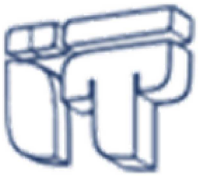


Seguridad

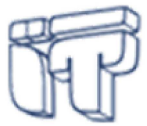
Visión general
Seguridad en componentes Web
Seguridad en clientes de consola
Seguridad en EJBS
Seguridad EIS

Pablo Basanta Val

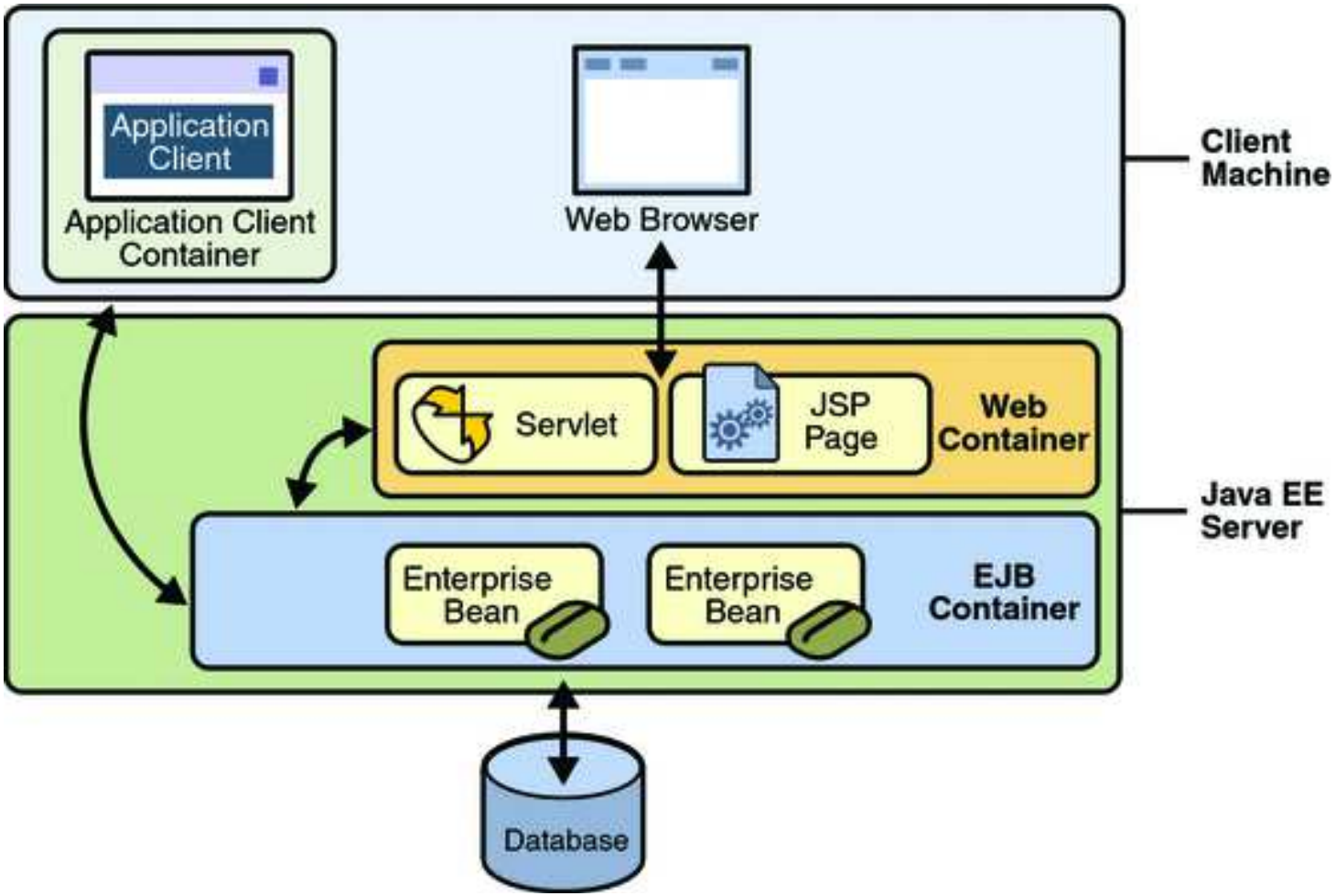
pbasanta@it.uc3m.es

La seguridad dentro de J2EE

- Muchas aplicaciones requieren proveer la identidad de los usuarios que están accediéndolas
  - Por ejemplo un banco, aula global, …
- J2EE y los EJBs proveen un modelo de seguridad híbrido (tanto programático como declarativo) basado en:
  - Autentificación (verificación de credenciales)
  - Autorización (con lo que puede interactuar un usuario)
  - Confidencialidad y integridad (soporte contra ataques, como por ejemplo SSL)
- En la arquitectura de J2EE se reparte la responsabilidad de la seguridad entre sus diferentes roles.
  - Administrador del sistema, proveedor del contenedor, el desarrollador de aplicaciones y el encargado de despliegue



