



Université Mohammed V de Rabat

Ecole Supérieure de Technologie de Salé

Chap 3

Sécurité des Applications

Pr R.ALAOUI

Introduction

Réserver des parties d'une application à certains utilisateurs implique de gérer:

- l'authentification des utilisateurs**
- la vérification des autorisations**

Les différents niveaux de sécurité

- **Sécurité physique**
 - ▶ Relative à la protection des locaux et des machines
- **Sécurité du personnel**
 - ▶ Relative à la protection physique des employés et à la protection du S.I. de l'entreprise contre ces employés
- **Sécurité des communications**
 - ▶ Relative à la protection du système de communication (réseau)
- **Sécurité des opérations**
 - ▶ Relative à la protection des échanges de données et des systèmes informatiques

La politique de sécurité

Définition : Une politique de sécurité informatique est une **stratégie** visant à **maximiser la sécurité informatique d'une entreprise**.

Elle est **matérialisée** dans un **document** qui reprend l'ensemble des **enjeux, objectifs, analyses, actions et procédures** faisant parti de cette stratégie.

à distinguer de la ***charte informatique***, qui est un **document de recommandations** concernant la **bonne utilisation des technologies informatiques**, et qui est destiné aux employés de l'entreprise

Le document de **politique de sécurité** est **unique et personnalisé**, qui tient en compte le **fonctionnement**, la **composition du système d'information** de l'entreprise et les **risques** informatiques qui lui sont propres

Nécessité de définir une politique de sécurité

Toute organisation doivent **respecter l'ensemble de règles formalisées** auxquelles les personnes ayant accès aux ressources technologiques et aux S.I.

Composantes d'une politique de sécurité

Politique d'achat

Politique de confidentialité

Politique d'accès

Politique de responsabilité

Politique d'authentification

Politique d'audit et de reporting

Objectifs

La sécurité informatique consiste à **assurer** que les **ressources matérielles ou logicielles** d'une organisation sont **uniquement utilisées** dans le cadre prévu.

La sécurité informatique vise généralement cinq principaux objectifs :

- **Authentification**
- **Identification**
- **Intégrité**
- **Non-répudiation**
- **Confidentialité**

Authentification

- **L'authentification** est la procédure qui consiste, pour un système informatique, à **vérifier l'identité d'une entité** afin **d'autoriser l'accès** de cette entité à des ressources (systèmes, réseaux, applications...).
- **L'authentification permet donc de valider l'authenticité de l'entité en question.**
- **Elle protège de l'usurpation d'identité**

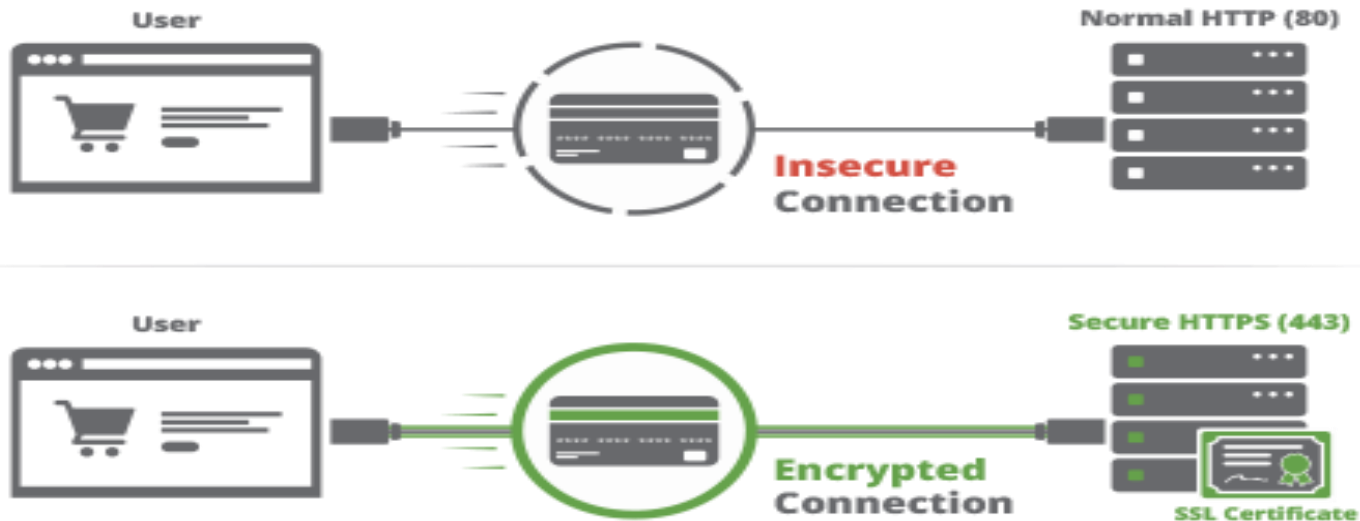
- **Les entités à authentifier peuvent être :**
- **Une personne**
- **Un programme qui s'exécute (processus)**
- **Une machine dans un réseau (serveur ou routeur)**

SSL (Secure Socket Layer)

Le protocole SSL (Secure Sockets Layer) était le protocole cryptographique le plus largement utilisé pour assurer la sécurité des communications sur Internet

L'adresse URL passe alors de HTTP a HTTPS, le 'S' signifiant 'sécurisé'.

HTTP vs HTTPS



Qu'est-ce qu'un certificat SSL ?

Un certificat SSL est un **fichier de données** qui lie une clé **cryptographique** aux informations d'une organisation, **installé sur un serveur**

Le certificat active le cadenas et le protocole « https », afin d'assurer une connexion sécurisée entre le serveur web et le navigateur.

Plusieurs entreprises spécialisées dans la sécurisation des échanges d'informations en ligne, **fournissent** aujourd'hui des **certificats SSL**.

Parmi les application, Le SSL est utilisé pour **sécuriser les transactions bancaires**, le transfert de données et les informations de connexions.

SSL est un complément à TCP/IP et permet (potentiellement) de sécuriser n'importe quel protocole ou programme utilisant TCP/IP. SSL est devenu la norme pour sécuriser la navigation sur les sites de réseaux sociaux.

Authentification

- **Une authentification simple** est une procédure d'authentification qui requiert un seul élément ou « facteur » d'authentification valide pour permettre l'accès à une ressource.

Ex. login/password sur Windows ou Linux

- **Une authentification forte** est une procédure d'authentification qui requiert au moins deux éléments ou « facteurs » d'authentification valides pour permettre l'accès à une ressource

Ex. carte bancaire (1. être en possession de la carte; 2. connaître le PIN)

- **Une authentification mutuelle** impose une double authentification entre les deux entités client et serveur

Exemple: Dans un environnement de réseau privé (entreprise)

l'authentification mutuelle est un moyen de permettre au client de vérifier ou d'authentifier le serveur de l'entreprise et qu'il pourra accéder à toutes les données du serveur autorisées avec ses informations d'identification d'accès. En même temps, le serveur authentifiera le client, en vérifiant les informations d'identification et les autorisations saisies par rapport au profil créé pour le client.

Identification

L'identification permet donc de connaître l'identité d'une entité alors que l'authentification permet de vérifier cette identité

L'authentification peut inclure une phase d'identification, au cours de laquelle l'entité indique son identité.

Cependant, cela n'est pas obligatoire ; il est en effet possible d'avoir des entités munies de droits d'accès mais restant anonymes.

Non-répudiation

Un mécanisme de non-répudation permet d'empêcher à une personne de nier le fait qu'elle a effectué une opération

(exemple : envoi d'un message, passage d'une commande).

Pour assurer la non-répudiation d'un message, on peut, par exemple,

Solution: utiliser la signature électronique

La signature électronique sert à authentifier et approuver officiellement des documents importants comme ceux liés au juridique et légal par exemple.

il existe des **logiciels appropriés** utilisant une **technique cryptographique spécifique**.

Exemple:

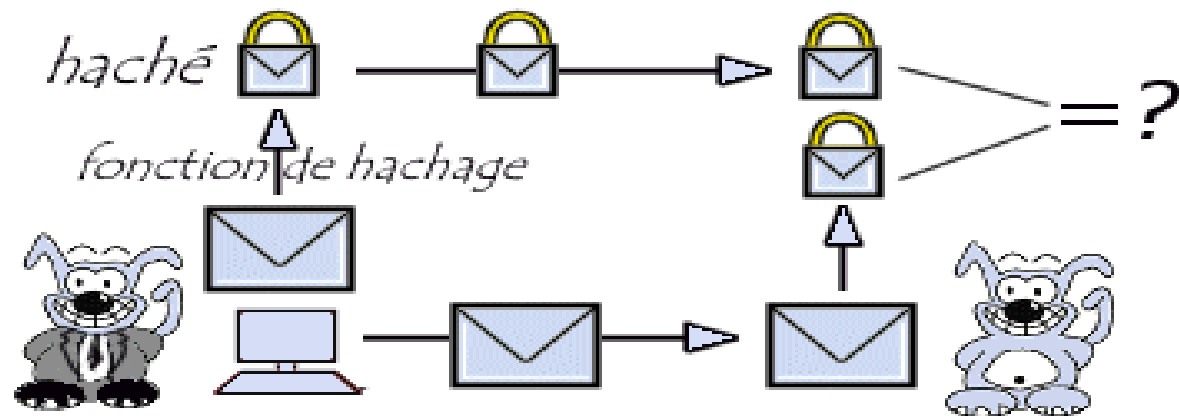
DocuSign, Eversign, Yousign, SignEasy...

