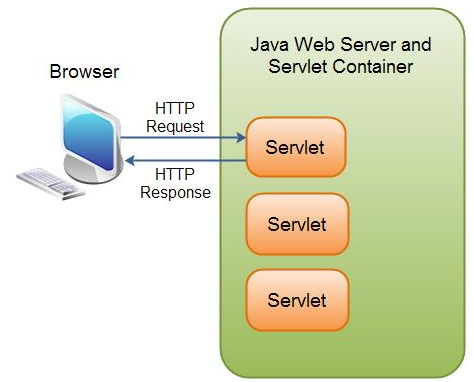
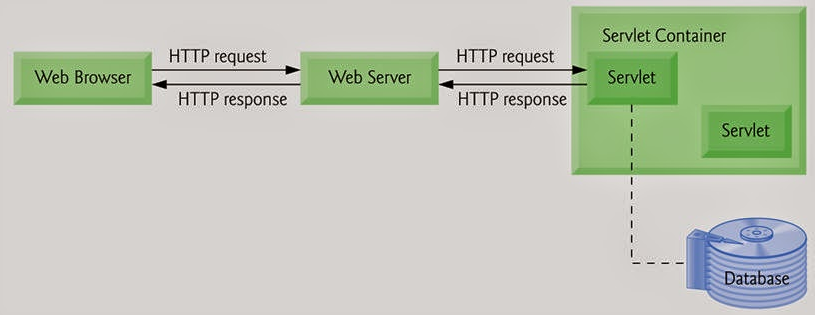
**INTRODUCTION**

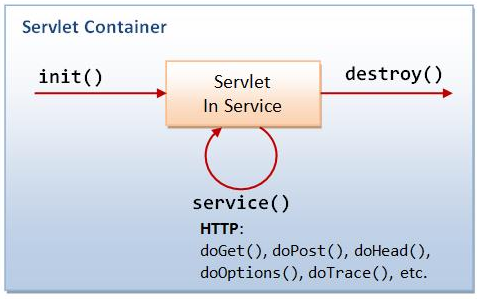
Le principe est d’avoir un Containeur connecté à un serveur

Les requêtes http sont statiques. Il n’y a pas de dynamisme garantie. Les servlets s’en chargeront



