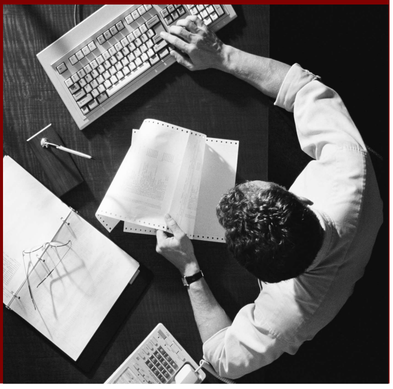
SAP NetWeaver Developer's Guide

Release: SAP NetWeaver 2004s



Protecting Access to a J2EE-Based Application Using J2EE Security Roles

Document Version 1.00 – October 2005

THE BEST-RUN BUSINESSES RUN SAP

SAP AG
Dietmar-Hopp-Allee 16
69190 Walldorf
Germany
T +49/18 05/34 34 24
F +49/18 05/34 34 20
www.sap.com

Disclaimer

Typographic Conventions

Type Style	Represents
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
Example text	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, F2 or ENTER.

Icons

Icon	Meaning
⚠	Caution
💬	Example
💡	Note
🧭	Recommendation
‹syn›	Syntax

Contents

1 Protecting Access to a J2EE-Based Application Using J2EE Security Roles

The Task

In this tutorial, you will include access protection to the quick car rental application that is provided with the SAP NetWeaver Developer Studio. In this application, a JSP and servlet serve as the frontend client. The business logic is implemented using EJBs.

You will provide access protection using authentication and J2EE security roles. The following rules apply:

- All car rental employees can access the application. They can view current reservations.
- Only those employees who work as booking agents can create or cancel reservations.

To perform the authorization check, you also have to require authentication. For this tutorial, you will use Basic Authentication (user ID and password).

Objectives

By the end of this tutorial, you will be able to:

- ✓ Use authentication to protect access to the application.
- ✓ Use J2EE security roles in the JSP to protect access to the application.
- ✓ Use J2EE security roles in the EJBs to distinguish between users with different authorizations for the different methods.
- ✓ Perform the administrative steps for assigning users to J2EE security roles using the Visual Administrator.

Prerequisites

Systems, Installations, and Authorizations

- ☐ The SAP NetWeaver Developer Studio is installed on your computer. The minimum stack level required is stack level 11.
- ☐ You can access the J2EE Engine from the SAP NetWeaver Developer Studio for deployment.
- ☐ You can log on to the J2EE Engine with an administrator user using the Visual Administrator.

Knowledge

- ☐ Java knowledge and basic knowledge of the J2EE programming model is advantageous.
- ☐ You have acquired some basic experience with the J2EE toolset in the Developer Studio.

Next Step:

Importing the Project for the J2EE-Based Car Rental Tutorial [Page 2]

2 Importing the Project for the J2EE-Based Car Rental Tutorial

Use

The project you will use for this tutorial is the J2EE-based quick car rental application that is provided with the SAP NetWeaver Developer Studio. Therefore, to begin working with this tutorial, you first have to import this project into the Developer Studio workspace.

Prerequisites

- ☐ You know the workspace directory for your SAP NetWeaver Developer Studio.
- ☐ You know the path to the examples provided with the Developer Studio. In a default installation this path is *C:\Program Files\SAP\JDT\eclipse\examples*.
- ☐ If you have previously worked on the J2EE quick car rental application using the Developer Studio, then this project is closed and its contents are removed from the Developer Studio workspace. In addition, the application does not exist as a deployed application on the J2EE Engine.
- ☐ The J2EE Engine and the Software Deployment Manager (SDM) are running.
- ☐ You can connect to the J2EE Engine from the Developer Studio for deployment. You also know the SDM password to use for the connection.

This tutorial assumes that you are working with the J2EE quick car rental project as it is provided with the J2EE Engine's example projects. Therefore, if you have previously worked on either the J2EE or the Web Dynpro car rental applications and deployed them to the J2EE Engine, we recommend you remove them not only from the Developer Studio's workspace, but also from the J2EE Engine and start with a completely new set of projects. Note the following:

- When deleting the projects from the Developer Studio, also delete the content from the workspace. Also delete the *Dictionary* project and the *Helperclasses*. (Switch to the *Navigator* to make sure that all projects have been removed.)
- To remove the application from the J2EE Engine, use the Deploy service in the Visual Administrator. Remove the application `sap.com/QuickCarRentalEar` and if applicable, `local/TutSD_CarRental`, and any of the corresponding components.
- You do not need to remove any entries from the database itself.

Procedure

Importing the project template into the SAP NetWeaver Developer Studio

1. Navigate to the location of the examples for the SAP NetWeaver Developer Studio.
2. Unpack the `J2EE_QuickCarRental.zip` archive into the workspace directory for your SAP NetWeaver Developer Studio.
3. Start the SAP NetWeaver Developer Studio.

4. Switch to the J2EE Development perspective and import the projects from the quick car rental application.

    a. To switch to the J2EE Development perspective, choose *Window* → *Open Perspective* → *J2EE Development*.

    b. Choose *File* → *Import.*

    c. Select `Multiple Existing Project into Workspace` and choose *Next*.

    d. In the *Select base folder* field, enter your workspace directory (or use the *Browse...* function).

       The projects for the car rental application appear. These are *Helperclasses*, *J2EE_QuickCarRentalEar*, *J2EE_QuickCarRentalEjb*, *J2EE_QuickCarRentalWeb* and *QuickCarRentalDictionary*.

    e. Select all of these projects. Also select the *Open projects after import* indicator and choose *Finish*.

       The projects are opened in the SAP NetWeaver Developer Studio.

5. Deploy the dictionary archive to the J2EE Engine.

    a. Switch to the Dictionary perspective. (Choose *Window* → *Open Perspective* → *Dictionary*.)

    b. Choose the *Dictionary Explorer*.

    c. Select the *QuickCarRentalDictionary* project, open the context menu with the right mouse button, and choose *Deploy*.

       If this is the first time you deploy from the SAP NetWeaver Developer Studio, then you are prompted for the SDM password. Enter this password to continue with deployment.

6. Deploy the EAR archive to the J2EE Engine.

    a. Switch back to the J2EE Development perspective and the *J2EE Explorer*.

    b. Expand the *J2EE_QuickCarRentalEar* project.

    c. Select the *J2EE_QuickCarRentalEar.ear* node, open the context menu and choose *Deploy to J2EE Engine*.

7. To test the deployment, start a Web browser and enter the URL for the application.

   `http://localhost:50000/QuickCarRental`

   If the application does not run, then make sure the J2EE Engine and the SDM are running and that the host and port are correct.

