

IBM WEBSPHERE WORKSHOP - LAB EXERCISE

WebSphere 4.0 - Security Lab

What This Exercise is About

This lab walks you through the administrative process for applying security, referring to the MyBank Application (from the previous lab). In this lab, you will add security information to the EAR file and then install and test the secured EAR file in WebSphere Advanced Edition.

User Requirement

Windows NT 4 ServicePack 6A
IBM DB2 Universal Database Version 7 with FixPack 2a
WebSphere Advanced Edition V4.0
MyBank Enterprise Application EAR file

What You Should Be Able to Do

You will use the application assembly tool (AAT) to declare and define J2EE security roles, as well as to control authorization on various J2EE modules. You will also enable security in the application server runtime and test your settings. Authentication will be performed using the local operating system user registry. This example only uses declarative security. It does not illustrate any of the programmatic methods supported by the J2EE programming model.

Introduction

J2EE 1.2 introduces the notion of security roles to encapsulate the grouping of method permissions. Authorization model is changed from Permission based model to Role based model. Security role is a semantic grouping of permissions that a given type of users of the application must have to use the application.

Exercise Instructions

This lab is divided into five parts:

Part One: Create the roles Banker_role and Customer_role and secure the EJB jar in the MyBank Application with these roles. Assign method permissions in the EJB jar such that

Banker_role is authorized to Create an Account and Transfer funds. Assign method permissions such that Customer_role is authorized to only Transfer funds.

Part Two: Secure the Web Module of the MyBank Application. Assign security constraints and web resource collections such that only Banker_role has access to the CreateAccount Servlet.

Part Three: Enable Global Security and Secure access to the Admin Console with local Operating System Authentication.

Part Four: Install Enterprise Application and test with a Java Client and a Servlet.

You will be performing two J2EE Roles - Assembler and Deployer - in this lab. You will be doing Part One and Part Two in the Application Assembler's role. Part Three and Part Four are the Deployer's responsibilities.

Part One: Create Roles and Secure the EJB module

__1. You will use WebSphere 4.0 Advanced Edition in this lab. Stop WebSphere 4.0 Single Server that you were using in the previous lab.

- __ Go to your command prompt.
- __ Enter **cd C:\WebSphere\SingleServer\bin.**
- __ Enter **stopserver.**

__2. Let's Create two new users : Banker_user and Customer_user. In your Start menu, click **Programs -> Administrative Tools -> User Manager.**

__3. Create user Banker_user:

- __ Select **Guest.** In the Menu bar, select **User -> Copy**
- __ In the Copy of Guest Dialog, Enter the following:

User Name: **Banker_user**
Password: **password**
Confirm Password: **password**

- __ Click **Add.**
- __ Click **Close.**

__4. Create user Customer_user:

- __ Select **Guest.** In the Menu bar, select **User -> Copy**
- __ In the Copy of Guest Dialog, Enter the following:

User Name : **Customer_user**
Password : **password**
Confirm Password : **password**

- __ Click **Add**.
- __ Click **Close**.

__5. Start the Application Assembly Tool (AAT) with the command:

- __ Go to your command prompt.
- __ Enter **cd C:\WebSphere\AdvancedEdition\bin**
- __ Enter **assembly**

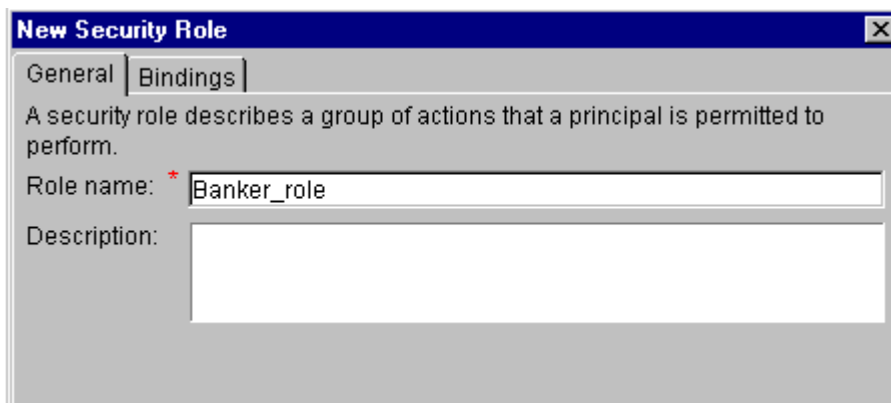
__6. Define Security Roles in the .ear File.

- __ Cancel the **Welcome to Application Assembly Tool** window
- __ Load the MyBank application into the tool. Select **File -> Open**, then navigate to **c:\MyBank\Solutions\MyBankApp.ear** from the MyBank Application scenario.

NOTE: The above EAR file is a solution to Lab 1, however, the file you created in Lab 1 can also be used if it was successful.