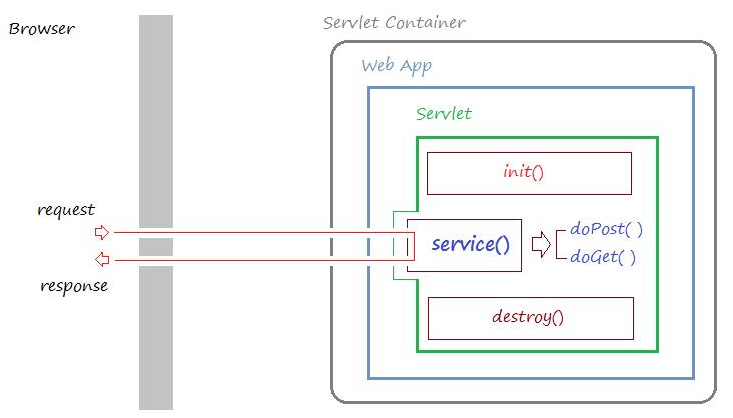


**ATTENTION : NE JAMAIS SURCHARGER LA METHODE SERVICE() DE LA SERVLET**



# LES SERVLETS CONTAINER PERMETTENT D’UTILISER LES SERVLET COMME TOMCAT

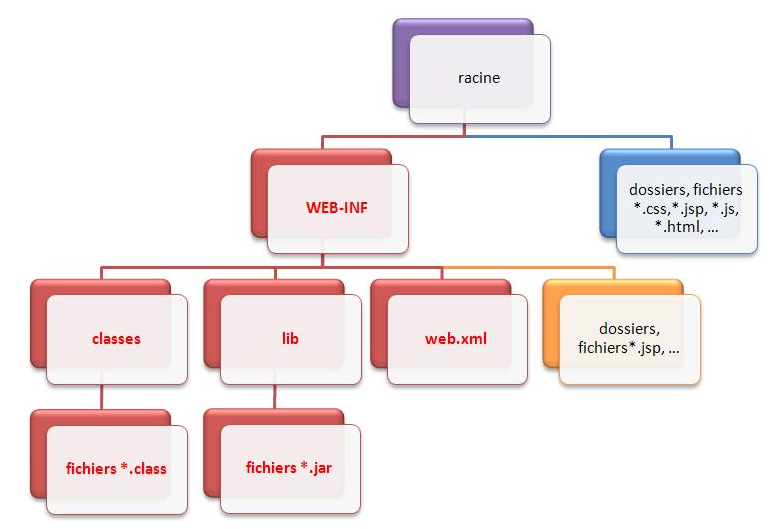
Pour chaque requête, un thread se lance. Ils vont travailler sur une même instance de SERVLET



# Structure d’une application WEB 🡺 Web App

IL n’y a pas de main dans une application JAVA EE, les servlets seront les points d’entrées.

Toutes les ressources rouges seront inaccessibles via le server http mais qu on pourra renvoyé via nos servlets (redirection etc..).



