Procesos de la Ingeniería Software
Tema 4

Soporte Java para construcción de aplicaciones empresariales

5. Seguridad en aplicaciones Java EE

Bibliografía

- Lectura obligada
  - Erik Jendrock et al. (2014): The Java EE 7 Tutorial
    - Capítulos 47 - 50

- Lectura complementaria
  - Antonio Goncalvez (2013): Beginning Java EE 7, Apress
    - Capítulo 8 (Sección “Authorization”)

Aspectos básicos de seguridad en Java EE

- La seguridad de una aplicación Java EE es gestionada por los contenedores (Web y EJB)

- Gestión de:
  - **Conexiones seguras**
    - Uso de HTTPS
  - **Autenticación**
    - Con usuario/contraseña y/o certificados digitales
    - Generalmente responsabilidad de la capa web o del cliente
  - **Autorización**
    - Permisos de acceso en base a roles
    - Generalmente responsabilidad de la capa de negocio

- Dos modos de gestión:
  - **Declarativa**
    - Se describen los requisitos de seguridad a través de anotaciones y/o descriptores de despliegue
  - **Programática**
    - Se codifica directamente en la aplicación
    - Utilizada cuando la seguridad declarativa no es suficiente

Terminología y conceptos básicos

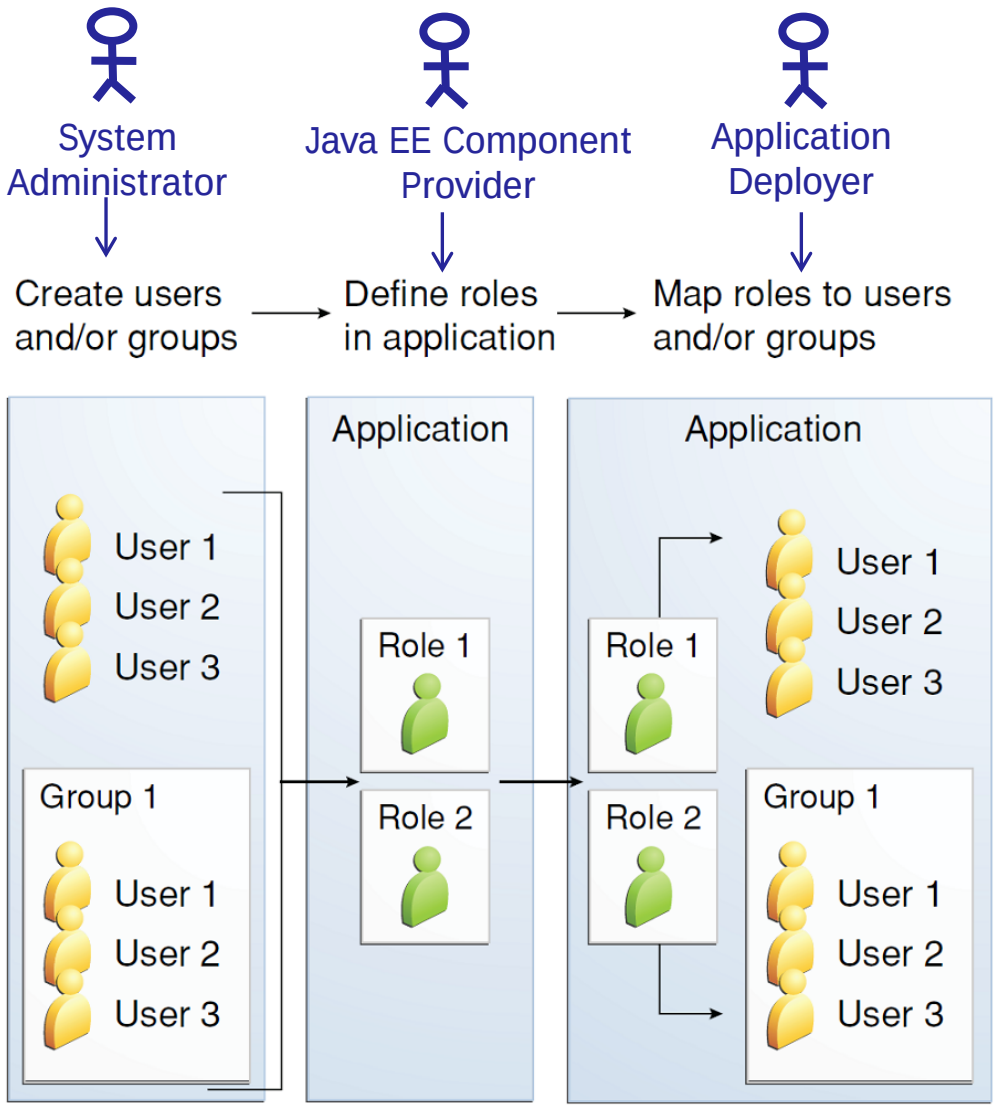
- **realm**: Dominio de seguridad dentro de un servidor de aplicación
  - Se pueden definir diferentes dominios, cada uno con sus propias políticas de seguridad y su conjunto de usuarios y grupos
- **user**: Identidad definida en el servidor de aplicación
  - Lleva asociada unas credenciales: usuario/contraseña o certificado digital
  - Los usuarios se definen dentro de los dominios de seguridad (realms)
  - Se pueden asociar en grupos
- **role**: Abstracción utilizada para distinguir permisos de acceso de los distintos usuarios a los recursos de una aplicación
- **principal**: Usuario autenticado por el servidor de aplicación
- **credential**: Objeto que contiene los atributos de seguridad asociados al principal, es decir, al usuario autenticado