Initial Project Structure

Once you have imported the project template into the Developer Studio, you will see the following structures in the *J2EE Explorer*.

J2EE Project Structure
J2EE application project: **J2EE_QuickCarRentalEar**
J2EE Enterprise Java Bean project: **J2EE_QuickCarRentalEjb**

J2EE Web project: **J2EE_QuickCarRentalWeb**

Next Step:

You can now begin with the tutorial steps.

If you are working with the tutorial for using J2EE security roles, see Using J2EE Security Roles in the Application - Steps [Page 4].

If you are working with the tutorial for using UME permissions, see Using UME Permissions in the Application – Steps [External].

3 Using J2EE Security Roles in the Application - Steps

The steps you will perform to protect access to the quick car rental application using J2EE security roles are:

1. You will first protect access to the application by specifying security in the JSP.
    a. To be able to perform authorization checks, you must know the user that is attempting to access the application. Therefore, you will specify that authentication is required to access the application.
    b. You will specify the policy domain that will use this authentication.
    c. You will create the J2EE security role to use for accessing the application.
    d. You will create a security constraint for the application that contains this J2EE security role.
2. You will protect access to the EJB methods by creating the J2EE security roles to use for the EJB methods and assigning them to the corresponding methods.
3. You will adjust the error handling to produce an access control error message if the user does not have the permissions for accessing the application.
4. You will rebuild and redeploy the application.
5. You will perform the administrative steps.
    a. You will create the users to use for this tutorial.
    b. You will assign the users to the J2EE security roles.
6. You will test the application and the role assignment.

Next Step:

Protecting Access to the Application in the JSP [Page 5]

4 Protecting Access to the Application in the JSP

Use

In the first part of this tutorial, you will protect access to the application by specifying access protection in the JSP.

Procedure

You will perform the following steps:

1. Require authentication for access to the JSP application. In this way, you obtain the user's ID to use for the authorization check.
2. Specify the policy domain that uses the authentication.
3. Create the J2EE security role `AccessQuickCarRental`. This role will protect access to the application.
4. Create a security constraint that specifies which set of resources are to be protected with this security role.

Next Step:

Specifying Authentication in the JSP [Page 5]

4.1 Specifying Authentication in the JSP

Use

The first step is to specify authentication in the JSP. Afterwards, only users who are successfully authenticated on the J2EE Engine can access the quick car rental application.

There are four types of authentication available: `BASIC`, `DIGEST`, `FORM`, and `CLIENT-CERT`. For more information about these types, see Authentication on J2EE Engine [External].

For this tutorial, you will specify `BASIC` (Basic Authentication).

Prerequisites

- ☐ The J2EE Development perspective is displayed in the SAP NetWeaver Developer Studio.
- ☐ The quick car rental application's Web client project, *J2EE_QuickCarRentalWeb*, is displayed in the *J2EE Explorer*.

Procedure

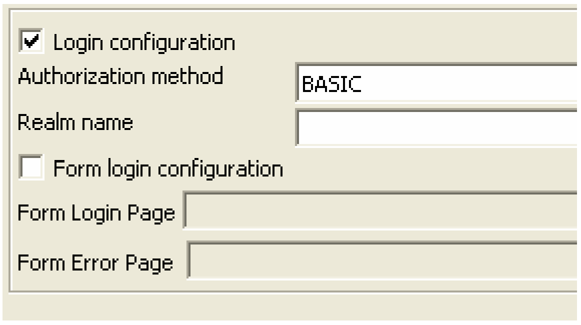
1. Expand the *J2EE_QuickCarRentalWeb* project.
2. Open the *web.xml* file by selecting it with a double-click.

   The Developer Studio's multi-page editor appears.
3. On the *General* tab:

a. Select the *Login configuration* indicator.

b. Select BASIC for the *Authorization method*.

See the preview below for an example of the input.



☑ Login configuration

Authorization method	BASIC
Realm name	

☐ Form login configuration

Form Login Page	
Form Error Page	

4. Save the data.

Result

The `<login-config>` element is added to the deployment descriptors for the Web application in the *web.xml* file as shown below:

```
      <login-config>
            <auth-method>BASIC</auth-method>
      </login-config>
```

Next Step:

Specifying the Policy Domain to Use for Authentication [Page 6]

4.2 Specifying the Policy Domain to Use for Authentication

Use

To apply the authentication mechanism to your application, you must also specify which policy domains on the J2EE Engine will use this authentication, for example, either the root domain for the application or a set of sub-domains. For more information about policy domains, see Configuring Authentication [External].

For this tutorial, you will specify that the authentication applies to the application's domain, which is `/QuickCarRental`. Other applications running on the J2EE Engine will not have access to the user's authentication information.

If all applications running on the J2EE Engine are to use the authentication, then specify the wildcard entry `/*`.

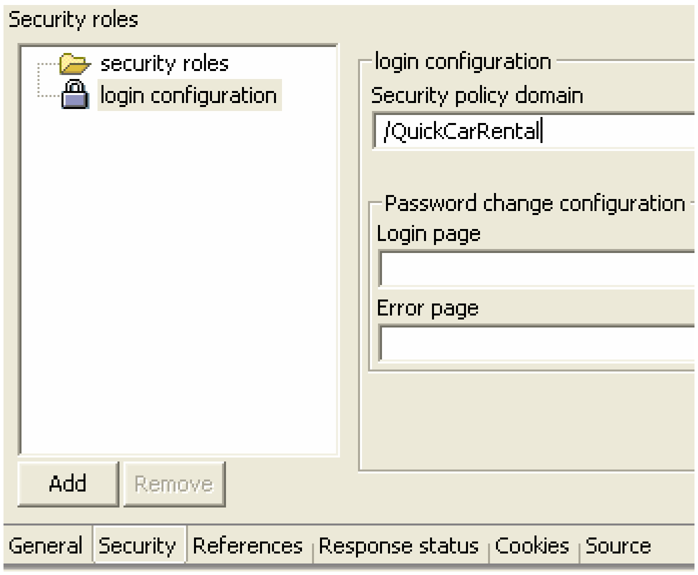
Prerequisites

☐ The quick car rental application's Web client project, *J2EE_QuickCarRentalWeb*, is displayed in the *J2EE Explorer*.

Procedure

1. Open the *web-j2ee-engine.xml* file by selecting it with a double-click.
2. Select the *Security* tab page.
3. Select the *login configuration* element.
4. In the *login configuration* section, enter **`/QuickCarRental`** in the *Security policy domain* field.

   See the figure below:

   login configuration
   Security policy domain: /QuickCarRental

   Do not enter anything in the *Password change configuration* section.
5. Save the data.

Result

The authentication specified applies to the `/QuickCarRental` domain for the application.