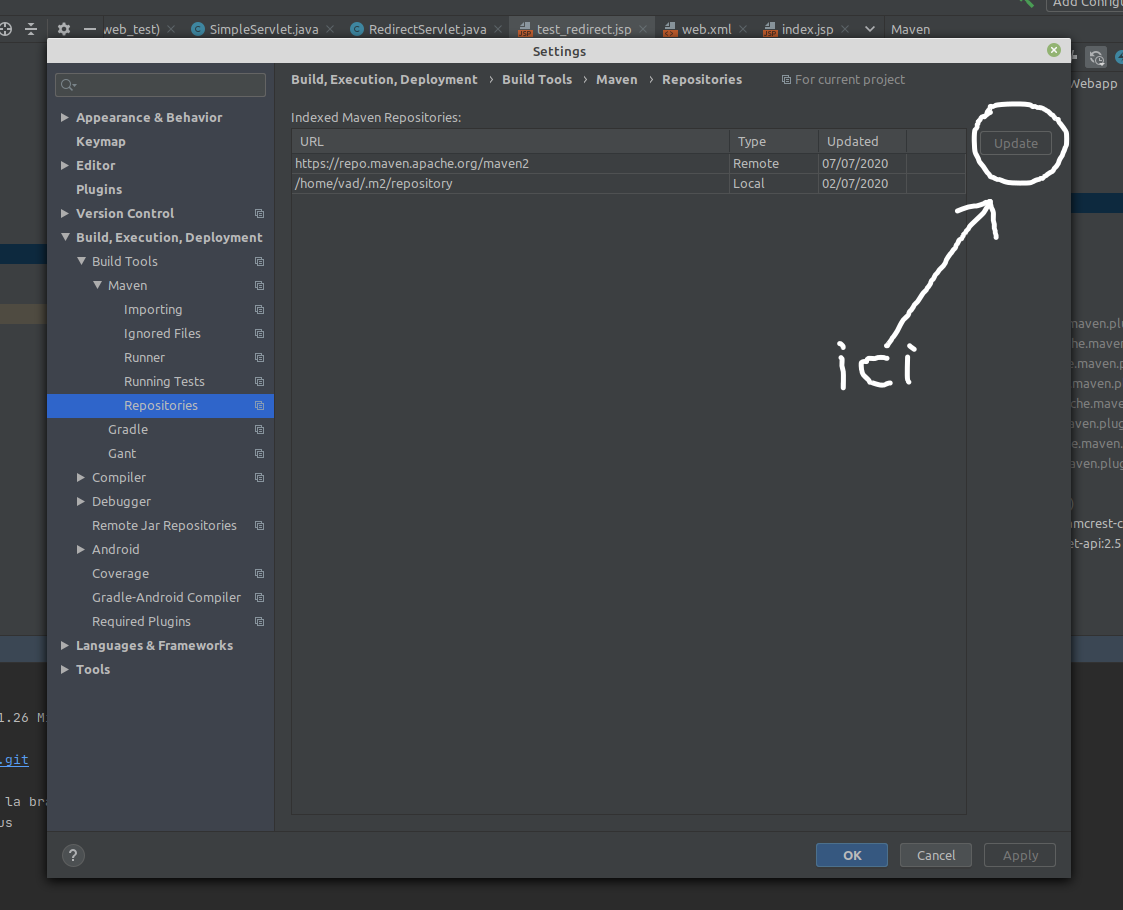
# SERVLET MAPPING

1 servlet => 1 mapping et ce mapping se déclare dans le fichier web.xml

|  |
| --- |
| <!DOCTYPE web-app PUBLIC  "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"  "[http://java.sun.com/dtd/web-app\_2\_3.dtd"](http://java.sun.com/dtd/web-app_2_3.dtd%22) > <web-app>   <display-name>Archetype Created Web Application</display-name>   <servlet>     <servlet-name>simple-servlet</servlet-name>     <servlet-class>nompackage.SimpleServletClassJava</servlet-class>   </servlet>   <servlet-mapping>     <servlet-name>simple-servlet</servlet-name>     <url-pattern>/simple</url-pattern>   </servlet-mapping> </web-app> |

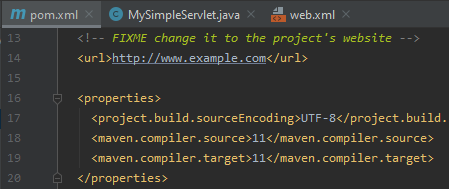
ETAPES :

1. Installer Tomcat : <https://www.liquidweb.com/kb/installing-tomcat-9-on-windows/#:~:text=Open%20your%20browser%20and%20head,the%20Windows%20Service%20Installer%20automatically.>
2. Installer INTELLIJ meme en verison community
3. Mettre à jour MAVEN (au plus récent)



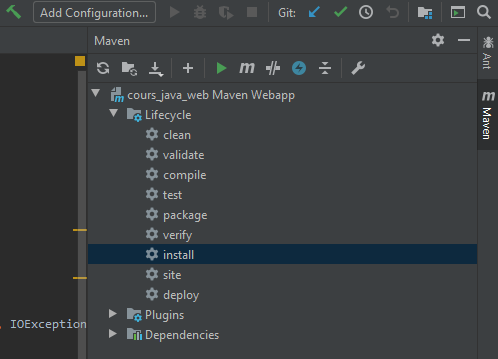
Important!!

🡺il faut que dans POM.XML la version java utilisée corresponde a la dépendance



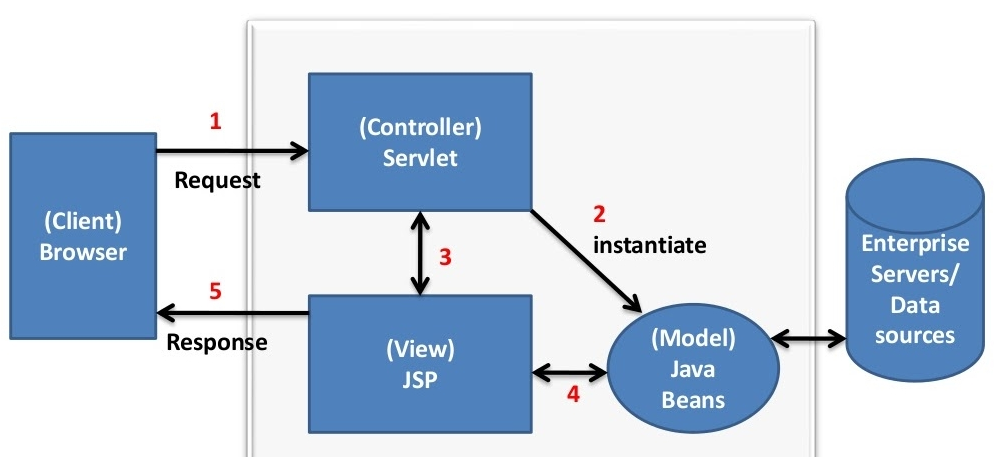
1. Voir la config dans le dossier CEDRIC 🡺 C:\Users\Mia Info\Documents\IPI\_COURS\JAVA\_WEB\COURS\CEDRIC\_lancement\_projet\_Maven
2. Mettre le fichier .WAR dans tomcat/WEBAPPS

