Administration Serveur J2EE

P.Mathieu

LP DA2I Lille
http://www.iut-a.univ-lille.fr
prenom.nom@univ-lille.fr

20 janvier 2020

TOMCAT

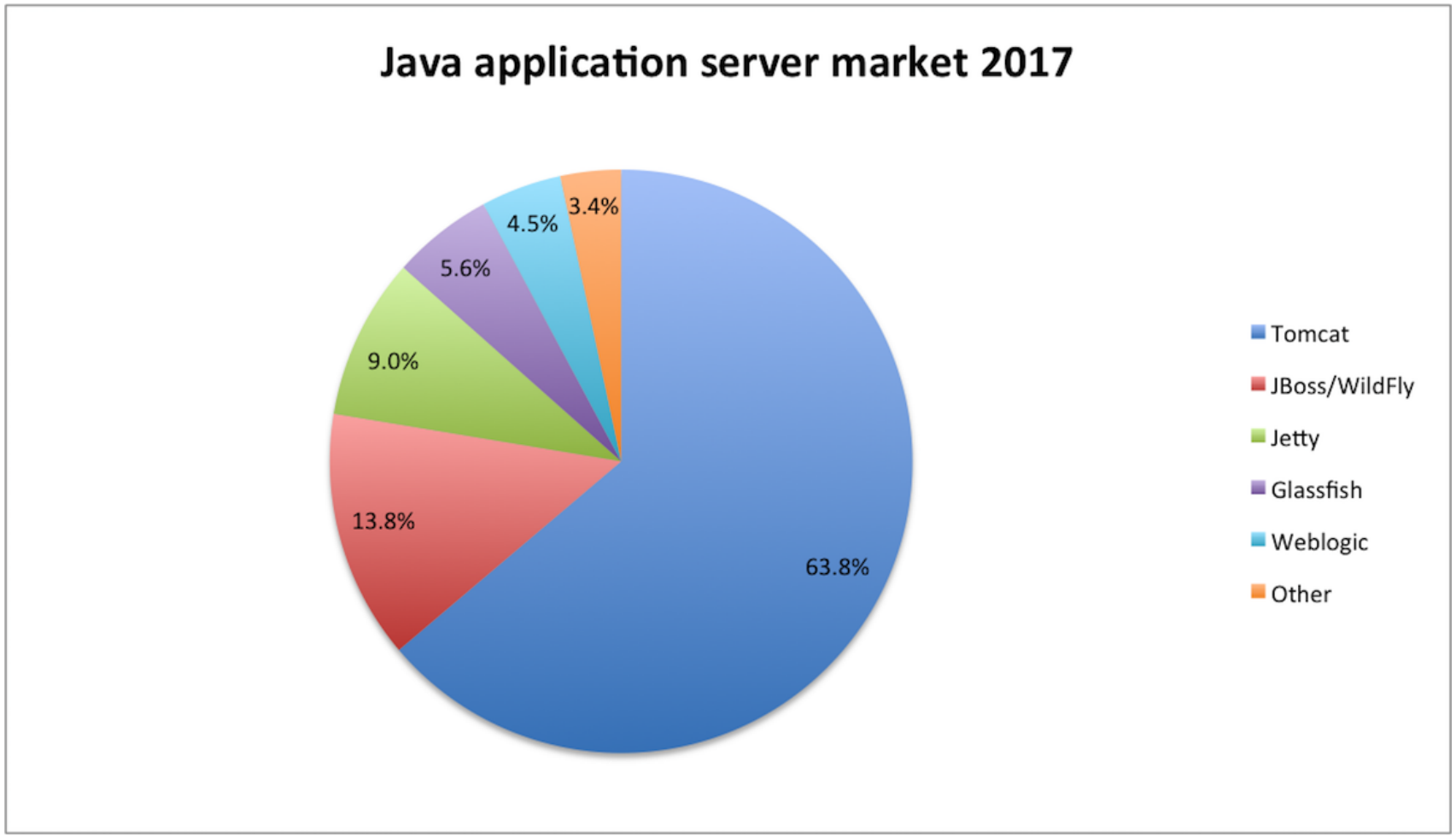
Il existe de nombreux serveurs compatibles ...

