L'utilisation de frameworks pour le développement avec Java EE

Module 8 – Tomcat

Objectifs

- Configurer Tomcat pour un accès via le protocole HTTPS
- Savoir sécuriser les applications web grâce à l'authentification
- Savoir gérer les utilisateurs et les rôles en fichier ou en base de données
- Savoir crypter les mots de passe

Tomcat

HTTPS

- SSL = Secure Socket Layer
  - Technologie qui permet à deux ordinateurs distants de communiquer à travers une connexion sécurisée
  - Chiffrement de la communication sur un réseau entre l'émetteur et le destinataire
  - Authentifie le serveur qui présente un certificat
  - Le serveur peut également demander l'authentification du client

Tomcat

HTTPS

- SSL = Secure Socket Layer
- Deux protocoles
  - SSL Handshake protocol
    - Les deux programmes SSL distants négocient les clés et les protocoles de chiffrement
  - SSL Record protocol
    - Les deux programmes SSL distants chiffrent les informations à échanger

Tomcat

HTTPS

- HTTPS : HTTP sous la couche SSL
- Protocole quasiment identique à l'exception :
  - Du nom (S en plus)
  - Du port par défaut : 443

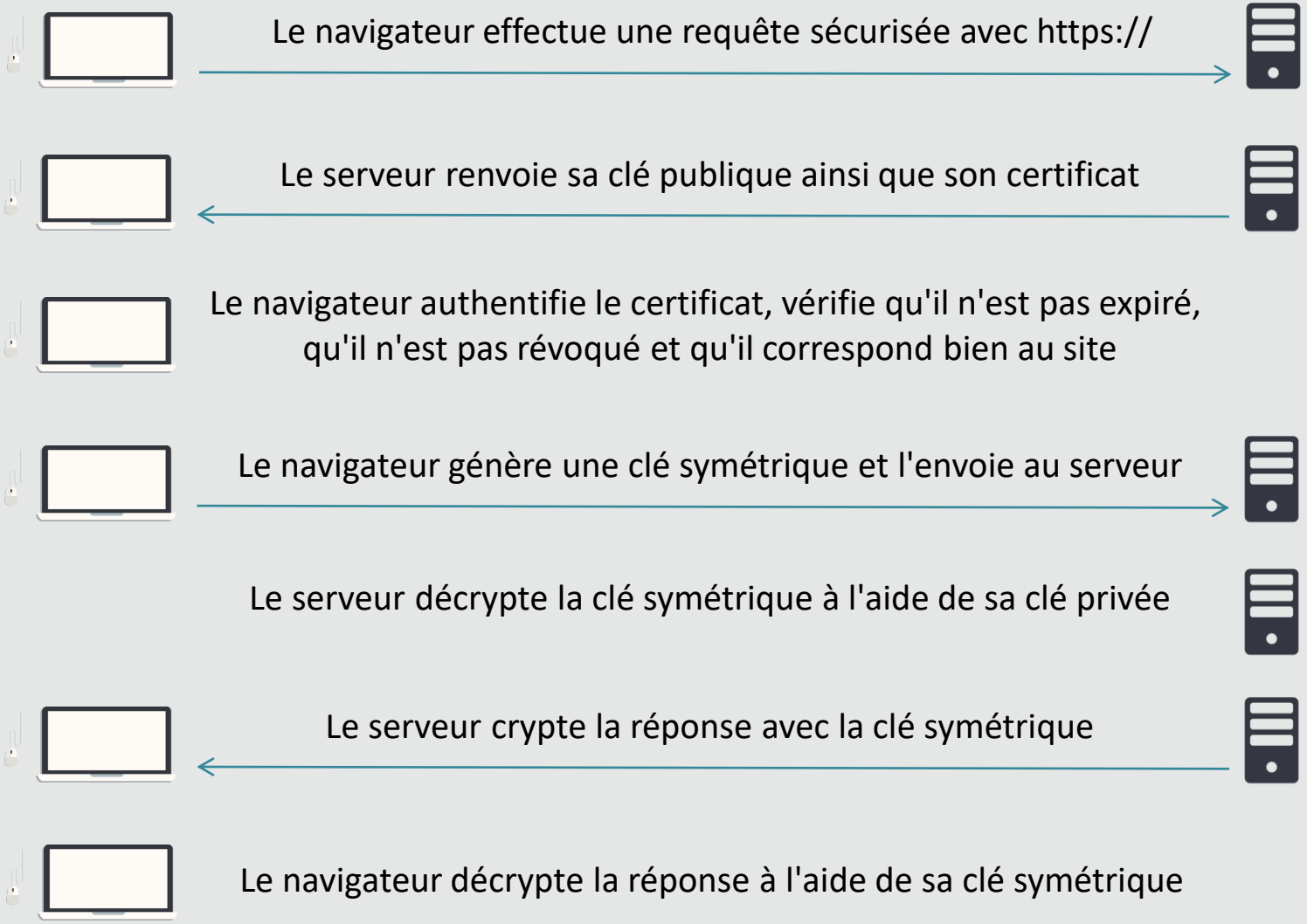
Tomcat

HTTPS

- TLS : Transport Layer Security
  - Couche similaire à SSL
  - S'appuie sur SSL 3.0
- TLS et SSL se servent d'un algorithme de chiffrement symétrique pour chiffrer le traffic

Tomcat

