

# 03) Introduction à Java EE

---

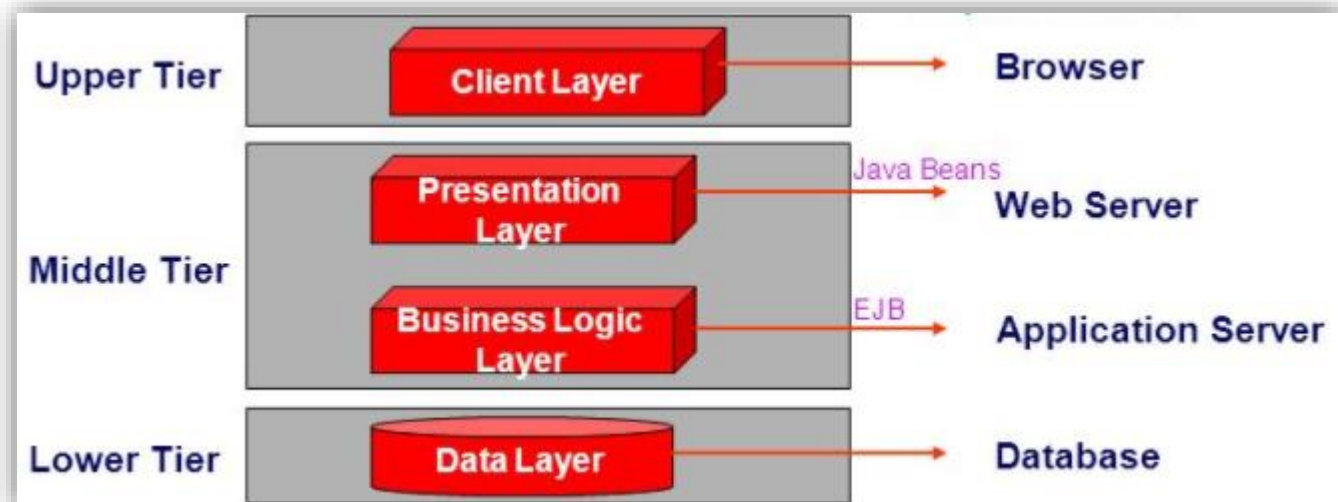
420-109-GG

LOTFI DERBALI

# La plateforme Java EE

Le rôle de la plateforme Java EE est d'apporter une infrastructure permettant d'accueillir des applications distribuées plus ou moins complexes.

- Le schéma suivant présente une vision générale de cette infrastructure composée de 3 couches principales.



# La plateforme Java EE

---

1. La couche "**client**" est la couche interagissant avec l'utilisateur. Cette couche peut prendre différentes formes : une application Java SE autonome, une applet, une page web accessible au travers d'un navigateur
2. La couche "**intermédiaire**" est la couche répondant aux requêtes de l'utilisateur. Elle peut se composer de deux sous-couches :
  - La couche **web** répondant aux requêtes HTTP avec les technologies centrales que sont les servlets et les JSP.
  - La couche **EJB** (Enterprise Java Bean) mettant à disposition des services centralisés comme l'accès aux données ou des traitements métier divers.
3. La couche "**données**" permet de stocker et lire les données manipulées par les applications. Elle peut être représentée par des bases de données relationnelles mais peut très bien être d'un autre type (fichiers XML, base de données NOSQL...).

# Conteneurs (*containers*)

---

C'est l'environnement d'exécution d'une application Java EE.

Il existe 4 conteneurs différents permettant d'exécuter et de gérer le cycle de vie des différents composants formant l'application.

- Le **conteneur d'applications** correspond tout simplement à la JVM installée sur la machine cliente
- Le **conteneur d'applets** correspond à un navigateur web embarquant un plug-in Java.
- Le **conteneur web** est fourni par un serveur d'applications. Il doit obligatoirement apporter une implémentation des spécifications des servlets et des JSP.
  - Il existe un certain nombre de serveurs d'applications. Un des serveurs les plus populaires est **Tomcat** de la fondation Apache (<http://tomcat.apache.org/>).
- Le **conteneur d'EJB** est aussi fourni par un serveur d'applications.
  - Tomcat est dépourvu de ce conteneur. Il faut utiliser un autre serveur d'applications comme JBoss, WebLogic... Ces serveurs proposent souvent eux-mêmes un conteneur web.

Ces conteneurs doivent respecter des règles et mettre à disposition des API standards.

- Par exemple, une application s'exécutant sur Tomcat doit ainsi pouvoir s'exécuter sur un autre serveur d'applications sans modification (mis à part certains points de paramétrage).

# Les spécifications Java EE

---

Plusieurs spécifications (trente-neuf au total !) forment la plateforme Java EE 7

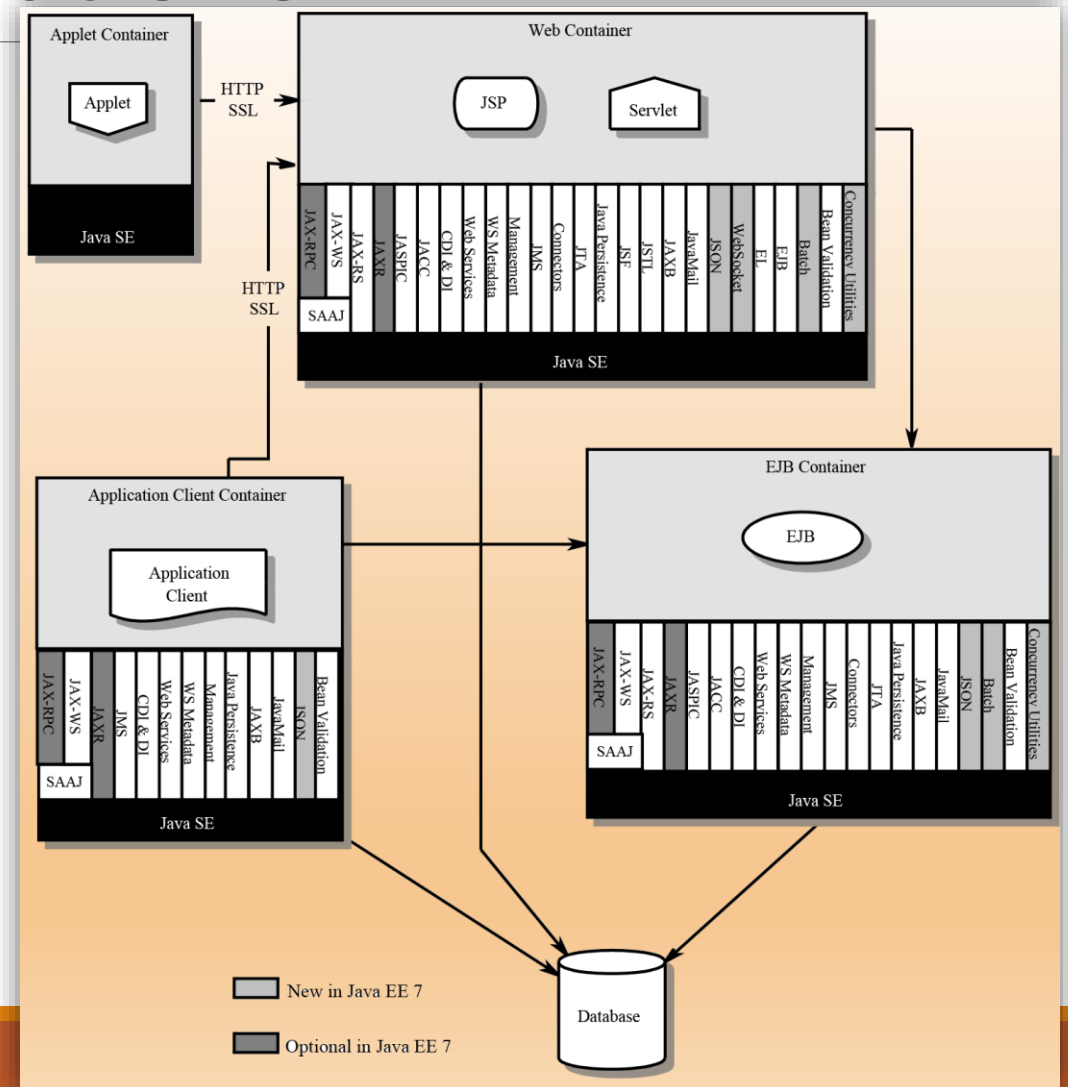
Certaines spécifications font partie du Web Profile et couvrent les technologies pour réaliser des applications web, la persistance des données et la mise en œuvre des services web REST.