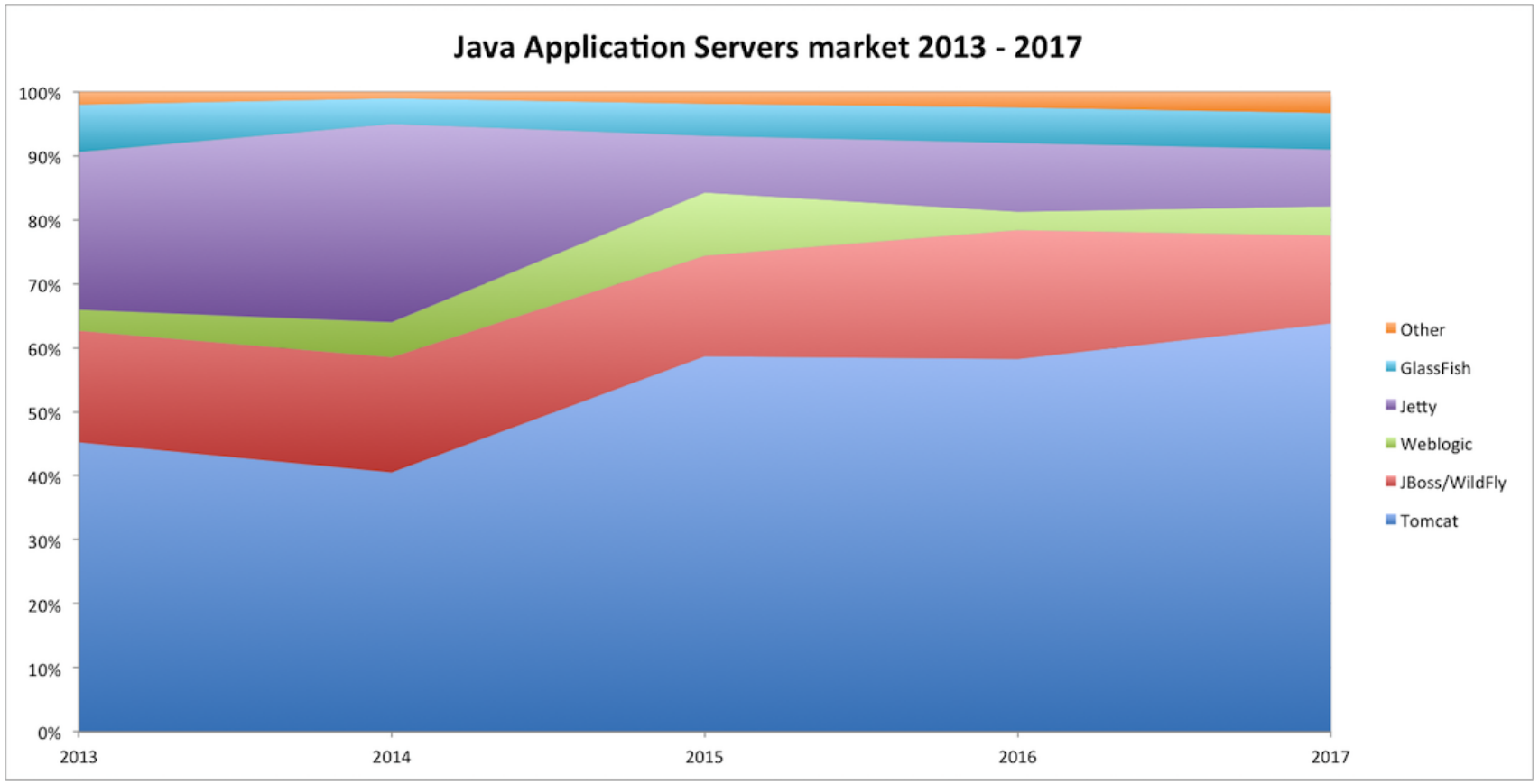
- Tomcat (Apache)
- Glassfish (Oracle)
- JBoss
- Resin (Caucho)
- WebLogic (BEA)
- WebSphere (IBM)
- Oracle Application Server
- Enhydra (Lutris)
- NewAtlanta
- ...