Vous pouvez signer tous les types de fichiers, tels que Word, PDF, jpeg ,XML ...

Intégrité

L'intégrité des données consiste à vérifier qu'elles n'ont pas été altérées d'une manière accidentelle ou avec fraude au cours de leur transmission ou de leur stockage.

Ce principe regroupe un ensemble de fonctionnalités mises en œuvre afin de s'assurer de leur intégrité, comme les **fonctions de hachage** telles que MAC (Message Authentication Code).



Principe de hachage

Lors de la réception du message, il suffit au destinataire de **calculer le haché du message reçu** et de le **comparer** avec le **haché accompagnant le document**.

Si le message (ou le haché) a été modifié durant la communication, les **deux empreintes ne correspondront pas**.

Confidentialité

La confidentialité est la propriété qui **assure** qu'une **information ne peut être lue que** par **des entités habilitées** (selon des contraintes précises)

Le chiffrement (parfois appelé à tort cryptage) est le procédé qui rendre la **compréhension d'un document impossible** à toute personne qui n'a pas la clé de (dé)chiffrement.

On distingue deux familles de systèmes de chiffrement :

- ☐ **Chiffrement symétrique ou à clé privé**
- ☐ **Chiffrement asymétrique (à clé publique)**
(en réalité utilisant une paire de clés)

Que peut-on sécuriser ?

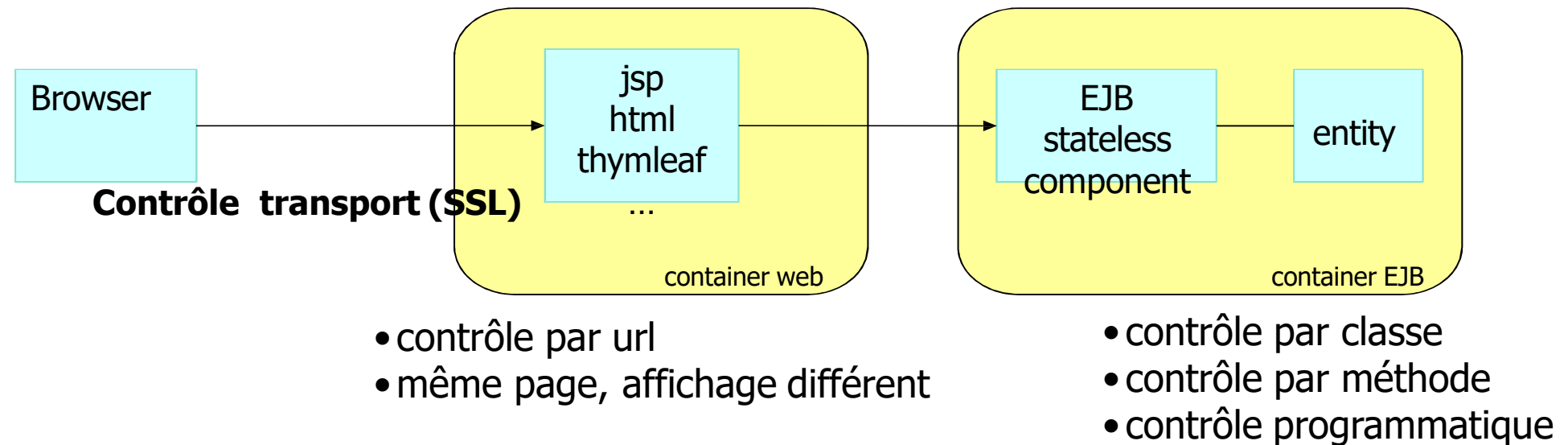
Cas d'utilisations (web dynamique)

- Un site offre des pages et des services personnalisés selon l'utilisateur
 - Le site permet aux utilisateurs de s'enregistrer
 - Le site permet aux utilisateurs de s'identifier
- Un site offre des pages accessibles uniquement si l'utilisateur est identifié

Cas d'utilisation (EJB | domaine)

- Certaines classes ou méthodes ne sont accessibles que par un « type » d'utilisateur

Que peut-on sécuriser ?



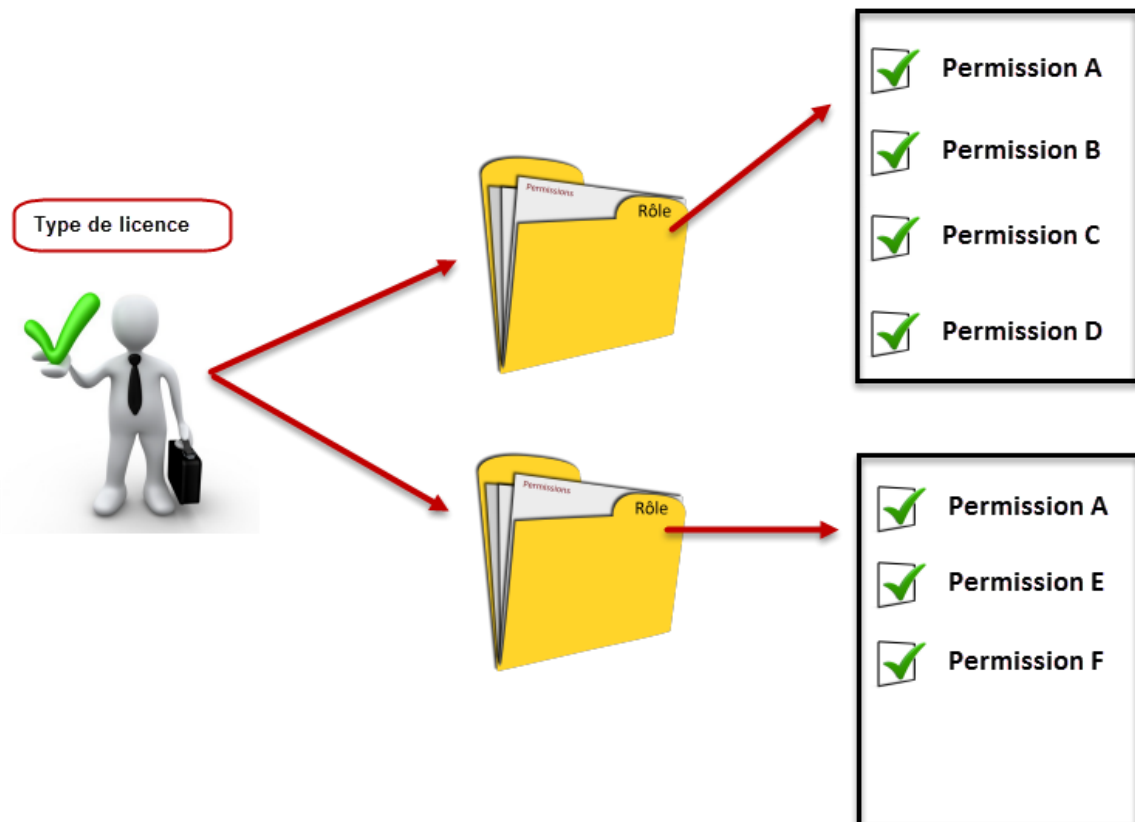
- Un site a des **pages spécifique selon le «rôle» de l'utilisateur** :
 - pages pour les **vendeurs**
 - pages **administrateurs**
 - pages pour les **clients**
- Une même page **affiche des données différentes selon le rôle de l'utilisateur** :
 - en **lecture seule** si client
 - en **lecture/écriture** si vendeur

Gestion des utilisateurs, des rôles et des accès

Dans la sécurité Java EE, les concepts d'**Utilisateur**, **Groupe** et **Rôle** sont essentiels pour gérer les contrôles d'accès.

Vous pouvez afficher et modifier les droits d'accès des utilisateurs et des groupes d'utilisateur sur les ressources.

Les droits d'accès sont déterminés par l'affectation de rôles.



Utilisateur

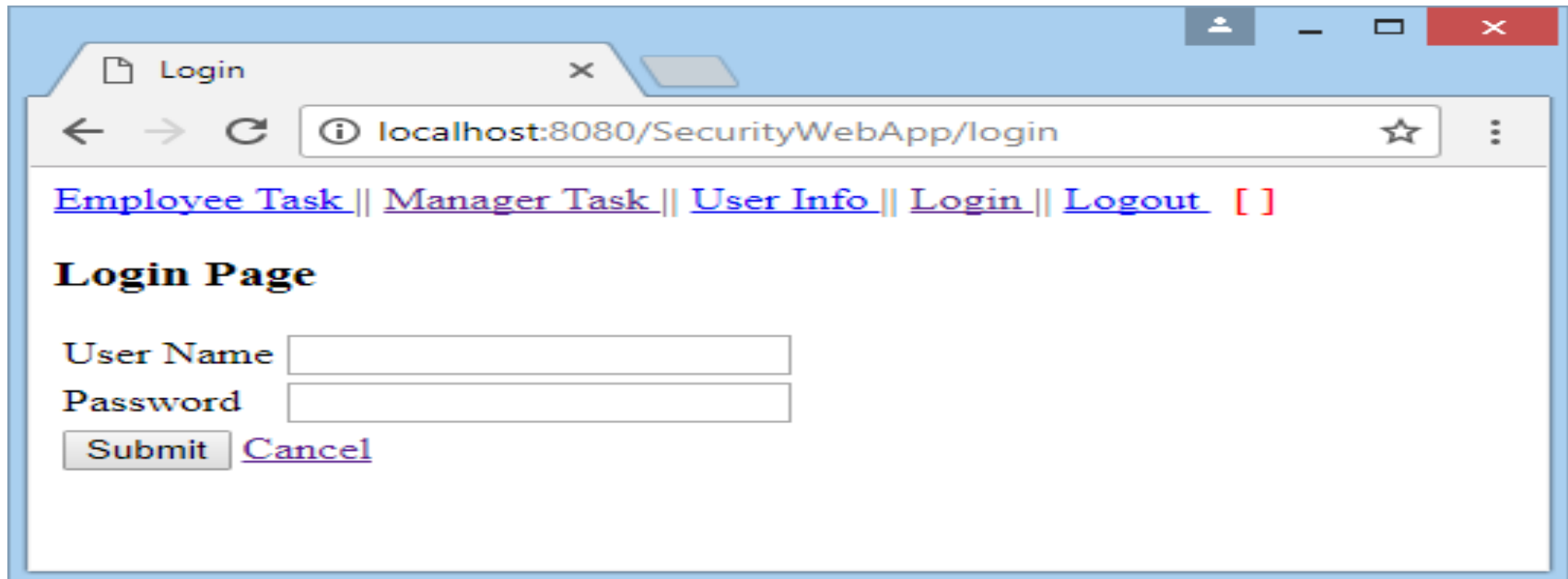
Un *utilisateur* est une identité individuelle authentifiée (ex : "alice", "bob").