- **Technologies pour les applications web**
  - Java API for WebSocket (*JSR 356*)
  - Java Servlet 3.1 (*JSR 340*)
  - JavaServer Faces 2.2 (*JSR 344*)
  - Expression Language 3.0 (*JSR 341*)
  - JavaServer Pages 2.3 (*JSR 245*)
  - Standard Tag Library for JavaServer Pages (JSTL) 1.2 (*JSR 52*)
- **Technologies pour les applications d'entreprise**
  - Java Persistence 2.1 (*JSR 338*)
- **Technologies des services web**
  - Java API for RESTful Web Services (JAX-RS) 2.0 (*JSR 339*)
- **Technologies Java SE fortement utilisées**
  - Java Database Connectivity 4.0 (*JSR 221*)

# Répartition des spécifications

Le schéma suivant présente l'interaction entre les différents conteneurs et les spécifications que chacun peut ou doit mettre à disposition.

- Les spécifications dans la partie haute sont obligatoires
- Les spécifications dans la partie basse (écrites verticalement) sont optionnelles.

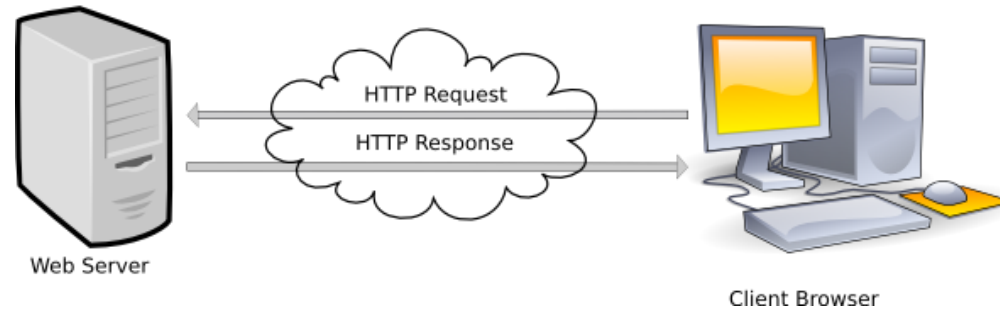


# Protocole HTTP

---

Le protocole **HTTP** (*HyperText Transfer Protocol*) est incontournable dans la réalisation d'application web.

- Le principe repose sur un couple requête/réponse comme le montre le schéma suivant :



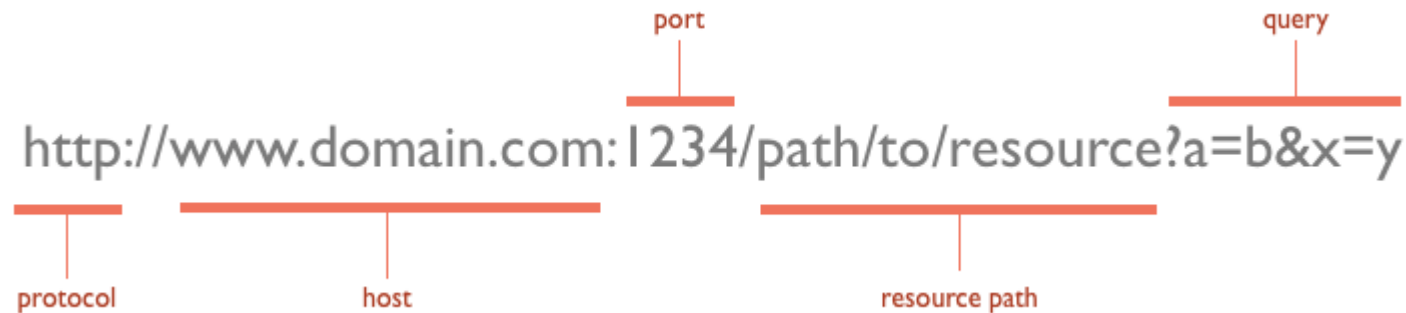
- Le client est toujours l'initiateur en envoyant une requête vers le serveur qui va, en retour, envoyer une réponse.
- Les requêtes et réponses HTTP sont transportées grâce au protocole **TCP/IP** (*Transmission Control Protocol/Internet Protocol*).
- Avant d'effectuer la requête, il faut donc établir une connexion TCP/IP.
- Depuis la version 1.1 du protocole HTTP, plusieurs couples requête/réponse peuvent être transportés sur la même connexion TCP/IP comme le montre le schéma suivant.

# URI

---

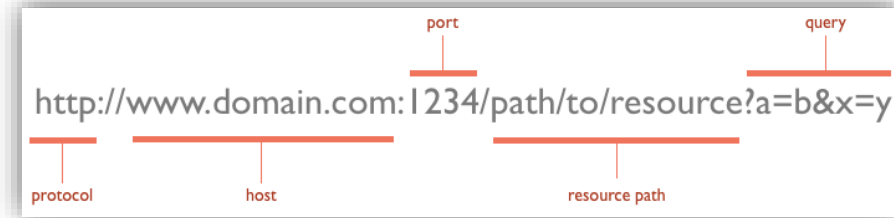
Un **URI** (*Uniform Resource Identifier*) est une chaîne de caractères formatée identifiant une ressource.