TOMCAT

- Dernière version Tomcat9, 2016 (Tomcat8 2014, Tomcat7 en 2010, Tomcat6 en 2006)
- Tomcat 9 : Java 1.8, compatible servlet4.0, jsp2.3 et EL 3.0
- Tomcat 8 : Java 1.7, compatible servlet3.1, jsp2.3 et EL 3.0
- Tomcat 7 : Java 1.6, compatible servlet3.0, JSP2.2 et EL 2.2
- Tomcat 6 : Java 1.5, compatible servlet2.4, JSP2.0
- Open-source entièrement écrit en Java
  (donc exécutable sur tout, y compris MacOs, Windows et Linux)
- Implémentation de référence offerte par le groupe Apache
- Fonctionnement :
  - en Stand-alone
  - en module Apache

Université de Lille




Java Application Servers market 2013 - 2017
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
2013
2014
2015
2016
2017
Other
GlassFish
Jetty
Weblogic
JBoss/WildFly
Tomcat

Université de Lille

Scrute par défaut sur le port 8080
Accès initial par `http://localhost:8080`

- `JAVA_HOME` mal configuré
- Un autre serveur utilise le port 8080
- Pb réseau, essayer un `ping localhost`
- Browser qui passe par un proxy
- Modif erronée des fichiers de configuration (voir `logs/catalina.out`)

Fonctionnement

Arborescence

Université de Lille

/webapps	les différentes applications web
/bin	scripts de démarrage (`*.sh` unix et `*.bat` windows)
/lib	jars à utiliser dans toutes les webapps (notamment servlet-api.jar ou jsp-api.jar)
/classes	même chose avec les classes
/conf	fichiers de configuration (notamment `server.xml`)
/conf/Catalina/localhost	fichiers additionnels à `server.xml`
/logs	logs d'accès (notamment `catalina.out`)
/work	zone de travail pour la compilation des JSP

Fonctionnement

Principe de base

Trois fichiers clés :

catalina.out Messages d'information et d'erreurs du serveur.

server.xml : Configuration générale du serveur.

- Définition des connecteurs et ports par défaut
- Définitions propres à tous les contextes
- Description de la gestion des contextes d'applications

web.xml : Fichier propre à chaque application web.