HTTPS



Tomcat

HTTPS

- Génération de la paire de clés asymétriques
  - Utilisation de keytool (fourni par le JDK)

```
keytool -genkey -keystore tomcatserver.jks -alias monsite -keyalg RSA
```

  - genkey : génère une paire de clés asymétriques
  - keystore : fichier qui stockera la paire de clés
  - alias : nom que possédera cette paire de clés dans le fichier (plusieurs paires possibles)
  - keyalg : algorithme de cryptage (RSA, DSA, DES…)

Tomcat

HTTPS

- Génération de la paire de clés asymétriques

```
$ keytool -genkey -keystore tomcatserver.jks -alias monsite -keyalg RSA

Entrez le mot de passe du fichier de clés :
Ressaisissez le nouveau mot de passe :
Quels sont vos nom et prénom ?
  [Unknown]:  www.monsite.fr
Quel est le nom de votre unité organisationnelle ?
  [Unknown]:  Service Informatique
Quel est le nom de votre entreprise ?
  [Unknown]:  MonSite SARL
Quel est le nom de votre ville de résidence ?
  [Unknown]:  Nantes
Quel est le nom de votre état ou province ?
  [Unknown]:  France
Quel est le code pays à deux lettres pour cette unité ?
  [Unknown]:  FR
Est-ce CN=www.monsite.fr, OU=Service Informatique, O=MonSite SARL, L=Nantes, ST=France, C=FR ?
  [non]:  oui

Entrez le mot de passe de la clé pour <monsite>
    (appuyez sur Entrée s'il s'agit du mot de passe du fichier de clés) :
Ressaisissez le nouveau mot de passe :

$
```