IMPORTANT !! pour chaque modification => redéployer dans tomcat/WEBAPPS



Puis copier le fichier .war et écraser l’ancienne version dans tomcat/WEBAPPS

A l’exécution de la requête



Une JSP est pour rappel l’équivalent du razor en ASP

Une Servlet est une compilation de la JSP. Il y a 4 niveaux de contexte dans la Servlet

1. Contexte page (peu utile) => uniquement dans la JSP
2. Contexte requête (http)
3. Contexte session utilisateur
4. Contexte application

## Affichage des bean dans la JSP

**SCRIPTLETS**

<%= session.getAttribute("pseudo") %>

**EXPRESSION LANGUAGE** (librairie **JSTL** à installer : création de librairie de tag)

Privilégier l’Expression Language => ${pseudo} qui va chercher directement l’attribut pseudo dans les contextes équivalents (ici Session)

Utilisation de JSTL

1. Mettre ceci en entête de jsp 🡪

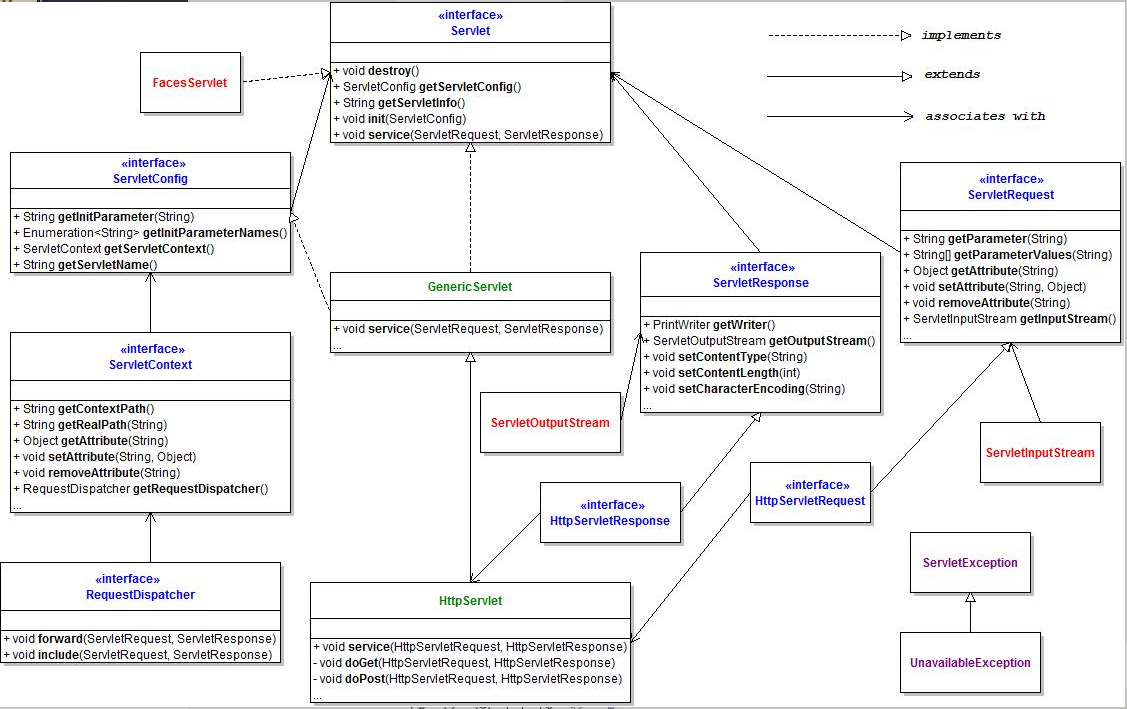
**<%@page contentType="text/html" pageEncoding="UTF-8"%>**