L'utilisateur peut correspondre à une personne ou à une (partie d'une) application informatique

- **Authentification** : Il se connecte via des mécanismes comme **FORM, BASIC ...**

Le plus souvent la saisie d'un nom de login et d'un mot de passe prouve l'identité d'un utilisateur

- **Stockage** : Les utilisateurs sont généralement gérés dans un **répertoire de sécurité** (LDAP, base de données, fichier file-realm).



The screenshot shows a web browser window with a single tab titled "Login". The address bar displays "localhost:8080/SecurityWebApp/login". The page content includes a navigation menu with links: [Employee Task](#), [Manager Task](#), [User Info](#), [Login](#), and [Logout](#), followed by a red "[]" indicator. Below the menu is the heading "Login Page". The login form consists of two input fields: "User Name" and "Password". At the bottom of the form are two buttons: "Submit" and "Cancel".

Les groupes

Un **groupe** est un **ensemble d'utilisateurs** partageant des **permissions communes** (ex : "admins", "clients").

Utilité : **Simplifie la gestion des droits** en regroupant des utilisateurs (ex : tous les membres du groupe "HR" ont accès aux ressources RH).

Hiérarchie : Les groupes sont définis dans le répertoire de sécurité, indépendamment de l'application.

USERS



GROUPS



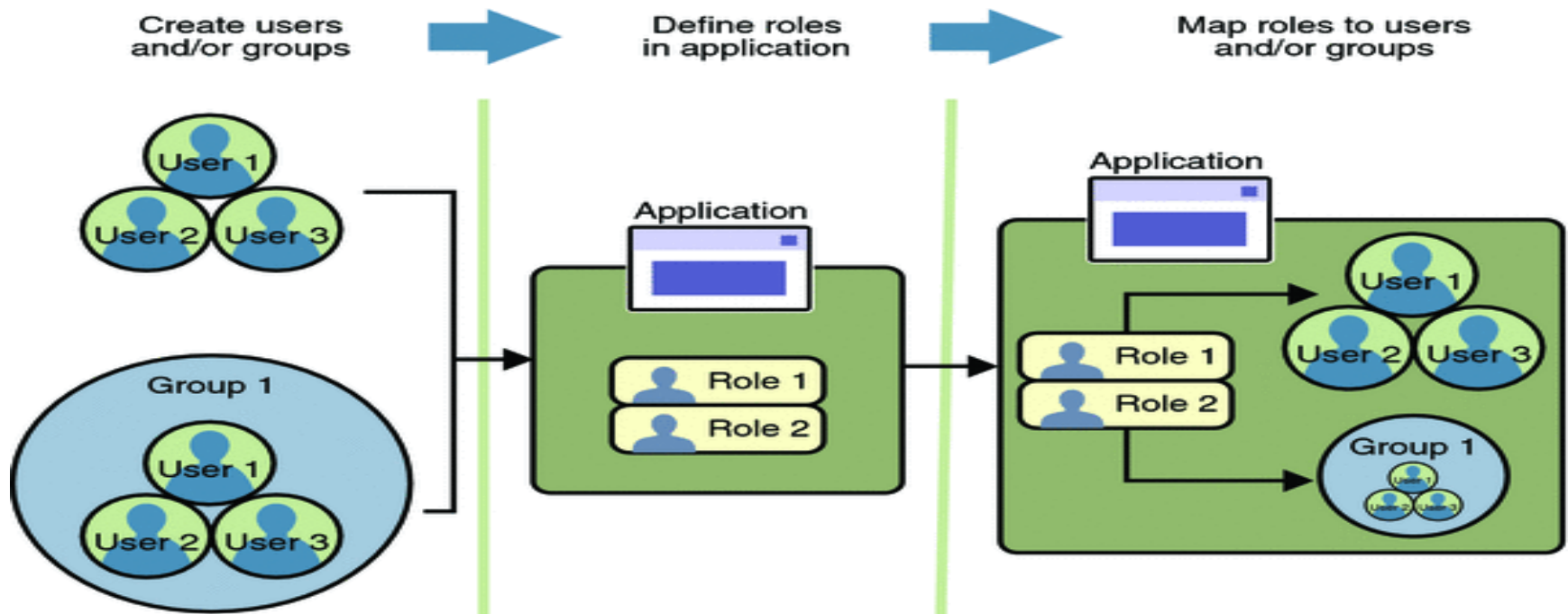
Rôle:

Un rôle est une **abstraction applicative** des **permissions** (ex : "ADMIN", "USER").

Découplage : Les rôles sont définis dans l'application (via web.xml, annotations ou @DeclareRoles), sans dépendre de l'infrastructure externe.

Utilisation : Les ressources (servlets, EJB) sont sécurisées en spécifiant les rôles requis (ex : @RolesAllowed("ADMIN")).

Des rôles différents pour chaque ressource peuvent être attribués aux utilisateurs et aux groupes d'utilisateurs.



Interaction entre Utilisateurs, Groupes et Rôles

1. Authentification :

- L'utilisateur fournit ses identifiants (ex : via un formulaire).
- Le serveur (ex : WildFly, GlassFish) vérifie son existence dans le répertoire de sécurité.

2. Mapping Groupe → Rôle :

Le serveur **mappe les groupes de l'utilisateur** aux rôles de l'application (via des fichiers de configuration comme **glassfish-web.xml** ou **jboss-web.xml**).

Exemple :

```
<!-- Fichier jboss-web.xml -->
<security-role-mapping>
  <role-name>ADMIN</role-name>
  <group-name>server-admins</group-name>
</security-role-mapping>
```

3. Autorisation :

L'application vérifie les rôles de l'utilisateur pour autoriser l'accès aux ressources.

Exemple :

```
@GET
@Path("/admin")
@RolesAllowed("ADMIN")
public String adminEndpoint() {
    return "Accès admin autorisé";
}
```

Exemple englobe les trois

Répertoire de sécurité :

- Utilisateur : "alice" → Groupes : "managers", "users".

Configuration serveur :

- "managers" → Rôle "ADMIN".
- "users" → Rôle "USER".

Application :

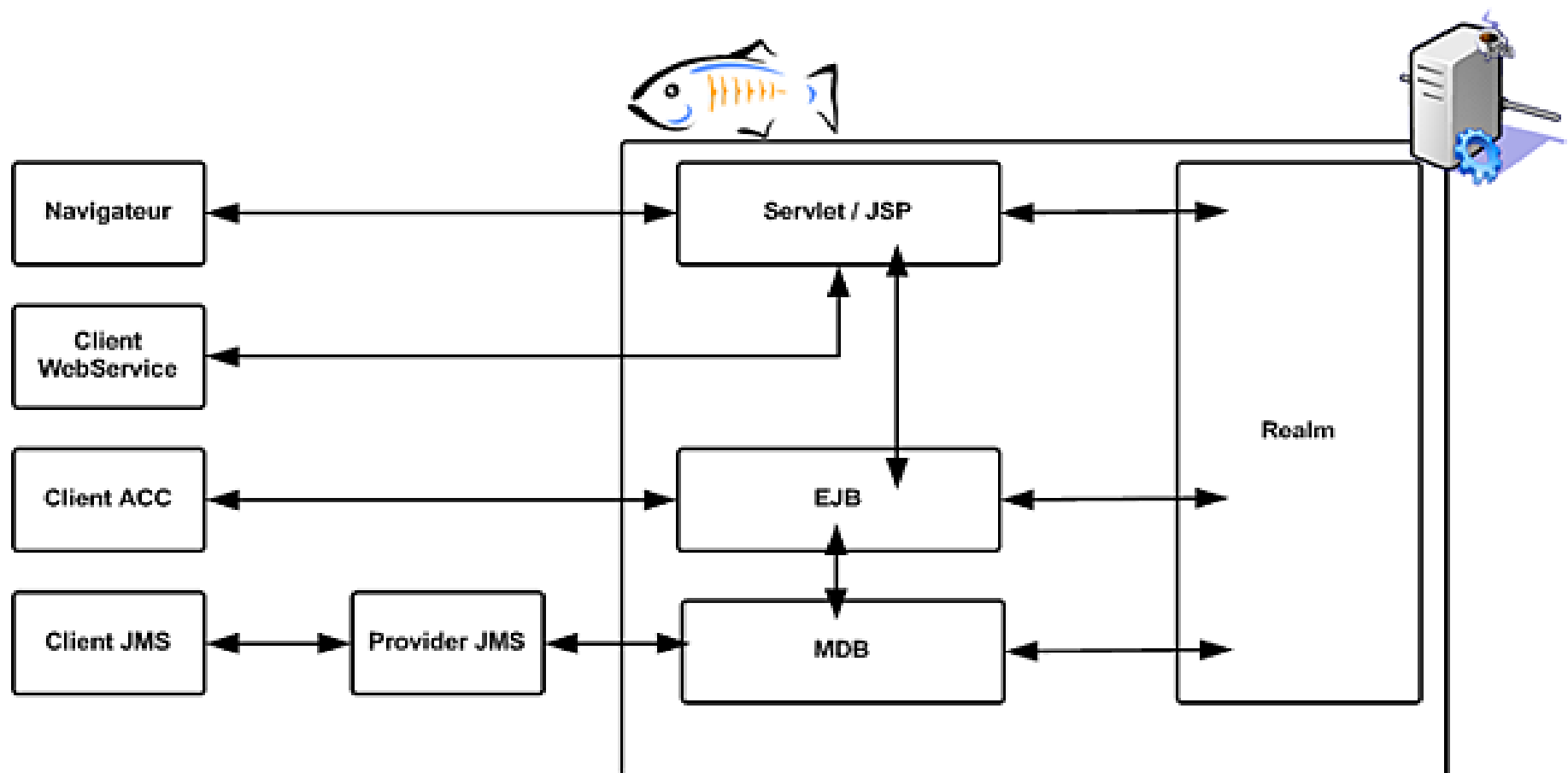
- @RolesAllowed("ADMIN") → Alice y aura accès via son groupe "managers".