Responsabilidad de la configuración

1. El administrador de sistemas es el encargado de configurar el servidor de aplicación con la base de datos de realms/usuarios/grupos

2. El desarrollador de componentes Java EE configura el control de acceso a los componentes y las necesidades de autenticación
   - Define roles y permisos para cada rol

3. El encargado del despliegue de la aplicación se encarga de mapear usuarios/grupos a roles



Seguridad en la capa web de Java EE

- Los componentes de la capa web pueden encargarse de aplicar autenticación y autorización

- La configuración de seguridad se realiza de forma declarativa
  - En base a la definición de restricciones de seguridad (`security-constraints`)
    - En Servlets se definen con anotaciones
    - En JSF sólo a través del descriptor de despliegue (web.xml + glassfish-web.xml)

- Un **`security-constraint`** consta de los siguientes elementos:
  - **`web-resource-collection`**
    - Identifica los recursos web (URL + método HTTP) a los que se les aplica la restricción de seguridad
    - No tienen porqué estar restringidos todos los URLs de una aplicación
  - **`auth-constraint`**
    - Indica qué roles tienen acceso a la colección de recursos web restringidos
  - **`user-data-constraint`**
    - Indica cómo se van a proteger los datos intercambiados entre cliente y servidor (a nivel de transporte)
      - Uso de HTTPS

Configuración de seguridad en web-xml

```
<web-app ....>

<security-constraint>
     <web-resource-collection>
             <web-resource-name>Calculadora</web-resource-name>
             <url-pattern>/calculator.jsf</url-pattern>
     </web-resource-collection>
     <auth-constraint>
             <role-name>ALUMNOS</role-name>
             <role-name>PROFESOR</role-name>
     </auth-constraint>
     <user-data-constraint>
             <transport-guarantee>CONFIDENTIAL</transport-guarantee>
     </user-data-constraint>
  </security-constraint>

  <security-role>
     <role-name>PROFESOR</role-name>
  </security-role>
  <security-role>
      <role-name>ALUMNOS</role-name>
   </security-role>

   <login-config>
      <auth-method>BASIC</auth-method>
      <realm-name>file</realm-name>
    </login-config>

</web-app>
```

