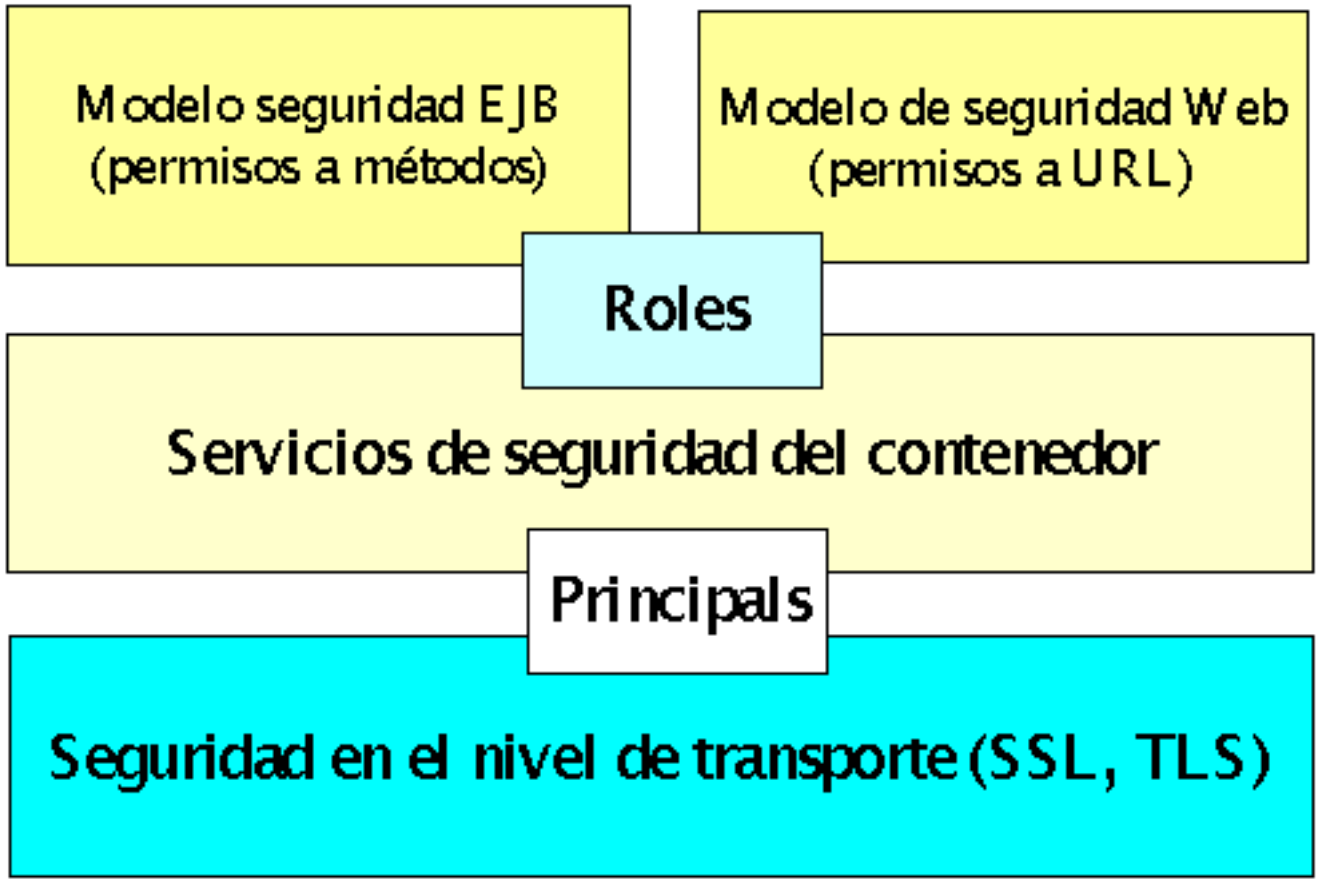
Especialista en Aplicaciones y Servicios Web con Java Enterprise

Enterprise JavaBeans

Sesión 6:

Seguridad

Arquitectura de seguridad en J2EE



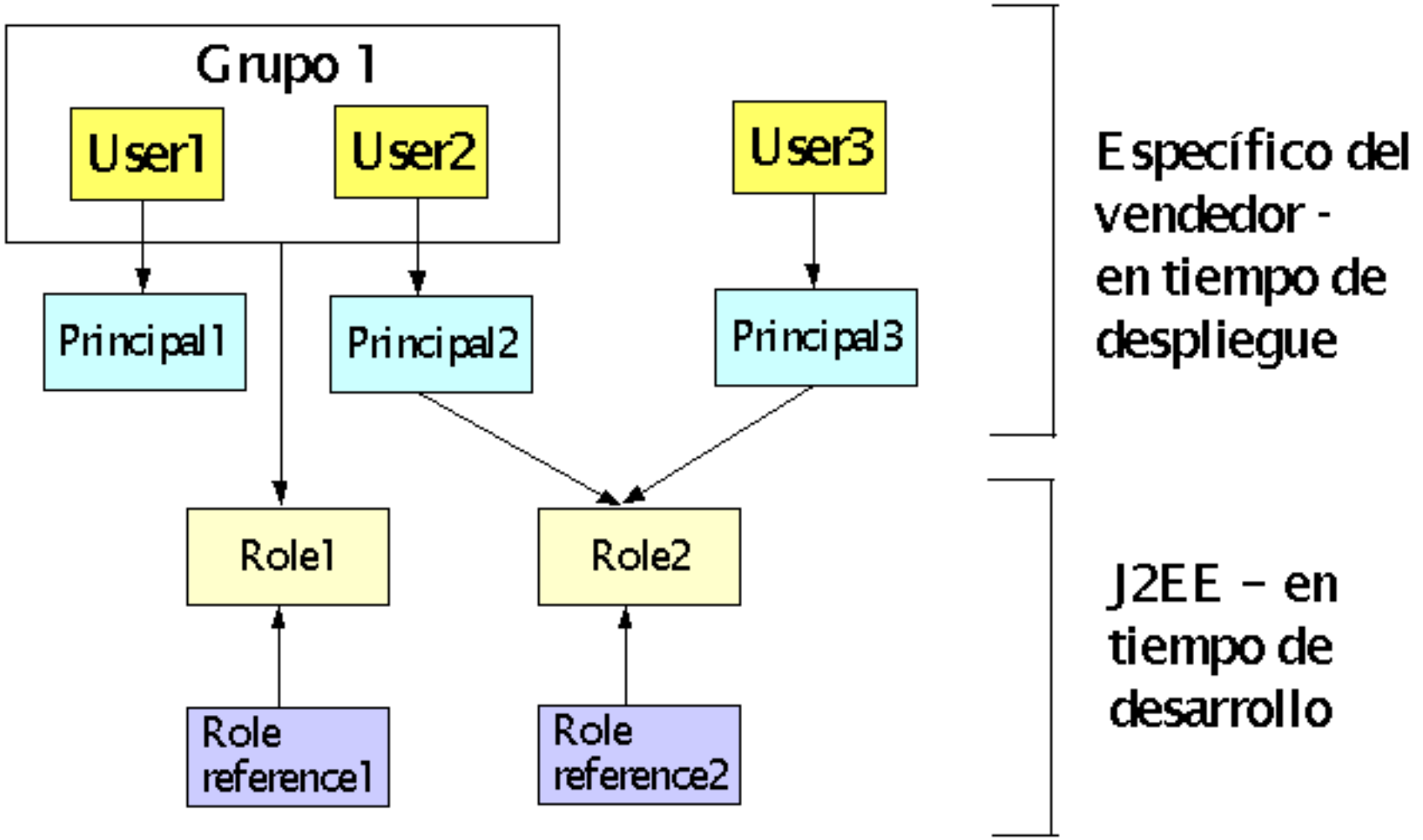
La seguridad en J2EE como abstracción

- Existen diferencias entre los distintos entornos operativos en los que va a desplegarse una aplicación de J2EE
- Distintas representaciones de las credenciales de los usuarios:
  - usuario/contraseña, certificados, kerberos
  - Servidor LDAP, base de datos, Sistema Operativo
- J2EE debe abstraer todas estas representaciones en un único marco común