Rôles d'un utilisateur

L'utilisateur déploie l'application qui associe selon ces rôles

La manière dont un rôle est associé à un utilisateur n'est pas standard et dépend du serveur d'applications

Un serveur comme GlassFish définissent des groupes d'utilisateurs et il est possible de les configurer pour qu'un groupe corresponde à un rôle.



Realm – domaine de sécurité

Un **realm** (ou un domaine en Français), est une **base de données des utilisateurs** habilités à utiliser une ou plusieurs applications déployées sur un même serveur web.

Le realm peut contenir les mots de passe de ses utilisateurs et leurs groupes d'appartenance.

Les **groupes** d'utilisateurs renseignés dans un realm font référence à une population d'utilisateurs présentant des caractéristiques communes au sein d'une organisation.

Chaque **utilisateur** d'un realm peut appartenir à un seul groupe, plusieurs groupes voire à aucun groupe.

- **Les Realm** font **partie d'un mécanisme de sécurité**, basé sur l'authentification pour **protéger l'accès** des ressources du serveur
- Le principe d'un Realm est de gérer une liste d'utilisateurs, avec des rôles.
- Un domaine de sécurité (*realm*) définit comment ces informations sont conservées, et comment on les utilise pour l'authentification
(voir balise <login-config> de web.xml) ;



Exemples : domaine défini par un fichier des mots de passe, par des tables relationnelles, par des certificats informatiques

User, Groupe, Rôle, ...

Un programme sécurisé doit être **indépendant des noms des utilisateurs** , selon les droits d'accès

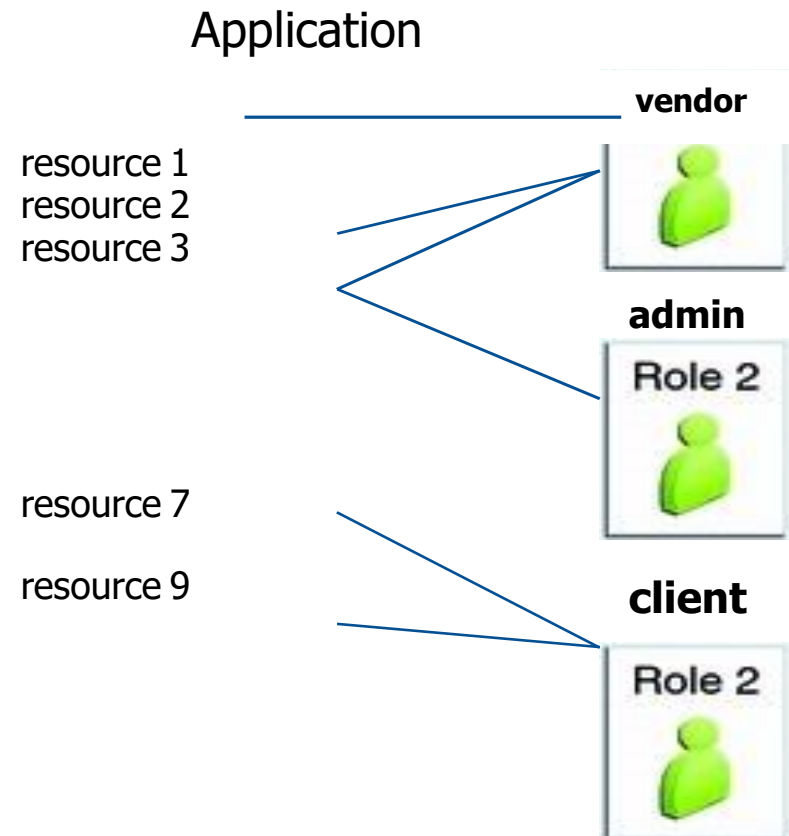
Un programme sécurisé doit être **dépendant des droits et autorisations**

- le programme vérifie des **rôles**

- if(isUserInRole("admin")) ok;

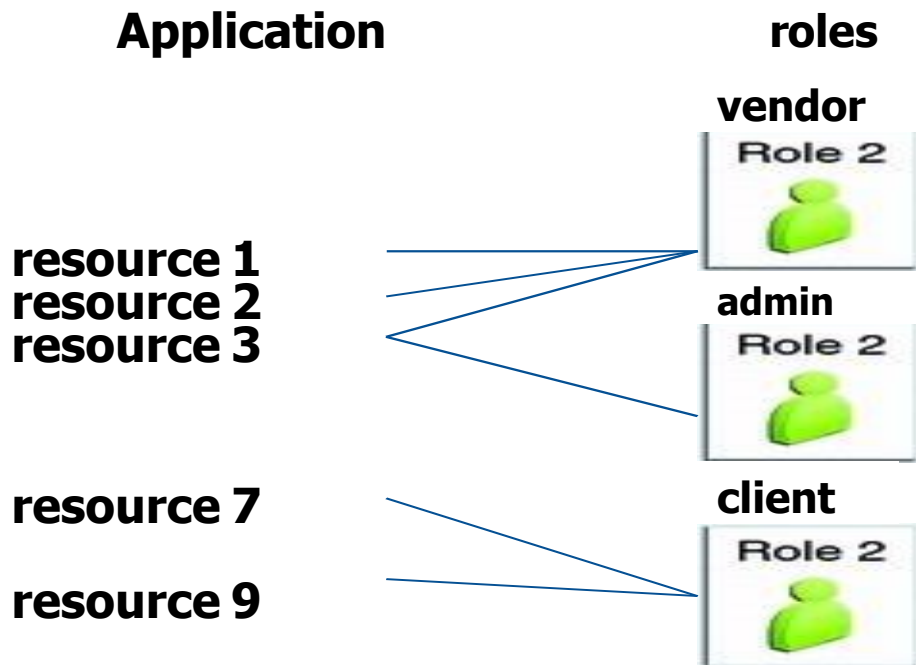
- // ok

- role** : rôle que peut prendre un utilisateur
 - ex: admin, vendor, client, visitor, secretary, ...
 - correspond souvent aux acteurs uml du système

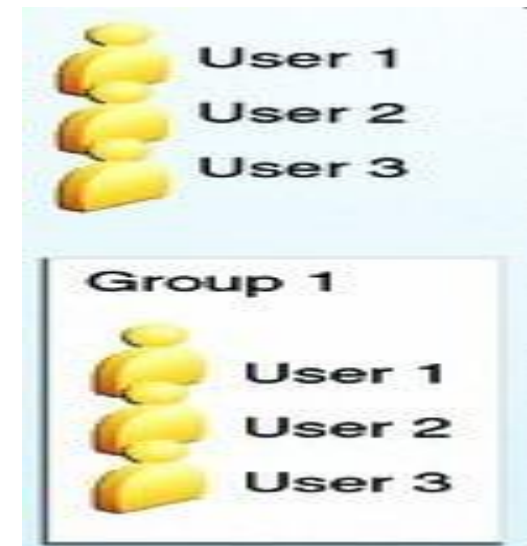


User, Groupe , Role

- Les utilisateurs sont enregistré en dehors de l'application
 - dans des fichiers
 - ou dans un base,
 - ...
- Les utilisateurs peuvent être regroupés en groupe
 - tout les utilisateur d'un même groupe partagent les même droits



Users and Groups



JAAS (Java Authentication and Authorization Service) est une API standard de java permettant de gérer des identifications et les droits associés (par rôles) au niveau du client et du serveur d'application.

JAAS est l'API standard utilisée par les serveurs d'application J2EE.

JAAS permet de séparer la gestion des droits d'accès aux composants J2EE du code métier.

JAAS valide l'identité du client, puis gère les autorisations, selon les rôles et les autorisations accordées par rôle.