The `<security-policy-domain>` element is added to the deployment descriptors for the Web application in the *web-j2ee-engine.xml* file as shown below:

```
<web-j2ee-engine>
      <security-policy-domain>/QuickCarRental</security-policy-
domain>
</web-j2ee-engine>
```

Next Step:

Creating a J2EE Security Role for Accessing the Application [Page 8]

4.3 Creating a J2EE Security Role for Accessing the Application

Use

In this step, you will create the J2EE security role `AccessQuickCarRental`. Only users that are assigned to this role will be able to access the quick car rental application.

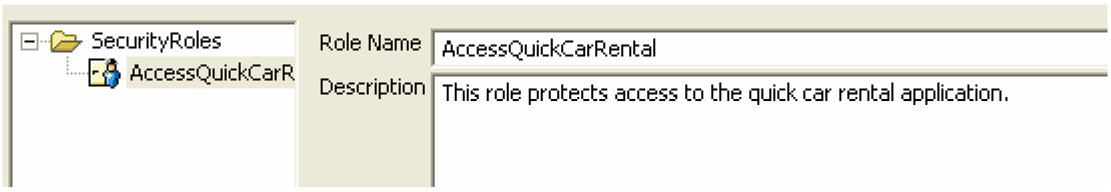
Prerequisites

☐ The quick car rental application's Web client project, *J2EE_QuickCarRentalWeb*, is displayed in the J2EE Explorer.

Procedure

1. Open or return to the *web.xml* file.
2. Choose the *Security Roles* tab page.
3. To create a role, select the *SecurityRoles* node and choose *Add*.

   A role with the name `DefaultSecurityRole` appears.
4. Select this role and enter the new name **`AccessQuickCarRental`** in the *Role Name* field.
5. Enter a description in the *Description* field, for example, **`This role protects access to the quick car rental application.`**.



Role Name: AccessQuickCarRental

Description: This role protects access to the quick car rental application.

6. Save the data.

Result

The `<security-role>` element with `<description>` and `<role-name>` sub-elements are added to the deployment descriptors for the Web application in the *web.xml* file.

```
      <security-role>
            <description>This role protects access to the quick car
rental
                                          application.</description>
            <role-name>AccessQuickCarRental</role-name>
      </security-role>
```

In addition, the `<security-role-map>` entry is added to the *web-j2ee-engine.xml* file as shown below:

```
<web-j2ee-engine>
      <security-role-map>
            <role-name>AccessQuickCarRental</role-name>
      </security-role-map>
      <security-policy-domain>/QuickCarRental</security-policy-
domain>
```

```
</web-j2ee-engine>
```

Next Step:

Creating a Security Constraint [Page 9]

4.4 Creating a Security Constraint

Use

Security constraints specify which set of resources are to be protected by the security role you just created. In addition, you can specify the level of transport layer security that is required when accessing this set of resources.

When specifying the security constraints, you need to consider the following:

- You need to determine the set of resources that are to be protected. For this purpose, you specify a URL pattern.
- You can also specify which HTTP methods are to be restricted. For example, you can specify that the HTTP `POST` method underlies the security constraint.
- You then specify an authorization constraint, which specifies the security role that a user must be assigned to in order to access this set of resources.
- You can also specify that protection at the transport layer is guaranteed.

For this tutorial, you will specify that the complete set of resources and HTTP methods are protected with the security role `AccessQuickCarRental`. You will not specify any transport layer security.

Prerequisites

- ☐ The quick car rental application's Web client project, *J2EE_QuickCarRentalWeb*, is displayed in the J2EE Explorer.
- ☐ You have created the security role `AccessQuickCarRental`.

Procedure

Specifying the Set of Resources to Protect

1. Open or return to the *web.xml* file.
2. Choose the *Security* tab page.

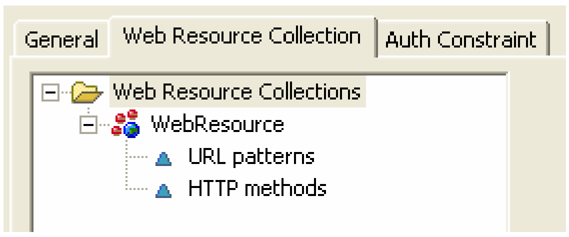
   The *Security Constraints* section appears.
3. Select *Security Constraints* and choose *Add*.

   The default *SecurityConstraint* appears.

   You can rename this constraint, however, for this tutorial, we will use the default name.
4. Choose the *Web Resource Collection* tab page.

   The Web resources appear.
5. Expand *Web Resource Collections* → *WebResource*.

Two items appear, one for URL patterns and one for HTTP methods as shown in the preview below.

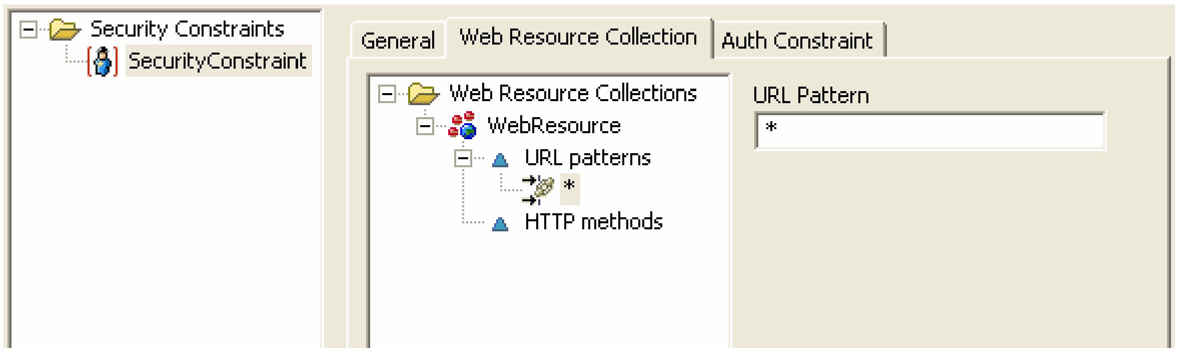


6. Select *URL patterns* and choose *Add*.

   The default URL pattern, which is a wildcard (`*`), is assigned to the security constraint. This means that all resources under the application's URL underlie this security constraint. For this tutorial, we will use the default.

   Also, for this tutorial, you will not specify any HTTP methods for the constraint.