Tomcat

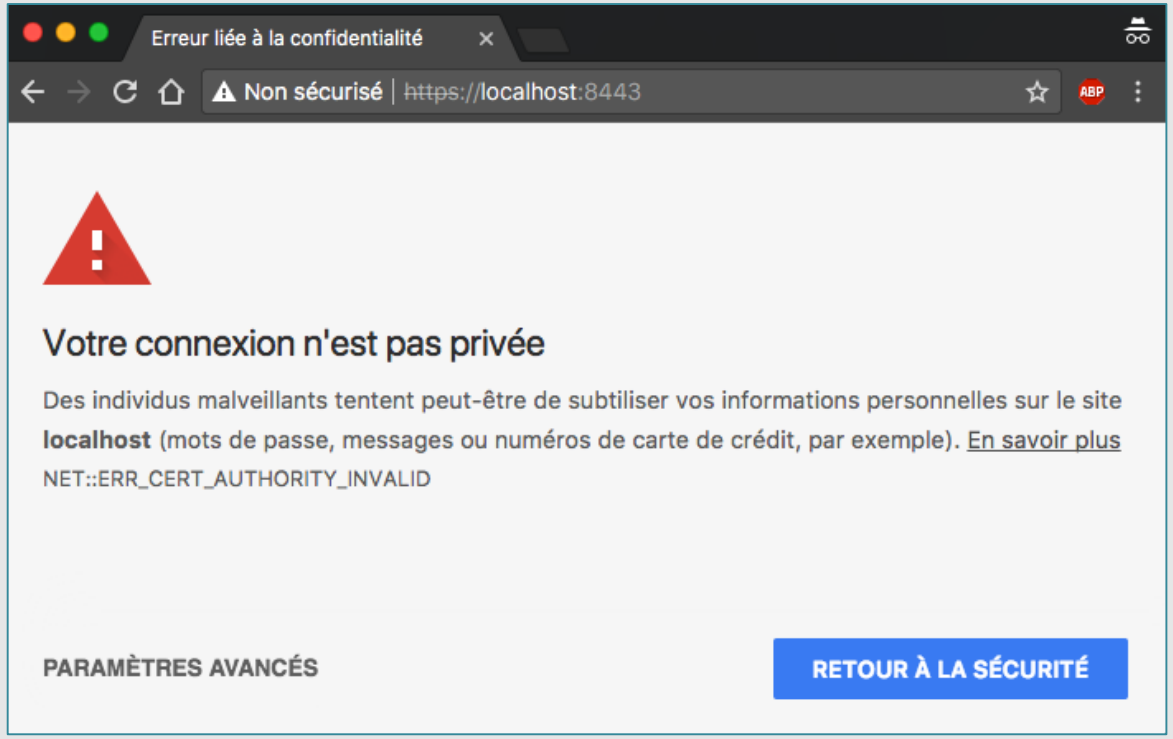
HTTPS

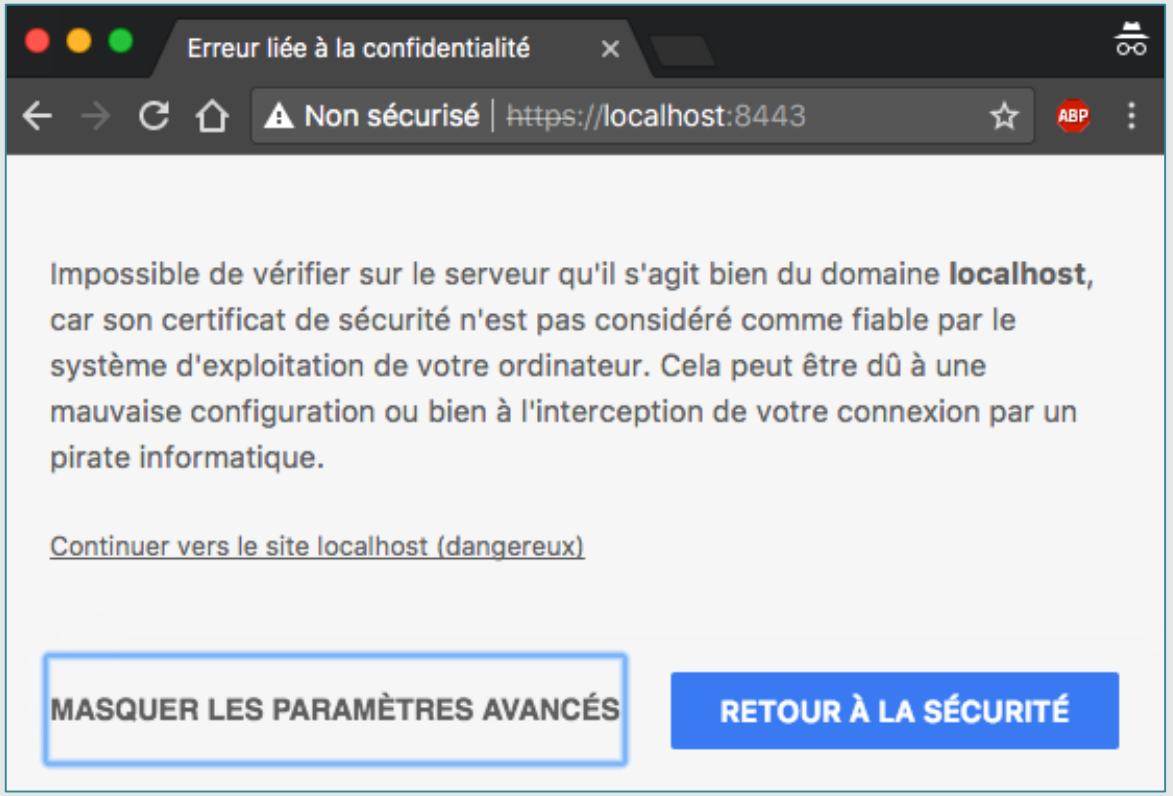
- Création du connecteur HTTPS
  - Dans CATALINA_HOME / conf / server.xml