- Un URL permet de localiser une ressource (*URL - Uniform Resource Locator*)
- Un URN permet de nommer une ressource (*URN - Uniform Resource Name*)
- La structure d'une URL HTTP doit être la suivante :





# URI



**Hôte** : ce paramètre permet de connaître la machine sur laquelle se situe la ressource. Il peut prendre la forme d'une adresse IP, d'un nom de machine (possible dans un réseau privé) ou d'un nom de domaine

**Port** : ce paramètre est optionnel. La valeur par défaut est 80. Il permet à une machine d'exposer différentes applications, chacune écoutant sur un port particulier. C'est une sorte d'extension à la notion d'adresse IP. L'adresse IP identifie une machine et le port identifie une application

**Chemin ressource** : ce paramètre définit le chemin d'accès à la ressource visée. Le slash "/" peut être utilisé pour séparer les différentes parties du chemin d'accès

**Paramètres** : ce paramètre sert à envoyer une ou plusieurs valeurs complémentaires. C'est utile dans le cas où la ressource est une ressource dynamique, c'est-à-dire une ressource qui déclenche l'exécution de code côté serveur. La structure est la suivante :

*param1=valeur1&param2=valeur2*

# Requête

---

Lorsque vous écrivez une URL dans votre navigateur préféré, celui-ci va envoyer une **requête HTTP** vers le serveur hébergeant l'application.

- La requête doit respecter la structure suivante :
  - En-tête de requête
  - Corps de requête (optionnel)
- Par exemple, lorsque vous écrivez l'URL ***http://tomcat.apache.org/index.html*** dans la barre d'adresse de votre navigateur, celui-ci crée la requête suivante :

```
GET /index.html HTTP/1.1
Host: tomcat.apache.org
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:49.0) Gecko/20100101
Firefox/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

# Les types de requêtes

---

Le protocole HTTP/1.1 propose différents types de requêtes (aussi appelés **méthodes** ou **verbes**). Cette information est obligatoirement présente dans une requête HTTP.

- **OPTIONS** : ce type de requête permet au client d'obtenir des informations sur les options disponibles pour obtenir la ressource visée par la requête.
- **GET** : ce type de requête permet au client d'obtenir la ressource visée par la requête.
- **HEAD** : ce type de requête est identique au type GET à part que la réponse ne devra jamais contenir de corps. Cette méthode est souvent utilisée pour tester la validité d'un lien.
- **POST** : ce type de requête permet au client d'envoyer un corps de requête contenant des informations que le serveur doit prendre en compte. Ce corps de requête est typiquement constitué d'informations saisies dans un formulaire HTML. Le serveur va enregistrer les informations reçues comme une nouvelle ressource.
- **PUT** : ce type de requête fonctionne comme les requêtes de type POST. La différence fondamentale est que ce type de requête vise une ressource existante. Le serveur va modifier la ressource existante avec les nouvelles informations reçues.
- **DELETE** : ce type de requête permet au client de supprimer la ressource visée par la requête.
- **TRACE** : ce type de requête permet au client de recevoir de la part du serveur le contenu de la requête telle qu'il l'a reçue. Ce type de requête peut être utilisé dans le cadre de tests ou de diagnostics.
- **CONNECT** : ce type de requête permet au client de se connecter à un proxy permettant de faire du tunneling.

# Les attributs de requêtes

---

Les attributes de requêtes ne sont pas tous obligatoires.

- **Connection:** permet d'indiquer si la connexion TCP/IP doit être maintenue (Keep-Alive) ou pas (close).
- **Content-Length:** définit la taille en nombre d'octets du corps de la requête ou de la réponse.
- **Content-Type:** définit le type de média correspondant au corps de la requête ou de la réponse.
- **Date:** définit la date à laquelle la requête ou la réponse a été générée.
- **Accept:** définit le ou les types de médias que le client accepte. Veuillez vous référer à la section Les types de médias.
- **Accept-Encoding:** définit les méthodes de compression acceptées par le client (ex : gzip, deflate).
- **User-Agent:** définit les informations du logiciel client à partir duquel la requête est émise. Cet attribut est intéressant pour faire des statistiques sur les navigateurs accédant à votre application.
- **Cookie:** permet d'envoyer au serveur tous les cookies associés au nom de domaine de la requête.
- ...

# Un mot sur l'attribut Content-Type

---

Voici une liste non exhaustive des principaux types de médias rencontrés.

Types de médias	Description
<code>text/plain</code>	Définit un contenu textuel brut.
<code>text/html</code>	Définit un contenu HTML.
<code>text/xml</code>	Définit un contenu XML.
<code>text/csv</code>	Définit un contenu CSV.
<code>text/css</code>	Définit un contenu CSS.
<code>image/jpeg</code>	Définit une image au format JPEG.
<code>image/png</code>	Définit une image au format PNG.
<code>image/svg+xml</code>	Définit un contenu SVG permettant de faire du dessin vectoriel.
<code>video/mpeg</code>	Définit une vidéo au format MPEG.
<code>video/mp4</code>	Définit une vidéo au format MP4.
<code>application/javascript</code>	Définit un contenu JavaScript.
<code>application/octet-stream</code>	Définit un contenu binaire non typé.
<code>application/pdf</code>	Définit un contenu au format PDF.
<code>application/json</code>	Définit un contenu au format JSON ( <i>JavaScript Object Notation</i> ).
<code>application/xml</code>	Définit un contenu au format XML.

# Réponse

---

Lorsque le serveur reçoit une requête, celui-ci va mettre en œuvre un ensemble de traitements pour retourner une **réponse** au client. Comme la requête, la réponse doit respecter la structure suivante :

- En-tête de réponse
- Corps de réponse (optionnel)

