Java Authentication and Auterization Service (JAAS)

CSOF-5201

Agenda

- Introducción
- Seguridad JEE
  - Seguridad Declarativa
  - Seguridad Programada
- Java Authentication and Authorization Services (JAAS)
  - Seguridad – Nivel Web
  - Seguridad – Nivel Ejb

Introducción

- Se requieren sistemas libres de peligro y que generen confianza a sus usuarios
- Asegurar confiabilidad, integridad y disponibilidad
- La seguridad es requerida en aplicaciones con datos sensibles y diversidad de usuarios

Introducción

- Funciones de la seguridad
  - Previene acceso de usuarios no autorizados a funciones de una aplicación
  - Permite identificar los usuarios y sus tareas (No repudio)
  - Protege al sistema de interrupciones que perjudiquen la calidad del servicio
  - Es transparente para el usuario

Introducción

- Java provee diferentes mecanismos de seguridad:
  - *Java Generic Security Services (Java GSS-API):* API basado en tokens usado para intercambiar mensajes de manera segura entre aplicaciones de comunicaciones
  - *Java Cryptography Extension (JCE):* Provee un framework para encripción, generación y acuerdo de claves y algoritmos de autenticación de mensajes.
  - *Java Secure Sockets Extension (JSSE):* Provee una implementación en Java de los protocolos SSL y TLS. Permite encripción de datos, autenticación de cliente-servidor e integridad de mensajes

Introducción

- *Java Authentication and Authorization Service (JAAS)*
  - Conjunto de APIs que proveen servicios de autenticación y de control de acceso de usuarios
  - Provee un framework portable y extensible para establecer autenticación y autorización de usuarios de manera programática
  - Garantiza acceso a varios programas, basado en permisos y políticas de seguridad
  - API de Java SE → Tecnología base para los mecanismos de seguridad de JEE

Introducción

- Características de seguridad para JEE:
  - **Autenticación:** Proceso de verificar que el usuario es quien dice ser, se apoya en el concepto de identificación. Se realiza a través de las credenciales del usuario, p.e. firmas digitales, usuario y contraseña

  - **Autorización o Control de Acceso**: Proceso que determina si un usuario tiene acceso a una tarea o a un recurso particular, este proceso es seguido de la autenticación
  - **Identificación:** Proceso que establece si una entidad es reconocida por el sistema

Introducción

- **Confidencialidad:** Define que los datos solamente estén disponibles para los usuarios autorizados
- **No Repudio:** El usuario debe ser responsable de las acciones que realiza, y no puede negarlo
- **Calidad del servicio (QoS):** Los servicios de seguridad deben garantizar la transmisión de información de manera confiable