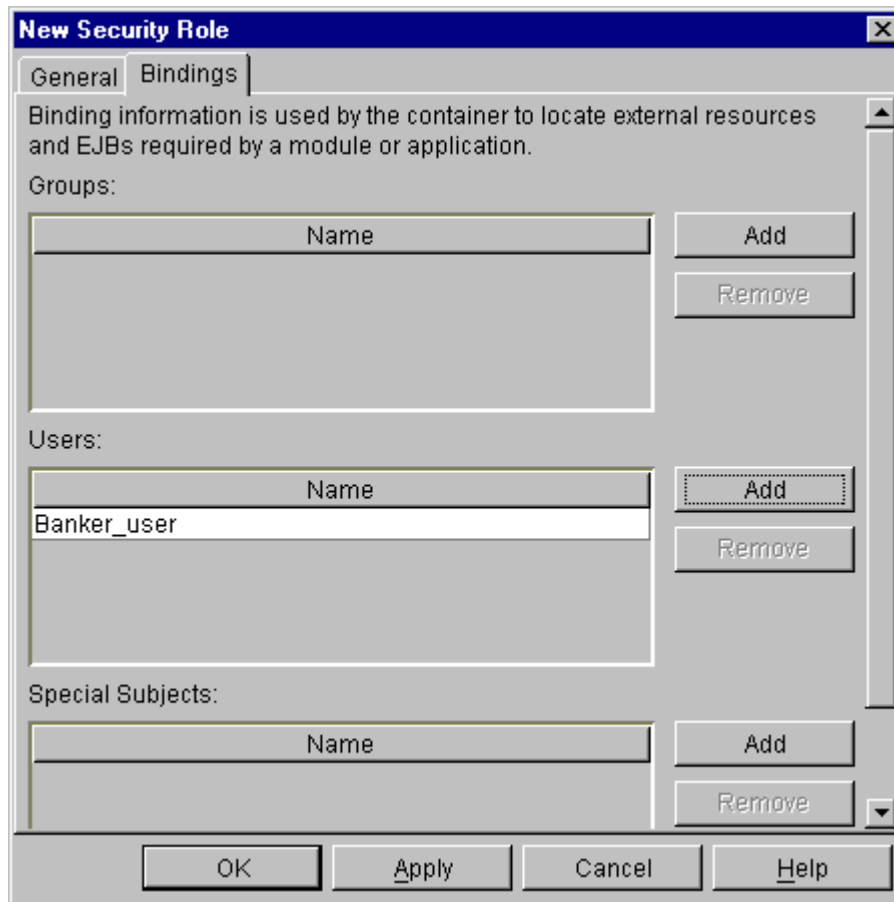
__7. Create a security role **Banker_role** to which you will be granting authorization to the EJBs and servlets in the application. Bind the user **Banker_user** to this role.

- __ From the .ear file, select the **Security Roles** folder. Right-click it.
- __ Select **New**.
- __ On the **General** tab, enter **Banker_role** for the **Role Name**. Optionally, enter a Description.



- __ On the **Binding** tab, associate the user ID **Banker_user** with this security role by selecting the **Add** button next to the **Users** area.

__ Specify **Banker_user** in the **Name** field. Click **OK**.



__ Click **OK** on the **Binding** tab.

__8. Create a security role **Customer_role**. Bind the user **Customer_user** to this role.

__ From the .ear file, select the **Security Roles** folder. Right-click it.

__ Select **New**.

__ On the **General** tab, enter **Customer_role** for the **Role Name**. Optionally, enter a Description.

__ On the **Binding** tab, associate the user ID **Customer_user** with this security role by selecting the **Add** button next to the **Users** area.

__ Specify **Customer_user** in the **Name** field. Click **OK**.

__ Click **OK** on the **Binding** tab.

__9. Authorize methods on the EJBs. You will authorize all home and remote methods in the Account bean and Transfer bean to security role Banker_role. You will authorize all

home and remote methods in Transfer bean and a few methods on Account bean to security role Customer_role.

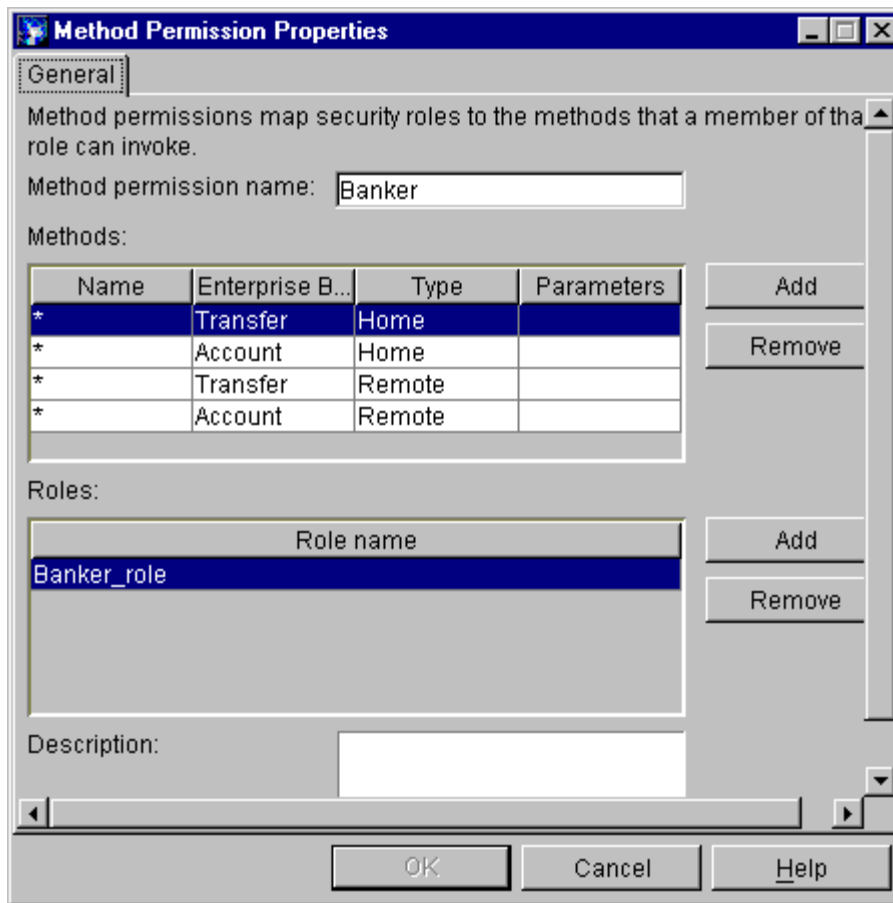
- ___ Expand the **EJB Modules** folder.
- ___ Expand the **MyBank EJB** Module.