```
HTTP/1.1 200 OK
Date: Wed, 05 Oct 2016 19:44:50 GMT
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Tue, 20 Sep 2016 06:56:13 GMT
ETag: "39b8-53ceaec9cf9c5-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 4065
Keep-Alive: timeout=30, max=100
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE html SYSTEM "about:legacy-compat">
<html lang="en">
...
</html>
```

# Réponse: Les codes de statut

---

Chaque réponse est associée à un **code de statut** sur trois digits. Ce code permet d'indiquer au client l'état de la réponse. Le client peut ainsi réagir en conséquence.

- Ces codes sont classés en cinq catégories. La catégorie est représentée par le chiffre des centaines :

Catégorie	Code	Description
Information	1xx	La requête est reçue par le serveur, le traitement se poursuit.
Succès	2xx	La requête a été reçue, comprise et traitée par le serveur.
Redirection	3xx	Des actions complémentaires sont nécessaires pour terminer la requête.
Erreur client	4xx	La requête ne peut pas être traitée par le serveur.
Erreur serveur	5xx	Le serveur est mis en échec sur le traitement d'une requête valide.

# Réponse: Les codes de statut

---

Voici la liste des codes les plus courants :

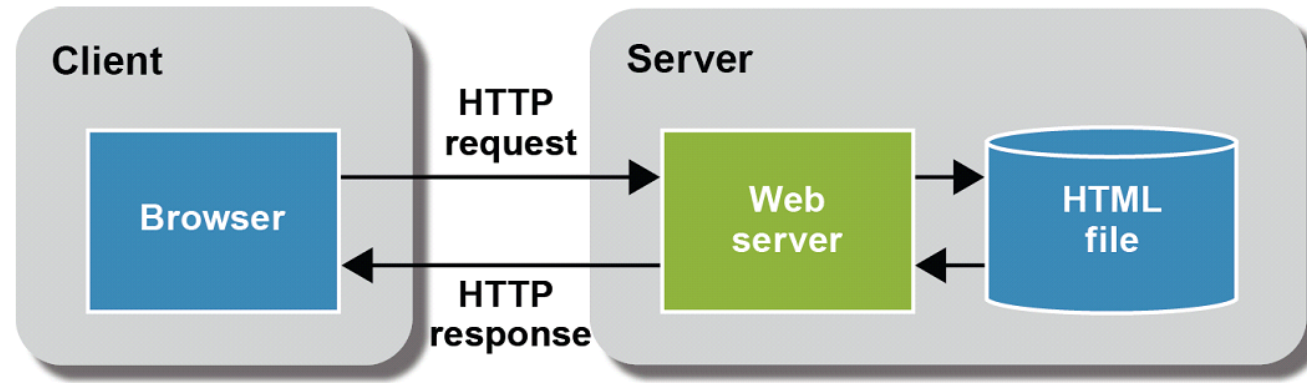
Code	Libelle	Description
100	Continue	Continue
101	Switching Protocols	Changement de protocole
200	OK	OK
201	Created	Créé
202	Accepted	Acceptée
204	No Content	Requête traitée avec succès mais aucune information en retour.
206	Partial Content	Contenu partiel (la suite arrive)
301	Moved Permanently	La ressource a changé d'adresse de façon permanente.
302	Found	La ressource a changé d'adresse temporairement.
304	Not Modified	La ressource n'a pas changé. Le client peut utiliser la version en cache.