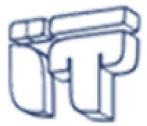
Arquitectura J2EE



Fuente:
Java EE Tutorial 1.4 , Fourth Edition

Autenticación

- Es el proceso de verificar si alguien es quien dice ser.
  - Típicamente mediante login y password
  - La clave puede estar en muchos formatos
  - Una vez autenticado, al usuario se le asigna un rol
- Los EJBs no especifican cómo se hace este proceso, es responsabilidad del servidor de aplicaciones
  - Se puede hacer con JNDI (por ejemplo contra LDAP de la universidad) o por ejemplo en Jboss se usa JAAS (Java Autentication and Autorización Service) o realms



Autorización

- Una vez autenticado el componente, se procede por un mecanismo de autorización, basado en roles
- En J2EE las políticas se aplican sobre roles y no sobre usuarios

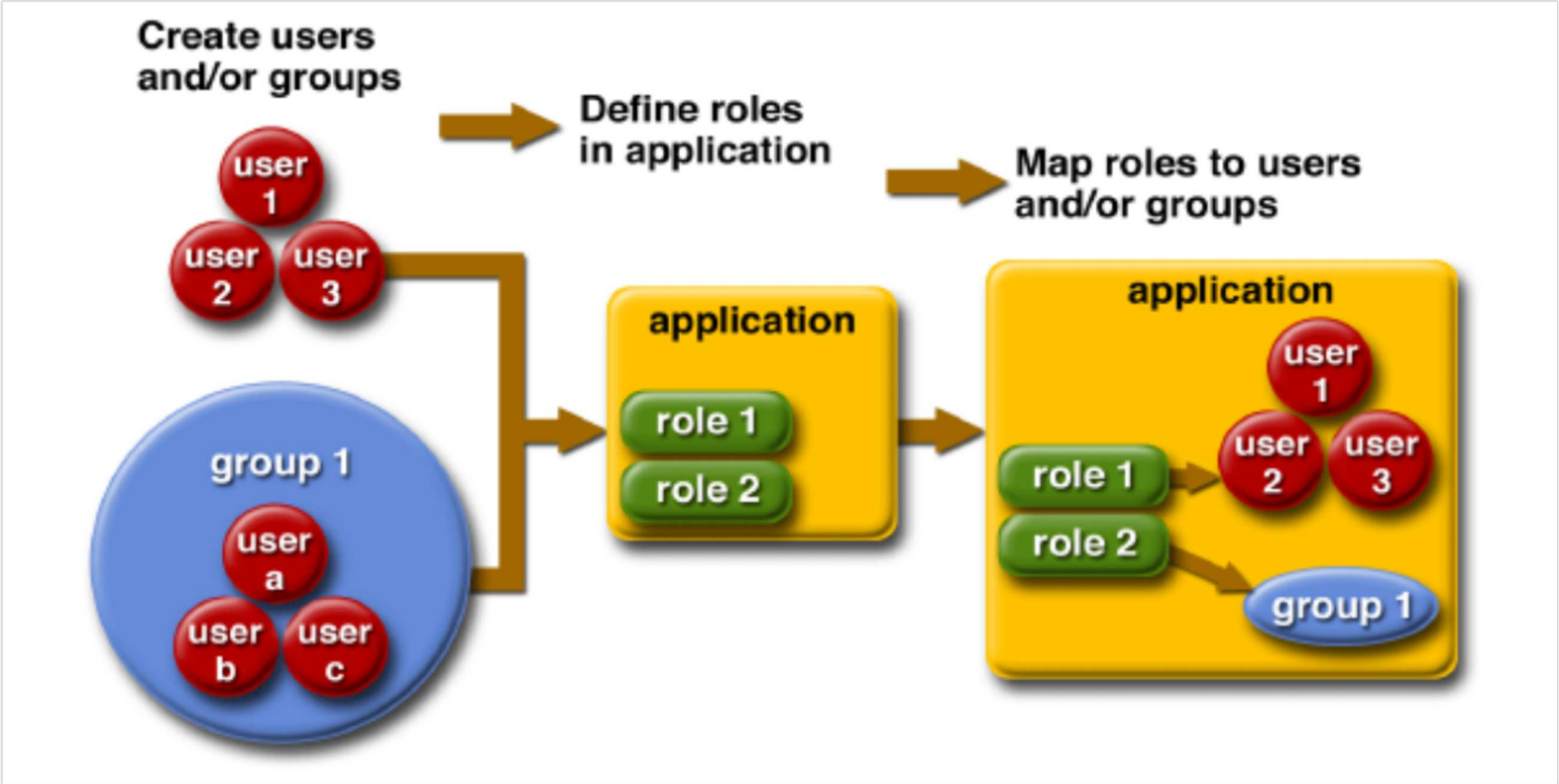
Reinos (Realms)