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
           maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
           clientAuth="false" sslProtocol="TLS"
           keystoreFile="conf/tomcatserver.jks"
           keystorePass="tomcatserver"
           keyAlias="monsite"
           keyPass="monsitepass"
           />
```

Tomcat

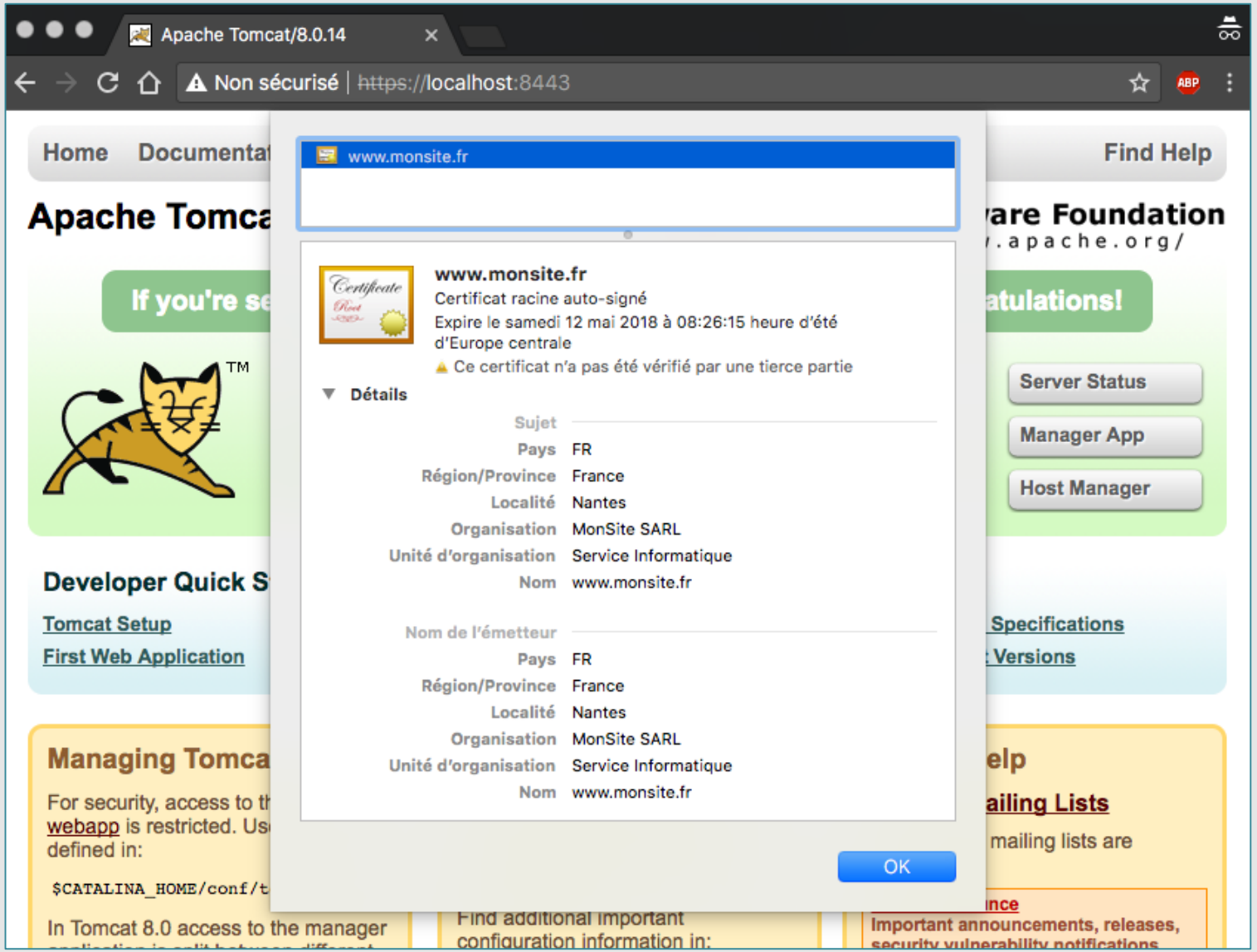
HTTPS





Tomcat

HTTPS



Tomcat

HTTPS

- Configurer une application afin de n'être accessible qu'avec HTTPS
  - Dans le fichier /WEB-INF/web.xml de l'application

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>monapplihttps</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Démonstration

Sécurisation des applications Java EE

- Utilisation de JAAS (Java Authentication and Authorization Service)
- JAAS est intégrée à Java SE et est l'API standard utilisée par les serveurs d'application Java EE
- Permet de séparer la gestion des droits d'accès aux composants Java EE du code métier

Tomcat

Sécurisation des applications Java EE

- Déclaration dans le fichier de configuration de l'application (web.xml) des parties à sécuriser

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>private</web-resource-name>
        <url-pattern>/private/*</url-pattern>
    </web-resource-collection>

    <auth-constraint>
        <role-name>administrateur</role-name>
    </auth-constraint>
</security-constraint>
```