Conceptos fundamentales J2EE

- La seguridad en J2EE se representa mediante:
  - Principals
  - Roles
  - Mapeado de Roles a Principals

Mapeado de roles



Tipos de seguridad en EJB (1)

- 1. Autentificación (¿cómo adquiere un Principal un cliente de un bean?)
  - No se dispone de ningún servicio de autentificación en EJB
  - La autentificación debe ya estar realizada y el cliente que la petición debe tener ya un Principal asociado
- 2. Comunicación (¿son seguras las comunicaciones entre los clientes y los beans?)
  - Es posible usar SSL para codificar las comunicaciones, pero depende del servidor de aplicaciones

Tipos de seguridad en EJB (2)

- 3. Autorización (¿está el cliente autorizado a acceder a un método del bean?)
  - Autorización declarativa (lo usual): se restringe el acceso a métodos mediante entradas en el fichero de despliegue ejb-jar.xml
  - Autorización por programación (no usual): en el código de los beans se pregunta por el usuario o el rol del cliente

Autentificación

- El siguiente código proporciona un ejemplo de autentificación de un cliente usando JNDI y BEA WebLogic Server

```
properties.put(Context.SECURITY_PRINCIPAL, "domingo" );
properties.put(Context.SECURITY_CREDENTIALS, ”12345678");
properties.put(Context.PROVIDER_URL, "t3://localhost:7001");
javax.naming.Context jndiContext =
   new javax.naming.InitialContext(properties);
```

- El usuario domingo debe estar creado en el servidor de aplicaciones con la contraseña “12345678”

Autorización programativa

- El principal (usuario) domingo se propagará al contexto de los beans creados en las siguientes llamadas

Cliente

```
Context jndiContext =
      new InitialContext(properties);
Object ref = jndiContext.lookup(
                        "TraderBean");
TraderHome home = (TraderHome)
   PortableRemoteObject.narrow(
                  ref, TraderHome.class);
Trader trader = home.create();
trader.compra("BEAS",500);
```

Contenedor EJB

```
public void compra(String symbol,
                   float amount) {
  ...
  Principal currentUser =
     ejbContext.getCallerPrincipal();
  // devuelve currentUser = domingo

  if (ejbContext.
        isCallerInRole("registered")) {
    ...
  }
}
```

¡Autorización por programación!

Autorización declarativa

Declaración de los roles

Declaración de los permisos

```
<asembly-descriptor>
   <security-role>
      <role-name>Administrador</role-name>
      <role-name>Usuario-registrado</role-name>
   </security-role>
   <method-permission>
      <role-name>Administrador</role-name>
      <role-name>Usuario-registrado</role-name>
      <method>
         <ejb-name>Trader</ejb-name>
         <method-name>compra</method-name>
      </method>
   <method-permission>
</asembly-descriptor>
```

Asignación de principals a roles

- La asignación de Principals a roles se hace en tiempo de despliegue, en el fichero weblogic-ejb-jar.xml:

```
<security-role-assignment>
   <role-name>Administrador</role-name>
   <principal-name>Administrators</principal-name>
   <principal-name>dmartinez</principal-name>
</security-role-assignment>
```

Asignación de principals a roles

- Los componentes EJB pueden ejecutarse como un rol determinado, ignorando el rol del cliente que les llama

ejb-jar.xml

```
<security-identity>
   <run-as>EJBAppAdmin</run-as>
</security-role-assignment>
```

weblogic-ejb-jar.xml

```
<run-as-role-assignment>
   <role-name>EJBAppAdmin</role-name>
   <run-as-principal-name>
      dmartinez
   </run-as-principal-name>
</run-as-role-assignment>
```

Anotaciones WebLogic

- Para declarar la autorización de acceso a un método del EJB:

```
@RemoteMethod(roles = "admin, bibliotecario")
public void addUsuario(String login) {
    ...
}
```

- Para declarar los principals asociados a roles:

```
@RoleMappings(
      {@RoleMapping(principals = "weblogic, admin",
                    roleName = "administrador"),
       @RoleMapping(principals = "dgl, admin, Bibliotecario",
                    roleName = "bibliotecario")})
public class UsuarioBean extends GenericSessionBean
                                  implements SessionBean {
```

¿Preguntas?