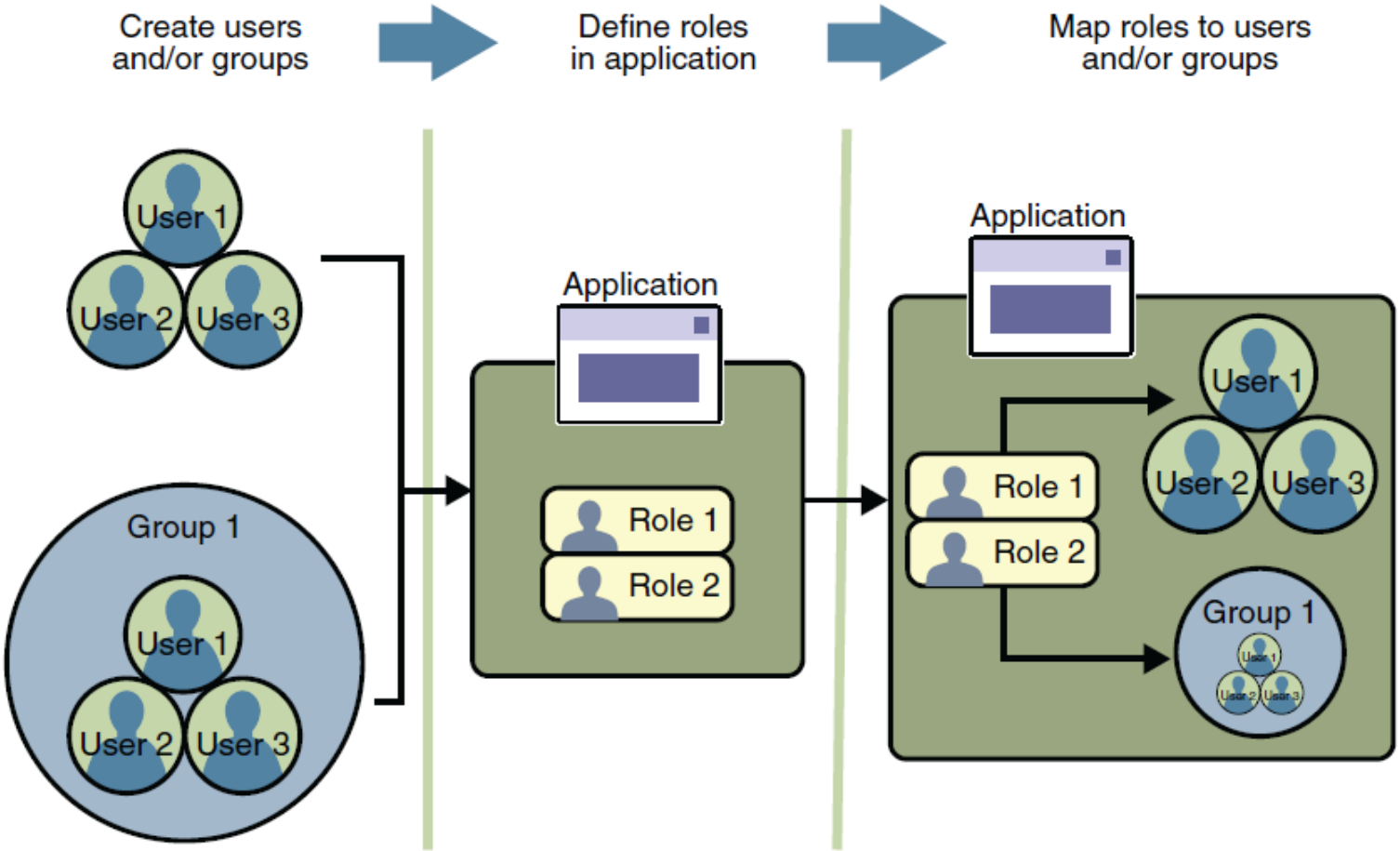
Introducción

- **Realm o Dominio de Seguridad:** Es un repositorio de *usuarios* y *grupos* para validar los usuarios de una aplicación Web que son controlados por la misma política de autenticación
  - El servicio de autenticación de JEE puede gobernar usuarios en multiples dominios de seguridad
  - En glassfish, vienen preconfigurados los realms *file*, *admin-realm* y *certificate*
  - En el realm file, el servidor almacena las credenciales de usuario locamlente en el archivo *keyfile*

Introducción

- **Usuario (Principal):** Es un individuo con una entidad registrado en el servidor de aplicaciones. Puede estar asociado a un grupo
  - Puede tener asociados un conjunto de *roles* a dicha identidad
- **Grupo:** Conjunto de usuarios autenticados con rasgos en común, p.e: grupo de vendedores
  - Puede tener asociados varios *roles* al igual que un usuario
  - Facilita el control de acceso → Gran cantidad de usuarios
  - Está relacionado al servidor de aplicaciones como tal
- **Rol**: Determina el permiso de acceso a un conjunto particular de recursos de una aplicación
  - Relacionado con una aplicación del servidor de aplicaciones p.e: administrador, empleado, estudiante
- **Credencial:** Contiene la información (atributos de seguridad) utilizada para autenticarse

Introducción



Tomado de [2]

Introducción

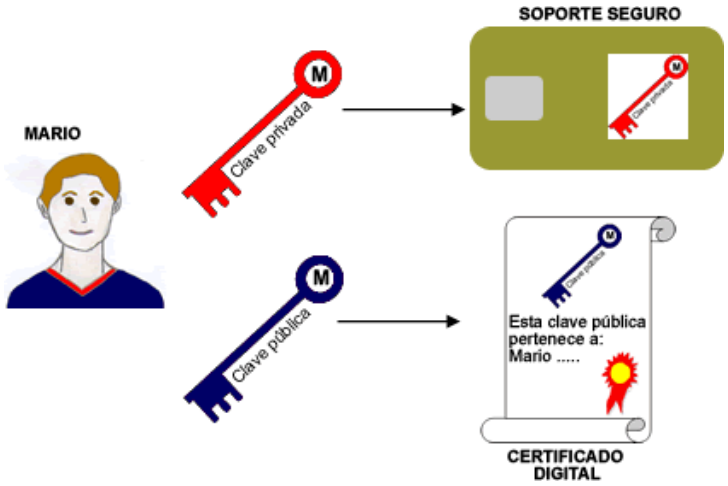
- Secure Socket Layer (SSL)
  - SSL es un protocolo para manejar la transmisión segura de mensajes en internet
  - SSL se ubica entre las capas de aplicación y transporte
  - Permite la comunicación entre navegadores Web y servidores Web sobre una conexión segura
  - En esta conexión, los datos son encriptados antes de ser enviados y desencriptados al momento de ser recibidos
  - Versión 3.0 desarrollada por netscape en 1996

Introducción

- Secure Socket Layer (SSL)
  - **Autenticación:** Se verifica la autenticidad del servidor por medio de un certificado digital. El servidor puede requerir un certificado del cliente → *Autenticación de cliente*
  - **Confidencialidad:** Los mensajes enviados entre cliente y servidor son encriptados → Un tercero no puede desencriptar los mensajes
  - **Integridad:** Se asegura que los datos no serán modificados durante el proceso de comunicación por un tercero

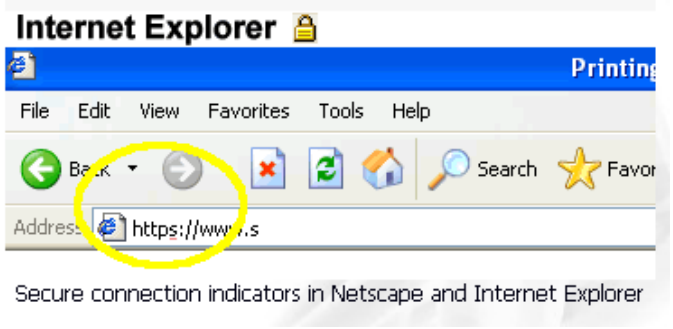
Introducción

- Certificado Digital
  - Forma digital de identificación
  - Emitido por entidades emisoras de certificados (CA)
  - Los certificados digitales permiten la criptografía mediante claves públicas
  - Cuando se emite un certificado digital, la entidad emisora firma el certificado con su propia clave privada