400	Bad Request	La requête est incorrecte.
401	Unauthorized	Le client doit s'authentifier pour accéder à la ressource.
403	Forbidden	Le client n'a pas le droit d'accéder à la ressource.
404	Not Found	La ressource demandée n'a pas été trouvée.
405	Method Not Allowed	Le type de la requête n'est pas autorisé.
500	Internal Server Error	Le serveur a rencontré une erreur inattendue.
501	Not Implemented	Le serveur ne sait pas traiter le type de la requête pour la ressource visée.
503	Service Unavailable	Le serveur n'est pas capable de répondre à cause d'une surcharge temporaire ou d'une maintenance.



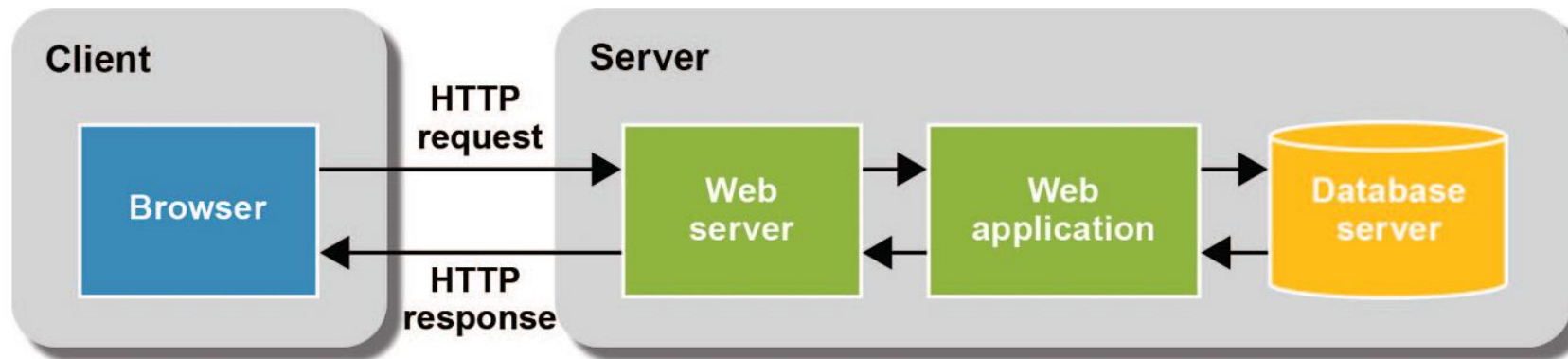
# Pages Web statiques et dynamiques

---

static web pages



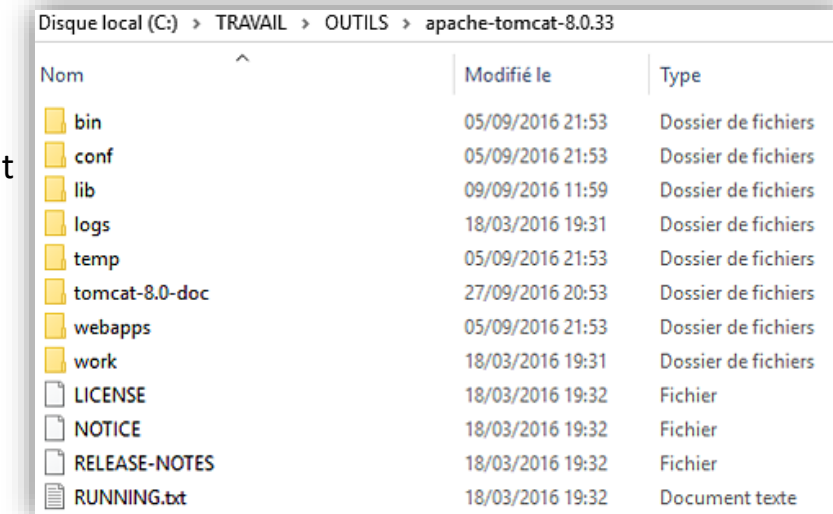
dynamic web pages



# Environnement de développement

## Étape 1: Tomcat

- Tomcat est le serveur d'applications choisi pour mettre en œuvre les technologies de la plateforme Java EE.
- Téléchargement
  - la dernière version téléchargeable est Tomcat 8 à l'adresse suivante <https://tomcat.apache.org/download-80.cgi>
  - Privilégiez la version compressée .ZIP
- Installation
  - L'installation est simple. Il suffit de décompresser l'archive correspondant à votre système d'exploitation dans le répertoire de votre choix.
  - Le répertoire **lib** contient les librairies composant Tomcat. Deux librairies sont rapidement identifiables : ***servlet-api.jar*** et ***jsp-api.jar***
    - Elles correspondent toutes les deux à l'implémentation des spécifications JSR340 et JSR245.
    - Tomcat est donc bien un conteneur web.

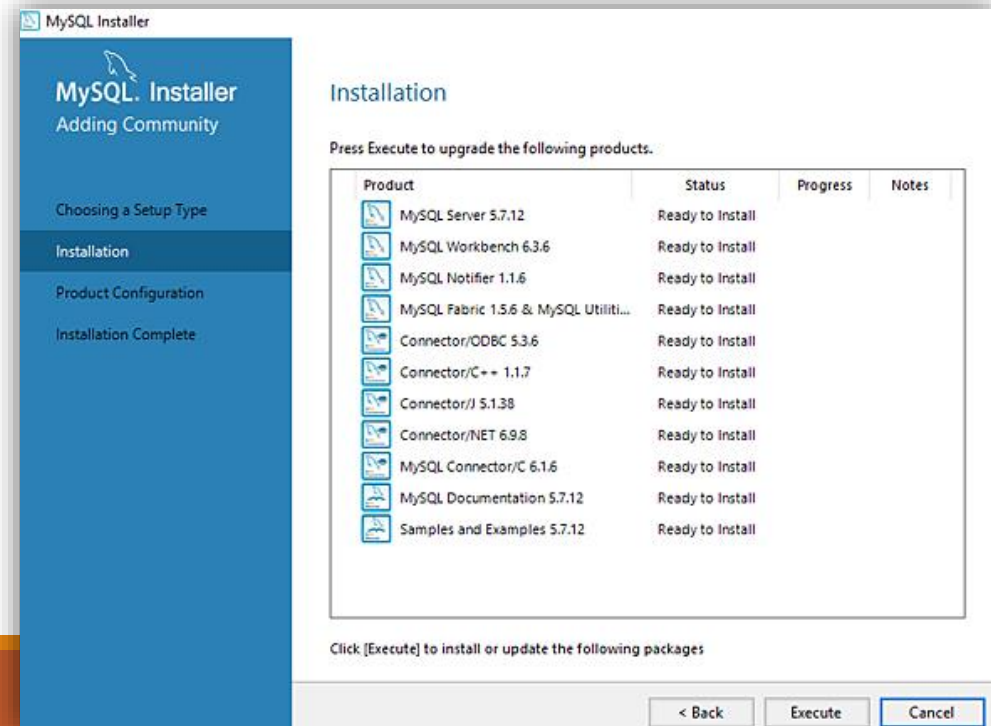


Nom	Modifié le	Type
bin	05/09/2016 21:53	Dossier de fichiers
conf	05/09/2016 21:53	Dossier de fichiers
lib	09/09/2016 11:59	Dossier de fichiers