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

```
<security-role>
    <description>Role utilise pour la partie private</description>
    <role-name>administrateur</role-name>
</security-role>
```

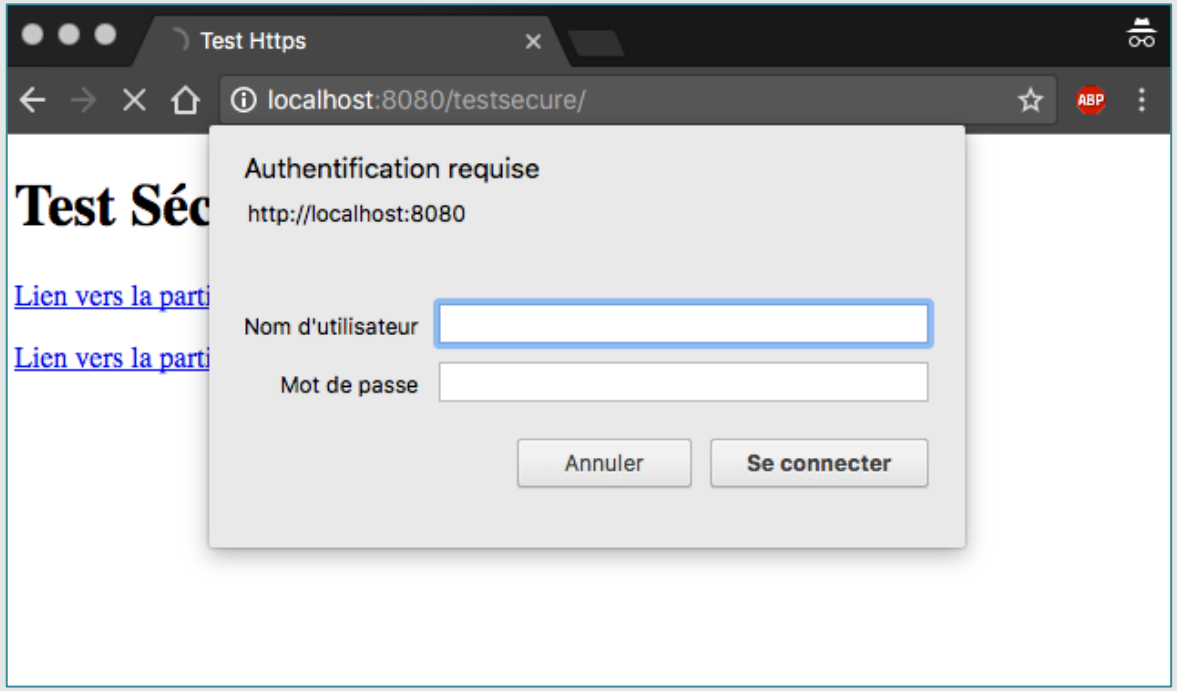
Tomcat

Sécurisation des applications Java EE

- Quatre mécanismes d'authentification
  - L'authentification de base (**BASIC**)
  - L'authentification codée (**DIGEST**)
  - L'authentification par certificat client (**CLIENT-CERT**)
  - L'authentification par formulaire (**FORM**)

