16

Securing Java EE Applications with JAAS

Objectives

After completing this lesson, you should be able to do the following:

- Explain the Java EE application security design
- Describe Java Authentication and Authorization Service (JAAS)
- Implement JAAS security for Web applications and Enterprise JavaBeans (EJB)

Goals of Java EE Security Architecture

- Lessen the burden of the application developer in securing applications
- Ensure fine-grained access control to EJB resources
- Enable portable and secured Web applications and EJBs

Overview of Java EE Security Architecture

Use JAAS APIs to:

- Authenticate a client to access the system
  - Define security identities (principals and users), groups, and roles to identify clients to the container.
  - Associate principals to the client to enable access to the bean methods.
- Authorize clients to access the bean methods
  - Define logical roles, set method permissions, and map roles to users in the deployment descriptors.
  - Use containers to authorize the client requests to the methods.

Java Authentication and Authorization Service (JAAS)

JAAS is a framework that:

- Provides a Java API package to enable applications to authenticate and enforce security
- Allows definition of logical security names that are mapped in deployment descriptors to users or roles defined in the run-time environment
- Controls access to Web applications based on URL patterns
- Allows fine-grained authorization to manage how clients can access bean methods

- A JAAS provider implements the JAAS framework and applies the Java 2 Security Model.

Java Authentication and Authorization Service (JAAS)

JAAS supports the following authorization, authentication, and user-community (realm) features:

- Principals
- Subjects
- Login module authentication
- Roles
- Realms
- Policies
- Permissions

Authorization of a Client

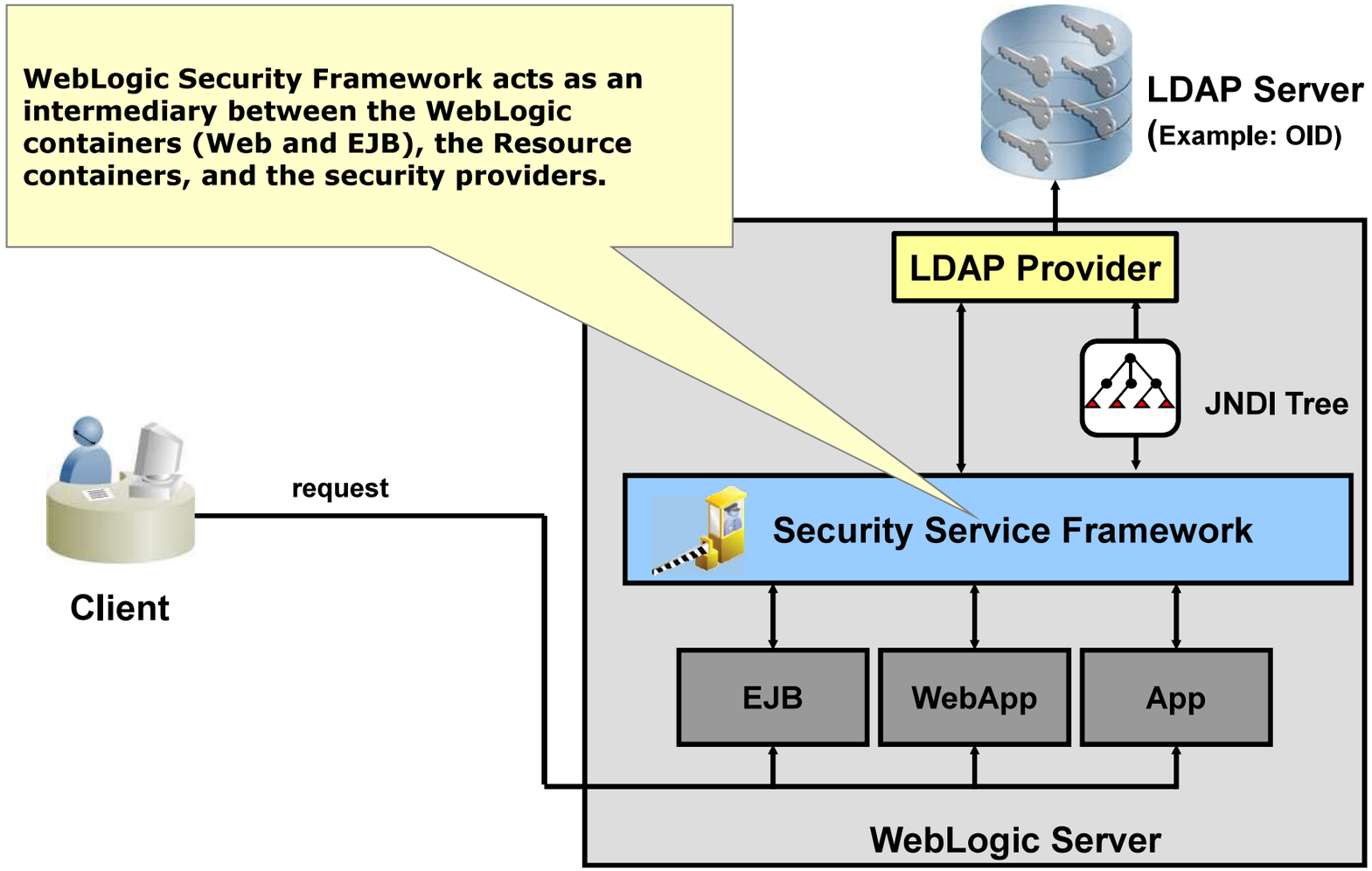
- The authorization is specified in the Java EE–specific deployment descriptors.
- Security roles:
  - Define the security view of the application to the deployer
  - Must be mapped to the appropriate security principals in the target environment
- Every client obtains a security principal.
- A client can invoke a URL or a method only if the client's role has the associated invocation rights.
- The container provider enforces the security policies and provides the tools for managing security.

