JavaEE
Développement d'application web

Aix-Marseille Université

Gaël Guibon, Omar Boucelma

2017-2018

Sommaire

Chapitre 1

Notions de bases et environnement

1.1 Notions de bases

1.1.1 Java et JavaEE

Vérification du niveau des étudiants en java standard
Topo sur l'intéret du JEE par rapport au java standard + distinction serveur / navigateur
Quelques rappels :

Java	**JavaEE**
Socle de base	Extension de Java
Programmes locaux	Programmes connectés
Entrée par méthode main()	Entrées par chaque servlet
HTML	**JavaEE**
Site web statique	Site web dynamique
Visible	Opaque et sécurisé

1.1.2 Le modèle MVC

Le principe Modèle Vue Controlleur (MVC) est une bonne pratique de programmation.

Modèle	**Vue**	**Controlleur**
Données	Affichage IHM	Actions / logique du code
Accès BDD	Adaptation de l'UI	Lien entre les éléments
Messages/contacts	Visualisation du message/contacts	Boutons "répondre à tous", envoyer emoji, ...

Exemple d'une répartition MVC :
De nombreux frameworks en différents langages sont désormais MVC :

- Java : Spring [2], Struts [3], Tapestry [4]
- Python : Django, Flask, Pyramid
- Javascript : Angular, ReactJS, EmberJS

1.2 Environnement

1.2.1 Installer Java8

Ouvrir un terminal et lancer les commandes suivantes :

2. http://spring.io/
3. http://struts.apache.org/
4. http://tapestry.apache.org/

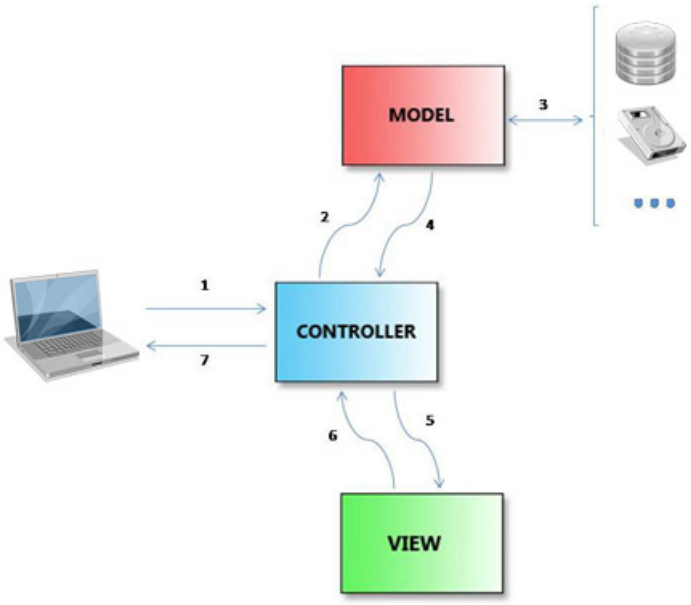

FIGURE 1.1 – Répartition MVC [1]

FIGURE 1.2 – Exemple : Messagerie SMS

1. sudo add-apt-repository ppa :webupd8team/java
2. sudo apt-get update -y

3. sudo apt-get install oracle-java8-installer
4. java -version

La dernière commande devrait vous afficher la version 1.8.x de Java.

1.2.2 Installer Maven

1. cd /opt/
2. wget http ://www-eu.apache.org/dist/maven/maven-3/3.5.0/binaries/apache-maven-3.5.0-bin.tar.gz
3. sudo tar -xvzf apache-maven-3.3.9-bin.tar.gz
4. sudo mv apache-maven-3.3.9 maven
5. sudo nano /etc/profile.d/mavenenv.sh
6. Dans ce fichier ajouter les lignes suivantes :

```
export M2_HOME=/opt/maven
export PATH=${M2_HOME}/bin:${PATH}
```

7. sudo chmod +x /etc/profile.d/mavenenv.sh
8. sudo source /etc/profile.d/mavenenv.sh
9. mvn –version

1.2.3 Installer l'IDE : Eclipse Oxygen

1. Télécharger Eclipse : https://www.eclipse.org/downloads/download.php?file=/oomph/epp/oxygen/R/eclipse-inst-linux64.tar.gz
2. Lancer l'installeur
3. Choisir Eclipse for JavaEE
4. Suivre l'installation avec les configurations par défaut

