J2EE Security and Enterprise Java Beans

Mrunal G. Dhond
Department of Computing and Information Sciences
Master of Science, Final Defense
February 26, 2003

Guidance by Dr. Daniel Andresen
Agenda

• J2EE and Enterprise Java Beans briefing
• Project description and goals
• EJB application security
• Provider types
• Declarative security
• Programmatic security
• Conclusion
Java™ 2 Platform, Enterprise Edition (J2EE™)

- Multitiered distributed application architecture specification developed by Sun Microsystems
- Component-based approach
- Scalability
- Unified security and flexible transaction control
- Platform independence
Enterprise Java Beans™ (EJB™)

- Implement business logic
- Provide an object oriented view of the database schema
Project Description

• Implemented university records system

• 9 bean classes. Each bean class implements interface javax.ejb.EntityBean

• EJB clients connected via
  ➢ Home interface (extends javax.ejb.EJBHome) and Remote interface (extends javax.ejb.EJBObjec)

• Bean managed persistence

• JDBC
Project Tools

- JDeveloper 9.0.3 – Integrated Development Environment
- Oracle 9i Application Server
- Oracle 9i Database
- OC4J – Oracle Containers for Java
- Deployment descriptors – XML
  - J2EE Deployment descriptor – <ejb-jar.xml>
  - Oracle 9iAS deployment descriptor – <orion-ejb-jar.xml>
EJB Structure in J2EE

- **EJB Client**
- **Home Interface (EJBHome)**
- **Remote Interface (EJBOBJECT)**
- **Enterprise JavaBean class (EntityBean)**
- **Persistent Store (Oracle9i)**
- **Container provided services – Java Security (JAAS) / RMI / JNDI**

**J2EE Container for Enterprise JavaBeans**
Project goals

- Examine security issues in an EJB application
- Implement user authentication and authorization
## EJB Application Security

<table>
<thead>
<tr>
<th>Roles</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Developer or Bean Provider</td>
<td>• Writes business logic</td>
</tr>
<tr>
<td></td>
<td>• Defines logical security roles</td>
</tr>
<tr>
<td>Application Assembler or Deployer</td>
<td>• Manages target operational environment</td>
</tr>
<tr>
<td></td>
<td>• Defines ‘user’ and ‘role’ information</td>
</tr>
</tbody>
</table>
EJB Application Security

- **Mapping** - Logical security roles are mapped on to the users/roles in the operational domain

(J2EE Deployment descriptor)  (Oracle 9iAS Deployment descriptor)  (XML-based provider type)

```xml
<security-role>
  STUDENT
</security-role>
```

```xml
<security-role-mapping>
  STUDENT
  <user name="joe"/>
</security-role-mapping>
```

Logical security role ‘STUDENT’  Logical role ‘STUDENT’ mapped to user ‘joe’ defined in the operational environment.
Provider Types

• Java Authentication & Authorization Service

• Oracle 9iAS (OC4J) supports JAAS by using repositories a.k.a provider types
  ➢ Secure, centralized storage & retrieval of admin. data
  ➢ LDAP-based: Oracle Internet Directory
  ➢ XML based: jazn-data.xml
    - DTD is provided in *Oracle9iAS for J2EE Services Guide Release 2*
XML-based provider type: jazn-data.xml

- Stores information required to grant access (authenticate) EJB clients

```xml
<jazn-data>
  <jaznrealm>
    <realm>
      <name>jazn.com</name>
      <users>
        <user>
          <name>joe</name>
          <description>student</description>
          <credentials>Welcome</credentials>
        </user>
      </users>
    </realm>
  </jaznrealm>
</jazn-data>
```

- `<credentials>` element denotes the password ‘welcome’ for user ‘joe’
Scenario – Undefined user accessing application

- The following is not defined in jazn-data.xml

```xml
<User>
  <name>dan</name>
  <description>faculty</description>
  <credentials>{903}6NVtuNP8PWcBOY7I/CzxmTcpAk5PkJrV</credentials>
</User>
```
Declarative Security