   Your security constraint is specified as shown in the figure below:



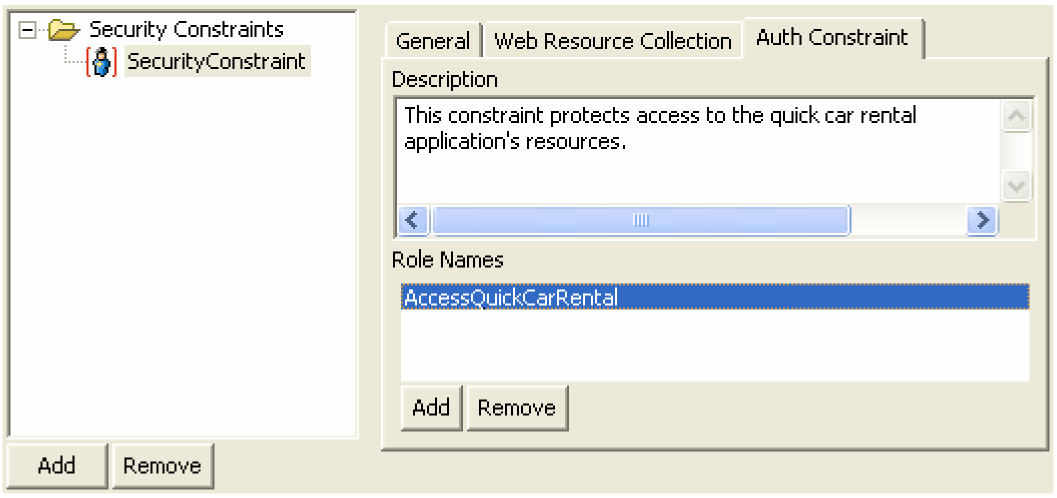
URL Pattern: *

Specifying the Authorization Constraint

1. Continue by choosing the *Auth Constraint* tab page.
2. Under *Role Names*, choose *Add*.

   The *Choose role-names* dialog appears.
3. Select the `AccessQuickCarRental` role and choose *OK*.
4. The role is added to the security constraint. Optionally, you can enter a short description for the authorization constraint.

   See the figure below:



Description
This constraint protects access to the quick car rental application's resources.

Role Names
AccessQuickCarRental

5. Save the data.

Result

The security constraint applies to the specified resources.

The `<security-constraint>` entries are added to the *web.xml* file as shown below:

```
      <security-constraint>
            <display-name>SecurityConstraint</display-name>
            <web-resource-collection>
                  <web-resource-name>WebResource</web-resource-name>
                  <url-pattern>*</url-pattern>
            </web-resource-collection>
            <auth-constraint>
                  <description>This constraint protects access to the
quick
               car rental application&apos;s resources.
                  </description>
                  <role-name>AccessQuickCarRental</role-name>
            </auth-constraint>
      </security-constraint>
```

Next Step:

Protecting Access to the EJB Methods Using J2EE Security Roles [Page 11]

5 Protecting Access to the EJB Methods Using J2EE Security Roles

Use

In the last section, you added authentication and authorization protection in the JSP component of the application. In this case, access to the application is restricted, but any user that can access the application can both view and maintain car reservations.

In the next section, you will separate the tasks for viewing and maintaining reservations. Users with the J2EE security role `CarRentalEmployee` will be able to view reservations and users with the role `BookingAgent` will also be able to create and cancel reservations.

Procedure

To set up access protection to the EJB methods, perform the following steps:

1. Create the J2EE security roles.
2. Select the methods that are to be protected by each J2EE security role.

Next Step:

Creating the J2EE Security Roles to Use for the EJB Methods [Page 12].

5.1 Creating the J2EE Security Roles to Use for the EJB Methods

Use

For this tutorial, you will create two security roles. The first role, `CarRentalEmployee`, will allow users to view current reservations. The second role, `BookingAgent`, will allow users to view, create and cancel reservations.

Prerequisites

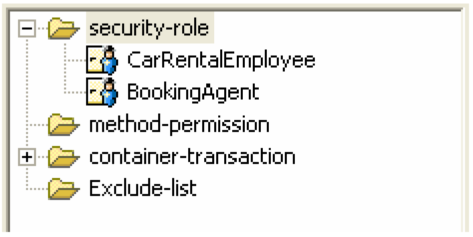
☐ The quick car rental application's EJB project, *J2EE_QuickCarRentalEJB*, is displayed in the *J2EE Explorer*.

Procedure

1. Expand the *J2EE_QuickCarRentalEjb* project.
2. Open the *ejb-jar.xml* file by selecting it with a double-click.
3. Choose the *Assembly* tab page.

   The elements for `security-role`, `method-permission`, `container-transaction`, and `Exclude-list` appear.
4. Select the `security-role` element and choose *Add*.
5. In the *Role name* field, enter **`CarRentalEmployee`**.
6. Choose *Add* again.
7. In the *Role name* field, enter **`BookingAgent`**.

   The two roles are created. See the figure below.



```
⊟ security-role
    CarRentalEmployee
    BookingAgent
  method-permission
⊞ container-transaction
  Exclude-list
```

8. Save the data.

Result

The `<security-role>` elements for each role are added to the deployment descriptor for the EJB in the *ejb-jar.xml* file.

```
            <security-role>
                  <description>
                  </description>
                  <role-name>CarRentalEmployee</role-name>
            </security-role>
            <security-role>
                  <description>
                  </description>
                  <role-name>BookingAgent</role-name>
            </security-role>
```

In addition, the `<security-permission>` entry is added to the deployment descriptor in *ejb-j2ee-engine.xml*.

```
      <security-permission>
            <security-role-map>
                  <role-name>CarRentalEmployee</role-name>
            </security-role-map>
            <security-role-map>
                  <role-name>BookingAgent</role-name>
            </security-role-map>
      </security-permission>
```

Next Step:

Selecting the EJB Methods for Each J2EE Security Role [Page 13]

5.2 Selecting the EJB Methods for Each J2EE Security Role

Use

In this step, you will select the methods that are to apply to each of the roles.

Prerequisites

☐ The quick car rental application's EJB project, *J2EE_QuickCarRentalEJB*, is displayed in the *J2EE Explorer*.

Procedure

Selecting the EJB Methods for the `CarRentalEmployee` Role

First, you will select the methods that apply to the `CarRentalEmployee` role.

1. Open or return to the *ejb-jar.xml* file.
2. Choose the *Assembly* tab page.
3. Select the `method-permission` element and choose *Add*.

   The *Choose methods* dialog appears, which shows the `QuickBookingBean` and the `QuickOrderProcessorBean`.