1.3 Comprendre Maven

1.3.1 Qu'est-ce que c'est ? ! !

Un outil d'automatisation :

— Très répandu en entreprise
— Pour la compilation
— Pour le déploiement
— Facilite le partage de code

Format d'usage :

```
usage: mvn [options] [<goal(s)>] [<phase(s)>]
```

1.3.2 Les éléments de base

```
<project>
  <modelVersion/>       VERSION DE MON PROGRAMME
  <groupId/>    ID DU GROUPE DE MON PROGRAMME : cnrs.amu.lsis ou com.usa.google ou autre
  <artifactId/> ID DE MON PROGRAMME : monprojet ou minecraft ou autre
  <packaging/>  TYPE DE PAQUET EN SORTIE : jar ou war ou autre
  <version/>    VERSION DE MON PROGRAMME : 0.0.9-beta ou 2.1.56
  <name/>       NOM EN INTERNE
  <url/>        URL DU PROGRAMME : par defaut mettre : http://maven.apache.org
  <dependencies/>       DEPENDANCES DU PROGRAMME : jsp, guava, et autres librairies externes
  <build>       EXECUTION LORS DE LA COMPILATION DU PROGRAMME
    <plugins/>  PLUGINS DIVERS
  </build>
  </project>
```

1.3.3 Architecture d'un projet maven

Générer un projet maven basique (java SE) :

```
mvn archetype:generate -DgroupId=test.project -DartifactId=superTest -DarchetypeArtifactId=maven-archetype-
    quickstart -DinteractiveMode=false
```

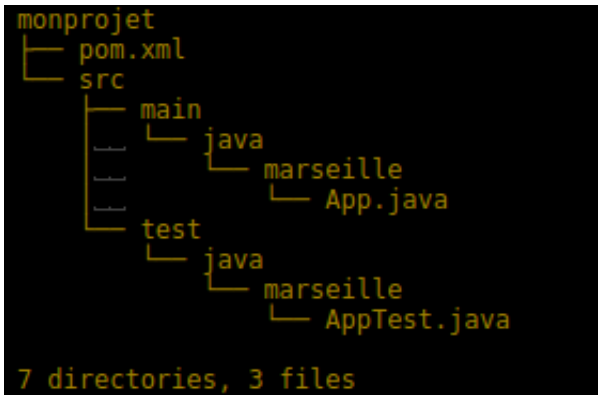

FIGURE 1.3 – Arborescence de base de maven

L'explication officielle de l'arborescence est présente ici : http://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html

1.3.4 Le fichier POM.xml

C'est le fichier central ! Celui qui permet de guider maven dans la compilation.
Un guide officiel du POM est présent à cette adresse : http://maven.apache.org/guides/introduction/introduction-to-the-pom.html

Les dépendances

```
 <dependencies>
        <dependency>
                <groupId>org.apache.tomcat.embed</groupId>
                <artifactId>tomcat-embed-core</artifactId>
                <version>${tomcat.version}</version>
        </dependency>
        .
        .
        .
</dependencies>
```

Les plugins

```
<plugins>
    <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.3</version>
            <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
            </configuration>
    </plugin>
    .
    .
    .
```

```
</plugins>
```

1.3.5 Accéder aux dépendances et plugins

Aller chercher sur maven central : https://search.maven.org/

1.4 Hello World

1. Cloner le projet de base avec Git : git clone [URL]
2. Terminal : mvn package
3. Terminal : java -jar target/basicHelloWorld-jar-with-dependencies.jar OU double clic sur le jar
4. Ouvrir un navigateur à l'adresse localhost :8080

Astuce : La commande htop vous indique les processus en cours (sudo apt install htop)

1.5 Sources utiles pour compléter

Documentation officielle de maven : http://maven.apache.org/guides/index.html
Tutoriel officiel : https://docs.oracle.com/javaee/7/tutorial/
Documentations et API officielle : http://docs.oracle.com/javaee/7/index.html
Tutoriel Site du Zero : https://openclassrooms.com/courses/creez-votre-application-web-avec-java-ee
Mémento et exemples d'annotations Servlet 3.0 : http://www.codejava.net/java-ee/servlet/servlet-annotations-