- Durée de vie de la session
- Welcome file
- Droits d'acces

Fonctionnement

conf/server.xml

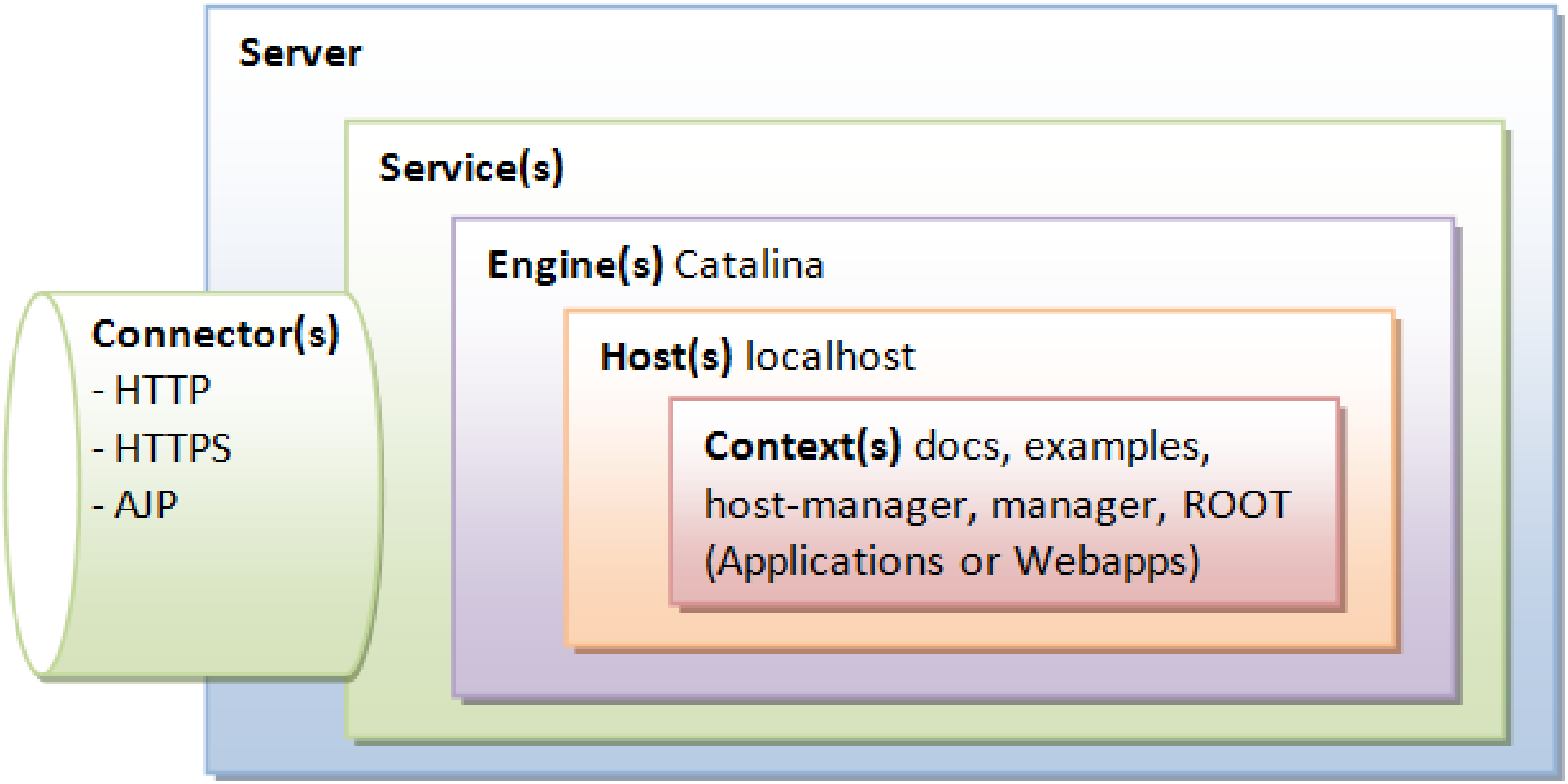
Fichier équivalent au `httpd.conf` de Apache

- <server> élément racine du fichier XML
- <service> type de service utilisé (standAlone ou connecté à Apache)
  - <connector> (un ou plusieurs) Type de connection (et port) utilisé (HTTP/1.1 ou SSL)
  - <engine> (un seul) système de traitement des requêtes d'un service.
    - <host> (1 ou plusieurs) virtual host. gère le système de déploiement des war et la gestion des alias.
      - <context> (1 ou plusieurs)

Description d'une webapp (partie à placer dans META-INF depuis

Fonctionnement

conf/server.xml



Université de Lille

Application WEB = WEB Appliction Resource (WAR)

```
contexte1
    page1.html
    mapage1.jsp
    META-INF
        context.xml
    WEB-INF
        web.xml
        lib
            postgresql.jar
        classes
            servlet1.class
```

Un contexte est archivable via `jar` pour obtenir un `.war`
Les `.war` sont automatiquement décompressés au démarrage.
Facilite grandement le déploiement !

Fonctionnement

Déploiement

- Pour un simple utilisateur (param par défaut, lancement manuel)

```
Tomcat
    bin
    conf
    lib
    work
    webapps
         contexte1
         contexte1.war
         contexte2
         contexte2.war
```

- Pour l'ensemble des utilisateurs (Tomcat en tant que démon)

```
<Host name="localhost" ...>
  ...
  <Listener className="org.apache.catalina.startup.UserConfig"
            directoryName="public_html"
            userClass="org.apache.catalina.startup.PasswdUserDatabase"/>
  ...
</Host>
```

- En fixant le chemin d'accès au war en mettant le descripteur directement dans `server.xml` via la variable `docBase`

Fonctionnement

Configuration d'une nouvelle application

Deux parties :

Dans Server.xml Creer un contexte avec les paramètres associés

- Logger : trace des connexions et authentifications
- Valve : trace des chargements de pages
- Realm : système d'authentification page/rôle
- Pool : pool de connexion

- Depuis Tomcat 4 peut aussi se mettre directement dans `conf/engine/host` avec pour nom `monappli.xml`).
- Depuis Tomcat 5 peut aussi se mettre directement dans `META-INF` avec pour nom `context.xml`). C'est encore mieux !