4. Expand the nodes for both of these beans until all of the methods are displayed.
5. Select the methods as shown in the table below:

Methods to Select for the `CarRentalEmployee` role

Bean	Method Type	Methods to Select

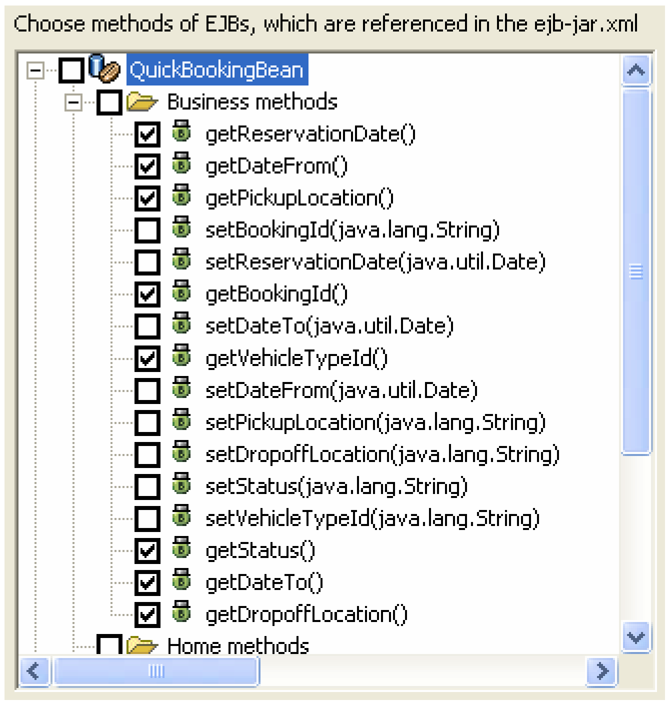
`QuickBookingBean`	`Business methods`	`getReservationDate()`
		`getDateFrom()`
		`getPickupLocation()`
		`getBookingId()`
		`getVehicleTypeId()`
		`getStatus()`
		`getDateTo()`
		`getDropoffLocation()`

	`Home methods`	None
	`Create methods`	`Create()`

	`Finder methods`	`findByPrimaryKey()`
		`findByStatus()`

`QuickOrderProcessorBean`	`Business methods`	`viewActiveBookings()`
	`Create methods`	`create()`

For an example of the business methods to select for the `QuickBookingBean`, see the figure below.


Choose methods of EJBs, which are referenced in the ejb-jar.xml

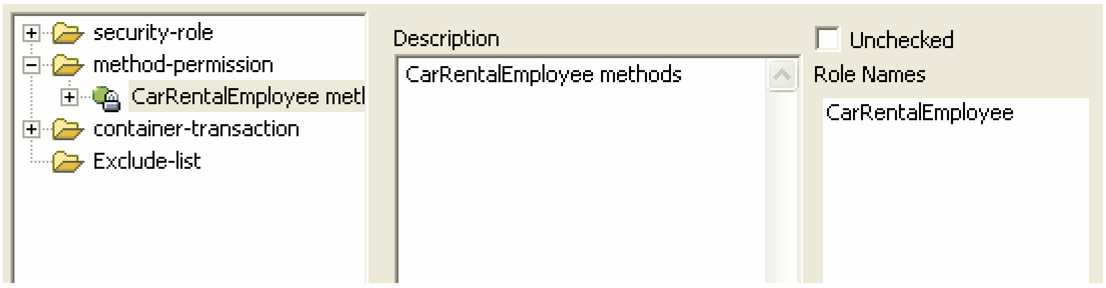
- ☐ QuickBookingBean
  - ☐ Business methods
    - ☑ getReservationDate()
    - ☑ getDateFrom()
    - ☑ getPickupLocation()
    - ☐ setBookingId(java.lang.String)
    - ☐ setReservationDate(java.util.Date)
    - ☑ getBookingId()
    - ☐ setDateTo(java.util.Date)
    - ☑ getVehicleTypeId()
    - ☐ setDateFrom(java.util.Date)
    - ☐ setPickupLocation(java.lang.String)
    - ☐ setDropoffLocation(java.lang.String)
    - ☐ setStatus(java.lang.String)
    - ☐ setVehicleTypeId(java.lang.String)
    - ☑ getStatus()
    - ☑ getDateTo()
    - ☑ getDropoffLocation()
  - ☐ Home methods

6. Choose *OK* to continue.

7. Rename this method by selecting the `method-permission` sub-node and entering a description in the *Description* text box, for example, **`CarRentalEmployee methods`**.
8. Deactivate the `Unchecked` indicator.

   The *Choose role-names* dialog appears.
9. Select the `CarRentalEmployee` role and choose *OK*.

   The role is added to the list of roles for this set of method permissions. See the figure below.



security-role
method-permission
CarRentalEmployee metl
container-transaction
Exclude-list

Description: CarRentalEmployee methods

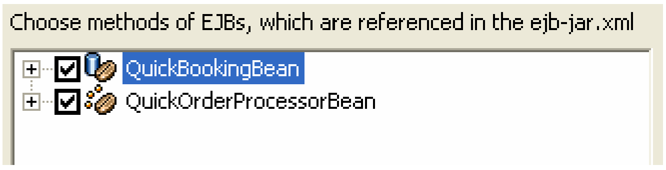
☐ Unchecked

Role Names: CarRentalEmployee

Selecting the EJB Methods for the BookingAgent Role

Repeat for the `BookingAgent` role:

1. Select the `method-permission` element and choose *Add*.
2. In the *Choose methods* dialog, select both of the beans and choose *OK*.



Choose methods of EJBs, which are referenced in the ejb-jar.xml

☑ QuickBookingBean
☑ QuickOrderProcessorBean

   This specifies that all of the methods for both beans apply to this set of method permissions.
3. Rename this set of method permissions by entering a description in the *Description* text box, for example, **`BookingAgent methods`**.
4. Deactivate the `Unchecked` indicator.

   The *Choose role-names* dialog appears.
5. Select the `BookingAgent` role and choose *OK*.

   The role is added to the list of roles for this set of method permissions.
6. Save the file.

Result

The `<method-permission>` elements are added to the deployment descriptor for the EJB. See the example below.

Note that some of the methods have been omitted in the example.

```
                <method-permission>
                        <description>CarRentalEmployee
methods</description>
```

```
                  <role-name>CarRentalEmployee</role-name>
                  <method>
                        <ejb-name>QuickBookingBean</ejb-name>
                        <method-name>getReservationDate</method-name>
                        <method-params/>
                  </method>
                  <method>
                        <ejb-name>QuickBookingBean</ejb-name>
                        <method-name>getDateFrom</method-name>
                        <method-params/>
                  </method>

         ...

                  <method>
                        <ejb-name>QuickBookingBean</ejb-name>
                        <method-name>getDropoffLocation</method-name>
                        <method-params/>
                  </method>

            </method-permission>
            <method-permission>
                  <description>BookingAgent methods</description>
                  <role-name>BookingAgent</role-name>
                  <method>
                        <ejb-name>QuickBookingBean</ejb-name>
                        <method-name>*</method-name>
                  </method>
                  <method>
                        <ejb-name>QuickOrderProcessorBean</ejb-name>
                        <method-name>*</method-name>
                  </method>
            </method-permission>
```

Next Step:

Catching the Access Control Error [Page 16]

6 Catching the Access Control Error

Use

Access to each of the EJB methods is now protected with a security role. Therefore, if a user who does not have the proper authorizations attempts to access the EJB methods, he or she will receive an error message.

However, the quick car rental servlet is currently designed to catch the `QuickCarRentalException` only, which is thrown by the EJB methods for various error conditions, while the access control error produced by the authorization check is thrown by the Web container. Therefore, you have to adjust the error control handling in the servlet to handle the error accordingly.

Prerequisites

- ☐ The quick car rental application's Web client project, *J2EE_QuickCarRentalWeb*, is displayed in the J2EE Explorer.

Procedure

1. Expand *J2EE_QuickCarRentalWeb* → *source* → *com* → *sap* → *engine* → *examples* → *servlets* → *quickcarrental*.
2. Open the *QuickReservationServlet.java* file with a double-click.
3. Insert a `catch (Exception e)` instruction block to each of the methods `viewAllBookings()`, `saveAction()`, and `cancelAction()`. See the code example below.

```
            try {

            ...

            } catch (QuickCarRentalException e) {

      session.setAttribute(Constants.CLIENT_MESSAGE,e.getMessage());
            } catch (Exception e) {
                        session.setAttribute(
                  Constants.CLIENT_MESSAGE,e.getMessage());
            }
```

4. Save the files.

Result

This instruction will catch the error messages returned from the Web container if the user cannot access the EJB methods.

Example

The following examples show the exception handling for each of the methods method.

Method viewAllBookings()

```
      private void viewAllBookings(
            HttpServletRequest request,
            QuickOrderProcessorLocal order) {
            HttpSession session = request.getSession(true);
            QuickBookingModel[] bookings;
            try {
                  bookings = order.viewActiveBookings();
                  session.setAttribute(
                  Constants.RESERVATIONS,
                  formatBookings(bookings));
            } catch (QuickCarRentalException e) {

      session.setAttribute(Constants.CLIENT_MESSAGE,e.getMessage());
            } catch (Exception e) {

      session.setAttribute(Constants.CLIENT_MESSAGE,e.getMessage());
            }
      }
```

Method saveAction()

```
      private void saveAction(
            HttpServletRequest request,
            QuickOrderProcessorLocal order) {
            HttpSession session = request.getSession(true);
            try {
            java.lang.String dateFrom =
request.getParameter("pickupDate");
            java.lang.String dateTo =
request.getParameter("dropoffDate");
            String vehicleTypeId =
request.getParameter("vehicleTypeId");

                  String pickupLocation =
request.getParameter("pickupLocation");
                  String dropoffLocation =
request.getParameter("dropoffLocation");
                  order.saveBooking(vehicleTypeId,dateFrom,
                        dateTo,
                        pickupLocation,
                        dropoffLocation);
            } catch (QuickCarRentalException e) {

      session.setAttribute(Constants.CLIENT_MESSAGE,e.getMessage());
            } catch (Exception e) {
                  session.setAttribute(
            Constants.CLIENT_MESSAGE,e.getMessage());
            }
      }
```

Method cancelAction()

```
      private void cancelAction(
            HttpServletRequest request,
            QuickOrderProcessorLocal order) {
            HttpSession session = request.getSession(true);

            String[] selectedBookings =
request.getParameterValues("check");
            for (int i = 0; i < selectedBookings.length; i++) {
                  try {
                        order.cancelBooking((String)
selectedBookings[i]);
                  } catch (QuickCarRentalException e) {

                        session.setAttribute(
                  Constants.CLIENT_MESSAGE,e.getMessage());
                  }  catch (Exception e) {

      session.setAttribute(Constants.CLIENT_MESSAGE,e.getMessage());
                }
            }
```

Next Step:

Rebuilding and Deploying the Application [Page 19]

7 Rebuilding and Deploying the Application

Use

After you have created the J2EE security roles for the JSP and the EJBs, you can rebuild and deploy your application.

Prerequisites

- ☐ The quick car rental application's projects are displayed in the *J2EE Explorer*.
- ☐ You have completed the steps for specifying the authentication methods, the security role information, and the security constraints.
- ☐ You have saved all files.

  ⚠ An asterisk (*) appears next to any file names for files that have not been saved.
- ☐ The J2EE Engine and SDM server are running and you can connect to the J2EE Engine from the Developer Studio.

Procedure

1. Choose *Project* → *Rebuild All*.

   This step rebuilds the EJB and Web projects.
2. You still have to rebuild the EAR file. Select the *J2EE_QuickCarRentalEar* project, open the context menu using the right mouse button, and choose *Build Application Archive*.
3. Confirm the message that the EAR has been built with *OK*.
4. Expand the *J2EE_QuickCarRentalEar* project.
5. To deploy the EAR file, select the *QuickCarRentalEar.ear* file, open the context menu and choose *Deploy to J2EE engine*.
6. If prompted, enter the SDM password.

   A message appears in the *Deploy Output View* section indicating the status of deployment.

Next Step:

Your application is now available on the J2EE Engine. However, before you can test the access protection, you have to create the test users and assign them the appropriate roles.

See Creating the Users [Page 19].

8 Creating the Users

Use

The quick car rental application is now protected with authentication and J2EE security roles. To access the application and the various methods, users must be assigned to the appropriate roles. In the next steps, you will act as the administrator to set up the users and corresponding role assignments.

In the first of the administrative steps, you will create the users:

- `Employee`
- `Agent`
- `OtherUser`

You will use these users for the role assignments and for testing the access protection.

Prerequisites

☐ The J2EE Engine is running.

☐ You have access to the Visual Administrator and can use it to connect to the J2EE Engine.

☐ You have a user ID with administrator rights on the J2EE Engine, for example, `Administrator`.

☐ There is a connection profile set up on the Visual Administrator that uses this user ID to connect to the J2EE Engine. For more information, see Logging on to SAP J2EE Engine [External].

Procedure

Connecting to the J2EE Engine Using the Visual Administrator

1. Start the Visual Administrator.
2. Select the connection profile that connects to the J2EE Engine using your administrator user and choose *Connect*.
3. Enter this user's password and choose *Connect*.