Chapitre 2

Les contrôleurs - servlets

2.1 Méthodes HTTP

GET	POST	HEAD
Pour ressources web	Pour envoi de fichiers	Pour méta-informations web
Répétée	Ponctuelle	Répétée
Taille limitée	Taille non limitée	Taille limitée
doGet()	doPost()	doHead()

2.2 Qu'est-ce qu'une servlet ?

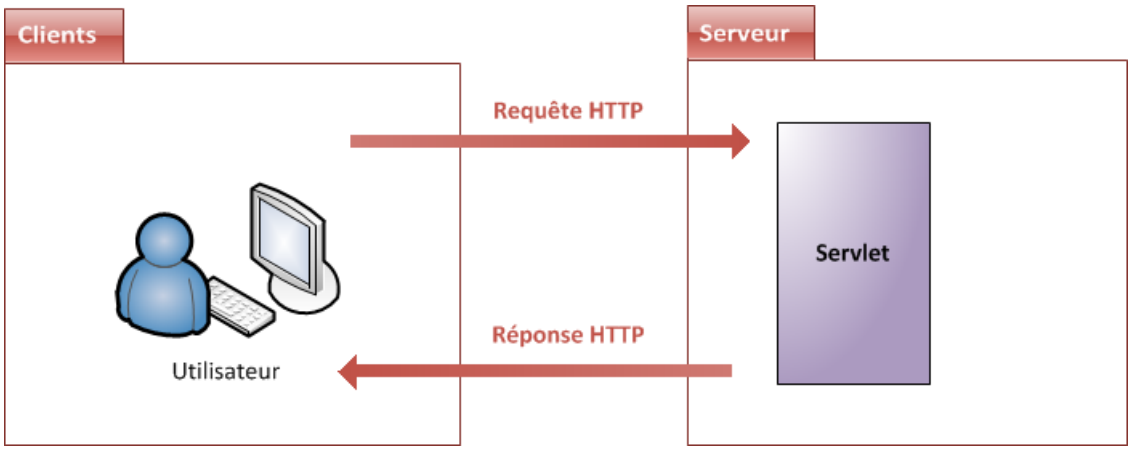

FIGURE 2.1 – Protocole HTTP et servlet [1]

Une servlet est :
- une classe java
- le lien central de l'application : requêtes, réponses, etc.
- un accès aux méthodes HTTP

2.3 Créer une servlet

2.3.1 Créer la servlet vide

Imports nécessaires :

```
import javax.servlet.http.HttpServlet;
```

1. Créer une classe src/mainjava/servlet/HelloServlet.java qui étend la classe HttpServlet
2. Sérialiser la classe

2.3.2 Déclarer la servlet à l'aide d'annotation

Imports nécessaires :

```
import javax.servlet.annotation.WebServlet;
```

1. Utiliser l'annotation @WebServlet
2. Par argument d'annotation, nommer la servlet "MaServlet"
3. Par argument d'annotation, lier la servlet aux urls "/hello" et "/bonjour"

Annotations possibles : https://docs.oracle.com/javaee/7/api/ section javax.servlet.annotation

- **@WebServlet** : pour créer une servlet et la déclarer
- **@WebFilter** : pour filtrer des catégories ("/admin/login"), des types de fichiers ("doc, txt, jpg, etc."),
- **@WebInitParam** : pour associer @WebServlet et @WebFilter
- **@WebListener** : pour créer un listener (exemple : listerner sur l'application pour savoir elle s'est arrêtée)
- **@HandlesTypes** : pour déclarer les classes
- **@MultipartConfig** : pour l'upload de fichier(s)
- **@HttpConstraint** : pour les contraintes de sécurité (SSL/TLS par exemple)
- **@HttpMethodConstraint** : pour les contraintes de sécurité sur les méthodes GET POST HEAD
- **@ServletSecurity** : pour les contraintes de sécurité sur toute la servlet

2.3.3 Remplir la servlet

Imports nécessaires :

```
import javax.servlet.http.HttpServletRequest; // pour req
import javax.servlet.http.HttpServletResponse; // pour resp
import javax.servlet.ServletOutputStream; // pour acceder au stream de sortie de la
    servlet
```

1. Créer une méthode doGet() ayant deux arguments : "req" de type HttpServletRequest et "resp" de type HttpServletResponse
2. Afficher la réponse (resp.getOutputStream() ;) dans le stream de sortie de la servlet (ServletOutputStream)
3. Ecrire "Bonjour le monde" dans le stream de sortie de la servlet
4. Fermer le stream
5. Dans un navigateur aller à "localhost :8080/bonjour"