Dans web.xml de l'appli . mapping servlet, durée de la session, welcomefile, autorisations sur les pages (realm), configurations JNDI : Pool, Mail

Fonctionnement

Définition d'un contexte d'application : `META-INF/context.xml`

```
<Context path="url" docBase="chemin disque" />

<Context path="/monappli" docBase="monappli" reloadable="true"
         crossContext="false" />

<Context path="/monappli2" docBase="monappli2" reloadable="true"
  crossContext="true">
  <Logger className="org.apache.catalina.logger.FileLogger"
          directory="webapps/monappli2/WEB-INF/log/"
          prefix="monappli2_log." suffix=".txt"
          timestamp="true"/>
</Context>
```

voir Documentation/Configuration/Context

Fonctionnement

Configuration d'un web.xml minimal

```
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
<web-app>

    <session-config>
        <session-timeout>30</session-timeout>
    </session-config>

    <welcome-file-list>
        <welcome-file>test.html</welcome-file>
    </welcome-file-list>

</web-app>
```

Fonctionnement

De nombreux autres attributs ...

Ordre à respecter !

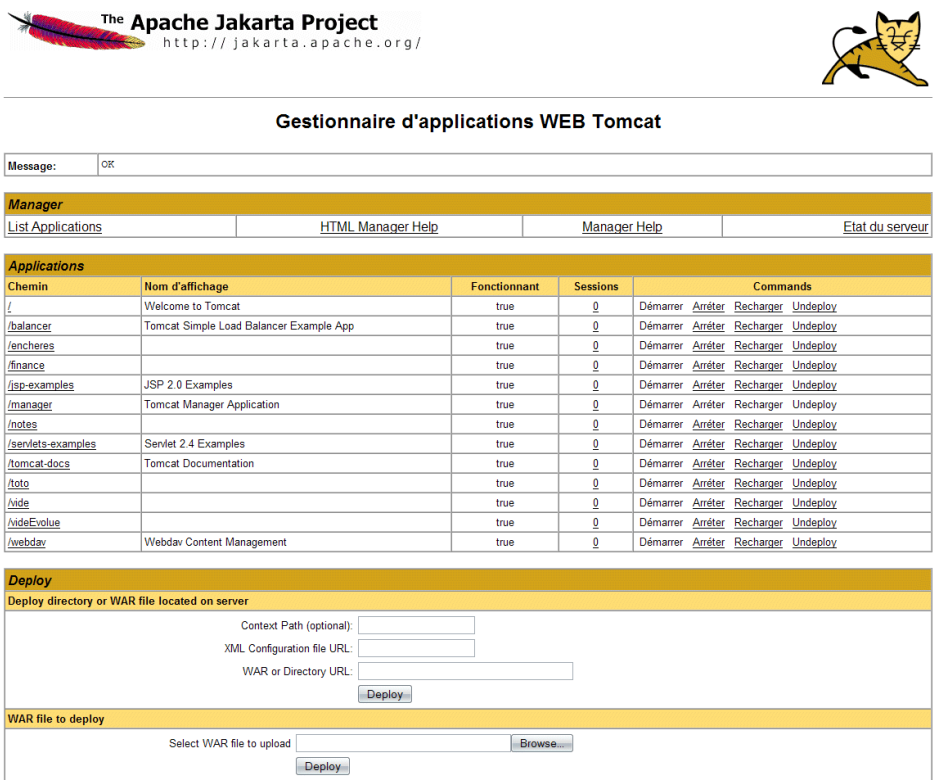
```
icon?
display-name?
description?
distributable?
context-param*
servlet*
servlet-mapping*
session-config?
mime-mapping*
welcome-file-list?
error-page*
taglib*
resource-ref*
security-constraint*
login-config?
security-role*
env-entry*
ejb-ref*
```

Fonctionnement

Console d'administration

Tomcat est fournit avec une console d'administration des contextes et de déploiement à chaud

- ajoutez vous le role `manager-gui` et/ou `manager-script` dans `conf/tomcat-users.xml`



Fonctionnement

Reload automatique et redémarrage ?