___ 10. Create a new Method Permission. You will create a method permission named Banker which will have all the methods in the home and remote interfaces of both Account bean as well as Transfer bean.

- ___ Select **Method Permissions**. Right click it.
- ___ Select **New**. Give the name **Banker** in the **Method Permission Name** box.
- ___ In the **Methods** area, click **Add**.
- ___ In the **Add** dialog, expand **Deployed_MyBankEJB.jar**. Expand **Transfer**.
- ___ Select the **Home** interface.
- ___ Click **Apply**. No pop-up panel will appear but changes will be added.
- ___ Select the **Remote** interface. Click **Apply**.
- ___ Expand **Account**.
- ___ Select the **Home** interface. Click **Apply**.
- ___ Select the **Remote** interface. Click **Apply**.
- ___ Click **Cancel** button to exit.

___ 11. In the **Roles** area, add the security role(s) for the method permission.

- ___ In the **Roles** area , click **Add**.
- ___ Select Security Role **Banker_role**.
- ___ Click **OK**.

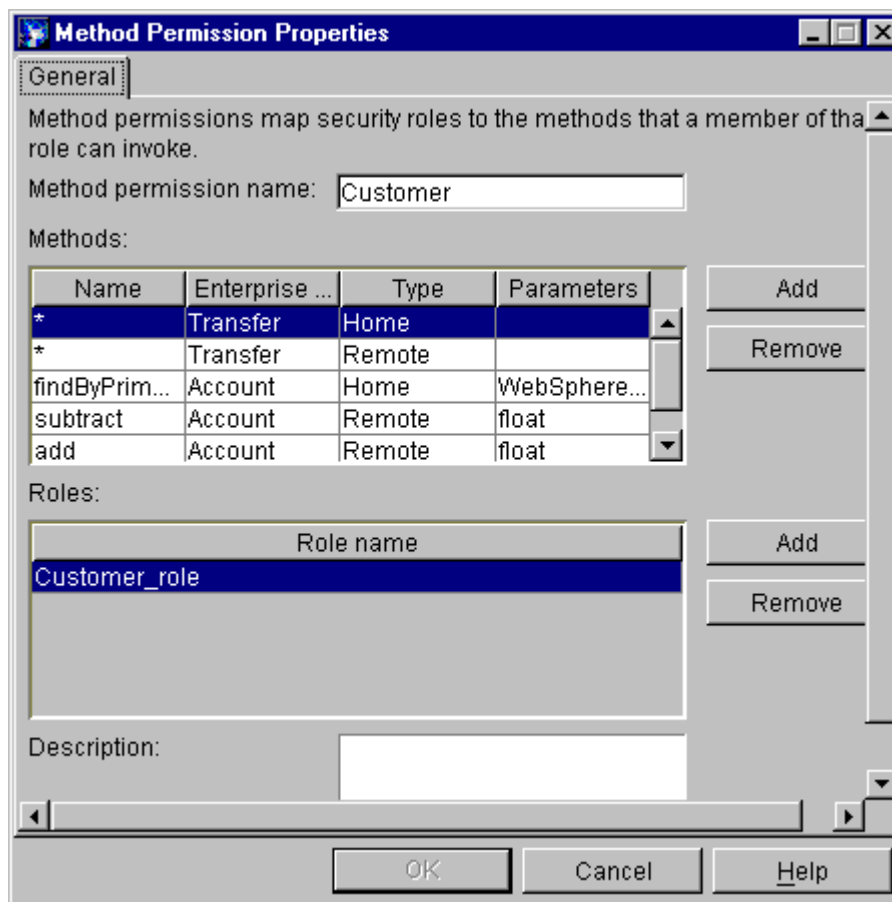


- __12. Click **OK** to close the Method Permissions dialog. You have now authorized **Banker_role** to all the methods in the home as well as remote interfaces of both Account and Transfer bean. Hence a user in this role is authorized to Create an Account, Get Balances and Transfer Amounts from one account to another.
- __13. Authorize methods to security role **Customer_role**. Create a new Method Permission. You will create a method permission named Customer which will have all the methods in the Transfer bean but does not have the methods in the Account bean that are required for Creating an Account.
- __ Select **Method Permissions**. Right click it.
 - __ Select **New**. Give the name **Customer** in the **Method Permission Name** box.
 - __ In the **Methods** area, click **Add**.
 - __ In the **Add** dialog, expand **Deployed_MyBankEJB.jar**. Expand **Transfer**.
 - __ Select the **Home** interface.
 - __ Click **Apply**. No pop-up panel will appear but changes will be added.
 - __ Select the **Remote** interface. Click **Apply**.
 - __ Expand **Account**.

- ___ Expand the **Home** interface. Select the method `findByPrimaryKey(..)`. Click **Apply**.
- ___ Expand the **Remote** interface. Select the method `add(float)`. Click **Apply**.
- ___ Select the method `getBalance()`. Click **Apply**.
- ___ Select the method `subtract(float)`. Click **Apply**.
- ___ Click **Cancel** button to exit.