2.4 Lier une servlet à une jsp (MVC)

1. Créer une nouvelle jsp basique nommée seconde.jsp dans src/main/resources/webapp
2. Ecrire un code html basique dans cette jsp

```
  <html>
<body>
        <h2>This is a superb text</h2>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
                eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
                     ad
                minim veniam, quis nostrud exercitation ullamco laboris nisi ut
```

```
                aliquip ex ea commodo consequat. Duis aute irure dolor in
                reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
                pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
                culpa qui officia deserunt mollit anim id est laborum.</p>
                <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
                eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
                    ad
                minim veniam, quis nostrud exercitation ullamco laboris nisi ut
                aliquip ex ea commodo consequat. Duis aute irure dolor in
                reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
                pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
                culpa qui officia deserunt mollit anim id est laborum.</p>
</body>
</html>
```

3. Créer une nouvelle servlet nommée "servletMVC" et comme url "/lorem"
4. Au lieu d'écrire directement dans la sortie de la servlet, transférer la requête et la réponse à une jsp existante

```
this.getServletContext().getRequestDispatcher( "/[NOM_DE_LA_JSP].jsp" ).forward(
    [REQUETE], [REPONSE] );
```

5. Aller à localhost :8080/lorem et voyez votre résultat en version MVC

2.5 Rediriger en fonction des paramètres de l'URL

1. Créer une nouvelle servlet nommée "ServletCondition" avec url "/condition"
2. Lire le paramètre "prenom" donné en requête url "/condition ?prenom=John"

```
// l'ensemble des parametres est une Map
Map<String, String[]> parameters = req.getParameterMap();

// il est possible d'acceder directement a un parametre
String paramPrenom = req.getParameter("MON_PARAMETRE");
```

3. Si le paramètre "prenom" est présent et non vide, transférer vers "seconde.jsp". Si non, transférer vers "index.jsp"

Cette solution ne respecte pas le modèle MVC !

Chapitre 3

La vue - EL et JSTL

3.1 EL : Expression Language

3.1.1 Qu'est-ce que l'EL ?

C'est un langage permettant de connecter facilement le contenu java et l'affichage HTML via une jsp.
Les EL permettent notamment de :

- Conditionner l'affichage
- Faire des calculs simples
- Accéder aux données / méthodes / paramètres de requête

3.1.2 Accéder aux paramètres et attributs

1. Créer une servlet nommée "ServletEL" avec url "/el" qui transfère vers "el.jsp"
2. Créer une jsp nommée "el.jsp" et y placer le HTML suivant

```
  <!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Test des expressions EL</title>
</head>
<body>
        <p>Je m'appelle ${ A REMPLIR }</p>
</body>
</html>
 
```

3. Afficher la valeur du paramètre "prenom". ("Je m'appelle XXX")
4. En fonction de la valeur du paramètre "pays" afficher la nationalité à l'aide d'un attribut ("Je m'appelle XXX et je suis de nationalité XXX").

```
  // servlet : donner un attribut a la requete
  req.setAttribute("test", "youhou");

  // jsp : acceder Ã  l'attribut
  ${ test }
```

5. Afficher plusieurs prénoms à partir de l'unique paramètre "prénoms". ("Mes prénoms sont XXX et XXX")

```
  // donner plusieurs valeurs a un seul parametre
  /el?prenoms=XXX&prenoms=XXX

  // acceder au parametre Ã  valeur multiple
  ${ paramValues.XXX }

  // acceder a une de ces valeurs
  ${ paramValues.XXX[index] }
```

3.2 JSTL

3.2.1 Qu'est-ce que JSTL ?

JSTL est une librairie regroupant plusieurs fonctions à utiliser dans la Vue. Permet de mieux respecter le modèle MVC !