- **Implémentations différentes** selon les fournisseurs
 - Glassfish
 - Tomcat
 - Jboss
 -

API Standard: JAAS

- JAAS est intégré à Java depuis Java 1.4
- Implémentations différentes selon les fournisseurs
 - Glassfish
 - Tomcat
 - Jboss
 -

JAAS

- **JAAS** peut être utilisé pour **deux choses** :
 - Pour **l'authentification** des utilisateurs,
Permet de déterminer, correctement, et en sûreté **qui est en train** d'exécuter du code Java,
Indépendamment du fait que le code fonctionne en tant que application, applet, bean ou servlet;
 - Pour **l'autorisation** des utilisateurs
Afin d'**assurer qu'ils ont les droits d'accès (permissions)** pour effectuer les actions accomplis.

Authentication Container Web JEE

La structure d'authentification HTTP est utilisée par plusieurs schémas d'authentification.

4 méthodes prédéfinies de communication avec l'utilisateur

basic, form-based, https, digest

se déclare dans `web.xml`:

```
<web-app>
```

```
...
```

```
<!-- Authentication -->
```

```
<login-config>
```

```
<auth-method>BASIC</auth-method>
```

```
</login-config>
```

```
</web-app>
```

Authentication basic

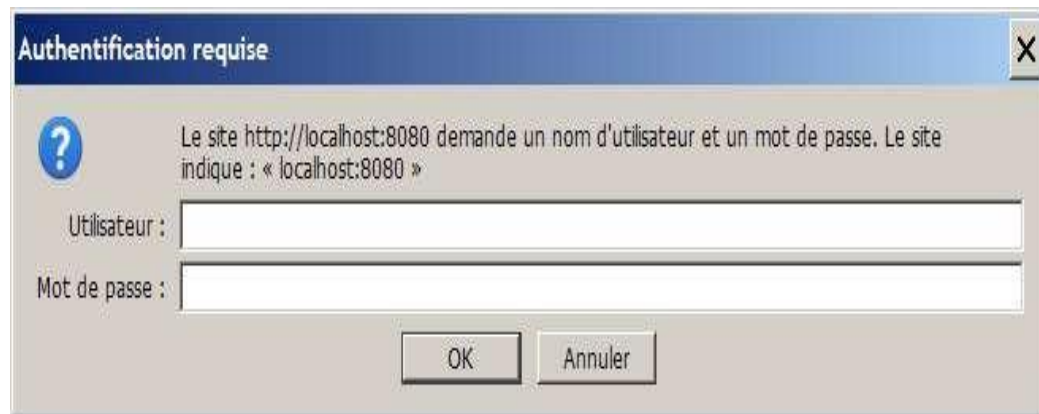
Par défaut, le mode d'authentification est le mode **BASIC** : le login et le mot de passe sont demandés à l'utilisateur par un formulaire affiché par le navigateur (ils transitent en clair sur le réseau)

La valeur par défaut pour **<realm-name>** dépend du serveur d'application

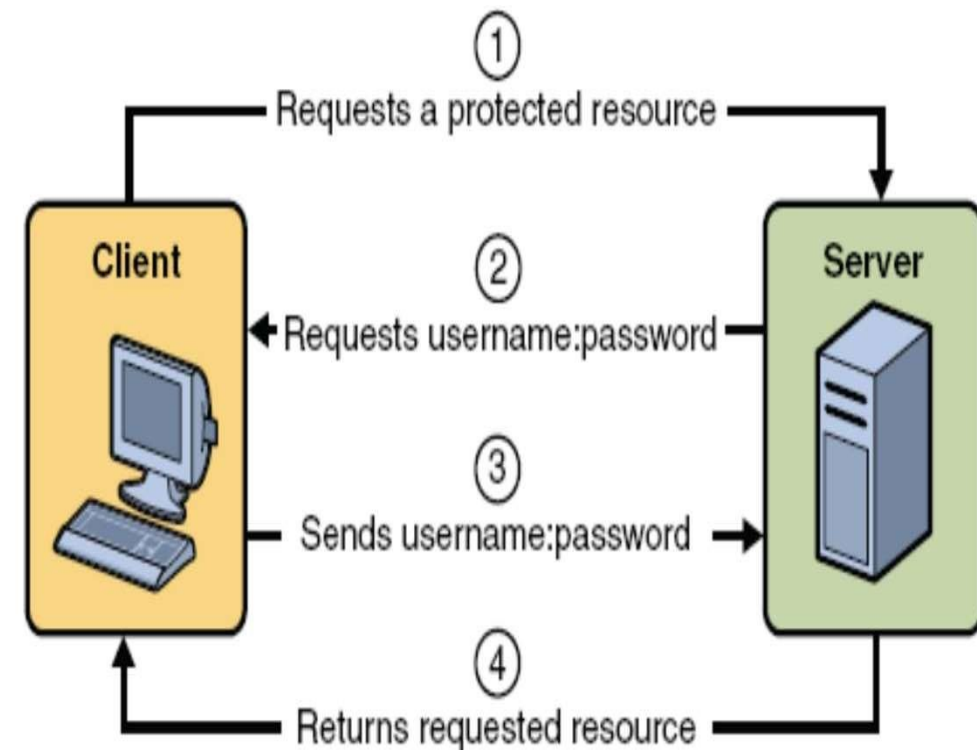
Ce type risque de poser des problèmes car le navigateur garde souvent des informations sur l'utilisateur qui vient de se connecter ; préférer le mode **FORM**

Authentication basic

- Le client web affiche une fenêtre demandant le password



```
<login-config>  
  <auth-method>BASIC</auth-method>  
</login-config>
```



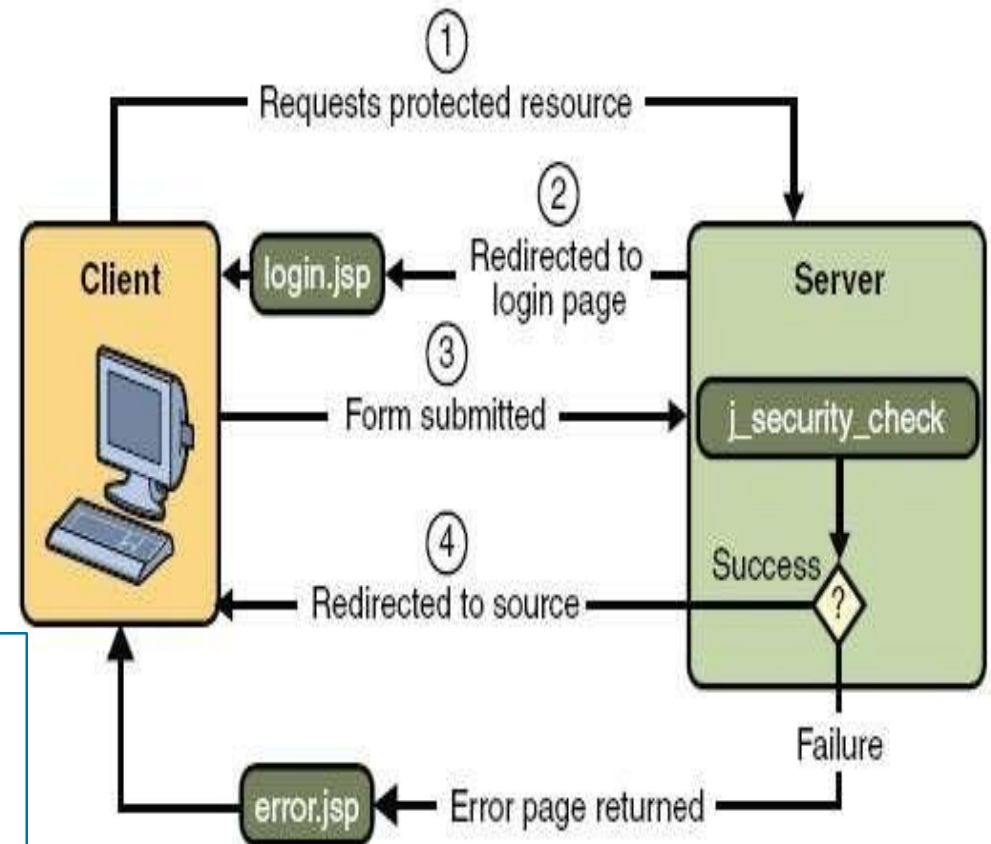
Authentification avec Form

FORM : utilisation d'un formulaire écrit par le développeur dans une page Web pour obtenir le login et le mot de passe

- L'application doit fournir le formulaire et la page d'erreur
- les urls dans la config doivent être absolues

The screenshot shows a Mozilla Firefox browser window titled 'Logon - Mozilla Firefox'. The address bar shows 'http://localhost:8080/s'. The page contains a login form with two input fields: 'Please Enter Your User Name:' and 'Please Enter Your Password:'. Below the fields are 'Submit' and 'Reset' buttons. The status bar at the bottom says 'Terminé'.

```
<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>
```



Authentication avec Form

- Exemple de formulaire
 - l'action est toujours `j_security_check`
 - il doit y avoir deux champs :
 - `j_username`
 - `j_password`