- Par défaut les JSP sont auto-reloadable
- Par défaut les Servlet ne sont pas auto-reloadable.
  Pour résoudre ce pb il faut définir un contexte avec
  `<Context path="/monappli" docBase="monappli" reloac`
- Les beans ne sont pas auto-reloadable.
  En cas de modif il faut recharger le contexte :

  `http://localhost:8080/manager/text/list`
  `http://localhost:8080/manager/text/start?path=/monc`
  `(start, stop, reload, sessions)`

Pool de connexion

Pool : principe

Objectif : Résoudre le difficile choix de la création, du controle et de la fermeture des connexions.

- `getConnection` + `close` dans chaque servlet
  pénalise nombreux hits sur la même servlet : ouvre des milliers de connections
- `getConnection` dans `init` et `close` dans `destroy`
  pénalise nombreuses servlets : Si 100 servlets ... 100 connexions bloquées
- `GetConnection` dans `contextListener`
  1 seule connexion pour tous : goulet d'étranglement

Il est impératif d'avoir une gestion fine de la connexion !

Pool de connexion

Pool : principe

Université de Lille

- Collection de connexions préallablement créées, réifiant les methodes d'ouverture et de fermeture de connexion
- C'est le Pool qui s'occupe de créer et de fermer physiquement les connexions indépendamment des programmes
- Chaque programme ne fait plus que des requêtes logiques au Pool. Plus rien de physique.
- Modèle singleton, partagé par l'ensemble des contextes.

Avantages :

- Le lien physique avec la base est externalisé
- on "ouvre" et "ferme" les connexions à chaque page sans soucis
- L'administrateur maitrise le nombre de connexions par contexte

Pool de connexion

API DBCP : DataBase Connection Pooling

Université de Lille

```
import org.apache.commons.dbcp2.BasicDataSource;
...

private static BasicDataSource ds;
ds = new BasicDataSource();
ds.setDriverClassName("org.postgresql.Driver");
ds.setUsername("mathieu");
ds.setPassword("xxx");
ds.setUrl("jdbc:postgresql://localhost/template1");
//
ds.setMinIdle(5);
ds.setMaxIdle(20);
ds.setMaxOpenPreparedStatements(180);
...
con = ds.getConnection();
con.close();
```

```
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-dbcp2</artifactId>
    <version>2.7.0</version>
```

Pool de connexion

Configuration du Pool de Tomcat

Université de Lille

server.xml ou META-INF/context.xml

```
<!-- Syntaxe tomcat 5.5 -->
 <Resource name="monpool" auth="Container" type="javax.sql.DataSource"
  maxTotal="8" maxIdle="4" maxWaitMillis="10000"
  username="duchemin" password="paul"
  driverClassName="org.postgresql.Driver"
  url="jdbc:postgresql://localhost/vide" />
```

et pour l'accès dans la JSP

```
Context initCtx = new InitialContext();
Context envCtx = (Context) initCtx.lookup("java:comp/env");
DataSource ds = (DataSource) envCtx.lookup("monPool");
Connection con = ds.getConnection();
```

Pool de connexion

Configuration du Pool avec JPA

Dans `persistence.xml`

```
<!-- Paramétrages du pool interne (par défaut min/max=32  initial=1) -->
<property name="eclipselink.jdbc.connections.initial" value="1"/>
<property name="eclipselink.jdbc.connections.min" value="2"/>
<property name="eclipselink.jdbc.connections.max" value="5"/>
```

Pool de connexion

Configuration du Pool avec Spring

Dans `application.properties`

```
spring.datasource.max-wait=5000
spring.datasource.max-active=3
spring.datasource.max-idle=1
```

Système d'authentification

Realm : Principe

Gérer automatiquement et de manière portable la sécurisation et l'authentification des pages (user,password,role)

Avantage : Le realm prend en charge tout le travail difficile
Attention : C'est du `lazy authentification`, pas du `on demand`

Quand un utilisateur tente d'accéder à une ressource protégée, le container vérifie s'il est déjà authentifié. Si c'est le cas et que cet utilisateur possède un role compatible avec la ressource, celle-ci lui est renvoyée.