El acceso a esta página es restringido

Sólo estos roles pueden acceder a la página indicada

El acceso debe ser a través de SSL

Conjunto de roles definidos en la aplicación

Configuración del modo de autenticación

Importante: Las credenciales que se proporcionan en la capa web, se propagan automáticamente a la capa EJB (pero sólo a través de comunicación SSL)

Tipos de autenticación y transporte

- Tipos de autenticación:
  - BASIC
  - FORM: Cómo BASIC pero permite customizar el aspecto de las pantallas de login y de error

```
<login-config>
    <auth-method>FORM</auth-method>
    <realm-name>file</realm-name>
    <form-login-config>
        <form-login-page>/login.xhtml</form-login-page>
        <form-error-page>/error.xhtml</form-error-page>
    </form-login-config>
</login-config>
```

  - DIGEST
  - CLIENT: Basado en certificados digitales (siempre con SSL)
  - MUTUAL: Tanto el cliente como el servidor se autentican

- Seguridad a nivel de transporte:
  - Valores: CONFIDENTIAL (SSL), INTEGRAL (SSL), NONE
  - Con autenticación BASIC o FORM es importante usar CONFIDENTIAL o INTEGRAL

Ejemplo de página de login en autenticación FORM

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>Login Form</title>
    </h:head>
    <h:body>
        <h2>Hello, please log in:</h2>
        <form name="loginForm" action="j_security_check" method="post">
            <p><strong>Please type your user name: </strong>
                <input type="text" name="j_username" size="25"/></p>
            <p><strong>Please type your password: </strong>
                <input type="password" size="15" name="j_password"/></p>
            <p>
                <input type="submit" value="Submit"/>
                <input type="reset" value="Reset"/></p>
        </form>
    </h:body>
</html>
```

Seguridad en componentes EJB - Control de accesos

- Los componentes EJB tienen capacidad para aplicar control de accesos (autorización) y, opcionalmente, autenticación

- Gestión de la seguridad en un EJB usando modo declarativo:
  1. El desarrollador del EJB, a través de anotaciones, define:
     - El conjunto global de roles asociados a la aplicación o el componente
     - Los roles permitidos en el acceso al EJB (de manera completa o por método)
  2. El encargado del despliegue mapea los roles definidos a usuarios y grupos
  3. Cuando un EJB anotado es invocado, el servidor de aplicación aplica automáticamente:
     1. Autenticación (si no la ha realizado la capa de presentación o el cliente):
        1. Pide credenciales (usuario/password o certificado digital)
        2. Comprueba que son válidos (frente a los datos almacenados en el servidor)
     2. Autorización:
        1. Comprueba que el usuario autenticado tiene asignado alguno de los roles permitidos de acceso al método invocado

- Gestión de la seguridad usando modo programático:
  - Las comprobaciones de seguridad se codifican directamente en el código del EJB usando la API **SessionContext**

Seguridad en componentes EJB - Anotaciones

- **DeclareRoles** (`@DeclareRoles`)
  - Define posibles roles de la aplicación
  - Anotación a nivel de clase
  - El conjunto de roles completo lo forman los declarados aquí y en todas las anotaciones RolesAllowed
- **RolesAllowed** (`@RolesAllowed`)
  - Define los roles permitidos en el acceso a la clase/método
  - Anotación a nivel de:
    - Clase (se aplica a todos los métodos del EJB)
    - Método (se aplica sólo al método anotado y sobrescribe a la anotación de clase)
- **DenyAll** (`@DenyAll`)
  - Anotación a nivel de clase o de método
  - Ningún rol tiene acceso a la clase/método
- **PermitAll** (`@PermitAll`)
  - Anotación a nivel de clase o de método
  - Todos los roles tiene acceso a la clase/método (no se chequea la identidad del usuario)
  - Valor por defecto

```
import javax.ejb.*;
import javax.annotation.security.*;