- También conocidos como *security policy domains* or security *domains*.
- Son ámbitos sobre los cuales se definen entidades de seguridad similares.
- El AppServer de Sun soporta:
  - File, Ldap, Certificate y Solaris
- Su configuración es dependiente de la plataforma
  - En el AppServer se puede utilizar la interfaz web a través del puerto 4848

Usuario, grupo y rol

- Usuario: identidad de un usuario
  - Ejemplo: Pablo Basanta
- Grupo: entidad colectiva de un grupo de usuarios
  - Ejemplo: Profesor de la asignatura de J2EE
- Rol: Desde el punto de vista lógico de la aplicación es un perfil
  - Ejemplo: profesor

Mapeo de roles a usuarios y/o grupos

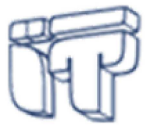


Fuente:
Java EE Tutorial 1.4 , Fourth Edition

Asignación de usuarios a grupos

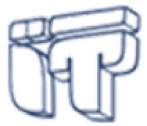
```
<sun-ejb-jar>
. . .
  <security-role-mapping>
    <role-name> AUTHORIZED_MERCHANT </role-name>
    <principal-name>JuanPalomo</principal-name>
    <group-name>TravelAgent</group-name>
  </security-role-mapping>
. . .
</sun-ejb-jar>
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

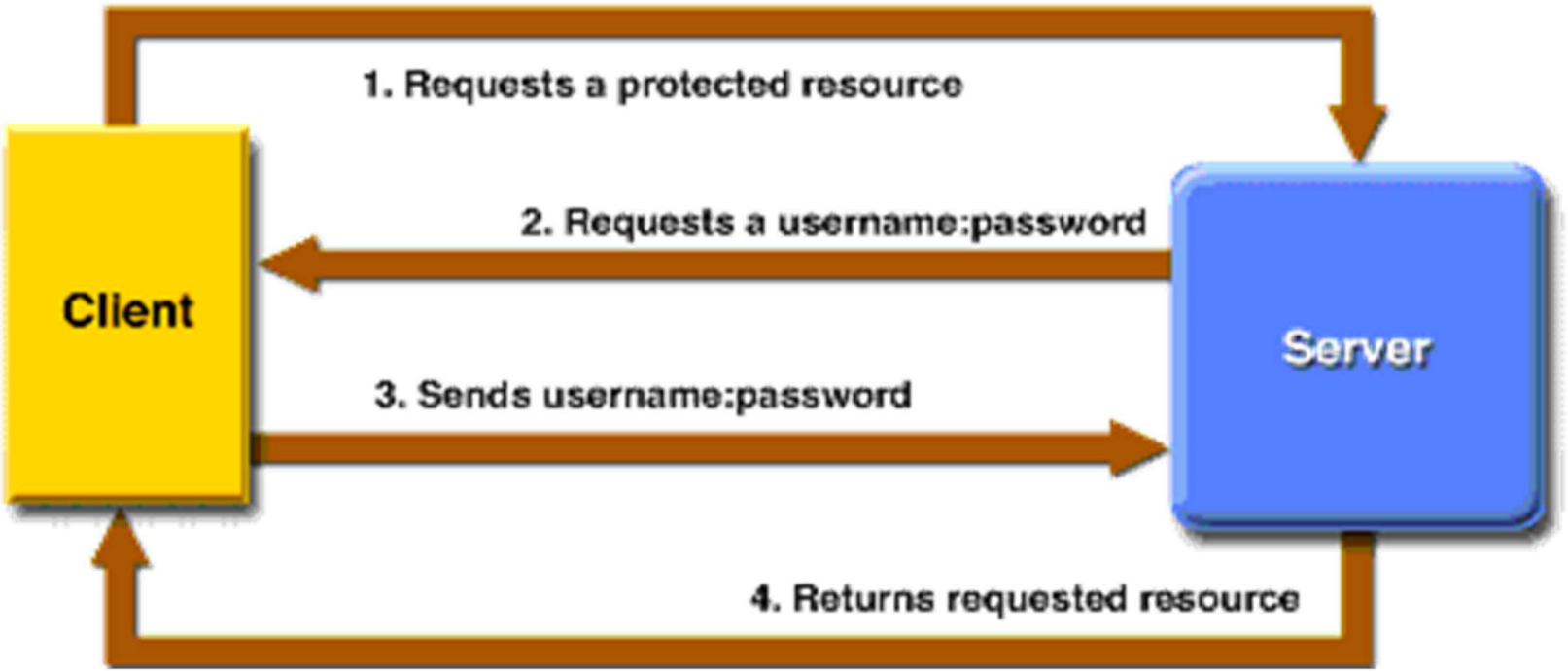


Mecanismos de autenticación

- Los clientes (web o appclients) se autentican antes de poder entrar
- En aplicaciones web se soportan varios tipos de autenticaciones
  - **HTTP basic authentication**
  - **Form-based login**
  - **Client certificate**
  - **Mutual authentication**
  - **Digest authentication**
- Mientras que para clientes de consola se suelen preferir otros mecanismos
  - **JAAS** (*Java Autorization and Autentication System*)

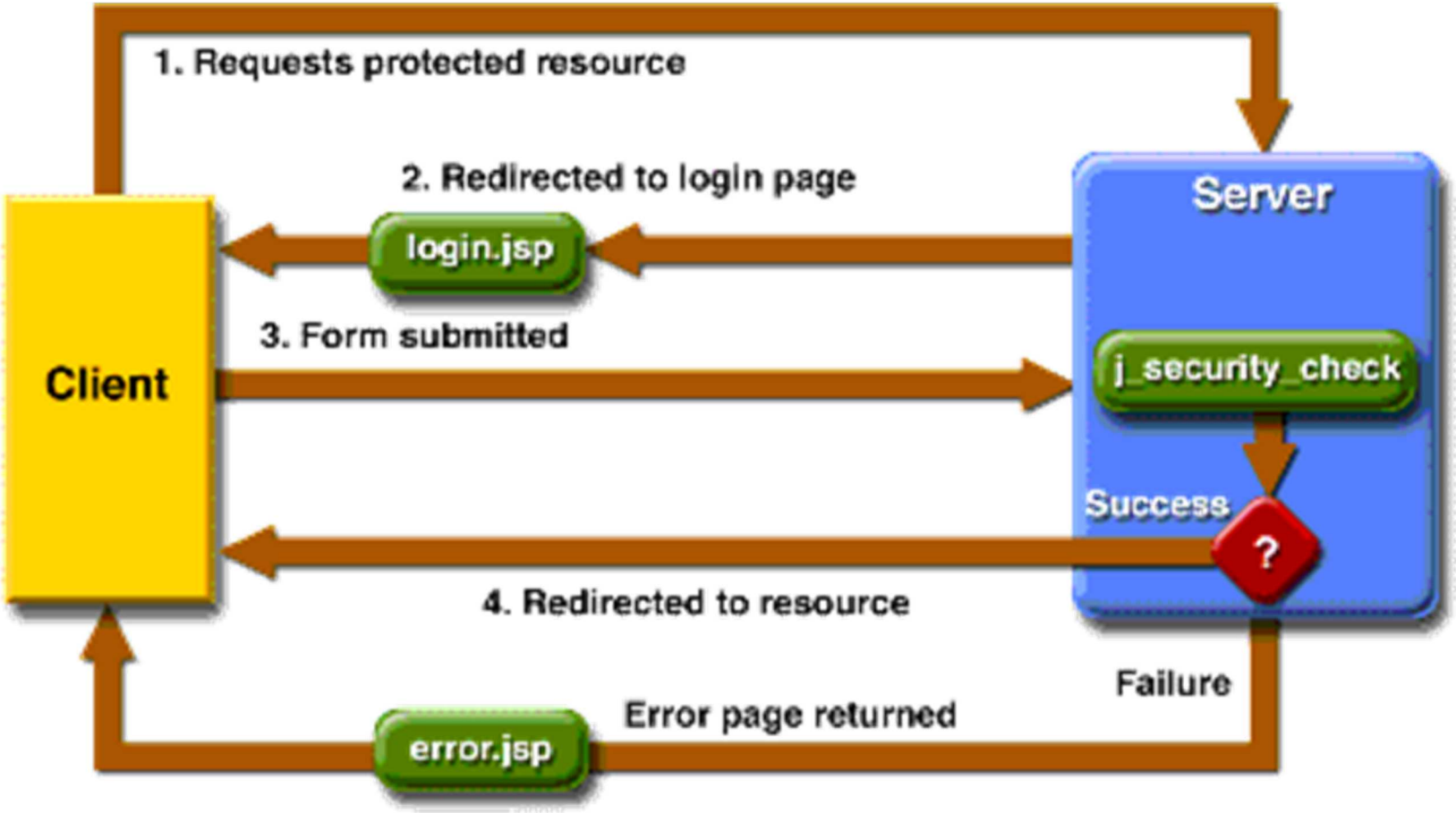


HTTP Basic



Fuente:
Java EE Tutorial 1.4 , Fourth Edition

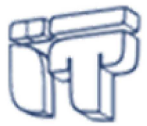