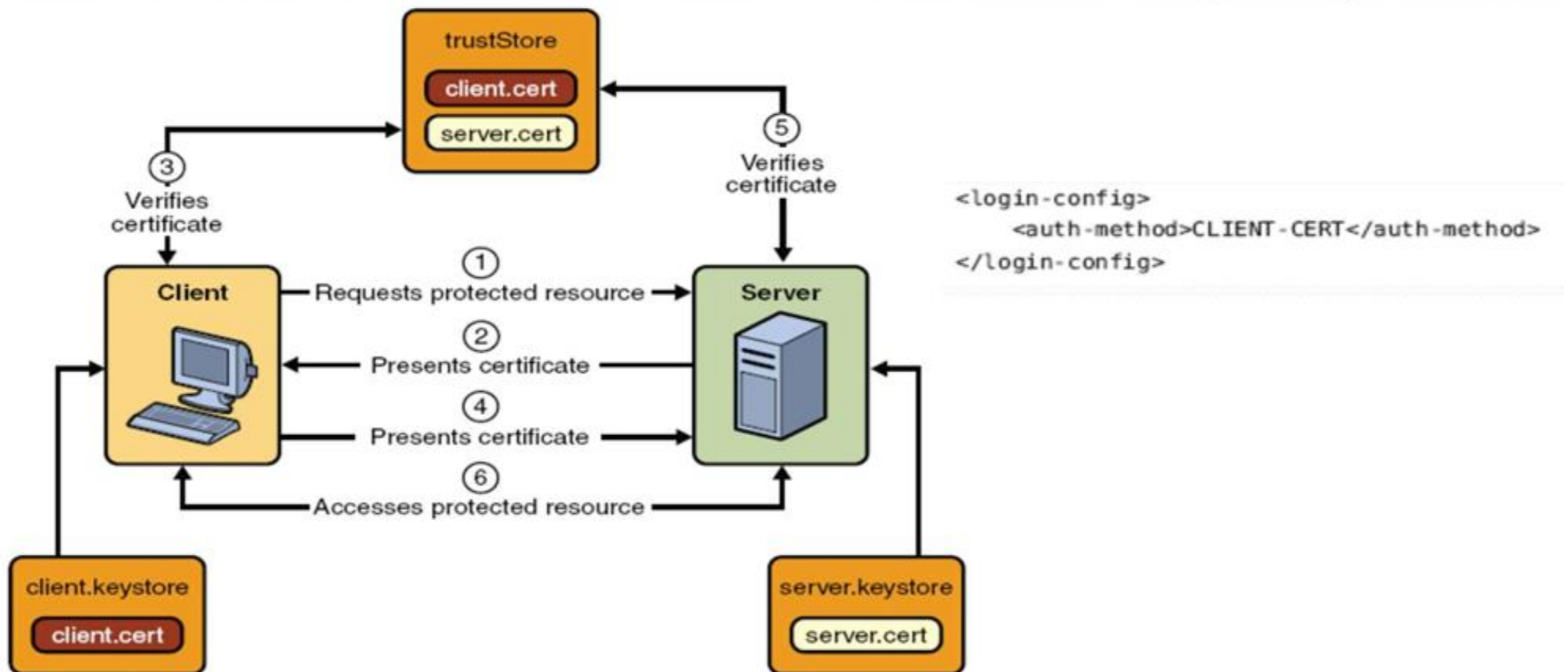
```
<form method="POST" action="j_security_check">  
<input type="text" name="j_username">  
<input type="password" name="j_password">  
</form>
```

Authentication client https certificate based

HTTPS: Combinaison du **HTTP** avec une **couche de chiffrement** comme **SSL**

Un certificat SSL est un **fichier de données** qui lie une **clé cryptographique**

SSL (Secure Socket Layer) est une **technologie** qui permet **d'établir un lien sécurisé** entre un **navigateur** et un **serveur Web** en **chiffrant les données échangées** entre ces derniers.

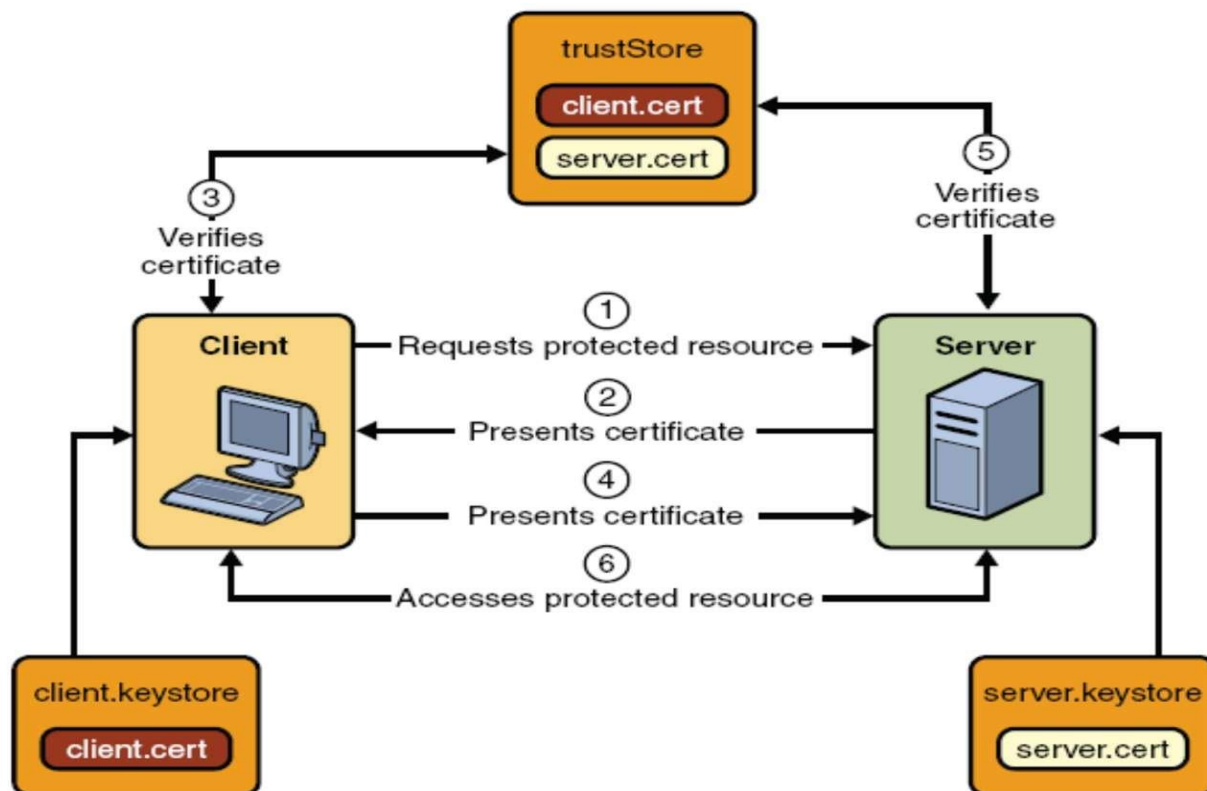


Authentication client https certificate based

Dans un schéma PKC (Public-Key Cryptography), Nous avons **deux clés**:

- La **clé publique** est utilisée par un expéditeur pour **chiffrer** des informations, elle peut être partagée en toute sécurité
- La **clé privée** est utilisée par un destinataire pour les **déchiffrer**.

Chaque **paire de clés asymétriques** est **unique**, garantissant qu'un message crypté à l'aide d'une clé publique ne peut être lu que par la personne qui possède la clé privée correspondante.



- Le client doit avoir une clé public (PKC)
- le serveur doit supporter SSL

```
<login-config>  
  <auth-method>CLIENT-CERT</auth-method>  
</login-config>
```

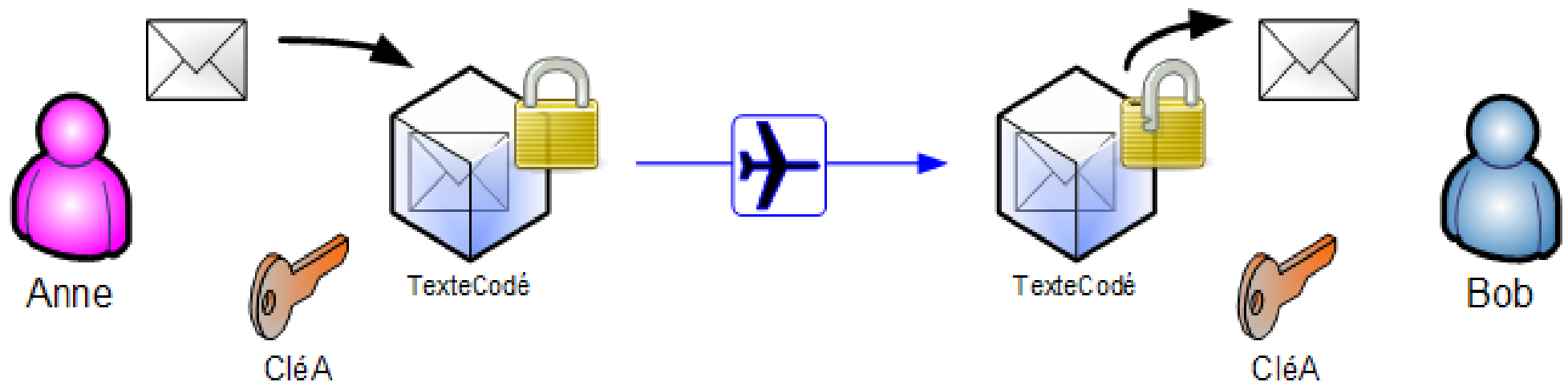
Chiffrement Symétrique

si **Anne** veut **envoyer** un **message chiffré** à **Bob**

elle doit lui communiquer un **mot de passe (clé de chiffrement)**.