___14. In the **Roles** area, add the security role(s) for the method permission.

- ___ In the **Roles** area , click **Add**.
- ___ Select Security Role **Customer_role**.
- ___ Click **OK**.



___15. Click **OK** to close the Method Permissions dialog. You have now authorized **Customer_role** such that a user in this role can transfer amounts from one account to another and also get balances from accounts. However, Method Permissions settings for **Customer_role** have restricted a user in this role from creating an account.

You have now created two roles `Banker_role` and `Customer_role`. You have bound the user `Banker_user` to `Banker_role` and the user `Customer_user` to `Customer_role`. You have also secured the EJBs in the MyBank Application.

Part Two: Secure the Web Module

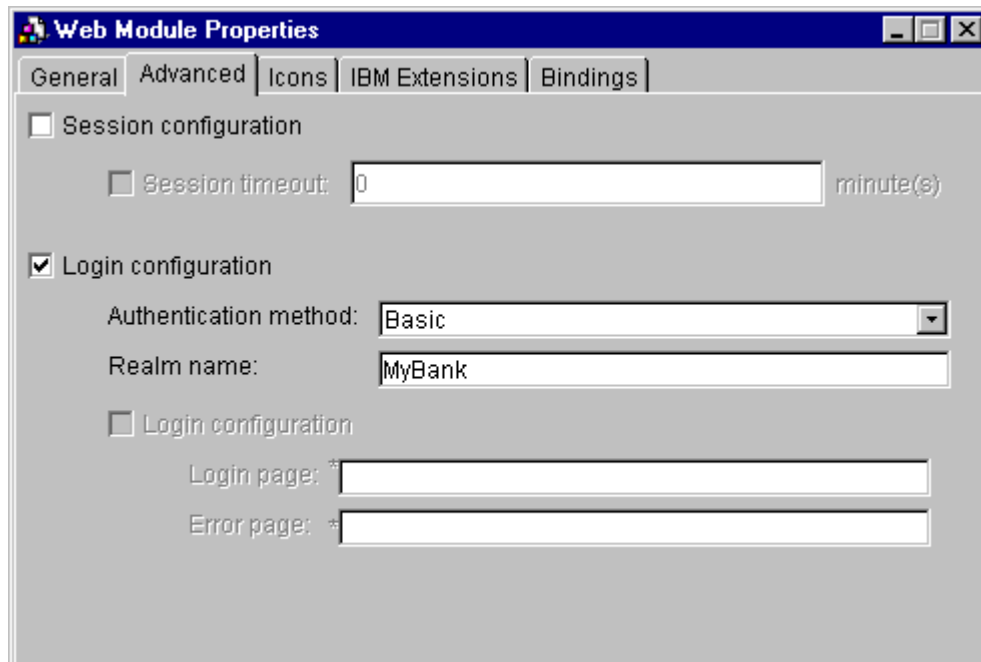
In this part you will secure the Web Module portion of the MyBank Application.

__1. Authorize the servlets.

- __ Expand **Web Modules**. Select **MyBank Web Module**.
- __ Right Click it. Select **Properties**
- __ Click on the **Advanced** tab.
- __ Verify that the **Login Configuration** box is checked.
- __ Enter the following:

Authentication Method: **Basic**
Realm Name: **MyBank**

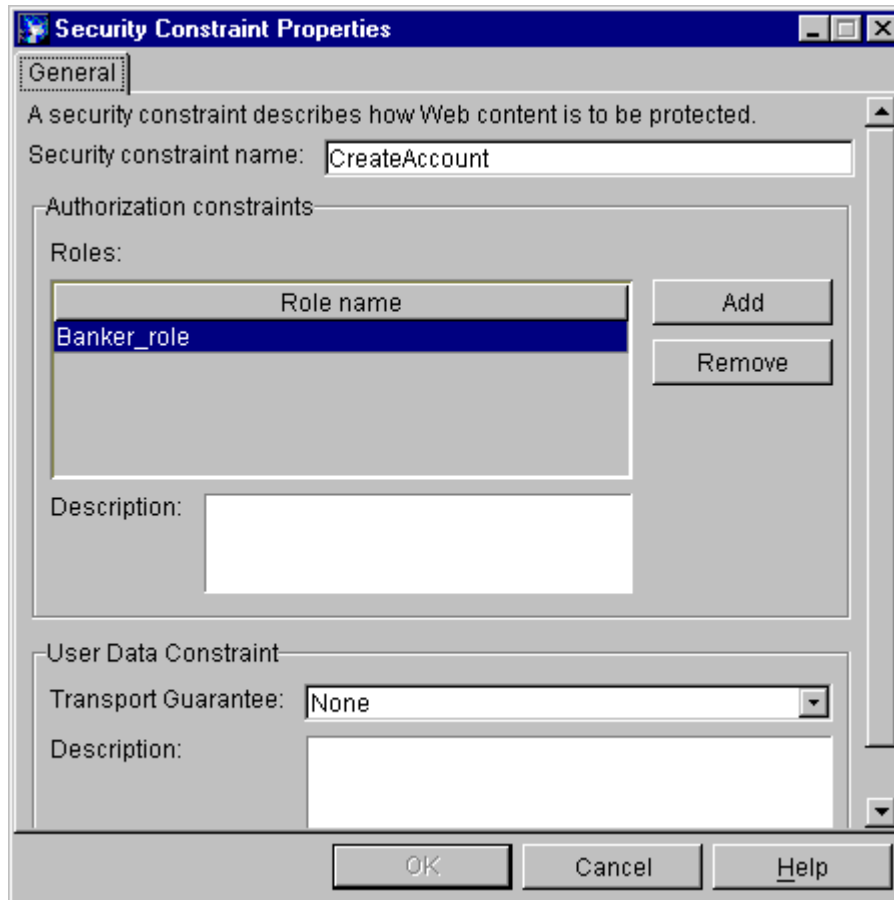
You have just selected the Basic Authentication method. Basic Authentication uses the challenge box that the browser pops up.