Autenticación basada en formulario



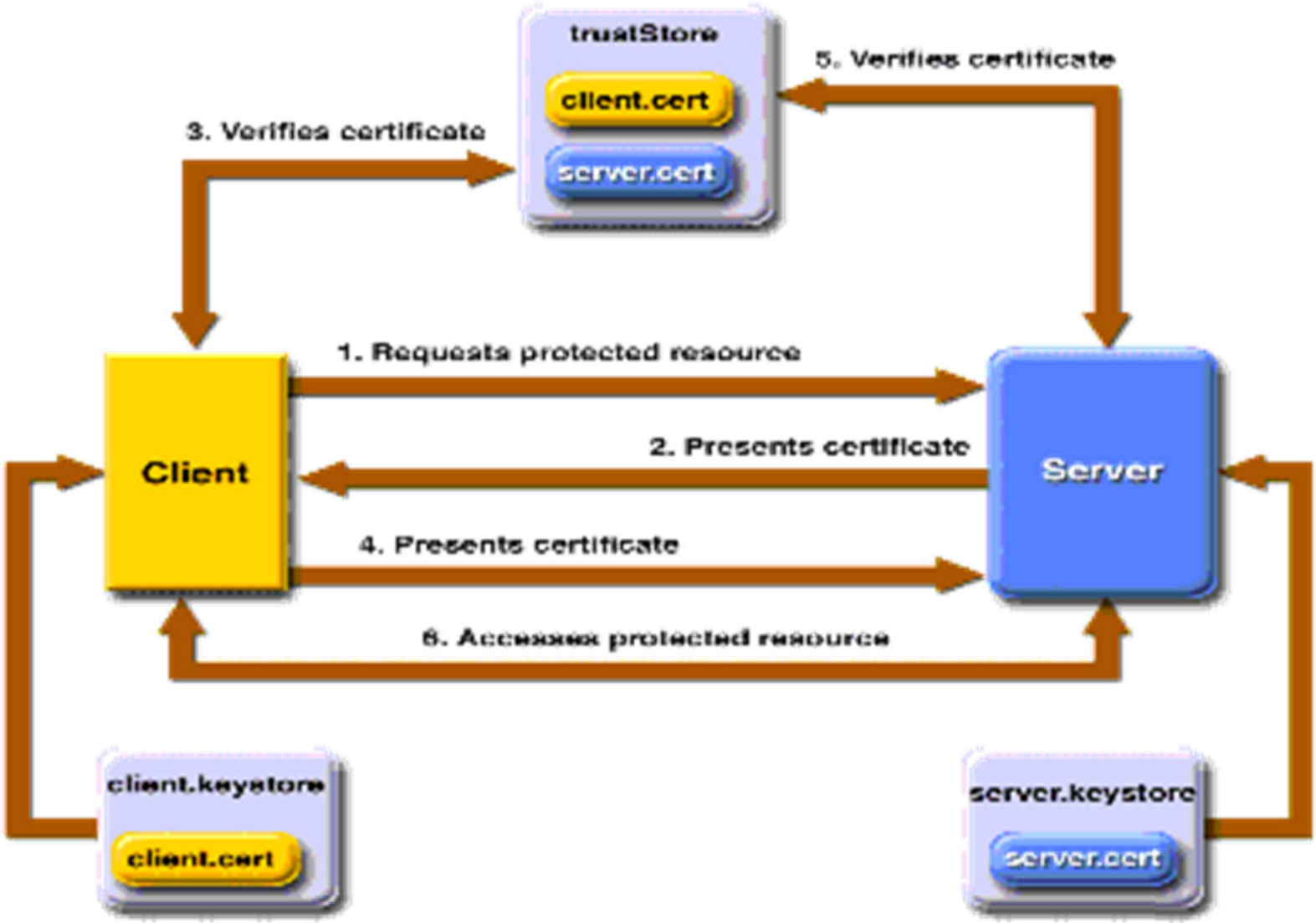
Fuente:
Java EE Tutorial 1.4 , Fourth Edition

Basada en certificado de cliente

- Utiliza HTTP sobre SSL (Secure Server Layer), por lo que es bastante segura
- SSL ofrece tanto encriptación, como autenticación del servidor, como integridad de mensaje como identificación opcional del cliente (opcional) para conexiones TCP/IP
- En caso de hacerse la autenticación del cliente se utiliza un certificado X.509
- Antes de usar esta configuración se ha de habilitar dicho tipo de seguridad en el servidor



Autentificación mutua



Fuente:
Java EE Tutorial 1.4 , Fourth Edition

Digest

- Es bastante parecida a la http basic pero más segura.
  - HTTP-basic utiliza 64-base encoding
- Pero sin embargo no es muy utilizad, aunque J2EE la mantiene por universalidad

Ejemplo de autenticación con formulario (1/2)