3.2.2 Les principales commandes

Les balises core

Super mémo complet ici –> https://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm
Quelques exemples :

```
<%— Affiche Hello World —>
<c:out value="Hello␣World">

<%— Declare une variable nommee "maVar" avec valeur 250 et une portee sur la session —>
<c:set var = "maVar" scope = "session" value = "${50+200}"/>

<%— Affiche le contenu de la variable "maVar" (i.e. 250) —>
<c:out value="${␣maVar␣}">

<%— Affiche la balise <p> si "maVar" est superieure Ã  50 —>
<c:if test = "${␣maVar␣>␣50␣}">
    <p>La variable "maVar" est supérieure Ã  50 ! Wahou!/><p>
</c:if>

<%— Boucle for sur un indice —>
<c:forEach var = "i" begin = "1" end = "5">
    Item <c:out value = "${i}"/><p>
</c:forEach>

<%— Iteration sur les elements d'une␣liste␣—>
<c:forEach␣items="${␣maListe␣}"␣var="element">
␣␣␣␣␣␣<c:out␣value="${␣element␣}"␣/>
</c:forEach>
```

Les fonctions JSTL

- fn :replace()
- fn :length()
- fn :join()
- etc.

3.2.3 Configurer JSTL

- Ajouter la librairie jstl en dépendance du projet maven (pom.xml) : https://mvnrepository.com/artifact/jstl/jstl/1.2
- Créer une nouvelle jsp nommée jstl.jsp et y ajouter tout en haut la ligne suivante :

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

3.2.4 Un affichage plus intelligent

1. Ajouter les lignes suivantes à jstl.jsp :

```
<html>
<head>
<meta charset="utf-8" />
```

```
<title>Test de JSTL</title>
</head>
<body>
        <c:out value="<p>Bonjour␣inconnu(e)</p>" />
</body>
</html>
```

2. Créer une servlet simple nommée "ServletJSTL" avec url "/jstl"
3. Afficher "Bonjour inconnu(e)" s'il n'y a pas de paramètre "prenom" donné.
4. Afficher le drapeau du pays donné en paramètre "pays"

C'est bien plus propre et respecte mieux le MVC !

Chapitre 4

Le modèle - beans

4.1 Qu'est-ce que c'est ?

Une manière de stocker les données. Un est :
- Réutilisable ! (objet java)
- Persistant ! (serialization)
- Paramétrable ! (propriétés)

Il respecte donc plusieurs règles :
- C'est une classe publique.
- Implémente Serializable pour pouvoir être sauvegardé.
- AUCUN champ public ! (getter et setters)
- Possède un constructeur par défaut public, sans paramètres ?

4.2 Exo : Agenda

4.2.1 Créer le bean Rendezvous

Créer un bean (un objet java) Rendezvous.java.

1. Ajouter les propriétés "duree"(int), "personnes"(liste de noms), "lieu"(string) et "type"(string)
2. Ajouter des getter et setter pour chaque propriété
3. Ajouter une méthode qui retourne le nombre de personnes

4.2.2 Créer le service RendezvousService

Un service est une classe qui va ouvoir être instancier et gérer un ou plusieurs beans, ainsi que leurs relations.

1. Créer un service RendezvousService.java
2. Ajouter une liste de rendezvous
3. Ajouter un constructeur qui initialise des rendez-vous par défauts
4. Ajouter une méthode d'ajout de rendez-vous
5. Ajouter une méthode qui récupère le nombre de rendez-vous
6. Ajouter une méthode qui récupère les rendez-vous en fonction de plusieurs types de rendez-vous différents

4.2.3 Utiliser le bean

1. Afficher un tableau auto généré pour lequel chaque ligne est un rendez-vous, et chaque colonne une propriété
2. Par requête http, ajouter des rendez-vous et rafraîchir la page automatiquement

Chapitre 5

Créer le mockup d'une application de messagerie (clone de twitter)

5.1 Les demandes du client

— Le client veut voir les messages envoyés (beans + service + jstl).
— Le client veut pouvoir envoyer un message (param + service).
— Le client veut pouvoir voir son nouveau message sans avoir à recharger la page (request + servlet).
— Le client veut pouvoir connaître en temps réel le nombre de messages envoyés (service).
— Le client veut pouvoir supprimer un message (service).
— Le client veut pouvoir ne pas avoir besoin de sélectionner le champ pour pouvoir écrire et d'envoyer directement le message en appuyant sur "entrée" (jquery)
— Le client veut une belle interface.... (css + ajax + servlet)

5.2 Quelques ressources utiles

— **Bootstrap** : http://getbootstrap.com/
— **W3School** (html, css, javascripts memos et tutoriels) : https://www.w3schools.com/

Chapitre 6

[WIP] Le stockage de données - ORM

à venir...

Chapitre 7

[WIP] Rendre persistance l'application

Chapitre 8

[WIP] Déploiement