__ Click **OK**.

__2. Add a security constraint. (for creating an account)

- __ Expand Web Modules.
- __ Expand the MyBank Web Module file.
- __ Select **Security Constraints**. Right-click it.
- __ Click **New**. Enter **CreateAccount** as the Security Constraint name.
- __ Click **Add** (next to Roles).
- __ Select **Banker_role**. Click **OK**.
- __ Set Transport Guarantee to **NONE**.

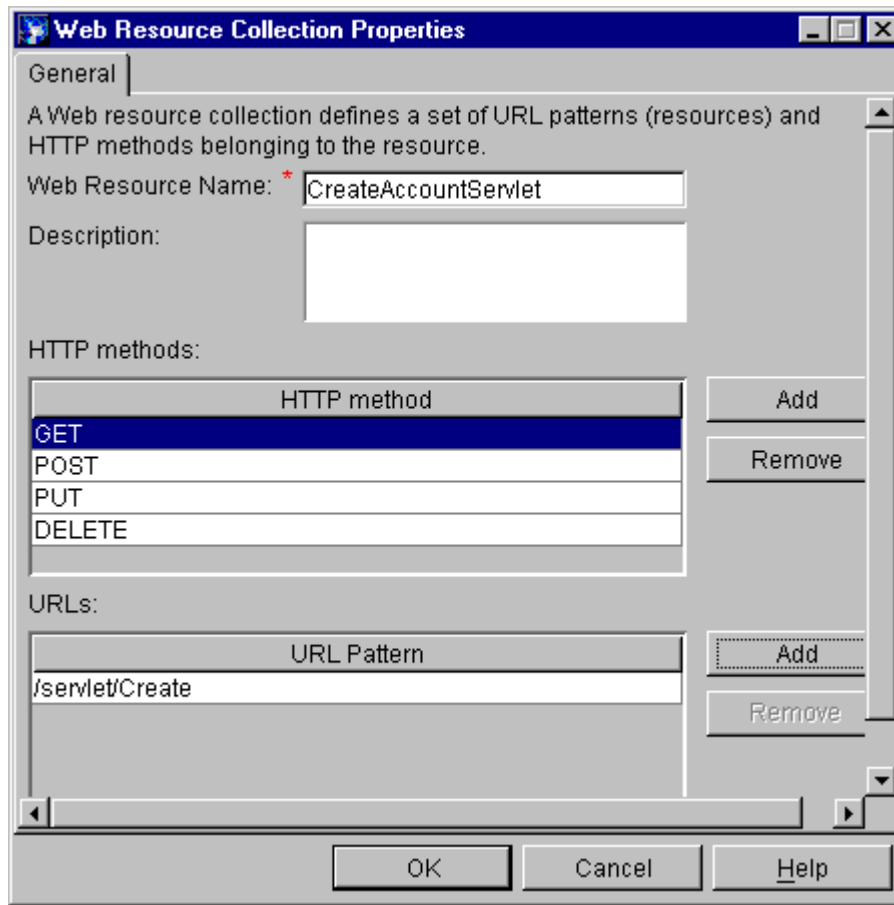


- __ Click **OK**.

__3. Add a resource collection to the constraint.

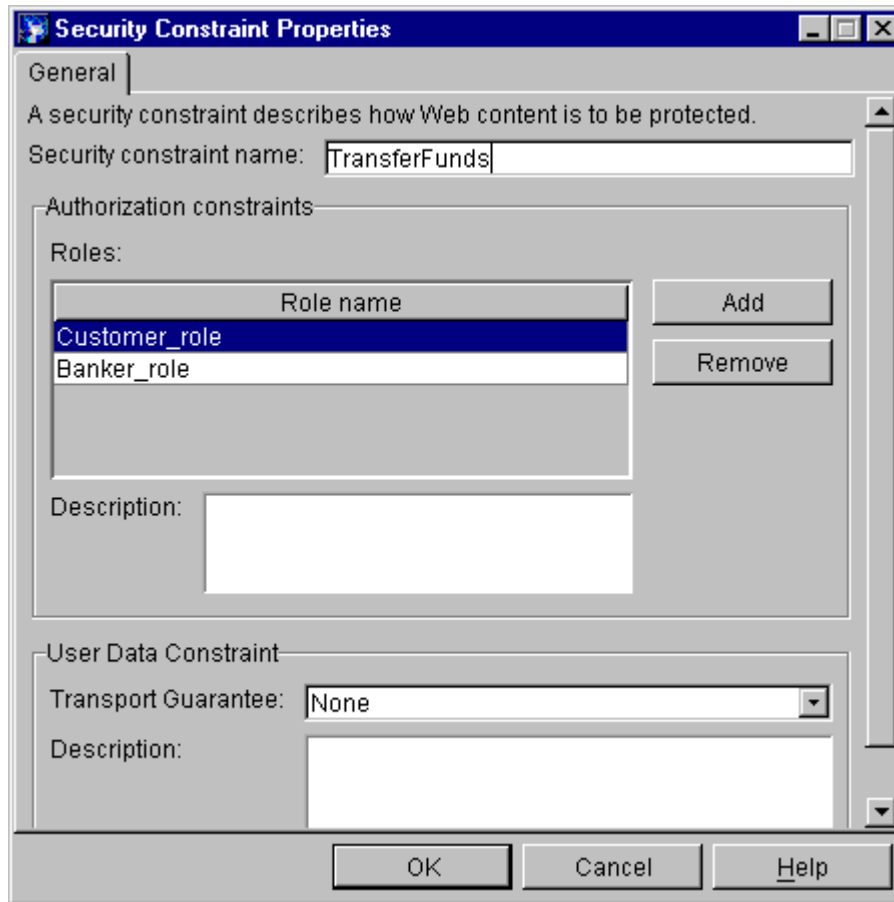
- __ Expand **Security Constraints**. Expand **CreateAccount**.
- __ Select **Web Resource Collections**. Right-click it.
- __ Click **New**.
- __ Enter a **Web Resource Name** for the collection, such as **CreateAccountServlet**.
- __ Click **Add** next to HTTP Methods
- __ Select **POST** and click **OK**
- __ Click **Add** next to HTTP Methods