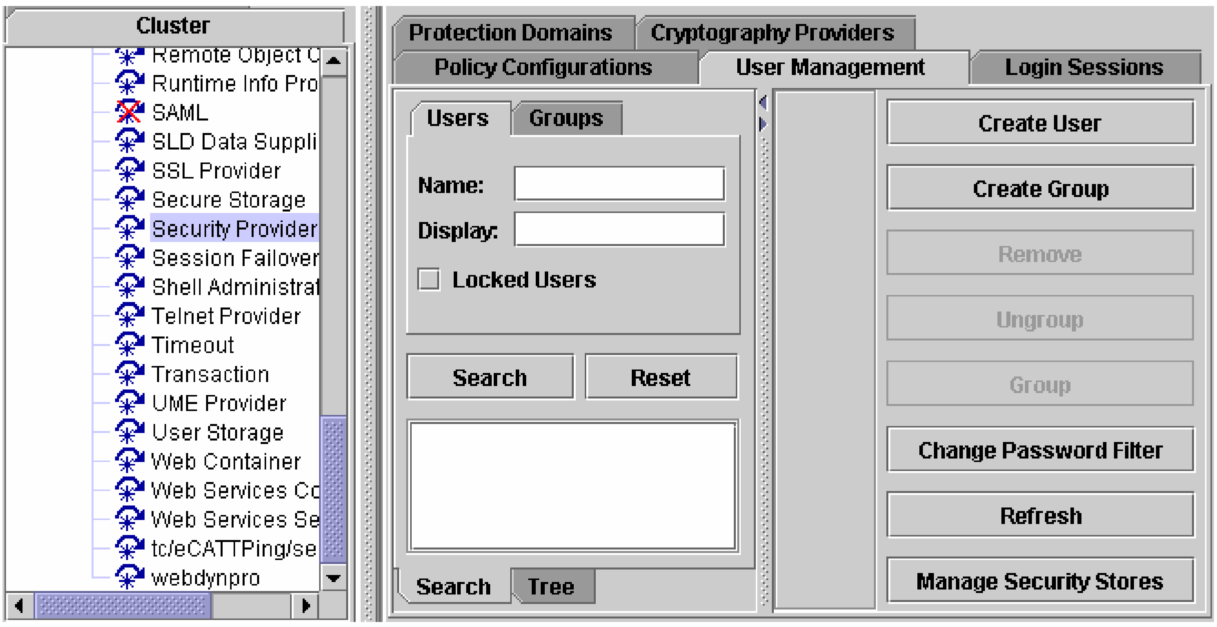
   The dispatcher and server entries appear in the Visual Administrator.

Creating the Users

1. Expand *<SID>* → *Server* → *Services*.
2. Select the Security Provider.

   The maintenance tab pages for the Security Provider appear in the right section of the screen.
3. Choose the *User Management* tab page.

   The *User Management* screen appears. See the figure below.

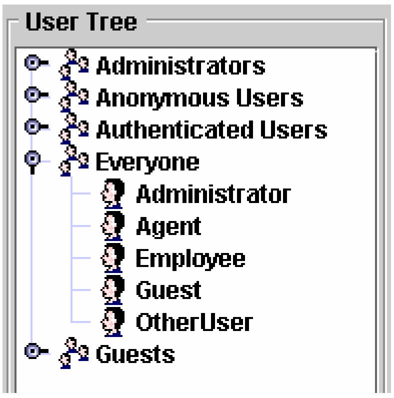


4. Search for the users `Employee`, `Agent` and `OtherUser` and create a user ID for each user that does not exist:
   a. For each user, enter the user ID in the *Name:* field and choose *Search*.
   b. If no ID for this user appears, then choose *Create User*.

      The *Create User* dialog appears.
   c. Enter the user ID for the user and an initial password. Repeat the password in the *Confirm password:* field.
   d. Choose *OK*.
   e. In the *Users* section, choose *Reset* to search for the next user.

   Repeat these steps for each user that does not exist.
5. To confirm that these users exist, choose the *Tree* tab page.
6. Expand the `Everyone` group.

These users should now exist in the group `Everyone`. See the figure below.



Next Step:

[Assigning Users to the J2EE Security Roles [Page 22]]

9 Assigning Users to the J2EE Security Roles

Use

Access to the quick car rental application is now protected with the J2EE security roles `AccessQuickCarRental`, `CarRentalEmployee`, and `BookingAgent`. Therefore, users must be assigned to the appropriate J2EE security roles to have access to the application and perform certain tasks.

In this step, you will assign the users you created in the last step to the corresponding J2EE security roles. The following rules apply:

- `Employee` and `Agent` are able to access the application. You will assign these users to the J2EE security role `AccessQuickCarRental`, which protects access to the JSP.
- `Employee` is able to view reservations. You will assign this user to the J2EE security role `CarRentalEmployee`, which protects access to the EJB methods. Users assigned to this role only have access to the methods used for displaying information.
- `Agent` is able to perform all actions. You will assign this user to the J2EE security role `BookingAgent`, which also protects access to the EJB methods. Users assigned to this role have access to all of the EJB's methods.
- `OtherUser` does not have access to the application.

Prerequisites

☐ The J2EE Engine is running.

☐ You are connected to the J2EE Engine as an administrator using the Visual Administrator.

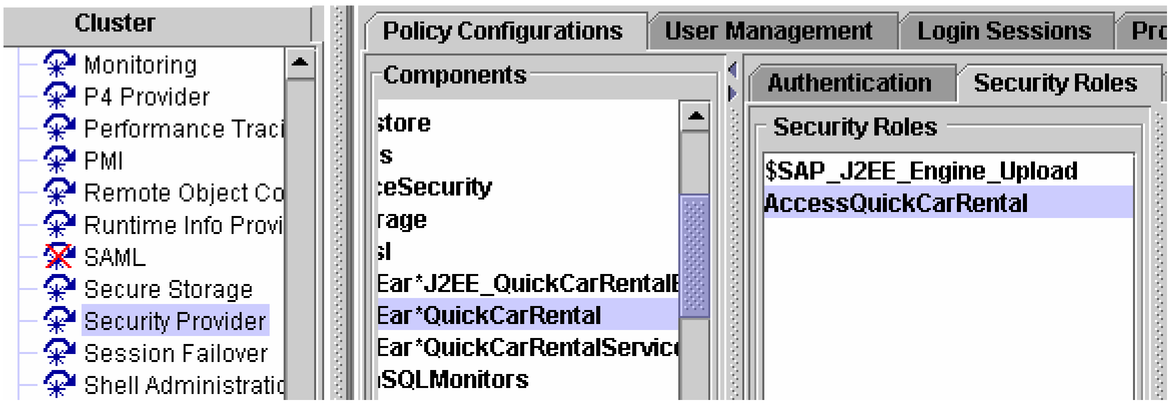
Procedure

Assigning the Users to the Role for Access to the JSP

1. Using the Security Provider, choose the *Policy Configurations* tab page.
2. Select the `QuickCarRental` application from the list of applications.
3. Choose the *Security Roles* tab page.

   The `AccessQuickCarRental` role appears in the *Security Roles* section.

