dc=com and in ou=PeopleB,ou=People,dc=mycompany,dc=com then you could set config/userDN to ou=People,dc=mycompany,dc=com and properties for these users would be retrieved from your LDAP server because the user search would start at the "People" ou and work its way down the branches (ou="PeopleA" and ou="PeopleB").

You should not create duplicate users in branches below your base userDN (or duplicate groups below your base groupDN) in your LDAP server. For example, your LDAP server will allow you to create a user with the uid="userA" under both your PeopleA and your PeopleB branches. The LdapPropertyManager in p13n_ejb.jar.jar will return property values for the first userA that it finds.

It is recommended that you do not enable this change (by setting config/objectPropertySubtreeScope to true) unless you need the flexibility offered by SUBTREE_SCOPE searches.

An alternative to SUBTREE_SCOPE searches (with or without multiple base DNs) would be to configure multiple base DNs and leave config/objectPropertySubtreeScope set to false. Each base DN would have to be the DN that contains the users (or groups) because searches would not go any lower than the base DN branches. The search would cycle from one base DN to the next until the first matching user (or group) is found.
The new ejb−jar.xml deployment descriptor is fully commented to explain how to set multiple DNs, multiple usernameAttributes (or groupnameAttributes), and how to set the objectPropertySubtreeScope flag.

3. Save and close the file.
4. Replace the original p13n_ejb.jar with the modified version:

a. Rename the original p13n_ejb.jar to use it as a backup. For example, rename it to p13n_ejb.jar.backup.
b. JAR the temporary version of p13n_ejb.jar to which you made changes. Name it p13n_ejb.jar.
c. Copy the new JAR to your application's root directory.

5. Start the server and re−deploy the application.

Related Topics

Using Multiple Authentication Providers in Portal Development