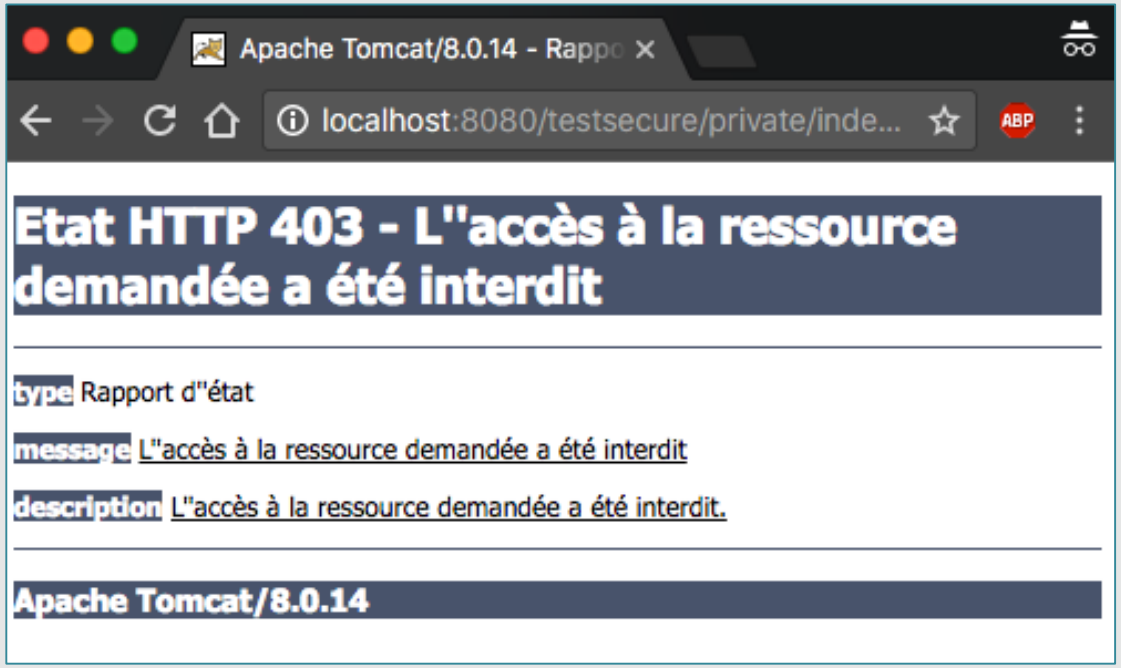
Sécurisation des applications Java EE

- L'authentification de base (**BASIC**)
  - La plus simple
  - Le navigateur demande le login et le mot de passe dans une boîte de dialogue (fenêtre popup)
  - Encodage avec l'algorithme Base64



Sécurisation des applications Java EE

- L'authentification de base (**BASIC**)
  - Authentification : la paire user/password existe-t-elle ?
    - Si non, la paire user/password est redemandée
  - Autorisation : si oui, a-t-elle les droits nécessaires
    - Si oui, accès à la page demandée
    - Si non, retour code 403 (accès interdit)



Sécurisation des applications Java EE

- L'authentification codée (**DIGEST**)
  - Identique à BASIC mais avec cryptage du mot de passe par hachage
  - Lors de la demande d'authentification, le serveur envoie une chaîne de caractères
  - Le navigateur ajoute cette chaîne au mot de passe, crypte le tout (SHA ou MD5) et envoie au serveur
  - Le serveur fait de même avec le mot de passe en clair stocké
  - L'authentification aboutit si les deux valeurs de hachage sont identiques

Tomcat

Sécurisation des applications Java EE

- L'authentification par certificat client (**CLIENT-CERT**)
  - Le client doit installer le certificat du serveur dans le navigateur
  - Les requêtes seront cryptées avec la clé publique du serveur
  - Données transmises en HTTPS

Tomcat

Sécurisation des applications Java EE

- L'authentification par formulaire (**FORM**)
  - Le navigateur n'intervient pas, c'est l'application qui fournit le moyen de s'authentifier
  - Utilisation d'un formulaire HTML

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

```
<form method="POST" action="j_security_check">
    <table>
        <tr>
            <td>Name:</td>
            <td><input type="text" name="j_username" /></td>
        </tr>
        <tr>
            <td>Password:</td>
            <td><input type="password" name="j_password" /></td>
        </tr>
        <tr>
            <td colspan="2"><input type="submit" value="Go" /></td>
        </tr>
    </table>
</form>
```

Les Realms

- Mécanisme de sécurité proposé par Tomcat basé sur l'authentification

- Cinq Realms sont proposés par Tomcat 8
  - UserDatabase Realm
  - JDBC Realm
  - DataSource Realm
  - JNDI Realm
  - JAAS Realm