logs	18/03/2016 19:31	Dossier de fichiers
temp	05/09/2016 21:53	Dossier de fichiers
tomcat-8.0-doc	27/09/2016 20:53	Dossier de fichiers
webapps	05/09/2016 21:53	Dossier de fichiers
work	18/03/2016 19:31	Dossier de fichiers
LICENSE	18/03/2016 19:32	Fichier
NOTICE	18/03/2016 19:32	Fichier
RELEASE-NOTES	18/03/2016 19:32	Fichier
RUNNING.txt	18/03/2016 19:32	Document texte

# Environnement de développement

## Étape 2: MySQL

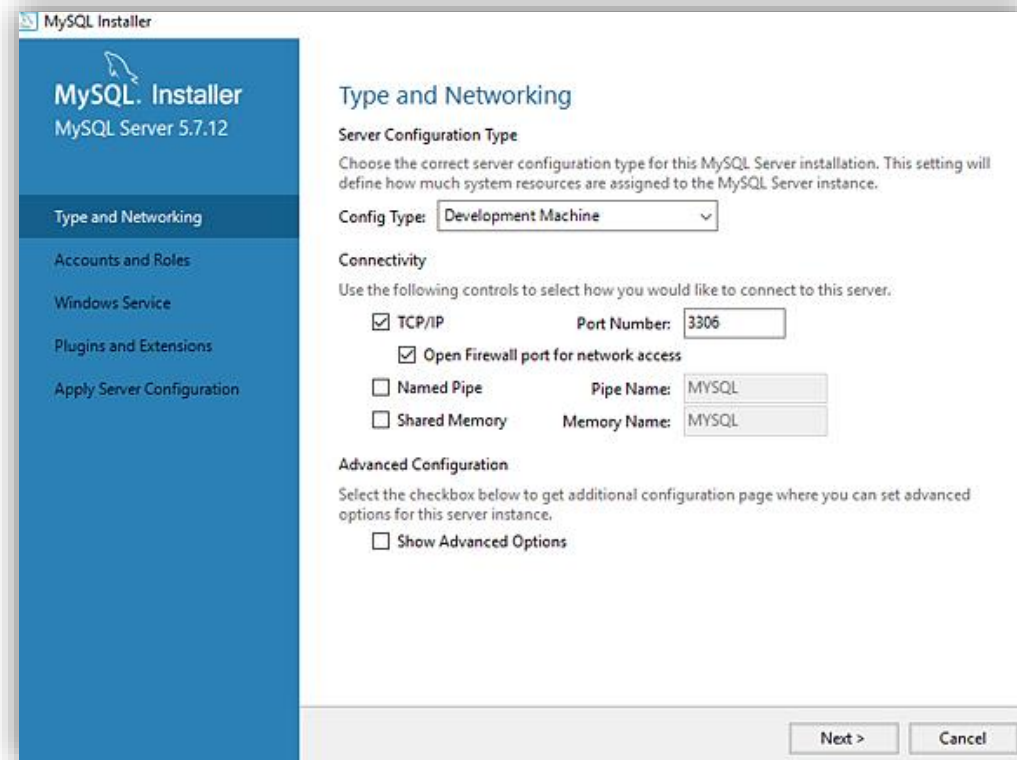
- Pour découvrir l'API JDBC, ce cours va s'appuyer sur le SGBDR (système de gestion de bases de données relationnelles) MySQL.
- Téléchargement
  - À partir de l'installeur MySQL Installer MSI <http://dev.mysql.com/downloads/mysql/>
  - L'outil utilisé pour administrer la base de données est MySQL Workbench CE.
- Installation
  - L'installation de MySQL et ses outils est guidée et ne pose pas de problème particulier.
  - Les éléments essentiels pour ce cours sont :
    - MySQL Server
    - Connector/J (le pilote JDBC).



# Environnement de développement

## Étape 2: MySQL

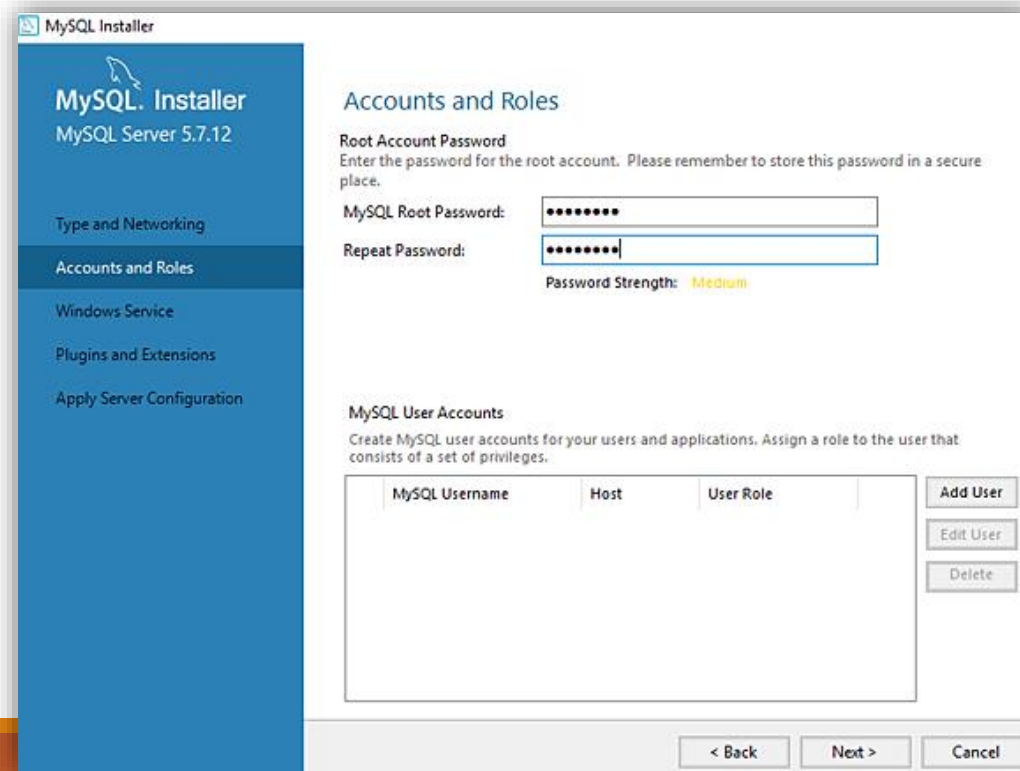
- La configuration par défaut est conservée. Elle permet une communication TCP/IP avec le serveur sur le port 3306



# Environnement de développement

## Étape 2: MySQL

- Le mot de passe du compte *root* est à définir et à ne pas oublier
- Vous pouvez ajouter un compte supplémentaire en cliquant sur le bouton « *Add User* »



The screenshot shows the 'MySQL Installer' window for 'MySQL Server 5.7.12'. The left sidebar lists the installation steps: 'Type and Networking', 'Accounts and Roles' (selected), 'Windows Service', 'Plugins and Extensions', and 'Apply Server Configuration'. The main area is titled 'Accounts and Roles' and contains two sections. The first