Quiz

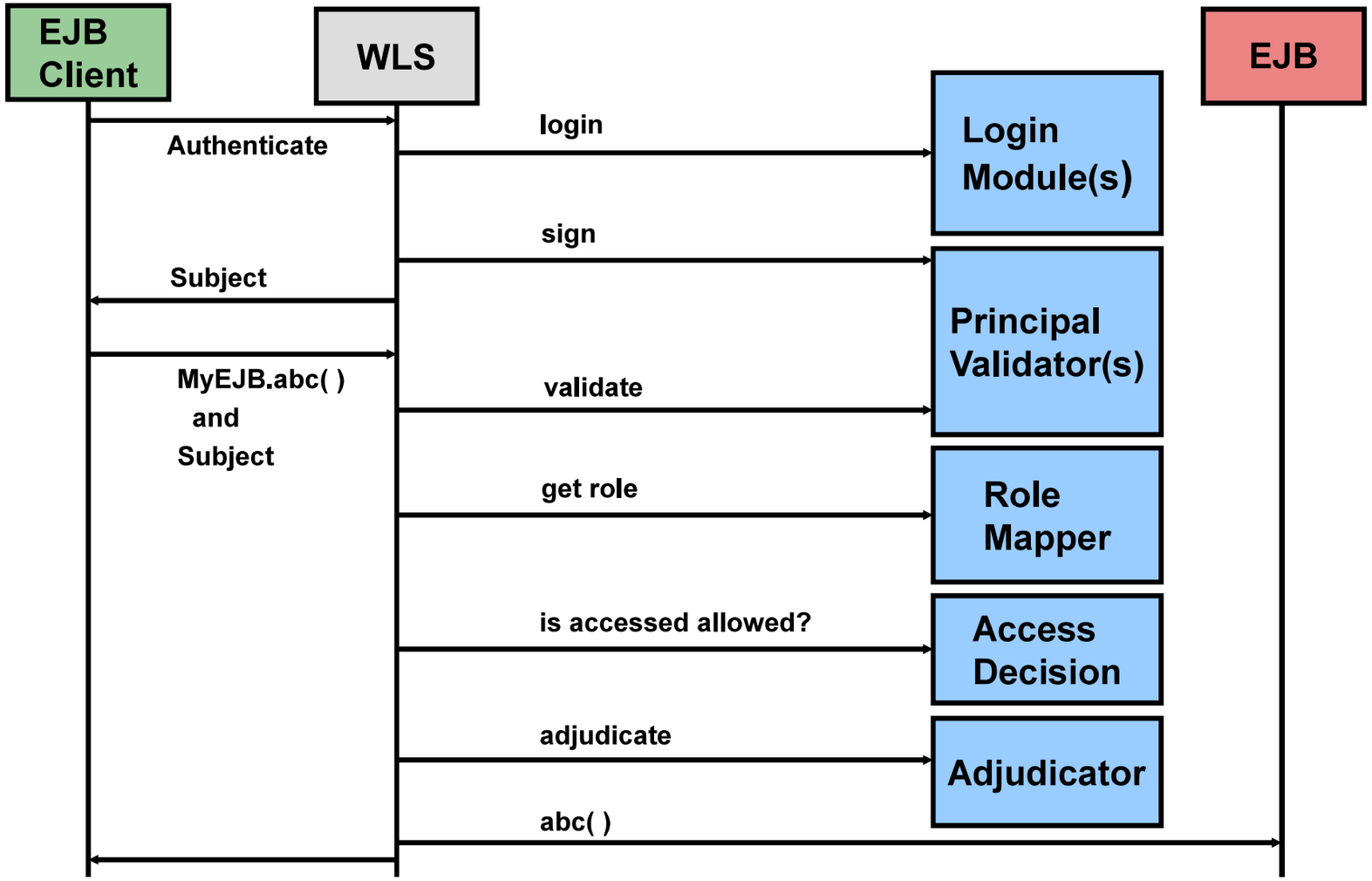
A principal is an identity assigned to a user or group as a result of authentication.