• J2EE deployment descriptor *ejb-jar.xml*
  - contains structural and referential information of the bean classes
  - Defines a set of ‘logical security’ roles.
  - DTD is provided at [http://java.sun.com/dtd/ejb-jar_2_0.dtd](http://java.sun.com/dtd/ejb-jar_2_0.dtd)

```xml
<assembly-descriptor>
  <security-role>
    <role-name>FACULTY</role-name>
  </security-role>
  <security-role>
    <role-name>STUDENT</role-name>
  </security-role>
</assembly-descriptor>

• Method permissions can be defined by specifying the methods of the enterprise bean that each security role is allowed to invoke

```xml
<method-permission>
  <role-name>FACULTY</role-name>
  <method>
    <ejb-name>Student</ejb-name>
    <method-name>findByPrimaryKey</method-name>
  </method>
</method-permission>
```
Declarative Security

- ejb-jar.xml

```xml
<assembly-descriptor>
  <security-role>
    <role-name>FACULTY</role-name>
  </security-role>
  <security-role>
    <role-name>STUDENT</role-name>
  </security-role>
  <method-permission>
    <description>Only FACULTY is allowed to create and remove a student bean instance</description>
    <role-name>FACULTY</role-name>
    <method>
      <ejb-name>Student</ejb-name>
      <method-name>create</method-name>
    </method>
    <method>
      <ejb-name>Student</ejb-name>
      <method-name>remove</method-name>
    </method>
  </method-permission>
  <method-permission>
    <description>FACULTY and STUDENT roles are allowed to find Student Bean object(s) by using findByPrimaryKey and findByCourse methods</description>
    <role-name>FACULTY</role-name>
    <role-name>STUDENT</role-name>
    <method>
      <ejb-name>Student</ejb-name>
      <method-name>findByPrimaryKey</method-name>
    </method>
    <method>
      <ejb-name>Student</ejb-name>
      <method-name>findByCourse</method-name>
    </method>
  </method-permission>
</assembly-descriptor>
```
Declarative Security

- ‘logical security’ roles defined in *ejb-jar.xml* have to be mapped on to the actual users and roles defined in *jazn-data.xml*
- mapping is done in *orion-ejb-jar.xml*
  - DTD is provided at *http://xmlns.oracle.com/ias/dtlds/orion-ejb-jar.dtd*

```xml
<assembly-descriptor>
  <security-role-mapping name="STUDENT">
    <user name="joe"/>
  </security-role-mapping>
  <security-role-mapping name="FACULTY">
    <user name="dan"/>
  </security-role-mapping>
</assembly-descriptor>
```
Scenario - Declarative security example

- EJB client 'dan' attempts to create a Student bean instance.
- Student table consists of only 1 record before the bean instance is created.
- EJB client 'dan' is able to access EJB methods 'create' and 'findByPrimaryKey' successfully.
Scenario - Declarative security example

- Runtime output which shows that the EJB Student bean identifies the ejb client ‘dan’ as being associated with the ‘FACULTY’ role.

- The Student bean instance is created and persistence occurs.
Scenario - Declarative security example

- EJB client ‘joe’ can access the student bean method ‘findByPrimaryKey’
- EJB client ‘joe’ cannot access method ‘remove’ as only ‘FACULTY’ role has permission to access this method and ‘joe’ is mapped on to the ‘STUDENT’ role.
Programmatic security

- EJB architecture provides programmatic access to EJB client’s security context.

- isCallerinRole(String roleName) and getCallerPrincipal() methods of javax.ejb.EntityContext

- to be used within bean methods to impose role base restrictions before data access

- if ejb client is not in desired security role then jdbc connection is closed and bean life cycle is terminated
Scenario – Programmatic security example

- EJB client is in ‘STUDENT’ security role.
- isCallerInRole (‘FACULTY’) tests whether EJB client is in ‘FACULTY’ role.
- Only ‘FACULTY’ role is allowed to access Student bean ‘create’ method.
Scenario – Programmatic security example

- EJB client is in ‘STUDENT’ security role.
Scenario – Programmatic security example

- EJB client is denied access to the EJB ‘create’ method and the jdbc connection is closed.
Conclusion

- EJB Application security can be implemented using a combination of declarative and programmatic security

- Knowledge base about deployment descriptors and their role

- Further work in managing OC4J security using JAAS API from Sun Microsystems

- Integrate web tier security with EJB application security management