- __ Select **GET** and click **OK**.
- __ Add the **PUT** and **DELETE** methods also in a similar manner.
- __ Click **Add** next to URL Pattern.
- __ Enter **/servlet/Create** in the URL Pattern field.
- __ Click **OK**.



- __ Click **OK**.

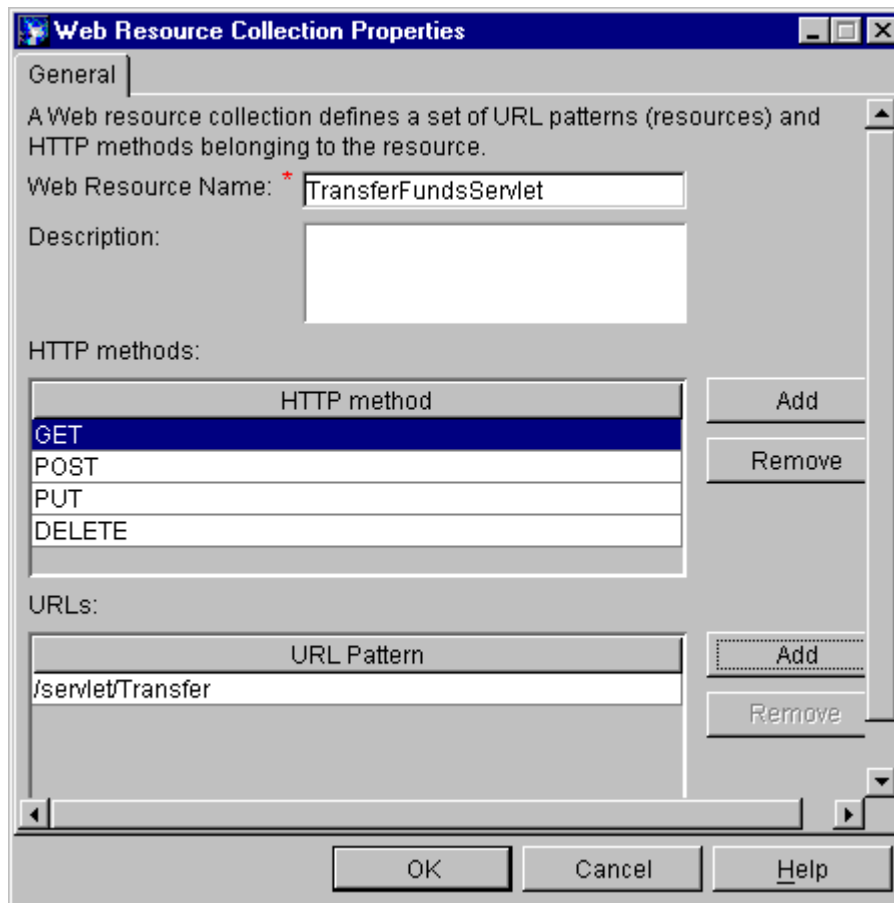
- __4. Add a security constraint. (for transferring funds between accounts)
 - __ Expand Web Modules.
 - __ Expand the MyBank Web Module file.
 - __ Select **Security Constraints**. Right-click it.
 - __ Click **New**. Enter **TransferFunds** as the Security Constraint name.
 - __ Click **Add** (next to Roles).
 - __ Select **Banker_role**. Click **OK**.
 - __ Click **Add** (next to Roles).
 - __ Select **Customer_role**. Click **OK**.
 - __ Set Transport Guarantee to **NONE**.



__ Click **OK**.

__5. Add a resource collection to the constraint.

- __ Expand **Security Constraints**. Expand **TransferFunds**.
- __ Select **Web Resource Collections**. Right-click it.
- __ Click **New**.
- __ Enter a **Web Resource Name** for the collection, such as **TransferFundsServlet**.
- __ Click **Add** next to HTTP Methods
- __ Select **POST** and click on **OK**
- __ Click **Add** next to HTTP Methods
- __ Select **GET** and click on **OK**.
- __ Add the **PUT** and **DELETE** methods also in a similar fashion.
- __ Click **Add** next to URL Pattern.
- __ Enter **/servlet/Transfer** in the URL Pattern field.
- __ Click **OK**.