1. True
2. False

Security Process Architecture



Security Services



Security Realms

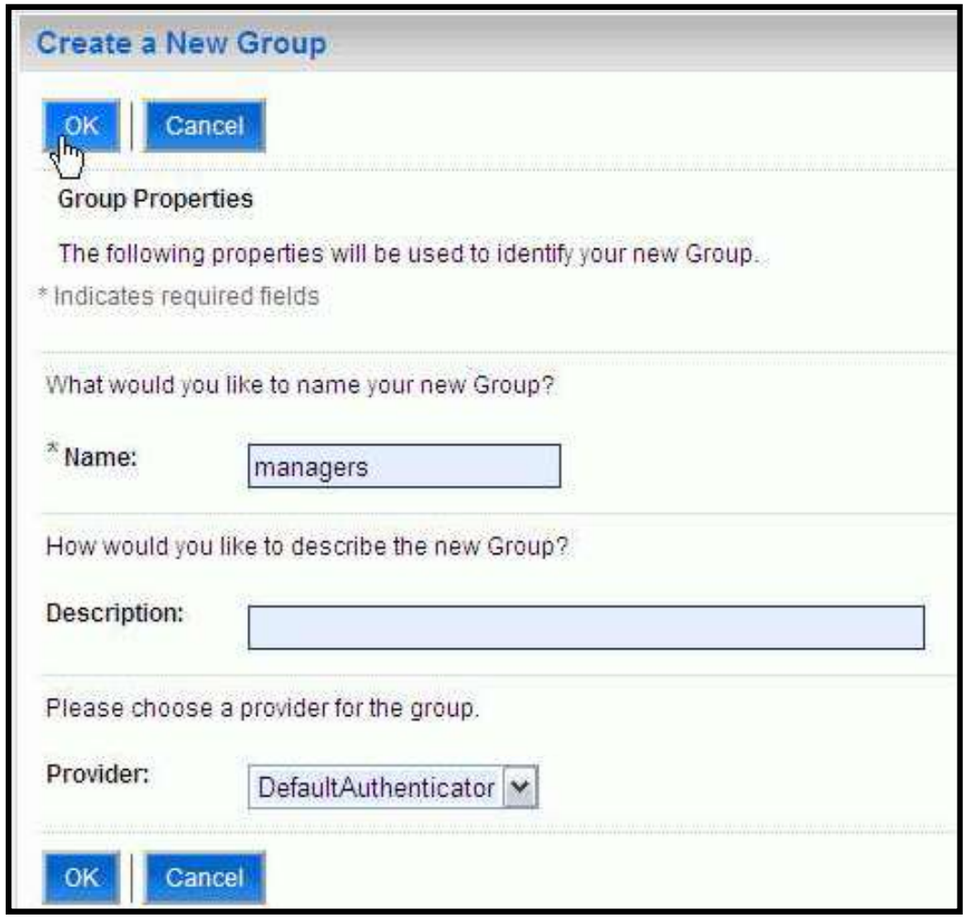
- A security realm is a collection of system resources and security service providers.
- Only one security realm can be active at a given time.
- A single security policy is used in any realm.
- Users must be recognized by an authentication provider of the security realm.
- Administration tasks include creating security realms.