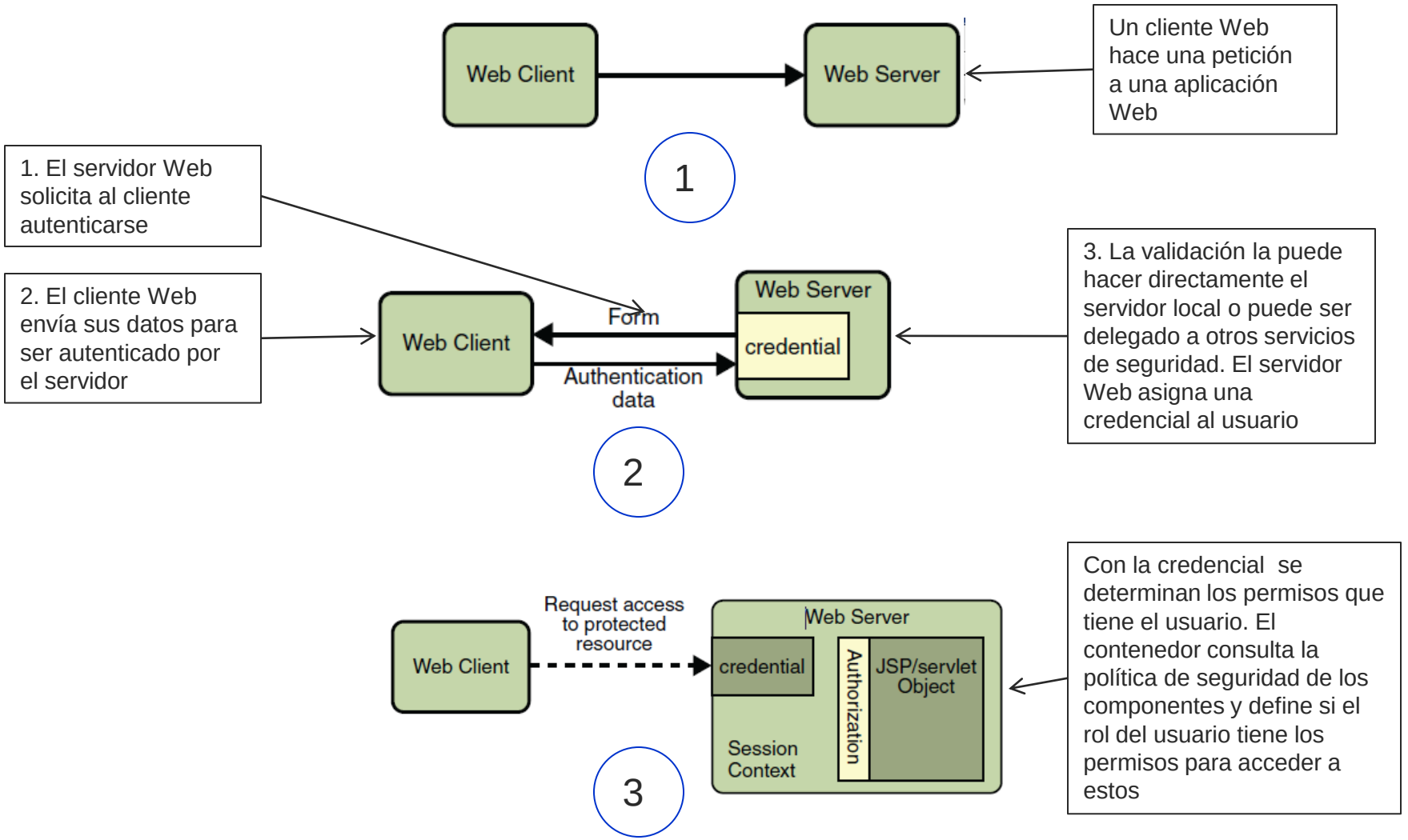
Introducción

- HTTPS
  - Hypertext Transfer Protocol over Secure Socket Layer
  - Versión segura del protocolo HTTP
  - Utilizado para manejar transacciones seguro sobre la Web utilizando certificados digitales
  - Las direcciones en el navegador comienzan con https://
  - El puerto estándar es 443
  - HTTPS sólo protege los datos en tránsito
  - La seguridad de los datos en el receptor y el emisor, dependen de la seguridad de computador de cada uno de ellos



Secure connection indicators in Netscape and Internet Explorer

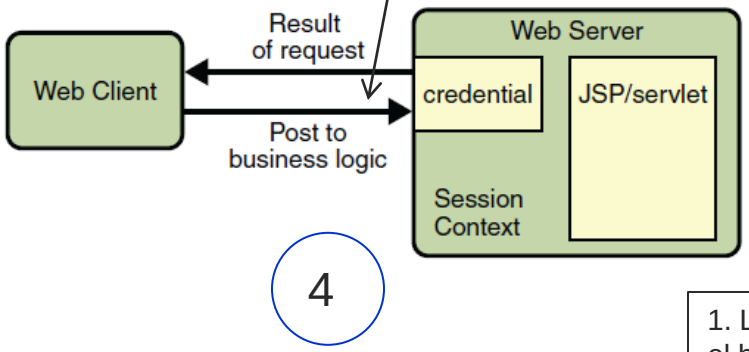
Introducción - Ejemplo



Tomado de [2]

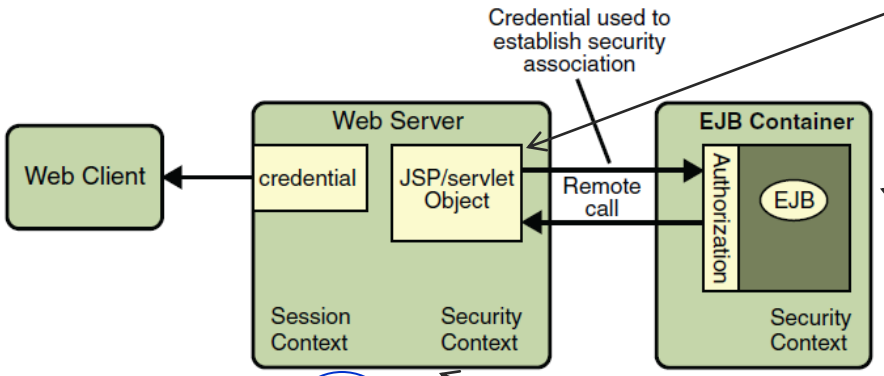
Introducción- Ejemplo

Si el usuario es autorizado, se acepta la solicitud



4

1. La página JSP localiza el bean de sesión y establece una asociación de seguridad a través de la credencial del usuario