__ Click **OK**.

- __6. Save your .ear file using **File->Save As**. Browse to c:\MyBank. Call it **Secured_MyBankApp.ear**
- __7. Click OK on the "Archive Saved Successfully" dialog box.
- __8. Exit the AAT. **File->Exit**.

You have secured the Web Module portion of the MyBank Enterprise Application. No additional assembly tasks are needed to secure J2EE application clients.

Part Three: Enable Global Security and Secure Access to Administrative Console

In this part you will secure access to the Admin Console with local operating system authentication

__1. Start the admin server: In your command prompt, Go to **c:\WebSphere\AdvancedEdition\bin**. Enter **adminserver**. Wait for the “open for e-business” message. You can also start the Admin Server from the Windows NT services or from the Windows **Start** button. Select **Programs->IBM WebSphere->Application Server V4.0 -> Start Admin Server**

__2. Open the Administrative console:

Start -> Programs -> IBM WebSphere -> Advanced Edition V4.0 -> Administrator's Console.

You can also start the Admin Console using the batch file **adminclient**.

__3. Enable security in the WebSphere Advanced Administrative Console.

__ Select **Console -> Security Center** .

__ On the **General** tab, check the **Enable Security** box.

__ On the **Authentication** tab, verify **Local Operating System** radio button is selected.

__ Enter **Security Server ID** and **Security Server Password**.

Security Server ID: **db2admin**

Security Server Password: **was1edu**

__ Click **Apply**.

__ Click **OK** on warning that the change will not take effect until the admin server is restarted. Click **OK**.

__4. Stop the **Admin Server** by right-clicking on the node and selecting **Stop**. Click **Yes** on the Confirmation Dialog box.

__5. Restart the Admin Server.

__6. Restart the Admin Console.

__7. When prompted, enter the username and password used when security was enabled.

The Admin Console will now use the local operating system for authentication.

Part Four: Install secured Enterprise Application and Test

In this part you will install the secured Enterprise Application and test it

- 1) From the servlet through a web browser
- 2) From the Java Application Client

__ 1. First, let us assign a database to be used by our enterprise application. Use the console to configure a new data source. In the tree on the left side of the console expand

WebSphere Administrative Domain->Resources->JDBC Providers->Sample DB Driver->Data Sources.

__ 2. Right-click **Data Sources**. Select **New**. Enter the following values.
(If a field value is not specified, leave the field blank)

__ Name: **BankDataSource**
__ JNDI Name: **jdbc/MyBank**
__ Database Name: **BankData**
__ User: **USERID**
__ Password: **PASSWORD**
__ Confirm Password: **PASSWORD**

Click **OK**. Click **OK** on the Information dialog box.

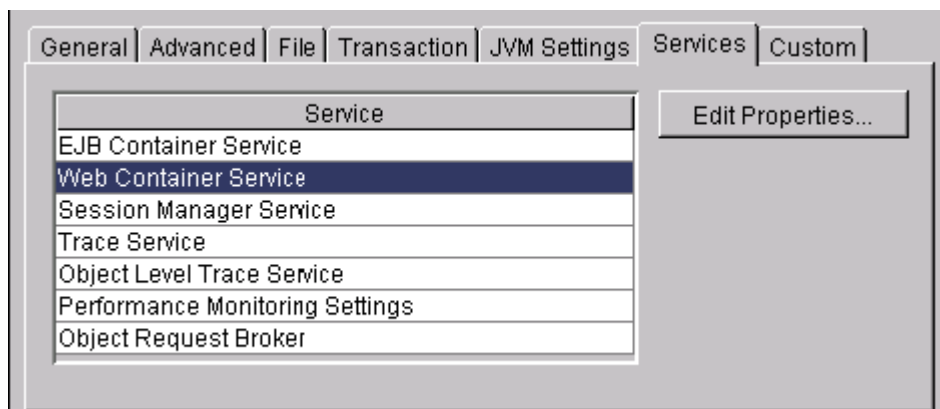
__ 3. Now, Let's create an Application Server where we will install Secured_MyBankApp.ear. Right click on the Application Servers folder and select **New**.

__ In the **General** tab, enter **MyBankServer** as your **Application Server Name**.

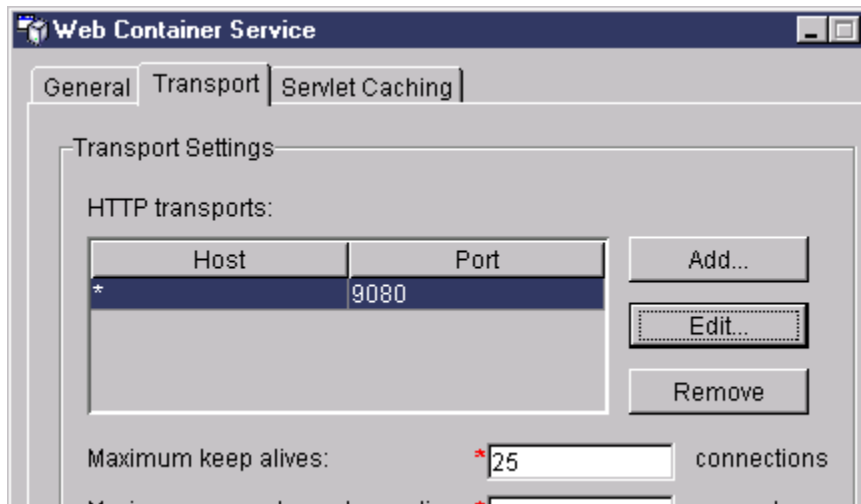
__ Click on the **File** tab. Enter the following:

Standard Output: **C:\WebSphere\AdvancedEdition\logs\MyBank_stdout.txt**
Standard Error: **C:\WebSphere\AdvancedEdition\logs\MyBank_stderr.txt**

__ Click on the **Services** tab. Select **Web Container Service** and click on **Edit Properties**



__ Click on the Transport tab - you should get to this screen:



- __ Select the “* 9080” HTTP Transport and click **Edit**.
 - __ Verify that the **Transport port** defaulted to **9081** (It could be different from 9081. Each Application server needs to listen on a unique port)
 - __ Click **OK** (or **Cancel**) to close the dialog and then **OK** again to close the transports.
 - __ Click **OK** to complete the creation of the Application Server.
 - __ You should get a successful completion message.
- __4. From the main tree in the admin console, select the **Virtual Hosts** folder.
- __ On the right pane, click on **Add** by the Aliases.
 - __ Specify “*:**9081**” (or any other port you may have used in the previous step)
 - __ Click Apply.
- __5. Let’s install the secured enterprise application. Select the Wizard icon and select **Install Enterprise Application**.
- __6. Verify the **Install Application (*.ear)** radio button is selected.
- __ Click **Browse** and navigate to **c:\MyBank** and select **Secured_MyBankApp.ear**.
 - __ Click **Open**.
 - __ Enter Application Name as **MyBankApp**.
 - __ Click **Yes** on the Unprotected Methods dialog box.
 - __ Since we have already given all the binding information using the Application Assembly Tool, we need not give this information again. Continue to click **Next** until the **Selecting the Application Server** window is displayed.
 - __ Hold the <shift key> down and select both MyBankEJB Module and MyBankWeb Module
 - __ Click on **Select Server** button.
 - __ Select **MyBankServer** and click **OK**. Click **Next**.

- __7. Click **Finish**. Select **No** on the Regenerate Code pop up box.
- __8. Once the informational box appears for Enterprise Application Install Completed Successfully, click **OK**
- __9. Start the MyBankServer

Expand **Nodes** -> **<Node-Name>** -> **Application Servers** -> **MyBankServer**.

Click **Start** button on the toolbar or right click and select **Start**. Click **OK** on the Informational Dialog box that reports successful completion.
- __10. Let's first test the application with the Servlet client. Open a browser window.
- __11. Type in <http://localhost:9081/MyBank/index.html>
- __12. Click on the **Create an Account** link. When the Create an Account page comes up, Enter 100 for Account number and 1000 for a starting balance.
- __13. Click **Create**.
- __14. You will be challenged for a userid and password. Enter **Banker_user** and **password**. Click **OK**.



- __15. Since the Banker_user belongs to Banker_role which is authorized to creating an account, the account will be created. Create one more account with Account number as 200 and 2000 as starting balance.

- __16. Now, click on the **Transfer funds** link. Since Banker_role is authorized to transfer funds, you will be able to transfer funds. Enter **10** in the **Amount** field, **100** in the **From Account** field and **200** in the **To Account** field.
- __17. Click **Transfer**. The transfer funds operation will be successful.
- __18. Now, exit the browser window. Since the browser caches the userid and password , we need to exit the browser in order to try with another userid and password.
- __19. Open a browser window. Type in <http://localhost:9081/MyBank/index.html>
- __20. Click on the **Create an Account** link. When the Create an Account page comes up, Enter 300 for Account number and 1000 for a starting balance.
- __21. Click **Create**.
- __22. You will be challenged for a userid and password. Enter **Customer_user** and **password**.
- __23. Since the Customer_user belongs to Customer_role which is not authorized to create an account, the account will not be created. The default error page will show up on the browser window.
- __24. Click on the browser's **Back** button.
- __25. Now, click on the **Transfer funds** link. Since Customer_role is authorized to transfer funds, you will be able to transfer funds. Enter **10** in the **Amount** field, **100** in the **From Account** field and **200** in the **To Account** field.
- __26. Click **Transfer**. The transfer funds operation will be successful.
- __27. Exit the browser window. You can try to invoke the servlet again. Give some other userid and password other than Customer_user and Banker_user. You will be unauthorized to all the operations.
- __28. Let's now test the application with the Java Application client. In your command prompt, change directories to **c:\WebSphere\AdvancedEdition\bin**
- __29. Launch the client in a client container by issuing the following command:
launchclient c:\MyBank\Secured_MyBankApp.ear.

- __30. You will be challenged for a userid and password. Enter **Banker_user** and **password**. Click **OK**.

- __31. Since the **Banker_user** is authorized to transfer funds and get balances, you will be able to do both.

- __32. Exit the Client. Launch the client in a client container again by issuing the following command:
launchclient c:\MyBank\Secured_MyBankApp.ear.

- __33. You will be challenged for a userid and password. Enter **Customer_user** and **password**. Click **OK**.

- __34. Since the Customer_user is also authorized to transfer funds and get balances, you will be able to do both.

What you did in this exercise

You applied security to the MyBank Application. You authorized two roles: Banker_role and Customer_role. Users belonging to Banker_role can create an account, get balances and Transfer funds between accounts. Users belonging to Customer_role can only get balances and Transfer funds between accounts.