Comme l'algorithme de **chiffrement est symétrique**, on a la relation suivante :

TexteCodé = chiffrement du message par la clé



Chiffrement Asymétrique

Un message chiffré par une clé **privée** sera **lisible** par **tous ceux** qui possèdent la clé **publique** correspondante.

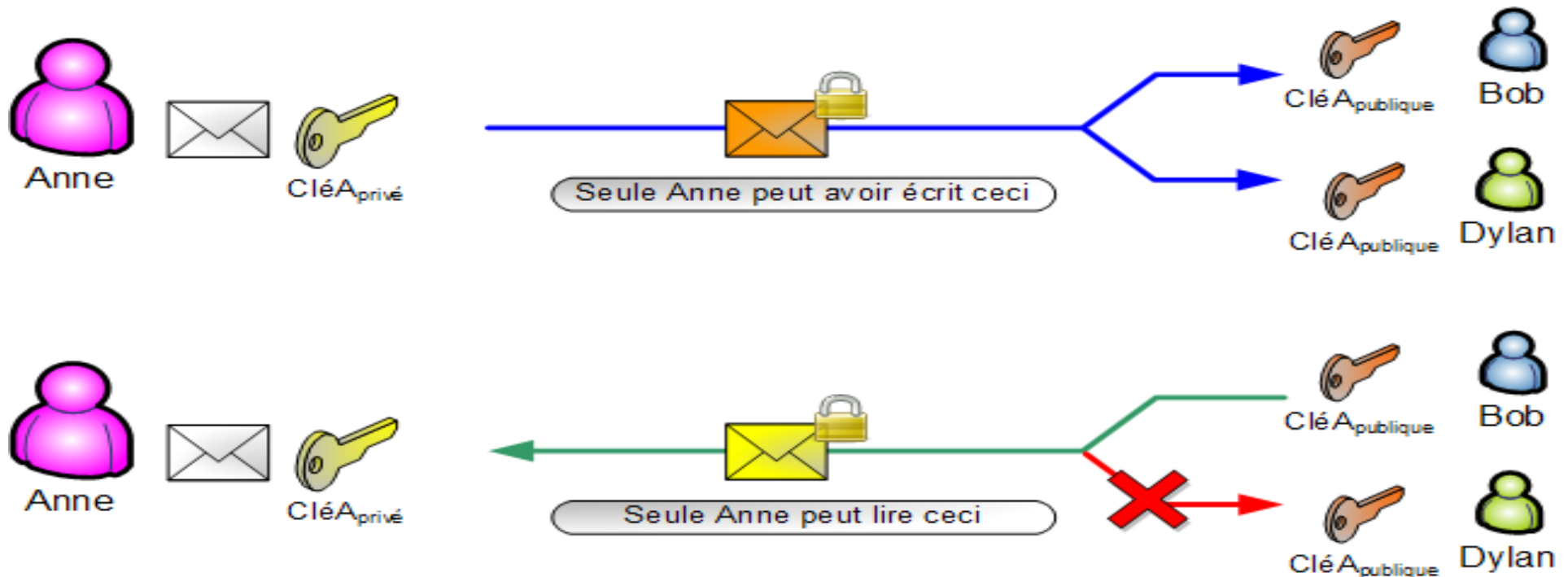
À l'inverse,

Un message chiffré par une clé **publique** n'est **lisible** que par le **propriétaire de la clé privée** correspondante.

Ainsi avec sa clé privée, Anne :

signe ses messages³ ;

lit (déchiffre) les messages qui lui sont adressés.



Déclaration des pages protégées

Dans l'application :

- Déclarer les rôles et les zones protégées
 - Au niveau de l'application :
 - Déclarer les rôles utilisés
 - Ex: admin, client, vendor
 - Déclarer les zones protégées
 - Spécifie quelles pages ou répertoires sont accessibles par quels rôles

Zone protégée	Accessible par
vendor/*	vendor, admin
client/clientPage.jsp	client

JEE - web.xml

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>not use</web-resource-name>
    <url-pattern>/vendor/*</url-pattern>
  </web-resource-collection>

  <auth-constraint>
    <role-name>vendor</role-name>
  </auth-constraint>
  <!-- do not encrypt. others: NONE INTEGRAL CONFIDENTIAL -->
  <user-data-constraint>
    <transport-guarantee>NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>

<!-- Authentication -->
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>myapp-realm</realm-name>
</login-config>

<security-role>
  <role-name>vendor</role-name>
</security-role>
```

La ressource
protégée

Le rôle autorisé

Encryption des
données

Methode
d'authentification

[option] le realm
à utiliser (ie le
fichier de users)

Rôles utilisés par
l'application

Realms

Un **realm** (ou un domaine en Français):
est une base de données des utilisateurs habilités à utiliser une ou plusieurs applications déployées sur un même serveur web.

Le realm peut contenir les **mots de passe** de **ses utilisateurs** et **leurs groupes** d'appartenance.

Les **groupes** d'utilisateurs renseignés dans un realm font référence à une population d'utilisateurs présentant des caractéristiques communes au sein d'une organisation.

Chaque **utilisateur** d'un realm peut appartenir à un seul groupe, plusieurs groupes voire à aucun groupe.

Le terme Principal est également utilisé pour désigner l'utilisateur dans un realm.

Dans **Glassfish**, un **realm** peut être stocké dans un fichier plat (**classe FileRealm**), dans une **base de données relationnelle** de type **MySQL** (**JDBCRealm**) ou dans une **base de données de certificats** (**CertificateRealm**).

La **configuration d'un realm** s'effectue dans le serveur d'application **GlassFish** par l'intermédiaire de sa **Console d'Administration**,
en ligne de commande via l'**outil asadmin** ou
par la mise à jour directe du fichier de configuration **domain.xml** de GlassFish.

The screenshot shows the GlassFish Administration Console interface. On the left is a tree view of the configuration hierarchy, with 'Domaines' (Domains) selected. The main panel is titled 'Nouveau domaine' (New Domain) and contains the following fields:

- Nom de configuration :** server-config
- Nom :** (empty text field)
- Nom de classe :** ☒ com.sun.enterprise.security.ee.auth.realm.jdbc.JDBCRealm (selected) and ☐ (empty) with a dropdown arrow.
- Propriétés propres à cette classe** (Properties specific to this class):
 - Contexte de JAAS :** (empty text field) with the label 'Identificateur du module de connexion' (Connection module identifier) below it.
 - JNDI:** (empty text field) with the label 'Nom JNDI de la ressource JDBC utilisée' (JNDI name of the JDBC resource used) below it.

Déclaration ou programmation

Java EE permet d'implémenter une politique de sécurité de façon déclarative ou par programmation Java



Le plus simple est d'utiliser les déclarations (annotations Java ou fichiers de déploiement XML) pour indiquer comment les ressources sont protégées



Si la façon déclarative ne suffit pas, il est possible de programmer les cas les plus complexes en Java

Déclaration

Les annotations permettent d'écrire les informations sur la politique de sécurité directement dans le code Java des des servlets ou EJB



Sans utiliser une annotation ; les informations de sécurité sont décrites dans des balises des fichiers de déploiement



Fichiers de déploiement

1- Fichiers standards qui ne dépendent pas du serveur d'application :

- **web.xml** (module Web),
- **ejb-jar.xml** (module EJB), **application.xml**

(module application regroupe plusieurs modules dans un fichier EAR)



2- Chaque serveur d'application a ses propres fichiers de déploiement pour les informations non standardisées ;

Pour GlassFish,

glassfish-web.xml,

glassfish-ejb-jar.xml,

glassfish-application-web.xml

Types de déclarations



Java EE permet de déclarer

- les rôles utilisés par l'application
- la façon d'authentifier les utilisateurs
- les pages Web protégées et les rôles qui y ont accès
- les classes ou méthodes Java protégées et les rôles qui y ont accès

Déclaration des rôles

- ❑ Un rôle doit être déclaré avant d'être utilisé
- ❑ On peut le faire avec une annotation ou dans un fichier de déploiement XML

Annotation pour déclarer des rôles

❑ L'annotation **@DeclareRoles** peut être mise sur les servlets ou EJB

Exemple :

```
@Stateless
```

```
@DeclareRoles("admin", "membre", "visiteur")
```

```
public class MonBean { ... }
```

Balise pour déclarer des rôles

Plusieurs balises **<security-role>** peuvent être mises directement sous la balise racine
(**web-app**, **ejb-jar** ou **application**)

Exemple:

```
<security-role>  
<description>Administrateur</description>  
<role-name>admin</role-name>  
</security-role>
```