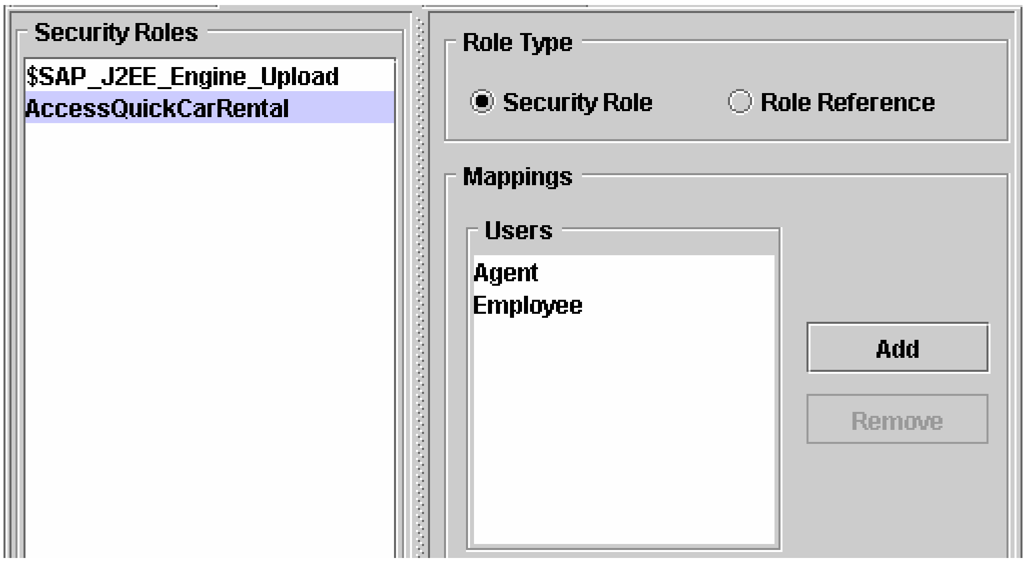
   See the figure below.



4. Select the `AccessQuickCarRental` role.
5. In the *Mappings / Users* section, choose *Add*.
6. In the dialog that follows, search for the user **`Agent`**.
7. Select `Agent` and choose *OK*.
8. Repeat steps 5-7 for the user `Employee`.

Do not assign the user `OtherUser` to the security role.

The users `Agent` and `Employee` are added to the *Users* section. See the figure below.



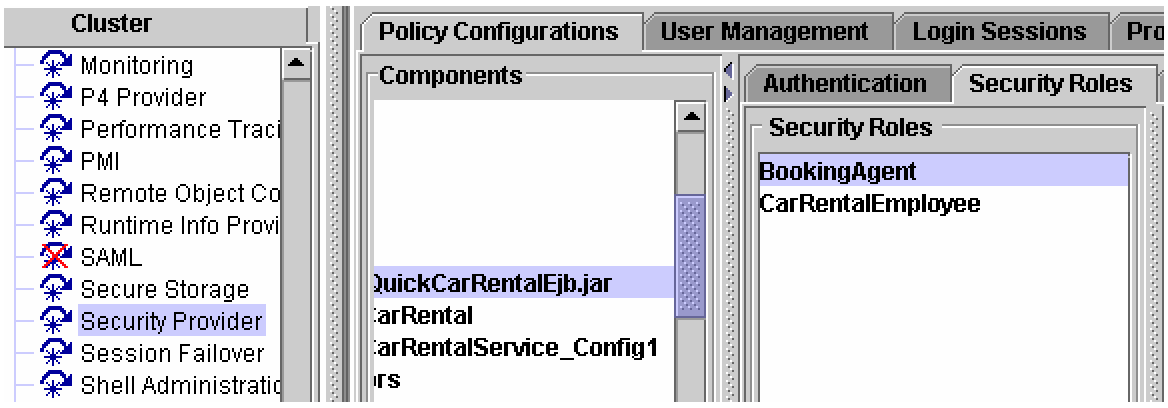
Assigning the Users to the Role for Access to the EJB Methods

1. Under *Components*, select the `J2EE_QuickCarRentalEjb.jar` application from the list of applications. (Keep the *Security Roles* tab page open.)

   The `BookingAgent` and `CarRentalEmployee` roles appear in the *Security Roles* section.

   See the figure below.

Selecting the EJB Methods for Each J2EE Security Role



2. Under *Security Roles*, select the `BookingAgent` role.
3. In the *Mappings / Users* section, choose *Add*.
4. In th dialog that follows, search for the user **`Agent`**.
5. Select `Agent` and choose *OK*.

   The user `Agent` is added to the *Users* section.
6. Select the `CarRentalEmployee` role.
7. In the *Mappings / Users* section, choose *Add*.
8. In the dialog that follows, search for the user **`Employee`**.
9. Select `Employee` and choose *OK*.

   The user `Employee` is added to the *Users* section.

Result

`Agent` and `Employee` can access the application. `Employee` can view reservations; `Agent` can also create and cancel reservations. No other users are allowed to access the application, for example, `OtherUser`.

Next Step:

Testing the Access Protection [Page 24]

10 Testing the Access Protection

Use

You are now ready to test the application for this part of the tutorial. You will make sure that `Employee` and `Agent` are able to access the application, but not the user `OtherUser`.

Prerequisites

- ☐ The J2EE Engine is running.
- ☐ The quick car rental application is deployed on the J2EE Engine.
- ☐ The users `Employee` and `Agent` are assigned to the `AccessQuickCarRental` security role for the JSP.

- ☐ The user `Employee` is assigned to the `CarRentalEmployee` security role for the EJB.
- ☐ The user `Agent` is assigned to the `BookingAgent` security role for the EJB.

Procedure

1. Execute the car rental application by entering its URL in the Web browser.

   `http://localhost:50000/QuickCarRental`

   You are prompted for user ID and password.
2. Enter the user ID and password for `Employee`.

   If the user's password is initial, then you must specify a new one for the user. Enter a new password and choose *Change password*.

   `Employee` has access to the JSP application. The Web browser displays the initial input screen for the car rental application.
3. Attempt to create a reservation. Enter data and choose *Add Reservation*.

   You receive an error message that the user `Employee` does not have access to the EJB method.
4. Close your Web browser.
5. Repeat the steps for the user `Agent`.

   The user `Agent` is also able to access the application.
6. Attempt to create a reservation. Enter data and choose *Add Reservation*.

   `Agent` is able to create and cancel reservations.
7. Close your Web browser and repeat the steps for the user `OtherUser`.

   The user `OtherUser` cannot access the application and receives an error.

Result

You are now finished with this tutorial. If you want to see how to use UME permissions for access protection, see one of the other tutorials:

Protecting Access to the J2EE-Based Application Using UME Permissions [External]

Protecting Access to the Web Dynpro Application Using UME Permissions [External]