Users and Groups

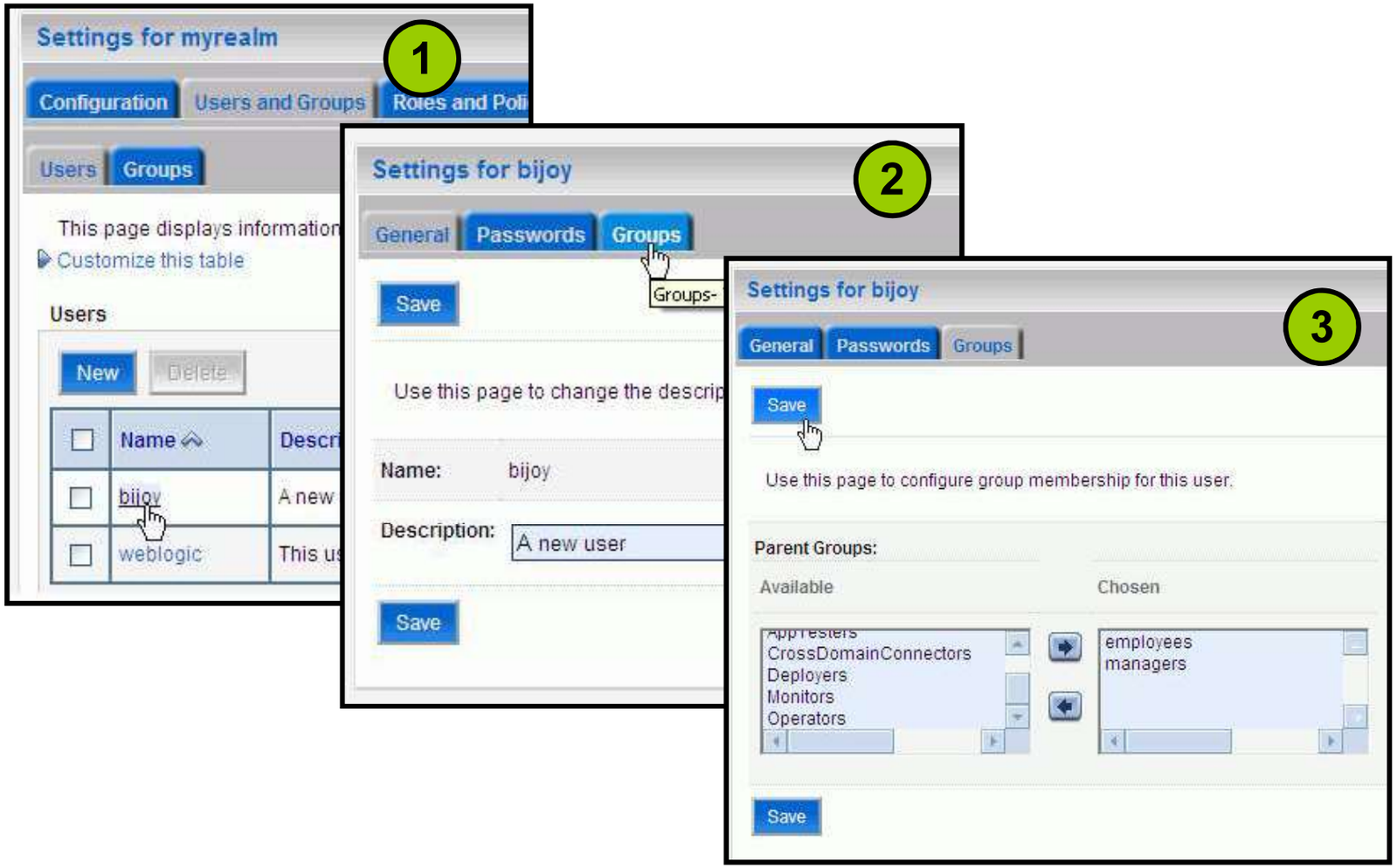
- Users are entities that use WebLogic Server such as:
  - Application end users
  - Client applications
  - Other WebLogic Servers
- Groups are:
  - Logical sets of users
  - More efficient for managing a large number of users

Configuring New Users in WebLogic Server

Configuring New Users in WebLogic Server



Adding Users to Groups



Logical Roles

- A role refers to a set of users who have the same permissions.
- A role differs from a group; a group has static membership; a role is conditional.
- A user and group can be granted multiple roles.
- There are two types of roles: global-scoped roles and resource-scoped roles.
- Roles defined in deployment descriptors can be inherited.
  - Occurs at deployment time
  - Can be disabled

Quiz

Security realm consists of a set of configured security providers, users, groups, security roles, and security policies.