@Stateless
@DeclareRoles("RESPONSABLE",
              "PROFESOR", "ALUMNO")
public class GestionAsignaturaBean{

  @RolesAllowed("RESPONSABLE")
  public calificaAlumno(Alumno a, int nota) {
      …
  }

  @RolesAllowed("PROFESOR",
                "RESPONSABLE")
  public Alumno consultaAlumno(String dni) {
     …
  }

  @PermitAll
  public Alumno consultaNota(String dniAlumno) {
     …
  }

}
```

Ejemplo de seguridad programática en EJB

- La gestión de la seguridad de forma programática se puede realizar a través de métodos proporcionados por el **SessionContext**:
  - `getCallerPrincipal()`: Retorna el "principal" que identifica el cliente
    - Se puede consultar desde él, por ejemplo, el nombre del usuario
  - `isCallerInRole(String role)`: Comprueba si el cliente que invoca (principal) pertenece al rol indicado

```
@Stateless
public class GestionCompras {

  @Resource
  private SessionContext ctx;

  @EJB
  private UsuariosDAORemote usuarios;

  public void anhadeCompra (double precio) {
     String userName = ctx.getCallerPrincipal().getName();
     Usuario u = usuarios.get(userName);
     double precioFinal = precio;
     if (ctx.isCallerInRole("EMPLEADO")) {
           precioFinal = precio – precio*DESCUENTO;
     }
     u.anhadeCompra(precioFinal);
}
```

Seguridad en Java EE – Mapeo roles/usuarios

- Mapeo de roles a usuarios/grupos:
  - Si los roles coinciden con los grupos no se hace nada
  - Si no coinciden, el mapeo se formula a través de los descriptores de despliegue específicos del servidor de aplicación (glassfish-ejb-jar.xml / glassfish-web.xml)
    - A través de elementos `<security-rol-mapping>`

- Ejemplo:
  - Suponer que añadimos al realm file de Glassfish dos usuarios (Patri y Alumnos), ambos al grupo PS1617
  - El descriptor de despliegue específico `glassfish-ejb-jar.xml` para el componente GestionAsignaturaBean (transparencia 23) sería

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE glassfish-ejb-jar PUBLIC "-//GlassFish.org//DTD
          GlassFish Application Server 3.1 EJB 3.1//EN"
          "http://glassfish.org/dtds/glassfish-ejb-jar_3_1-1.dtd">
<glassfish-ejb-jar>

   <security-role-mapping>
      <role-name>PROFESOR</role-name>
      <principal-name>Patri</principal-name>
   </security-role-mapping>

   <security-role-mapping>
      <role-name>ALUMNO</role-name>
      <group-name>PS1617</group-name>
   </security-role-mapping>

   <enterprise-beans/>
</glassfish-ejb-jar>
```

Seguridad Java EE – Definición de roles/usuarios/grupos

- La configuración de los realms/usuarios/grupos y el mapeo de roles a usuarios dependen del servidor de aplicaciones utilizado

- En Glassfish:
  - Configuración de realms/usuarios/grupos
    - Se realiza a través de la consola de gestión (localhost:4848)
    - Glassfish proporciona tres realms predefinidos:
      - `file:` Para definir usuarios con usuario/contraseña (realm por defecto)
        - Los usuarios se almacenan localmente (fichero keyfile)
      - `certificate:` Para definir usuarios con certificados digitales (X.509)
        - Los usuarios se almacenan en una base de datos certificada
        - Glassfish proporciona certificados autofirmados válidos para desarrollo (no para producción)
      - `admin-realm:` Solo para definir usuarios con permiso de administrador
        - Los usuarios se almacenan localmente (fichero admin-keyfile)
    - Se pueden definir además realms JDBC
      - Usuarios almacenados en una base de datos estándar

  - Mapeo de roles a usuarios:
    - En los descriptores glassfish-ejb-jar.xml o glassfish-web.xml (ver trasparencia anterior)