2. El contenedor EJB se encarga de definir si un cliente tiene acceso a sus métodos. Esto lo realiza a través de las políticas de seguridad asociadas con él

5

3. Se tienen dos contextos de seguridad, el primero para el servidor Web y el segundo para el contenedor EJB

Agenda

- Introducción
- Seguridad JEE
  - Seguridad Declarativa
  - Seguridad Programada
- Java Authentication and Authorization Services (JAAS)
  - Seguridad – Nivel Web
  - Seguridad – Nivel Ejb

Seguridad JEE

- Las aplicaciones JEE son desplegadas en contenedores, los cuales ofrecen dos tipos de seguridad:
  - Declarativa *(declarative authorization)*
  - Programada (*programmatic authorization*)

Seguridad Declarativa

- Declarativa
  - La seguridad requerida por los componentes es definida en descriptores de despliegue
  - Los descriptores son externos a la aplicación
  - Se definen los roles y requerimientos de acceso
  - Se mapean los roles con el ambiente específico, usuarios y políticas de seguridad
  - Un archivo descriptor es un XML

Seguridad Declarativa

- Descriptores
  - Enterprise JavaBeans → Descriptor de despliegue EJB
    - *ejb-jar.xml* → Debe estar ubicado en META-INF/ y debe estar ubicado en un archivo JAR de EJB
  - Web Services → *jaxrpc-mapping-info* definido en JSR109. Provee un mapeo entre el WSDL y Java
  - Componentes Web → Descriptor de despliegue Web
    - *web.xml*

Seguridad Programada

- Programada:
  - Está embebida en el código fuente de la aplicación
  - Es útil cuando no es suficiente la seguridad declarativa para definir el modelo de seguridad de la aplicación
  - El API contiene los siguientes métodos:
  - Interface **`javax.ejb.EJBContext`**
    - **`java.security.Principal getCallerPrincipal();`**
    - **`boolean isCallerInRole(String roleName);`**
  - Interface **`javax.servlet.http.HttpServletRequest`**
    - **`boolean isUserInRole(String roleName);`**
    - **`java.security.Principal getUserPrincipal();`**

Anotaciones

- Las anotaciones permiten un estilo *declarativo* de programación
- Los desarrolladores pueden especificar información de seguridad dentro de una clase utilizando anotaciones
- Información adicional debe estar ubicada en archivos descriptores
- Cualquier información explícita especificada en un descriptor de despliegue ***sobrescribe*** cualquier valor especificado con anotaciones.

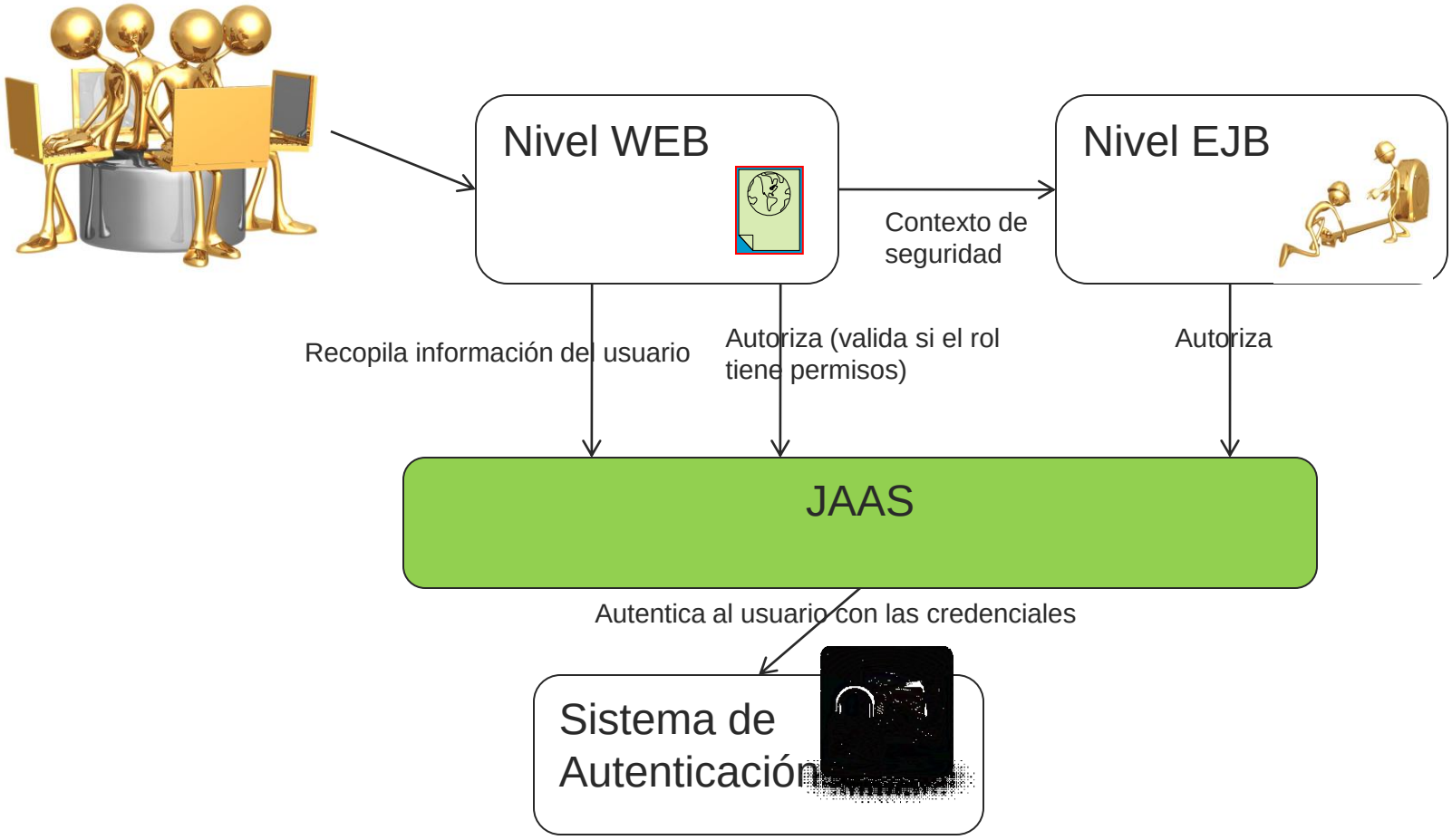
Agenda

- Introducción
- Seguridad JEE
  - Seguridad Declarativa
  - Seguridad Programada
- Java Authentication and Authorization Services (JAAS)
  - Seguridad – Nivel Web
  - Seguridad – Nivel Ejb