```
<security-role>  
<role-name>membre</role-name>  
</security-role>
```

Exemple

Un utilisateur authentifié est lié à un principal qui a un identifiant unique et qui peut être associé à plusieurs rôles.

Le principal de l'utilisateur Frank, par exemple, est lié aux rôles Employé et Admin.

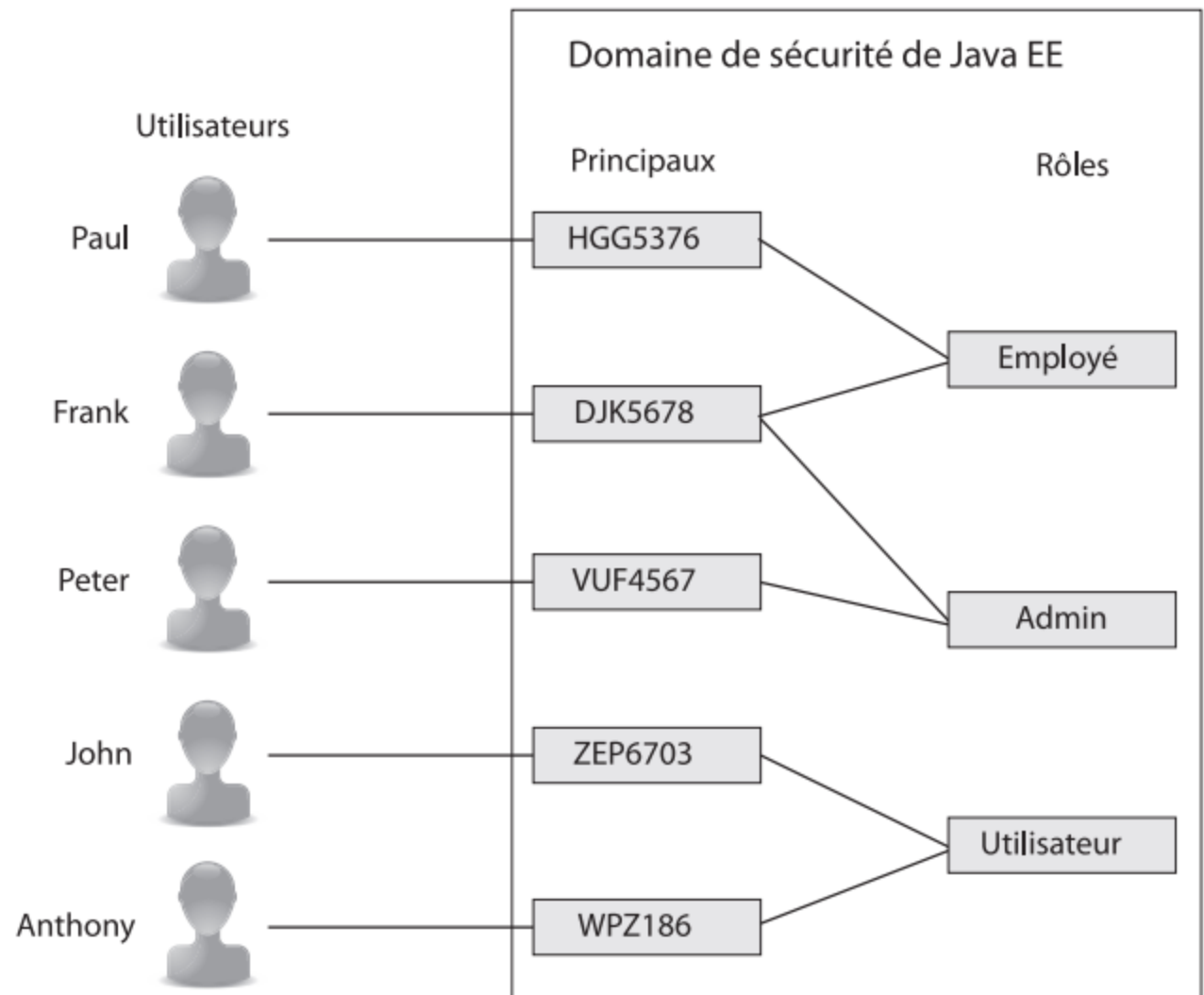


Fig. Principaux et rôles

Zones protégées SpringBoot

- Dans une classe de config
- `authorizeRequests()`

```
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
                .antMatchers("/", "/client").permitAll() // accès sans login
                .anyRequest().authenticated() // toutes les pages demandent authentication
                .and()
            .formLogin()
                .loginPage("/login")
                .permitAll()
                .and()
            .logout()
                .permitAll();
    }
}
```

Zones protégées SpringBoot

- Protection par rôle

```
@Configuration
@Order(SecurityProperties.BASIC_AUTH_ORDER - 10)
public class ApplicationConfigurerAdapter extends
WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws
    Exception {
        http.antMatcher("/foo/**")
            .authorizeRequests()
                .antMatchers("/foo/bar").hasRole("BAR")
                .antMatchers("/foo/spam").hasRole("SPAM")
                .anyRequest().isAuthenticated();
    }
}
```

Protection de classes ou de méthodes

- Permet de **restreindre l'accès** :
 - d'une **classe** – toutes les méthodes sont protégées
 - d'une **méthode**
- Peut se faire
 - par **@notation**
 - ou dans les **fichiers de configuration**
 - prédomine sur les @annotations

Annotations standard (JSR)

JEE et Spring

- 3 @nnotations :

- `@RolesAllowed("list-of-roles")`

The value of the `@RolesAllowed` annotation is a list of security role names to be mapped to the security roles that are permitted to execute the specified method or methods. Specifying this annotation on the bean class means that it applies to all applicable business methods of the class.

- `@PermitAll`

The `@PermitAll` annotation specifies that all security roles are permitted to execute the specified method or methods. Specifying this annotation on the bean class means that it applies to all applicable business methods of the class.

- `@DenyAll`

The `@DenyAll` annotation specifies that no security roles are permitted to execute the specified method or methods.

Déclaration des rôles

- Nécessaire dans JEE
- Implicite dans SpringBoot

JEE

- Déclarer les rôles et les zones protégées
 - Au niveau de l'application :
 - Déclarer les rôles utilisés
 - Ex: `admin`, `client`, `vendor`
 - Déclarer les zones protégées
 - Spécifie quelles pages ou répertoires sont accessibles par quels rôles

Zone protégée	Accessible par
<code>vendor/*</code>	<code>vendor, admin</code>
<code>client/clientPage.jsp</code>	<code>client</code>

-

JEE web.xml

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>not use</web-resource-name>
    <url-pattern>/vendor/*</url-pattern>
  </web-resource-collection>

  <auth-constraint>
    <role-name>vendor</role-name>
  </auth-constraint>
  <!-- do not encrypt. others: NONE INTEGRAL CONFIDENTIAL -->
  <
  <user-data-constraint>
    <transport-guarantee>NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>
<!-- Authentication -->
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>myapp-realm</realm-name>
</login-config>

<security-role>
  <role-name>vendor</role-name>
</security-role>
```

La ressource
protégée

Le rôle autorisé

Encryption des
données

Methode
d'authentification

[option] le realm
à utiliser (ie le
fichier de users)

Rôles utilisés par
l'application

Déclarer les utilisateurs et les groupes

- Au niveau du serveur
 - Déclarer les utilisateurs, les associer à des groupes
 - Ex:

Utilisateur	Groupe
dumoulin	admin
dejanvier	vendor, admin

- Peut se faire dans un fichier, une BD, un LDAP ...
- Configuration dépendante du serveur
- Généralement, il existe une configuration 'fichier' (pour les tests)

Exemple de fichiers

ipint_users.properties

```
#login=passwd  
cedric=cedricpass  
vendor1=vendor1pass  
client1=client1pass
```

ipint_groups.properties

```
#group=login1, login2  
admin=cedric  
vendor=cedric,vendor1  
client=cedric,client1
```


Mise en œuvre avec
glassfish

Configurer le domaine de sécurité (realm)

Au niveau du serveur, il faut configurer un domaine de sécurité (realm)

choisir file, db, ...

Permet de déclarer les groupes et users



Déclarer un fichier d'utilisateurs

- Choisir un domaine de type fichier
 - Soit réutiliser 'file'
 - Soit créer un nouveau domaine de ce type (FileRealm)



Domaines

Créez, modifiez ou supprimez des domaines de sécurité (d'authentification):

Nom de configuration : server-config

Domaines (3)

<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Nouveau...	Supprimer
	Nom		Nom de classe	
<input type="checkbox"/>	admin-realm		com.sun.enterprise.security.auth.realm.file.FileRealm	
<input type="checkbox"/>	certificate		com.sun.enterprise.security.auth.realm.certificate.CertificateRealm	
<input type="checkbox"/>	file		com.sun.enterprise.security.auth.realm.file.FileRealm	

Domaine de sécurité (Realm) par application

- Il est possible d'utiliser un domaine (realm) différent pour chaque application
 - Définir un realm
 - A l'aide de l'application d'admin
 - Configurations->server-config-> Security->Realms
 - Puis spécifier son utilisation dans web.xml :
 - `<realm-name>...</realm-name>`

```
<!-- Authentification -->  
<login-config>  
  <auth-method>BASIC</auth-method>  
  <realm-name>myapp-realm</realm-name>  
</login-config>
```

The screenshot shows the 'Edit Realm' interface in a web console. At the top, there are 'Save' and 'Cancel' buttons. Below them is a 'Manage Users' button. A note indicates that an asterisk (*) denotes a required field. The 'Configuration Name' is set to 'server-config'. The 'Realm Name' is 'myapp-realm' and the 'Class Name' is 'com.sun.enterprise.security.auth.realm.file.FileRealm'. Under the 'Properties specific to this Class' section, 'JAAS Context' is set to 'fileRealm' (with a description: 'Identifier for the login module to use for this realm'), 'Key File' is set to 'myapp-keyfile' (with a description: 'Full path and name of the file where the server will store all user, group, and password information for this realm'), and 'Assign Groups' is an empty text box (with a description: 'Comma-separated list of group names').

Edit Realm Save Cancel

Edit an existing security (authentication) realm. Manage Users

* Indicates required field

Configuration Name: server-config

Realm Name: myapp-realm

Class Name: com.sun.enterprise.security.auth.realm.file.FileRealm

Properties specific to this Class

JAAS Context: * fileRealm
Identifier for the login module to use for this realm

Key File: * myapp-keyfile
Full path and name of the file where the server will store all user, group, and password information for this realm

Assign Groups:
Comma-separated list of group names