Les Realms

- UserDatabase Realm
  - Les informations d'authentification sont stockées dans un fichier XML
  - Realm par défaut de Tomcat
  - Le fichier XML par défaut est CATALINA_HOME / conf / tomcat-users.xml

```
<tomcat-users>
    <role rolename="administrateur" />
    <role rolename="client" />
    <user username="user1" password="pass1" roles="administrateur" />
    <user username="user2" password="pass2" roles="client" />
    <user username="user3" password="pass3" roles="administrateur,client" />
</tomcat-users>
```

Les Realms

- JDBC Realm
  - Les informations d'authentification sont stockées en base de données
  - La structure des tables est la suivante

```
mysql> desc TomcatUsers;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| user_name | varchar(15) | NO   | PRI | NULL    |       |
| user_pass | varchar(15) | NO   |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

```
mysql> desc TomcatUsersRoles;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| user_name | varchar(15) | NO   | PRI | NULL    |       |
| role_name | varchar(15) | NO   | PRI | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

Les Realms

- JDBC Realm
  - Les informations d'authentification sont stockées en base de données
  - Le Realm doit contenir l'ensemble des paramètres de connexion

```
<Context docBase="/Chemin Vers Le War/application.war" >
    <Realm className="org.apache.catalina.realm.JDBCRealm"
       driverName="com.mysql.jdbc.Driver"
       connectionURL="jdbc:mysql://localhost:3306/javaavance"
       connectionName="java"
       connectionPassword="avance"
       userTable="TomcatUsers"
       userNameCol="user_name"
       userCredCol="user_pass"
       userRoleTable="TomcatUsersRoles"
       roleNameCol="role_name"/>
</Context>
```

Tomcat

Les Realms

- DataSource Realm
  - Les informations d'authentification sont stockées en base de données
  - Une DataSource est préalablement définie
  - Le Realm doit contenir la description des tables

```
<Resource   name="jdbc/poolConnexionGlobal"
      type="javax.sql.DataSource"
      driverClassName="com.mysql.jdbc.Driver"
      url="jdbc:mysql://localhost:3306/javaavance"
      username="java"
      password="avance"
      initialSize="4"
      maxActive="20"
      minIdle="3"
      maxIdle="10"
      removeAbandoned="true"
      removeAbandonedTimeout="60" />
```

```
<Context docBase="application" path="/application">

  <Realm  className="org.apache.catalina.realm.DataSourceRealm"
       dataSourceName="jdbc/poolConnexionGlobal"
       userTable="TomcatUsers"
       userNameCol="user_name"
       userCredCol="user_pass"
       userRoleTable="TomcatUsersRoles"
       roleNameCol="role_name"/>

</Context>
```

Les Realms

- JNDI Realm
  - Permet l'utilisation d'un service d'annuaire de type LDAP
  - Par exemple Microsoft Active Directory, OpenLDAP
  - Nécessité d'un paramétrage spécifique du Realm

```
<Realm    className="org.apache.catalina.realm.JNDIRealm"
      connectionURL="ldap://localhost:389"
        userPattern="uid={0},ou=people,dc=mycompany,dc=com"
           roleBase="ou=groups,dc=mycompany,dc=com"
           roleName="cn"
         roleSearch="(uniqueMember={0})"
/>
```

Tomcat

Les Realms

- JAAS Realm
  - Il est possible de créer son propre Realm
  - Par exemple, liste des utilisateurs et rôles dans un fichier Excel, dans un fichier à plat au format JSON…
  - Créer une classe qui implémente javax.security.auth.spi.LoginModule
  - Créer un fichier jaas.config contenant la description du Realm

Cryptage des mots de passe

- Par défaut, les mots de passe sont stockés en clair
- Tomcat fournit l'utilitaire digest qui permet de crypter les mots de passe
- Algorithmes de cryptages supportés
  - SHA
  - MD2
  - MD5

```
$ ./digest.sh -a sha password
password:5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8

$ ./digest.sh -a md2 password
password:f03881a88c6e39135f0ecc60efd609b9

$ ./digest.sh -a md5 password
password:5f4dcc3b5aa765d61d8327deb882cf99
```

Tomcat

Cryptage des mots de passe

- Modification des mots de passe dans le fichier XML ou en base de données

```
<Resource name="MyResource" auth="Container"
  type="org.apache.catalina.UserDatabase"
  factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
  pathname="conf/tomcat-users-cryptes.xml" />
```

```
<Context docBase="application" path="/application">

   <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
          resourceName="MyResource" digest="MD5"/>

</Context>
```

Tomcat

Sécurisation d'une application Web

Démonstration