1. True
2. False

Configuring Security

To protect a Web application with declarative security:

1. Determine Web Application resources that must be protected.
2. Define roles that should access the protected resources.
3. Map protected resources to roles that should access them.
4. Map roles to users/groups in the WLS security realm.
5. Set up an authentication mechanism.

Determining Protected Resources

- Web resources are defined based on URL patterns.
- URL patterns provide a flexible way to define a single resource or a group of resources.
- Examples of URL patterns:

URL Pattern	Role Name
`/*`	Some role name (such as director, manager, guest)
`/*.jsp`	...
`/context/*`	...

Defining the Logical Roles

The logical security roles defined in the Java EE deployment descriptors are:

- Specified in the `web.xml` file for Web applications, or in the `ejb-jar.xml` file for EJB components
- Defined in the `<security-role>` element for the application (One or more roles can be specified.)
- Authorized to invoke methods that are listed in the `<method-permissions>` element for the security role
- Scoped at the level of the application or all the enterprise beans in the `jar` file

Defining and Using Logical Roles

1. Define a logical role in the `<security-role>` element.
2. Use the role in the `<security-constraint>` element.

```
<security-role>
 <role-name>managers</role-name> <!--define-->
</security-role>
<security-constraint>
 <web-resource-collection>
  <web-resource-name>UpdEmployee</web-resource-name>
  <url-pattern>/UpdEmployees.jsp</url-pattern>
 </web-resource-collection>
 <auth-constraint>
  <role-name>managers</role-name> <!--apply-->
 </auth-constraint>
 <user-data-constraint>
  <transport-guarantee>NONE</transport-guarantee>
 </user-data-constraint>
</security-constraint>
```

Defining and Using Logical Roles in EJBs

1. Define a logical role in the `<security-role>` element.
2. Use the role in the `<method-permission>` element.

```
<assembly-descriptor>
 <security-role>
  <description>Manager</description>
  <role-name>managers</role-name>
 </security-role>
 <method-permission>
  <role-name>managers</role-name>
  <method>
   <ejb-name>HrApp</ejb-name>
   <method-name>incrementSalary</method-name>
   <method-params><method-param>int</method-param>
   <method-param>int</method-param></method-params>
  </method>
 </method-permission> ...
</assembly-descriptor>  ...
```

Mapping Logical Roles to Users and Groups

Mapping is done in the WebLogic-specific deployment descriptors.

- Mapping a logical role to a group or a specific user:

```
<?xml version = '1.0' encoding = 'windows-1252'?>
<weblogic-web-app ... >
    <security-role-assignment>
        <role-name>managers</role-name>
        <principal-name>joe</principal-name>
        <principal-name>bijoy</principal-name>
    </security-role-assignment>

    ...

</weblogic-web-app>
```

weblogic.xml

Setup Authentication

- Configure how a Web application determines users' security credentials:
  - `BASIC`: Uses the Web browser to display a dialog box
  - `FORM`: Uses a custom HTML form
  - `CLIENT-CERT`: Uses a client certificate to authenticate requests
- Configuring authentication:

```
<login-config>
   <auth-method>BASIC, FORM, or CLIENT-CERT</auth-
   method>
<form-login-config>
      <form-login-page>login.jsp</form-login-page>
      <form-error-page>badLogin.jsp</form-error-page>
   </form-login-config>
</login-config>
```