```
<body>
  <form action="j_security_check">
  Enter your user name: <input type="text" name="j_username"/>

  Enter your password: <input type="text" name="j_password"/>
 </form>
```

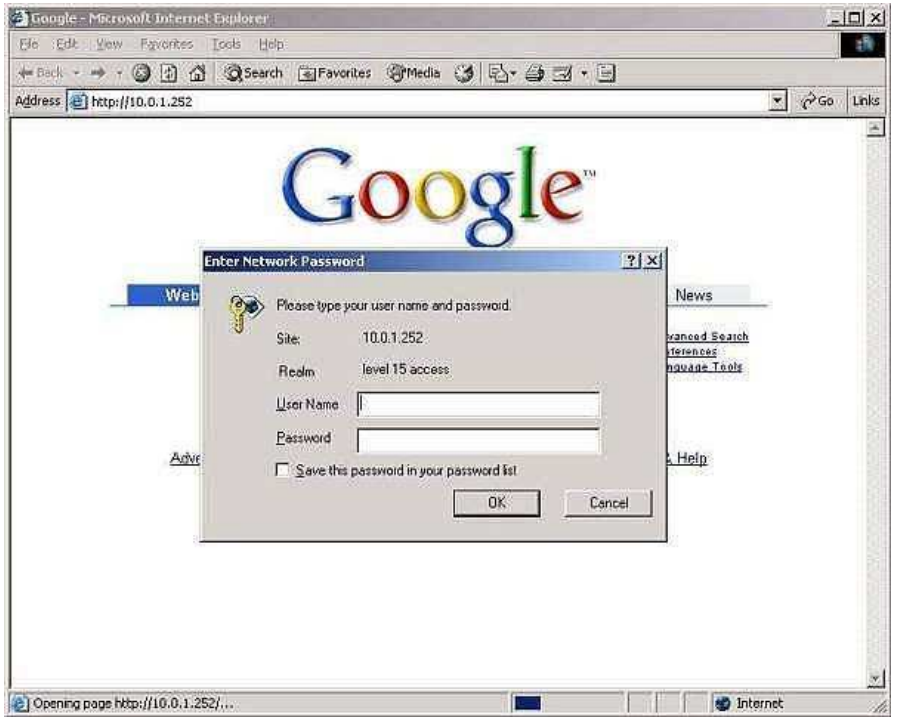
Fuente:
Java EE Tutorial 1.4 , Fourth Edition

Ejemplo de autenticación con formulario (2/2)

```
<login-config>
  <auth-method>FORM</auth-method>
    <form-login-config>
       <form-login-page>login.jsp</form-login-page>
       <form-error-page>login-invalid.jsp</form-error-page>
  </form-login-config>
</login-config>
```

Fuente:
Java EE Tutorial 1.4 , Fourth Edition

Ejemplo de configuración HTTP-basic (1/2)

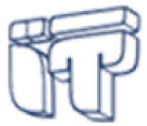


Fuente:
Java EE Tutorial 1.4 , Fourth Edition

Ejemplo de configuración HTTP-basic (1/2)

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

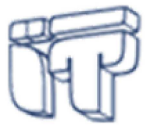
Fuente:
Java EE Tutorial 1.4 , Fourth Edition



Asignación de restricciones a componentes web

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Protected Area</web-resource-name>
    <url-pattern>/private <url-pattern>
    <http-method>PUT</http-method>
    <http-method>DELETE</http-method>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>role1</role-name>
    <role-name>employee</role-name>
 </auth-constraint>
</security-constraint>
```

Fuente:
Java EE Tutorial 1.4 , Fourth Edition



Seguridad (extra) en la autenticación

- Tanto la autenticación básica http como la basada en formulario carecen de seguridad y las claves enviadas pueden ser interceptadas por otros clientes
- Se puede hacer que esta autentificaciones se cifren con SSL haciendo que sean confidenciales o integrales
  - CONFIDENCIAL= nadie leer la comunicación
  - INTEGRAL= nadie puede alterarlo

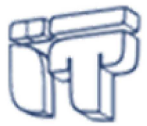
Autenticación de clientes

- Una primera alternativa basada en JAAS
  - El cliente implementa un clase especial javax.security.auth.callback.CallbackHandler
- Una segunda alternativa basada en login programático
  - Solo para clientes EJB en Appserv de Sun (uso restringido) com.sun.appserv.security.Programmatic

Fragmento de autenticación de cliente con JAAS

```
LoginContext ctx = null;
//Creación de contexto
try {
ctx = new LoginContext("WeatherLogin",
                       new   MyCallbackHandler());
} catch(LoginException le) { System.exit(-1);
} catch(SecurityException se) { System.err.println("Error en
el contexto "}
//Login
try {
ctx.login(); } catch(LoginException le) {
    System.out.println("Authentication failed. "}
System.out.println("Ya estás autenticado");
```

Fuente:
Java EE Tutorial 1.4 , Fourth Edition

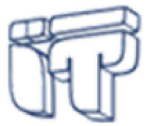


Seguridad (extra) en los clientes

- Se puede configurar a IIOP para que utilice características avanzadas (seguridad, confidencialidad, confianza), relacionadas con las seguridad de los clientes

```
<ior-security-config>
<transport-config>
     <integrity>NONE</integrity>
     <confidentiality>NONE</confidentiality>
     <establish-trust-in-target> NONE </establish-trust-in-target>
     <establish-trust-in-client> NONE </establish-trust-in-client>
</transport-config>
</ior-security-config>
```

Fuente:
Java EE Tutorial 1.4 , Fourth Edition

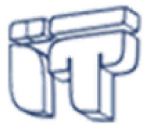


Seguridad programática (dentro del componente)

- En el API consta de dos métodos:
  - `getCallerPrincipal`, el cual permite obtener la identidad del invocante
  - `isCallerInRole`, que permite asociar una sesión a un cliente

Method Summary	
`java.security.Principal`	**`getCallerPrincipal``()`** Obtain the `java.security.Principal` that identifies the caller.
`boolean`	**`isCallerInRole``(java.lang.String roleName)`** Test if the caller has a given security role.

http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/ejb/EJBContext.html



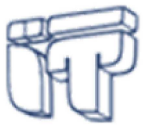
Ejemplo de seguridad programática

```
[…]
InitialContext ctx =new InitialContext();
Principal caller= ctx.getCallerPrincipal();
String travelAget =caller.getName();

if ctx.isCallerInRole("JUNIOR_TRAVEL_AGENT")
   throw new IllegalRoleh();

[…]
```