**<**[**%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**](mailto:%25@taglib%20uri=%22http://java.sun.com/jsp/jstl/core%22%20prefix=%22c%22%20%25)**>**

**<%@page isELIgnored="false" %>**

1. Mettre cette dépendence dans la POM.xml fichier de paramétrage Maven

**<dependency>  
      <groupId>javax.servlet</groupId>  
      <artifactId>jstl</artifactId>  
      <version>1.2</version>  
    </dependency>**

Resumé de la doc de la servlet

EXERCICE CEDRIC

|  |
| --- |
| Création d'un nouveau projet "whatevernameyoulike" Java sous IntelliJ Create Maven project from archetype : org.apache.maven.archetypes:maven-archetype-webapp JDK 11    Ajout du .gitignore    Maven task / install Afin de générer le .war (contenu dans package target) Et déploiement de celui-ci sous Tomcat =>le copier dans Apache\tomcat\Webapps  => vérifier que l'app est OK    Modifications du pom.xml Changement de la version de Java (par la 11)     <properties>         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>         <maven.compiler.source>11</maven.compiler.source>            <maven.compiler.target>11</maven.compiler.target>       </properties> Ajout d'une dépendence pour JEE     <dependency>       <groupId>javax</groupId>       <artifactId>javaee-web-api</artifactId>       <version>8.0.1</version>     </dependency>     1. Création d'un répertoire pour les sources java     /src/main/java mark directory as source source root (contextual menu)      1. Creation d'un package pour les servlets fr.cviaud.javawebcourse.controller      1. Ajout de la librairie servlets au projet via une librairie globale, pointant sur     /home/vad/tools/apache-tomcat-9.0.36/lib/servlet-api.jar      1. Codage d'une première servlet qui renvoie une page HTML très simple     Déclaration de cette servlet dans web.xml   <servlet>     <servlet-name>simple-servlet</servlet-name>     <servlet-class>fr.cviaud.javawebcourse.controller.SimpleServlet</servlet-class>   </servlet>   <servlet-mapping>     <servlet-name>simple-servlet</servlet-name>     <url-pattern>/simple</url-pattern>   </servlet-mapping>    Codage d'une servlet simple Affichant l'heure, via println() sur le flux "response"    Récupérer d'un paramètre depuis la requête Création d'une page formulaire, pour demander un pseudo Récupération de ce pseudo dans la servlet, et affichage dans la réponse    Redirection servlet --> jsp view    Exploitation de données de contexte depuis la jsp 1. Affichage de "bonjour <pseudo>   lecture du pseudo dans la requete depuis la jsp     <%= request.getParameter("nom\_param") %> 2. Affichage de l'heure de connexion   Calcul dans la servlet   stockage dans un attribut de la requete     req.setAttribute(LOGIN\_TIME\_PARAMETER\_NAME, loginTime);   Lecture de l'attribut depuis la jsp     <%= request.getAttribute("login\_time") %>    Stockage d'infos dans la session utilisateur (pseudo)     final HttpSession session = req.getSession();     session.setAttribute(PSEUDO\_PARAMETER\_NAME, pseudo); Exploitation de ces données depuis la jsp     <%= session.getAttribute("pseudo") %> |