Système d'authentification

S'il n'est pas authentifié :

1. Le formulaire de login est envoyé au client et la ressource demandée est mémorisée
2. Le client renvoie la forme remplie au serveur qui essaye d'authentifier cet utilisateur
3. Si l'authentification donne un succès, le role de l'utilisateur est comparé avec celui nécessaire pour la ressource. S'il a la permission, le client est redirigé vers la ressource précédemment mémorisée
4. Si l'authentification donne un échec, l'utilisateur est automatiquement redirigé vers la page d'erreur définie pour le Realm

Système d'authentification

Configuration d'un Realm

Deux parties :

server.xml ou META-INF/context.xml la déclaration du type de realm utilisé.

- memory realm (login/mdp/role stocké dans un fichier XML)
- JDBC realm (stocké une BDD)
- DataSource realm (accès via un pool)
- LDAP realm (stocké dans un annuaire LDAP)

web.xml Définit les roles(`<security-role>`), le lien entre les pages et les roles (`security-constraint`) ainsi que le type de fenetre d'identification (`login-config`).

- BASIC : gérée par le navigateur, en clair

Système d'authentification

Configuration d'un JDBC Realm

```
<Realm  className="org.apache.catalina.realm.JDBCRealm"
        driverName="org.postgresql.Driver"
        connectionURL="jdbc:postgresql://localhost/vide"
        connectionName="duchemin" connectionPassword="******"
        roleNameCol="role" userCredCol="password" userNameCol="login"
        userRoleTable="users" userTable="users" localDataSource="true"
/>
```

Ici le realm se connecte directement à la base

Système d'authentification

Configuration d'un DataSource Realm avec Hashage

```
<Realm className="org.apache.catalina.realm.DataSourceRealm"
       dataSourceName="myDataSource"
       roleNameCol="role" userCredCol="password" userNameCol="login"
       userRoleTable="users" userTable="users" localDataSource="true"
       <CredentialHandler
          className="org.apache.catalina.realm.MessageDigestCredentialHand
          algorithm="md5"
       />
</Realm>
```

Ici le realm se connecte directement à une source de données et utilise des mdp chiffrés

Utiliser un `NestedCredentialHandler` quand il y a plusieurs méthodes utilisées simultanément.

Système d'authentification

Exemple complet trivial (part1)

```
Arborescence

test
   |- index.jsp
   |- admin
   |- classic
   |- WEB-INF
         |- web.xml
```

```
Dans ce cas, les infos sont recherchées
dans conf/tomcat-users.xml
le Realm activé par défaut.
```

```
<?xml version="1.0" encoding="utf-8"?>
<web-app>
    <!-- lien Pages/Roles -->
        <security-constraint>
        <web-resource-collection>
            <web-resource-name>admin</web-resource-name
            <url-pattern>/admin/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>admin</role-name>
        </auth-constraint>
    </security-constraint>

    <security-constraint>
        <web-resource-collection>
            <web-resource-name>classic</web-resource-na
            <url-pattern>/classic/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>classic</role-name>
        </auth-constraint>
    </security-constraint>

    <!-- La maniere dont la page de login est affichee
    <login-config>
        <auth-method>BASIC</auth-method>
        <realm-name>Nom affiché sur le formulaire</real
    </login-config>
</web-app>
```

Système d'authentification

Exemple complet trivial (part 2)

On crée ici son propre Memory Realm avec son propre fichier d'utilisateurs.

```
test
   |- index.jsp
   |- admin
   |- classic
   |- WEB-INF
   |         |- web.xml
   |- META-INF
             |- context.xml
             |- utilisateurs.xml
```

```
<Context path="/test">
    <Realm className="org.apache.catalina.realm.MemoryRealm"
          pathname="webapps/test/META-INF/utilisateurs.xml" />
</Context>
```

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="classic"/>
  <role rolename="admin"/>