Fuente:
Java EE Tutorial 1.4 , Fourth Edition



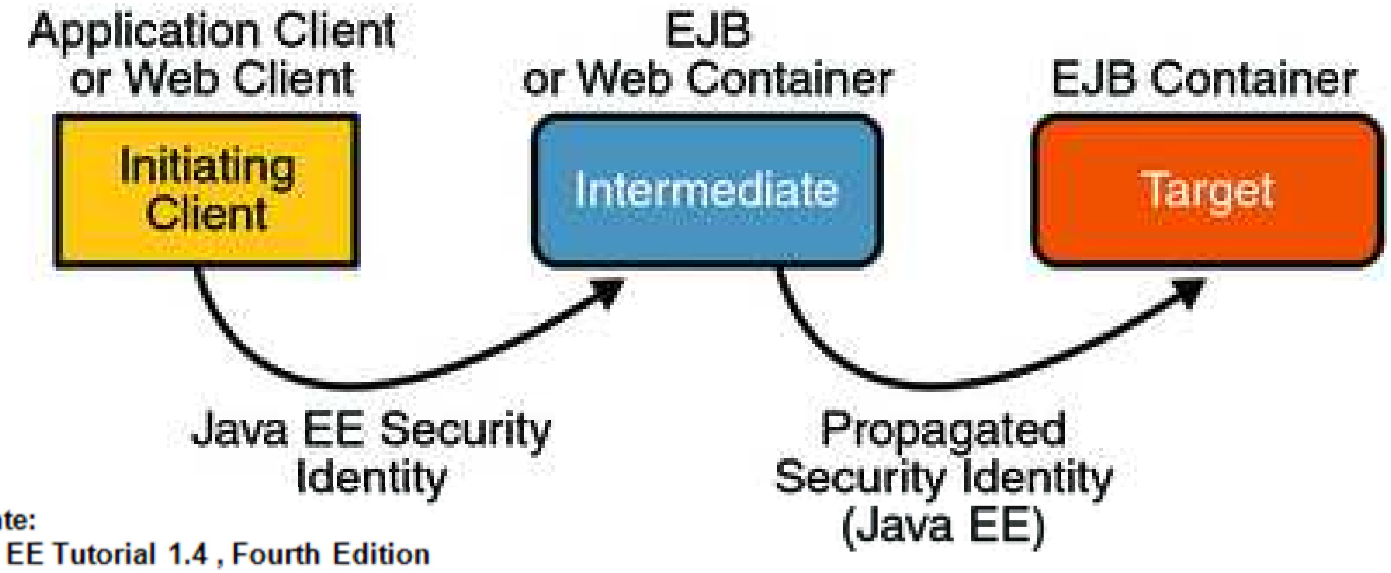
Seguridad declarativa en EJBs

- En los descriptores xml se pueden definir restricciones en los accesos (en función de los roles)

```
<ejb-jar version=2.0>
 <assembly-descriptor>
  <security-role>
   <description/>
   <role-name>AUTHORIZED_MERCHANT</role-name>
  <security-role>
 </method-permission>
   <role-name>AUTHORIZED_MERCHANT</role-name>
   <method>     <ejb-name>ProcessPaymetnBean</ejb-name>
                <method-name>byCredit</method-name>
   </method>
  <method-permission>
. . .
```

Fuente:
Java EE Tutorial 1.4 , Fourth Edition

Propagación de roles



Fuente:
Java EE Tutorial 1.4 , Fourth Edition

- Por defecto, la identidad se propaga dentro de entornos seguros (por ejemplo de la aplicación web al contenedor de EJBs). Esto sucede cuando hay confianza en el otro extremo.
- Se puede cambiar esta identidad con el elemento `<run-as>`.

Seguridad EIS

- La capa EIS puede requerir una conexión (con seguridad) a un elemento
  - Por ejemplo a una base de datos
- J2EE ofrece dos alternativas:
  - **Container-Managed Sign-On**
    - **Donde el programador es responsable de enviar datos al servidor sobre la atentificación**
  - **Component-Managed Sign-On**
    - **Donde el programador es responsable de enviar datos al servidor sobre la atentificación**

Cuestiones para reflexionar

Bloque I y II: Transacciones

- ¿Porqué cree que J2EE ofrece un doble soporte, basado en un modelo declarativo y otro programático para trabajar con transacciones?
- ¿Describa un caso en el llamar a un método transaccional etiquetado como **requiresNew** surta el mismo efecto que llamar a uno **required**?
- ¿Qué son los atributos transaccionales de J2EE?¿Cuantos hay? Enumérelos
- ¿Cuál es la diferencia existente entre JTS y JTA?

Bloque III, IV: Seguridad

- Discuta sobre cual puede haber sido el motivo que ha empujado a los diseñadores de J2EE a incluir dentro de sus modelo soporte directo para la seguridad.
- ¿Qué es un *Realm*?
- ¿En qué se diferencian autorización y autenticación?
- ¿Cómo se llama el principal mecanismo de autenticación utilizado para clientes de consola?
- ¿Qué significa JAAS?