web.xml

Authentication Examples

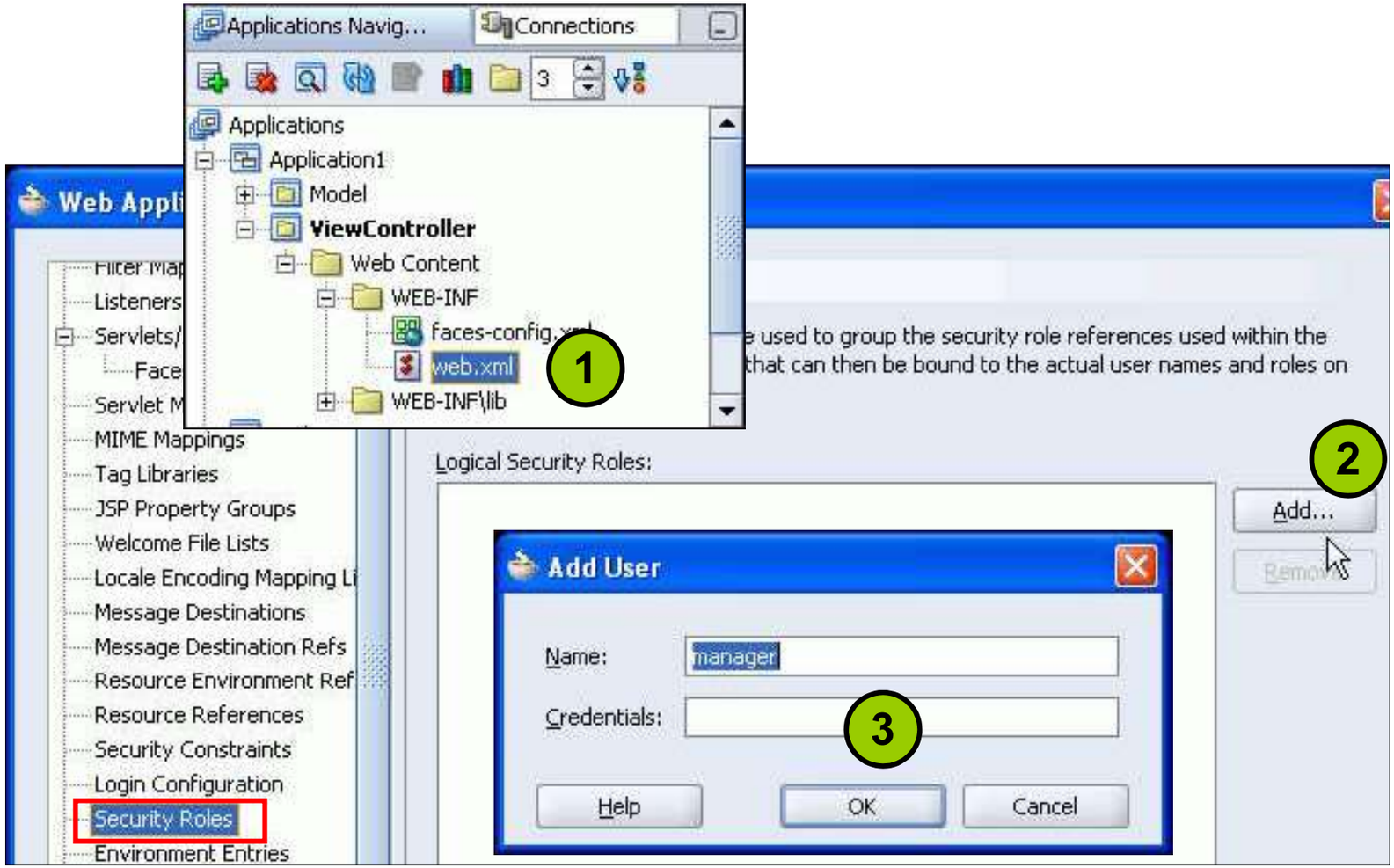
BASIC Authentication:

FORM-based Authentication:

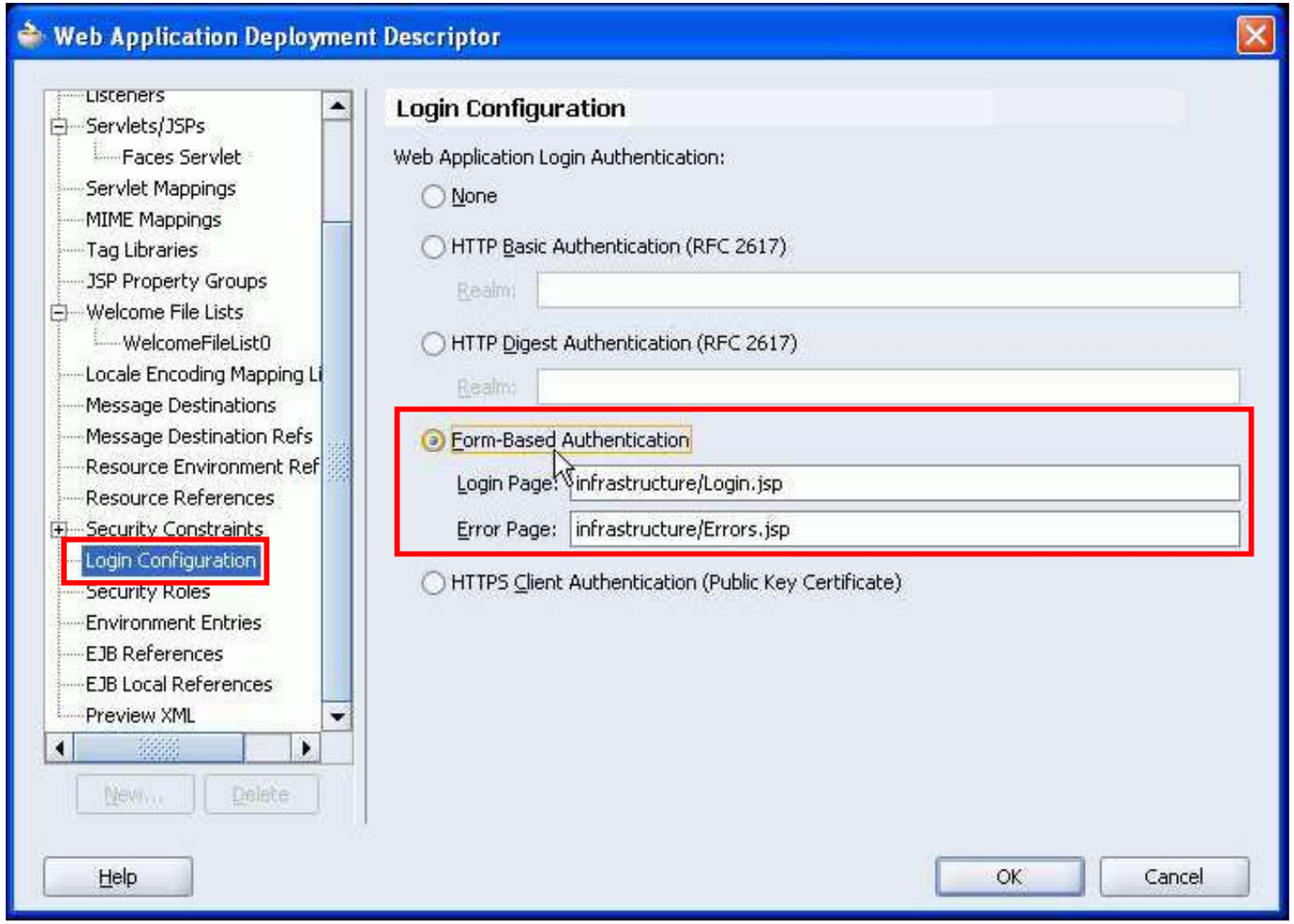
Setting Access Control with JDeveloper

- You can set security roles to access a Web application or an EJB by using JDeveloper deployment descriptor editors:
  - Use the Web Application Deployment Descriptor editor for Web applications.
  - Use the EJB Module Editor for EJBs.
- For Web applications, set the access permissions as constraints defining URL patterns for a Web resource.
- For EJBs, set the access permissions for either individual methods or all the methods of the bean.