  <user username="jean" password="jean" roles="admin"/>
  <user username="paul" password="paul" roles="classic"/>
</tomcat-users>
```

Système d'authentification

REALM Documentation

la meilleure documentation :

tomcat9.0doc/realm-howto.html

Autres composants

Configuration d'un accès mail

Trois parties :

Ajouter l'API `javax.mail.jar` dans `tomcat/lib`

server.xml ou META-INF/context.xml

```
<Resource name="mail/Session" auth="Container"
          type="javax.mail.Session"
          mail.smtp.host="mailserv.univ-lille1.fr"
          username="..." (ici prenom.nom)
          password="..."
/>
```

web.xml

```
<resource-ref>
    <description> Connexion a un serveur SMTP </description>
    <res-ref-name> mail/Session </res-ref-name>
    <res-type> javax.mail.Session </res-type>
    <res-auth> Container </res-auth>
</resource-ref>
```

Autres composants

et l'appel ...

```
Context initCtx = new InitialContext();
Context envCtx = (Context) initCtx.lookup("java:comp/env");
javax.mail.Session sess = (javax.mail.Session) envCtx.lookup("mail/Session

Message message = new MimeMessage(sess);
message.setFrom(new InternetAddress("alain.terrieur@free.fr"));
InternetAddress to[] = new InternetAddress[1];
to[0] = new InternetAddress("paul.duchemin@gmail.com");
message.setRecipients(Message.RecipientType.TO, to);
message.setSubject("test");
message.setContent("taratata taratata", "text/plain");
Transport.send(message);
```

Autres composants

Le connecteur SSL

- SSL est une technique permettant à un client et un serveur de communiquer de manière sécurisée.
- Ce processus est symétrique entre client et serveur qui chacun envoie à l'autre sa clé publique.
- Coté serveur cela se concrétise par la mise en place d'un "certificat"
- Coté client, visualisable via un cadenas indiqué sur les pages sécurisées.
- Une fois activé, toutes les pages peuvent être accédées par le port 443 par defaut
- Sous Tomcat, par défaut il n'est pas actif, et il pointe sur le port 8443

Autres composants

Le connecteur SSL

Deux étapes :

- Creer (ou récupérer) un certificat
- Ouvrir et paramétrer le connecteur SSL du serveur

1. Créer un certificat.
   - via JSSE qui fournit la commande `keystore` ⇒ uniquement Certificat (clé "changeit" par défaut)
   - via Thawte ou VeriSign ⇒ Cetificat+signature
2. `keytool -genkey -alias tomcat -keyalg RSA`
3. `keytool -list`
4. Attention : le fichier `.keystore` est rangé là où la commande a été tapée. Préféréz la racine du compte.

Autres composants

Le connecteur SSL

Coté serveur ...

1. Décommenter le connecteur SSL dans `server.xml`
2. Paramétrer la clé

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector
           protocol="org.apache.coyote.http11.Http11NioProtocol"
           port="8443" maxThreads="200"
           scheme="https" secure="true" SSLEnabled="true"
           keystoreFile="${user.home}/.keystore" keystorePass="change
           clientAuth="false" sslProtocol="TLS"/>
```

3. Pour accéder aux pages, url : `https://...:8443/...`

Autres composants

Le connecteur SSL

Et pour rediriger certaines pages sur certains ports

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Servlet4Push</web-resource-name>
        <url-pattern>/*</url-pattern>
        <http-method>GET</http-method>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

`CONFIDENTIAL` indique que `HTTPS` sera utilisé. `NONE` indique que `HTTP` sera utilisé.

Autres composants

Le connecteur SSL

la meilleure documentation :

tomcat9.0doc/sslhowto.html#Configuration

Autres composants

Evolution des Releases

- Apparition des Realms : `4.0+`
- Apparition des Pools : `4.1+`
- Apparition du DataSourceRealm : `4.1.?`
- Suppression du invoker servlet par défaut : `4.1.12+`
- Configuration des contexts dans catalina/localhost : `5.0.0+`
- Prise en compte de la norme JSP 2.0 : `5.0+`
- Suppression de la console d'admin dans le package initial : `5.5.0`
- Changement de syntaxe des Pools : `5.5.0+`
- Définition des contextes dans `META-INF/context.xml` : `5.5.0+`
- Suppression des loggers : `5.5.0+`