section, 'Root Account Password', prompts the user to enter a password for the root account, with fields for 'MySQL Root Password' and 'Repeat Password', and a 'Password Strength' indicator showing 'Medium'. The second section, 'MySQL User Accounts', provides instructions to create user accounts and assign roles, featuring a table with columns for 'MySQL Username', 'Host', and 'User Role', and buttons for 'Add User', 'Edit User', and 'Delete'.

MySQL Installer

MySQL. Installer  
MySQL Server 5.7.12

Type and Networking

Accounts and Roles

Windows Service

Plugins and Extensions

Apply Server Configuration

### Accounts and Roles

**Root Account Password**  
Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password:

Repeat Password:

Password Strength: **Medium**

**MySQL User Accounts**  
Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL Username	Host	User Role
----------------	------	-----------

< Back Next > Cancel

# Environnement de développement

---

## Étape 3: Eclipse

- Téléchargement
  - Il est important de télécharger la version pour développeur Java EE (*Eclipse IDE for Java EE Developers*).
  - <http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/oxygen1arc1>



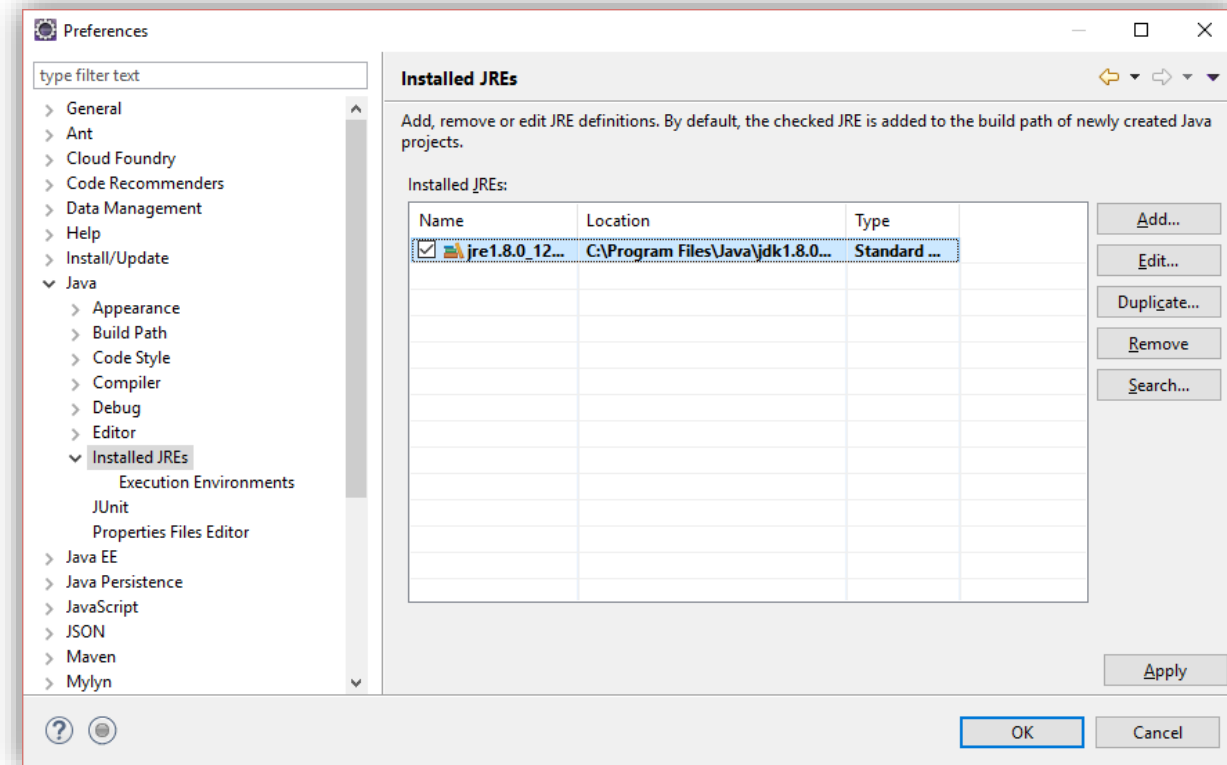
## Eclipse IDE for Java EE Developers

### Package Description

Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn, EGit and others.

This package includes:

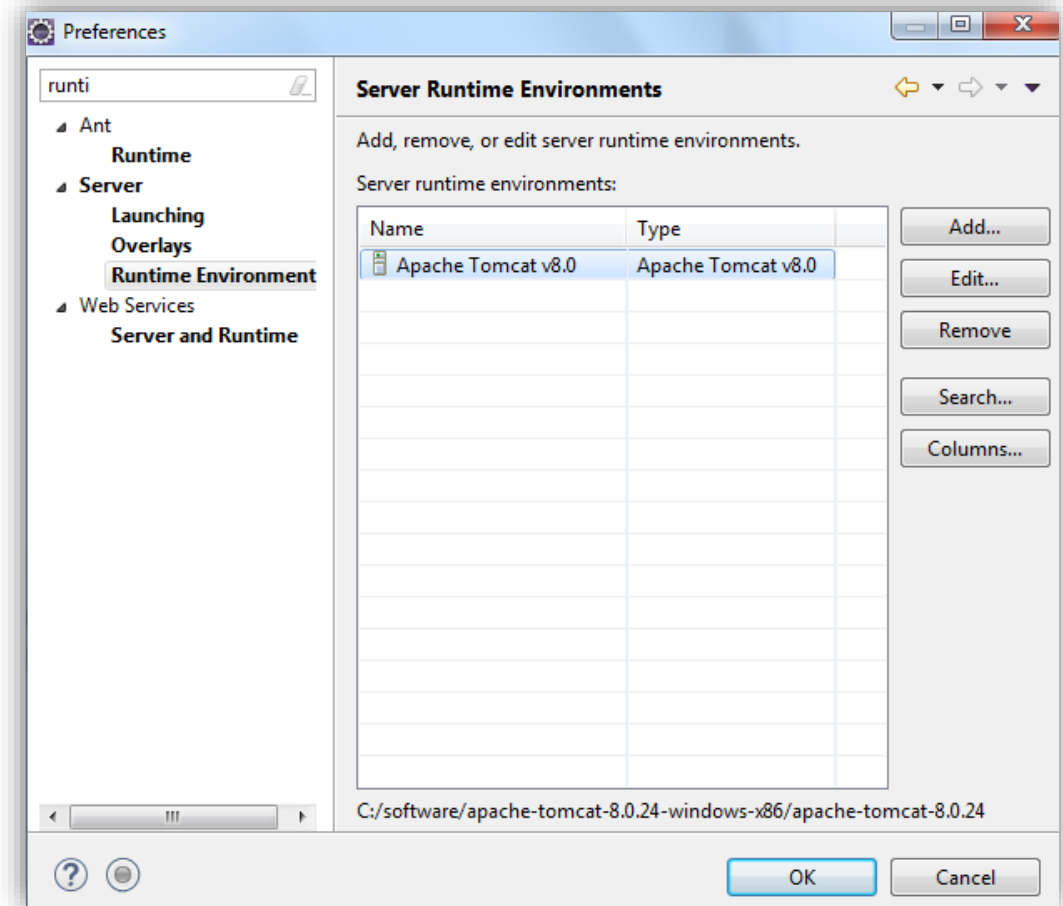
- Data Tools Platform
- Git integration for Eclipse
- Eclipse Java Development Tools
- Eclipse Java EE Developer Tools
- JavaScript Development Tools
- Maven Integration for Eclipse
- Mylyn Task List
- Eclipse Plug-in Development Environment
- Remote System Explorer
- Code Recommenders Tools for Java Developers