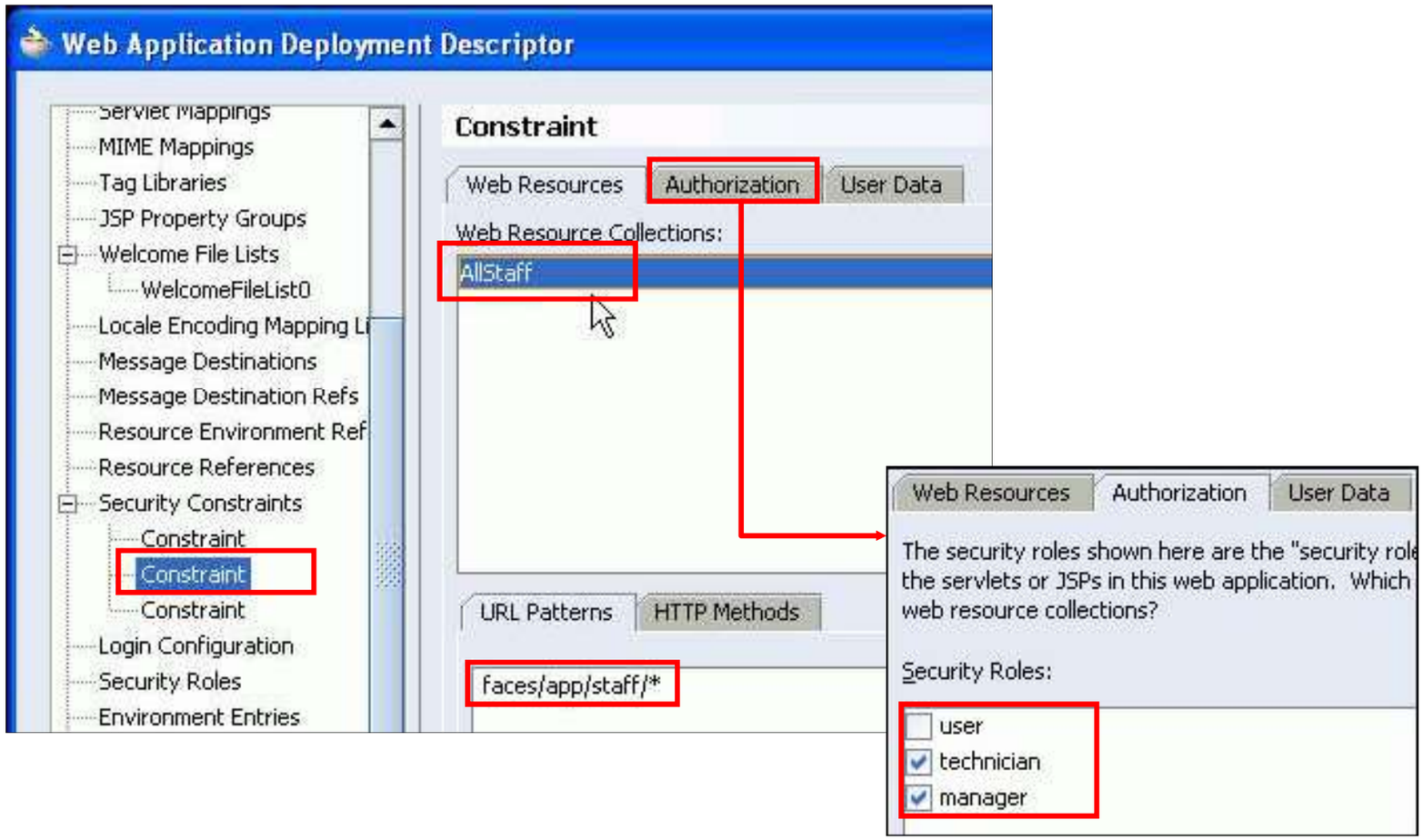
Creating Web Application Security Roles



Web Application Login Authentication



Web Application Authorization



EJB Security Roles

EJB security roles:

- Provide role-based access control of EJBs and their methods
- Are implemented through security annotations
  - `@DeclareRoles`
  - `@RolesAllowed`
  - `@PermitAll`
  - `@DenyAll`
  - `@RunAs`

Security Annotation: Example

```
@Stateless
@RolesAllowed("admin")
public class AdminServiceBean implements AdminService {

 public void adminTask() {System.out.println("Admin");}

 @RolesAllowed("user")
 public void sharedTask() {
  System.out.println("Shared admin/user method called");
 }

 @PermitAll
 public void safeTask() {System.out.println("Safe");}

 @DenyAll
 public void badTask() {System.out.println("Error");}

 @EJB SudoBean bean;
 @RunAs("admin")
 public void privilegedTask() {bean.sudoTask();}
}
```

1 2 3 4 5

Summary

In this lesson, you should have learned how to:

- Describe the principles behind Java EE application security design
- Describe the Java Authentication and Authorization Service (JAAS)
- List the security attributes of the Java Naming and Directory Interface (JNDI) Context interface
- Implement JAAS security for Web applications
- Use security annotations to implement JAAS security for EJBs

Practice : Overview

This practice covers the following topics:

- Implementing login authentication
- Using JAAS to restrict access to JSF pages