JAAS

- La seguridad de las aplicaciones JEE está basada sobre Java Authentication and Authorization Services (JAAS)
- API JAAS
  - Separa el sistema de autenticación de la aplicación JEE
  - La aplicación sólo debe conocer como comunicarse con el API de JAAS
  - JAAS conoce como comunicarse con sistemas de autenticación como:
    - Lightweight Directory Access Protocol (LDAP): Microsoft Active Directory or Oracle Internet Directory (OID)
  - Es diseñado para autenticación y autorización, incluyendo las capas web y EJB.
  - La autenticación se puede realizar a nivel web y se mantiene el contexto de seguridad en todos los niveles. (Contexto compartido)

JAAS



Escenario de Seguridad usando JAAS [1]

Agenda

- Introducción
- Seguridad JEE
  - Seguridad Declarativa
  - Seguridad Programada
- Java Authentication and Authorization Services (JAAS)
  - Seguridad – Nivel Web
  - Seguridad – Nivel Ejb

Seguridad - Nivel WEB

- El contenedor WEB se encarga de la seguridad. Es necesario definir:
  - Los componentes que se aseguran
  - La forma de asegurarlos
  - El manejo de credenciales de autenticación
  - Los roles que tienen acceso a cada componente
- La autenticación y la autorización a nivel Web se configura a través de:
  - **`web.xml`**
    - **`login-config`**
    - **`security-constraint`**
  - Es necesario definir los usuarios, dominios y grupos en el contenedor Web o un sistema de usuarios

Seguridad - Nivel WEB

- Ejemplo **`web.xml`**

```
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>ActionBazaarRealm</realm-name>
</login-config>

<security-role>
    <role-name>intranet</role-name>
</security-role>

<security-constraint>
    <web-resource-collection>
        <web-resource-name>ActionBazaar Administrative Component
        </web-resource-name>
        <url-pattern>/admin/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>CSR</role-name>
    </auth-constraint>
</security-constraint>
```

Autenticación: Mecanismo de autenticación

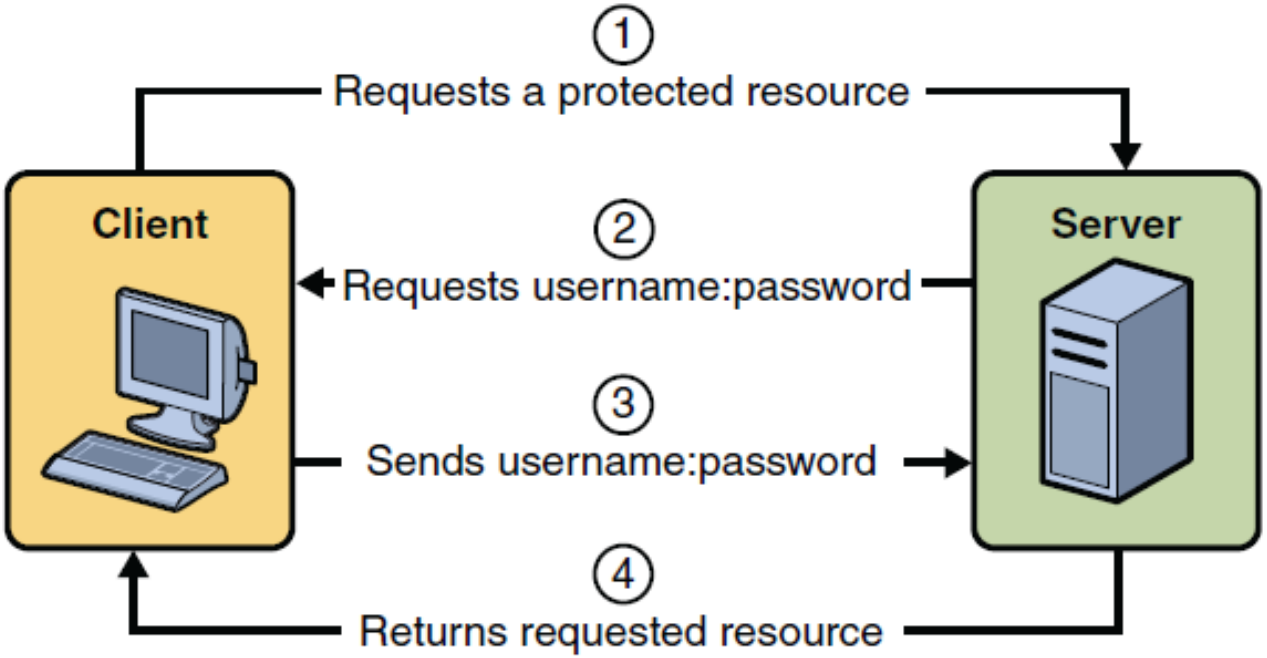
Define los roles de la aplicación en un descriptor de despliegue

Autorización: Define los roles que tienen acceso a la aplicación

Tomado de [1]

Descripción web.xml

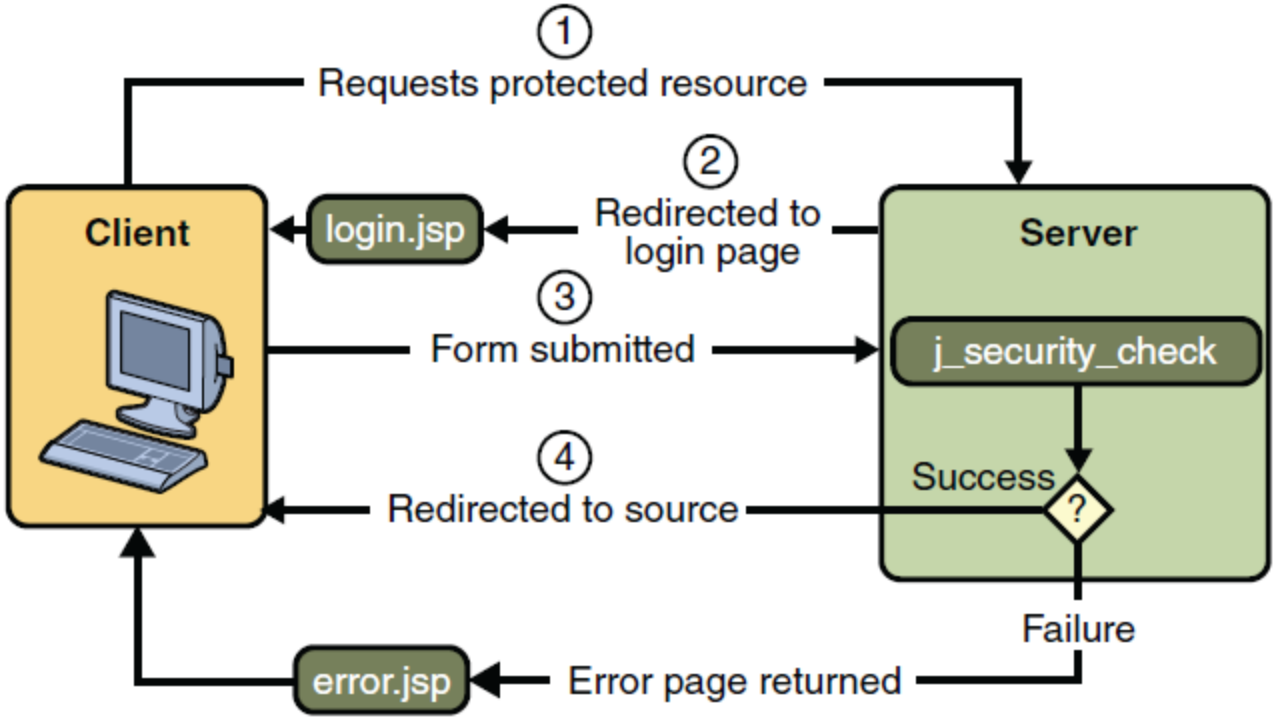
1. Mecanismo de Autenticación
   a. **BASIC:** Utiliza un esquema de autenticación en la cabecera HTTP. El browser toma la información de usuario/contraseña



Tomado de [2]

Descripción web.xml

1. Mecanismo de autenticación
   b. **FORM:** Tiene el mismo esquema de BASIC, con la diferencia de usar un formulario HTML personalizado



Tomado de [2]

Descripción web.xml

1. Mecanismo de autenticación
   b. **FORM:** Tiene el mismo esquema de BASIC, con la diferencia de usar un formulario HTML personalizado

Ejemplo:

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

Tomado de [2]

Descripción web.xml

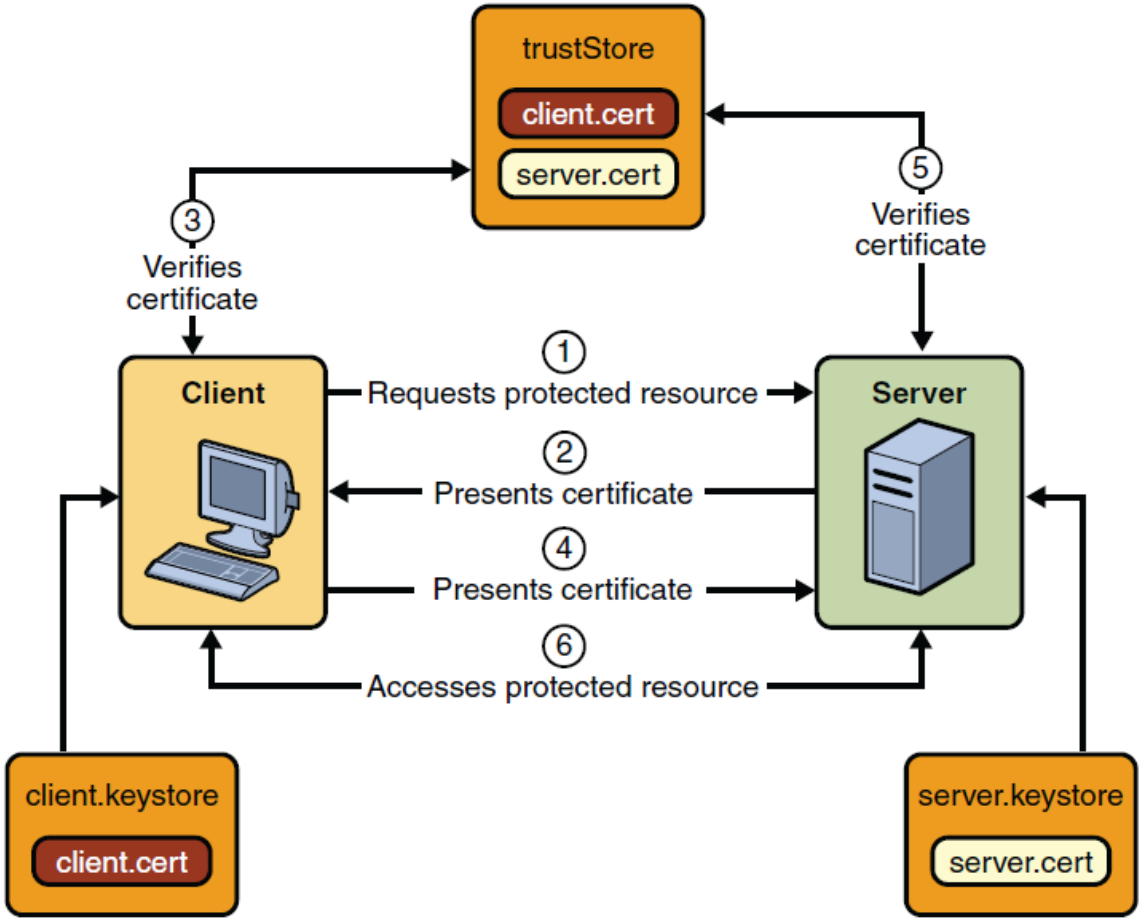
1. Mecanismo de autenticación

   c. **CLIENT-CERT:** Maneja certificados

   - El cliente envía un certificado de llave pública, almacenado en el navegador del cliente a través de Secure Socket Layer (SSL)
   - El servidor autentica el contenido del certificado
   - Estas credenciales son validadas por el proveedor de JAAS

Descripción web.xml

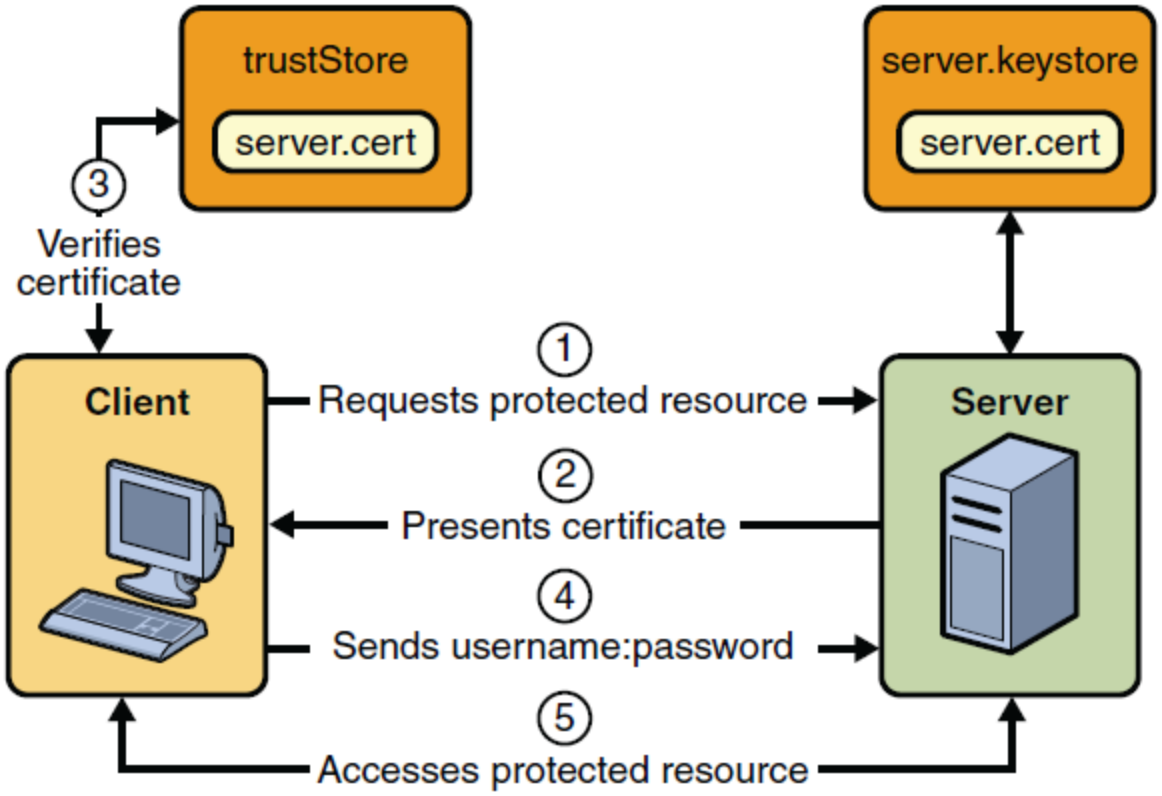
1. Mecanismo de autenticación
   c. **CLIENT-CERT:** Maneja certificados



Tomado de [2]

Descripción web.xml

1. Mecanismo de autenticación
   c. **CLIENT-CERT:** Maneja certificados



Tomado de [2]

Descripción Web.xml

2. **<realm-name>** → Especifica el nombre del dominio que se utiliza para la autenticar las credenciales del usuario.
   - Este dominio está sobre el sistema de autenticación de JAAS
3. **<security-constraint>** → Define los privilegios de acceso a una colección de recursos
   - **<web-resource-collection>** → Define los componentes de la aplicación Web que aplican para este *constraint*
     - **<web-resource-name>** → El nombre de la colección de recursos web
     - **<url-pattern>** → Describe un patrón para identificar los elementos que aplican al constraint de seguridad
   - **<auth-constraint>** → Define los grupos que tienen acceso a una colección de recursos Web, definidos en el *constraint*
     - **<rol-name>** → Debe corresponder con alguno de los roles definidos en el descriptor de ejecución para la aplicación

Seguridad - Nivel WEB

- En Glassfish para mapear un usuario con un grupo se utiliza el tag **<security-role-mapping>** en el archivo descriptor *sun-ejb-jar.xml*

```
<security-role-mapping>
     <role-name>employee</role-name>
     <group-name>bankemployee</group-name>
</security-role-mapping>
```

Tomado de [3]

Agenda

- Introducción
- Seguridad JEE
  - Seguridad Declarativa
  - Seguridad Programada
- Java Authentication and Authorization Services (JAAS)
  - Seguridad – Nivel Web
  - Seguridad – Nivel Ejb

Seguridad - Nivel EJB

- Nivel EJB → Autenticación
  - Aplica para los beans de sesión y los beans de mensaje
  - No aplica para los beans de entidad
  - Cuando se despliega una aplicación, el administrador del sistema local debe mapear cada rol a los grupos definidos

Seguridad - Nivel EJB

- La autenticación para EJB se puede realizar a través de un descriptor por ejemplo: *sun-ejb-jar.xml*

```
<sun-ejb-jar>
    <enterprise-beans>
        <ejb>
            <ejb-name>BankServiceBean</ejb-name>
            <ior-security-config>
                <as-context>
                    <auth-method>USERNAME_PASSWORD</auth-method>
                    <realm>default</realm>
                    <required>true</required>
                </as-context>
            </ior-security-config>
        </ejb>
    </enterprise-beans>
</sun-ejb-jar>
```

Un cliente EJB necesita usar autenticación IOR (Interoperable Object Reference)

Servicio de autenticación

Método de autenticación

Dominio de seguridad en el cual es autenticado el usuario

Define si es requerida la autenticación de los usuarios

Tomado de [3]

Seguridad - Nivel EJB

- Para la Autorización a nivel de EJB, se puede utilizar la seguridad declarativa
  - El contenedor valida los roles definidos
  - Especificación JSR-250
    - **javax.annotation.security.DeclareRoles**
    - **javax.annotation.security.RolesAllowed**
    - **javax.annotation.security.PermitAll**
    - **javax.annotation.security.DenyAll**

```
@DeclareRoles("BIDDER" "CSR", "ADMIN")
@Stateless
public class BidManagerBean implements BidManager {
        @RolesAllowed("CSR, ADMIN")
        public void cancelBid(Bid bid, Item item) {...}

        @PermitAll
        public List<Bid> getBids(Item item) {...}
}
```

Declara roles

Especifica roles que tienen acceso al método (Autorización)

Cualquier rol puede ejecutar el método

Tomado de [1]

Seguridad - Nivel EJB

- **`@DeclareRoles`**
  - Declara los roles de seguridad utilizados en la aplicación
  - Cuando no se definen, el sistema por defecto toma los roles de **`@RolesAllowed`** y define una lista de roles
- **`@RolesAllowed`**
  - A nivel de métodos y a nivel de clase
  - Define los roles que pueden utilizar un método o los métodos de una clase

Seguridad - Nivel EJB

- **`@PermitAll`**
  - El método o la clase pueden ser accedidos por cualquier rol
- **`@DenyAll`**
  - Sólo se implementa a nivel de método
  - Su funcionalidad es no accesible para cualquier rol
- NOTA: Las anotaciones **`@PermitAll`**, **`@DenyAll`** y **`@RoleAllowed`** no pueden aplicarse en la misma clase simultáneamente

Seguridad - Nivel EJB

- **@RunAs**
  - Se aplica a nivel de clase e indica el rol con el que se ejecuta el Bean
  - Se puede utilizar para agregar algún rol en particular durante la ejecución del bean o del método

```
@RunAS("ADMIN")
@RolesAllowed("CSR")
public void cancelBid(Bid bid, Item item) {...}
```

Tomado de [1]

Nivel EJB - Autenticación y Autorización

- Para la autorización a nivel EJB se puede utilizar la seguridad programada
  - Permite manejar características específicas para los usuarios
  - Se validan los roles a través del código
  - Se trabaja sobre el contexto de seguridad
  - Interfaz **`javax.ejb.EJBContext`**. Provee dos métodos:
    - **`public java.security.Principal getCallerPrincipal()`** Permite que los métodos del bean obtengan el nombre del *Principal*. Ej: El nombre del *Principal* puede ser utilizado para consultar una base de datos
    - **`public boolean isCallerInRole(String roleName)`** Valida si el rol tiene permiso

Seguridad - Nivel EJB

- Ejemplo autorización programada

```
@Stateless
public class BidManagerBean implements BidManager {
   @Resource SessionContext context;
    ...
    public void cancelBid(Bid bid, Item item) {
        if (!context.isCallerInRole("CSR")) {
            throw new SecurityException("No permissions +
               to cancel bid");
        }
        ...
     }
     ...
}
```

Tomado de [1]

Seguridad - Nivel EJB

```
@Stateless public class
EmployeeServiceBean implements EmployeeService{

    @Resource SessionContext ctx;
    @PersistenceContext EntityManager em;
    public void changePhoneNumber(...) { ...

        // Obtain the caller principal
        callerPrincipal = ctx.getCallerPrincipal();

        // Obtain the caller principal's
        name.callerKey = callerPrincipal.getName();

        // Use callerKey as primary key to find EmployeeRecord
        EmployeeRecord  myEmployeeRecord =
                em.findByPrimaryKey(EmployeeRecord.class, callerKey);

        // Update phone number
        myEmployeeRecord.setPhoneNumber(...); ...
    }
}
```

Se puede retornar el nombre del *Principal*

Busca el registro a partir del nombre del *Principal*

Tomado de [4]

Bibliografía

1. EJB 3 in action. Panda Debu, Rahman Reza, Lane Derek. Manning. 2007.

2. The Java™ EE 5 Tutorial Third Edition. For Sun Java System Application Server Platform Edition 9. Eric Jendrock. 2006.

3. EJB 3 Developer Guide. Michael Sikora. Packt Publishing. BIRMINGHAM – MUMBAI. 2008

4. http://java.sun.com/javaee/5/docs/tutorial/doc/bnbyl.html#bnbyn