- Eclipse XML Editors and Tools



# Environnement de développement

## Étape 3: Eclipse

- Configuration
  - [serveur d'applications](#)
    - Il faut se positionner sur le menu *Server - Runtime Environment*
    - Cliquez sur le bouton *Add*
    - Sélectionnez le serveur *Apache Tomcat 8.0* dans le nœud Apache et cliquez sur le bouton *Next*
    - En cliquant sur le bouton *Browse* sélectionnez le répertoire racine de Tomcat (sélectionnez le bon JRE)
    - Validez et cliquez sur *Finish*

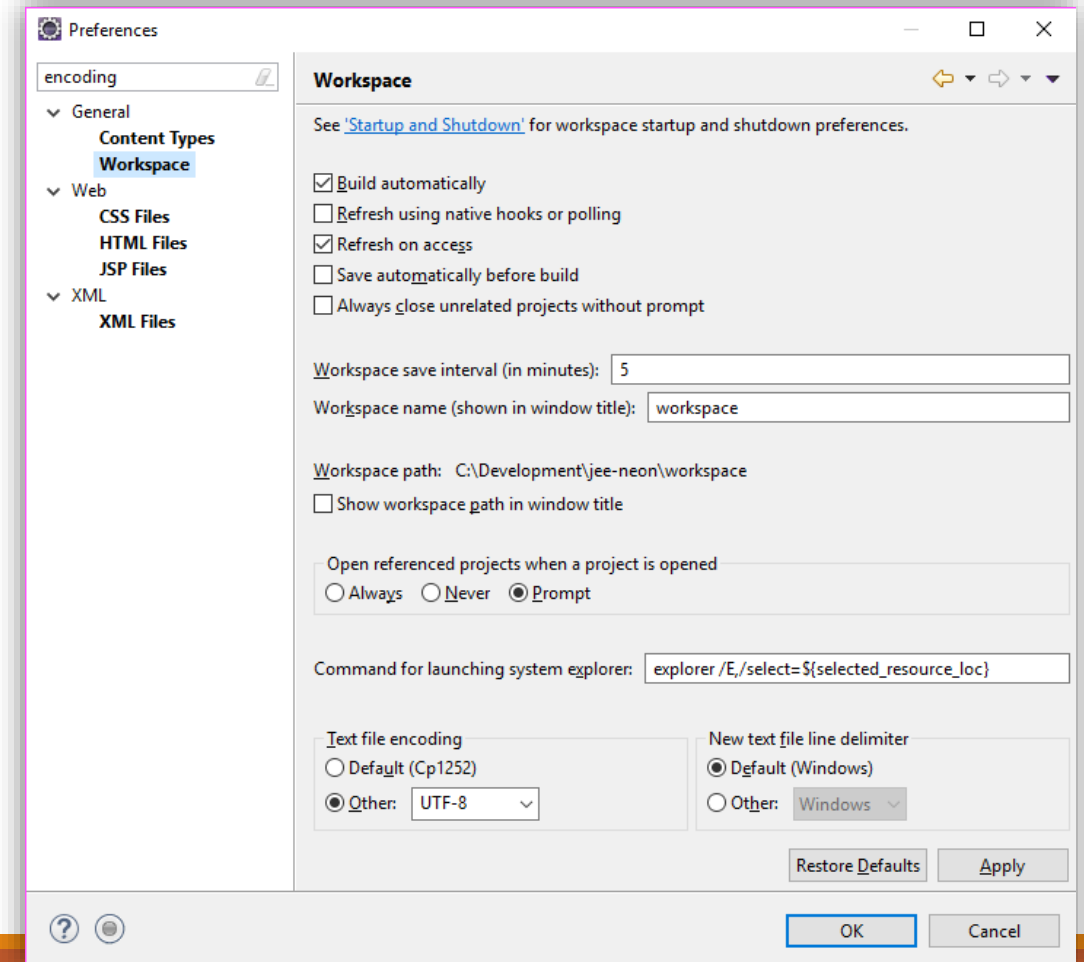




# Environnement de développement

## Étape 3: Eclipse

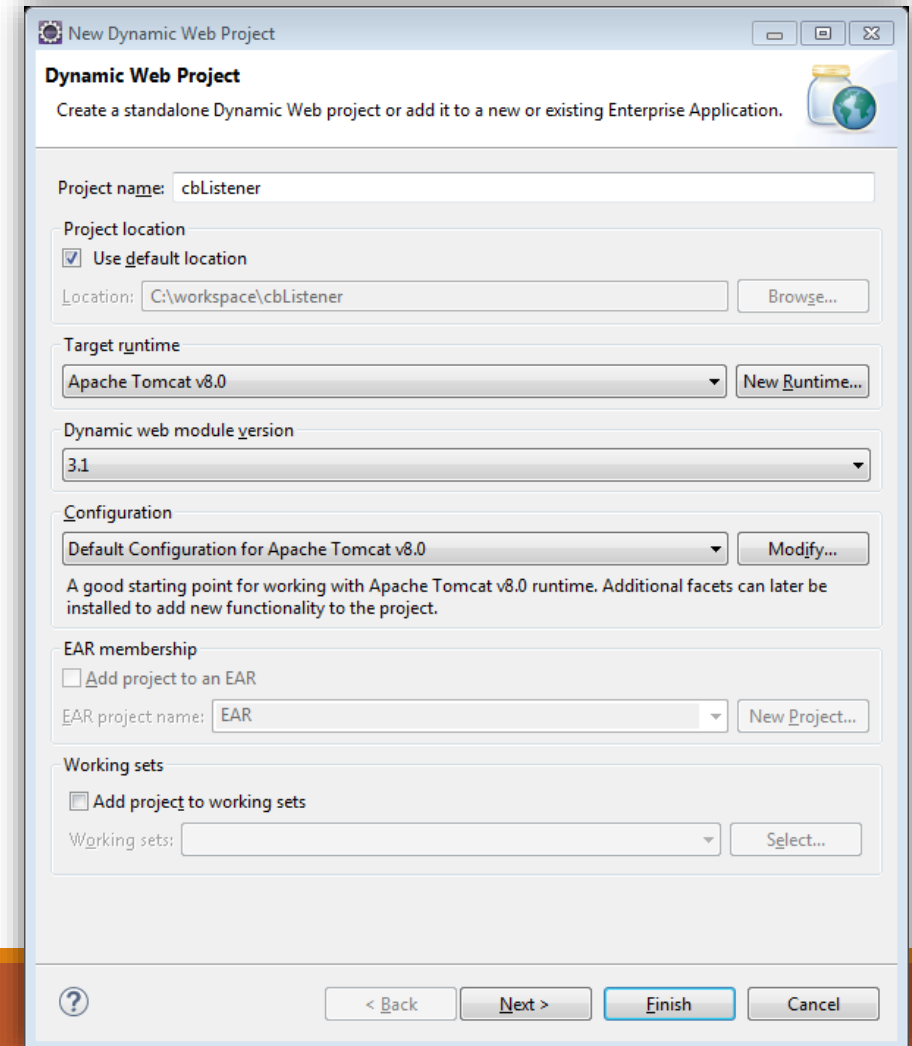
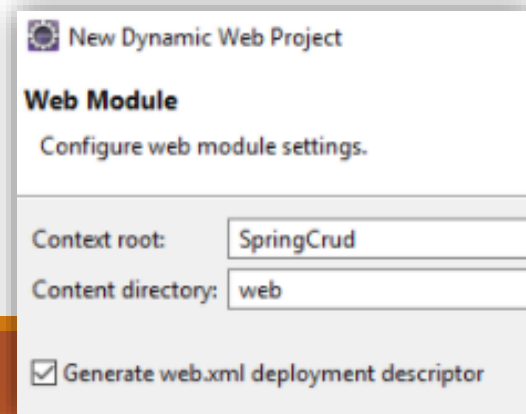
- Configuration
  - Encodage
    - Cliquez successivement sur les menus suivants
      - *General - Workspace*
      - *Web - CSS Files*
      - *Web - HTML Files*
      - *Web - JSP Files*
      - *XML - XML*
    - Modifiez toutes les zones de saisie nécessaires pour utiliser uniquement l'*UTF-8*



# Environnement de développement

## Étape 4: Création d'un projet d'application Web

- Cliquez sur le menu *File - New - Dynamic Web Project*
- Donnez un nom à votre projet dans la zone *Project name*
- L'environnement d'exécution (*Target runtime*) doit être le serveur d'applications paramétré précédemment à savoir Apache Tomcat v8.0.
- La version de la spécification des servlets doit être positionnée à 3.1 (*Dynamic web module version*).
- Cliquez deux fois sur *Next* pour arriver sur l'écran *Web Module*
- Cochez *Generate web.xml deployment descriptor*
- Cliquez sur *Finish* pour terminer.



# Environnement de développement

---

## Étape 4: Création d'un projet d'application Web

- La structure classique de ce type de projet est la suivante:

