# TP1 – Servlets HTTP (Tomcat 9, `javax.servlet.*`)
# Application GreetingServlet

Abdessabour Ahzab
Dadda Mohamed laghdaf

Année universitaire 2025–2026

## Table des matières

# Introduction générale

L'objectif de ce TP est de manipuler le protocole HTTP en utilisant la plateforme Java EE (JEE) et plus particulièrement les servlets HTTP. On met en place une petite application web composée :

— d'un formulaire HTML (`greetings.html`) pour saisir un nom,
— d'une servlet `GreetingServlet` qui traite la requête,
— d'un descripteur de déploiement `web.xml` qui relie l'URL `/hello` à la servlet.

Le but fonctionnel est simple : l'utilisateur saisit son nom, soumet le formulaire, et reçoit une page qui lui souhaite la bienvenue et lui annonce un gain virtuel à la loterie. :contentReference[oaicite :0]index=0

# 1 Environnement et structure du projet

## 1.1 Prérequis

L'application a été développée et exécutée avec la configuration suivante :

— JDK 11+ (tests réalisés avec JDK 21 en ciblant Java 11),
— Maven 3.x pour la gestion du projet,
— Tomcat 9.0.x comme conteneur web compatible `javax.servlet.*`,
— IDE Eclipse EE (ou IntelliJ IDEA) pour l'édition et le déploiement.

## 1.2 Structure Maven du projet

Le projet Maven `HttpServletLab-TP1` suit l'arborescence standard :

```
HttpServletLab-TP1/
 pom.xml
 src/
   main/
      java/
        exple1/
           GreetingServlet.java
      webapp/
         greetings.html
         WEB-INF/
            web.xml
 docs/
    screenshots/
```

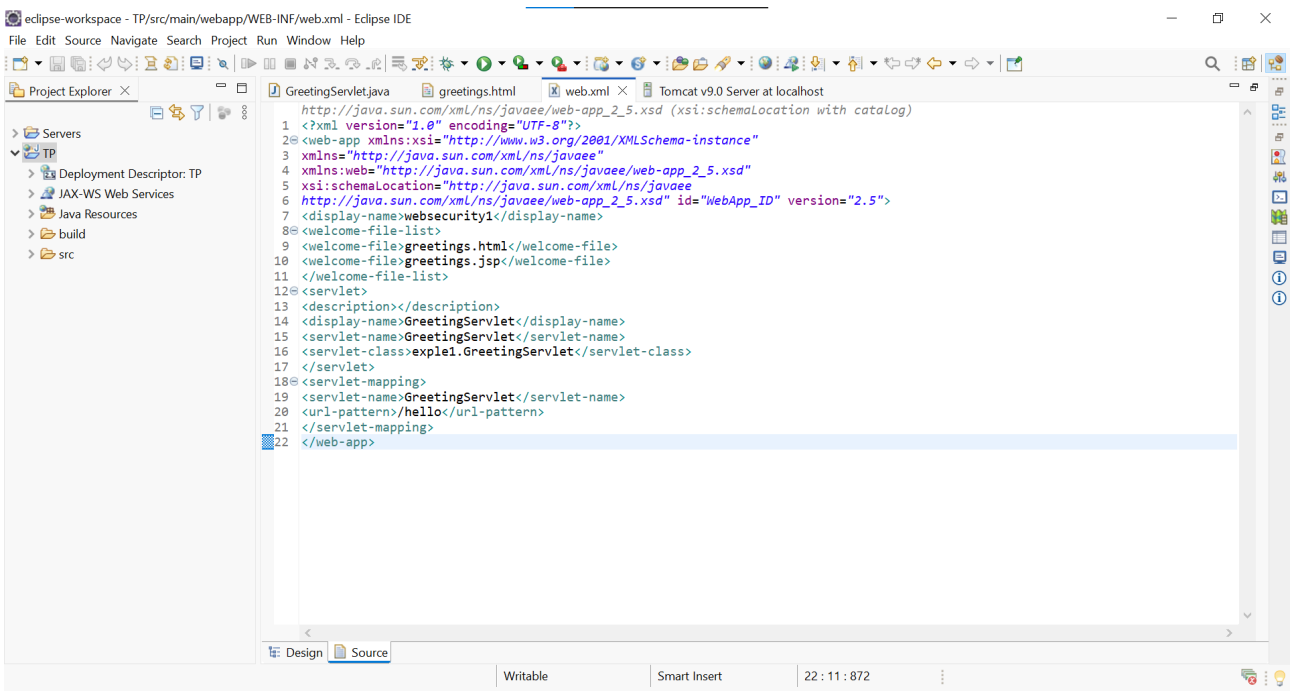


FIGURE 1 – Structure du projet HttpServletLab-TP1 dans l'IDE

# 2 Formulaire HTML `greetings.html`

## 2.1 Objectif

Afficher un formulaire HTML permettant à l'utilisateur de saisir son nom et d'envoyer une requête HTTP (`POST`) vers l'URL `/hello`.

## 2.2 Description

Le fichier `greetings.html` contient :
— un titre pour la page,
— un formulaire avec un champ texte `nom`,
— un bouton de soumission.
Extrait du code HTML : :contentReference[oaicite :1]index=1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Welcome to this amazing web application</title>
</head>
<body>
<CENTER><H1>Tenter votre chance a cette lotterie virtuelle!</H1></CENTER>
<FORM ACTION="./hello" METHOD="post">
  Votre nom svp: <INPUT TYPE="TEXT" NAME="nom"><BR><BR>
  <CENTER><INPUT TYPE="SUBMIT" NAME="Submit Query"></CENTER>
</FORM>
</body>
</html>
```

## 2.3 Interface utilisateur

La page d'accueil affiche le formulaire centré avec un champ « Votre nom svp » et un bouton « Submit Query ».


FIGURE 2 – Page d'accueil greetings.html sur localhost

### 2.4 Résumé

Cette première étape met en place la couche présentation côté client. Aucun traitement serveur n'est encore réalisé, le formulaire se contente d'envoyer les données vers l'URL `/hello`.

## 3 Servlet `GreetingServlet`

### 3.1 Objectif

Implémenter une servlet HTTP qui reçoit le paramètre `nom`, construit une réponse HTML personnalisée et renvoie un montant de gain aléatoire.

### 3.2 Description

La classe `GreetingServlet` :
— hérite de `HttpServlet`,
— redéfinit `doGet()` et `doPost()`,
— lit le paramètre `nom` dans la requête,
— met le nom en majuscules et l'insère dans le message,
— génère un nombre aléatoire avec `Math.random()` pour simuler le gain.
Extrait de code simplifié : :contentReference[oaicite :2]index=2

```
package exple1;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class GreetingServlet extends HttpServlet {

  @Override
```

```
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    String nomPrenom = "Anonymous";
    String votreNom = request.getParameter("nom");
    if (votreNom != null) {
      nomPrenom = votreNom.toUpperCase();
    }

    out.println("<html><head><title>Greetings Servlet</title></head>");
    out.println("<body bgcolor=\"#FDF5E6\">");
    out.println("<h1>Greetings " + nomPrenom + "!</h1>");
    out.println("<p>Vous avez gagné : " + (Math.random()*10)
                + " millions de dollars !</p>");
    out.println("</body></html>");
  }

  @Override
  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
      throws ServletException, IOException {
    doGet(request, response);
  }
}
```

### 3.3 Résumé

Cette servlet représente la couche métier très simple du TP : elle illustre l'accès aux paramètres HTTP, la génération de HTML côté serveur et la réutilisation de `doGet()` dans `doPost()`.

## 4 Descripteur de déploiement `web.xml`

### 4.1 Objectif

Configurer l'application web pour que l'URL `/hello` soit prise en charge par la classe `exple1.GreetingServlet` et définir la page d'accueil `greetings.html`. :contentReference[oaicite :3]index=3

### 4.2 Extrait de configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation=
          "http://java.sun.com/xml/ns/javaee
```

```
          http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
        version="2.5">

  <display-name>HttpServletLab-TP1</display-name>

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

  <servlet>
    <servlet-name>GreetingServlet</servlet-name>
    <servlet-class>exple1.GreetingServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>GreetingServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>
</web-app>
```

### 4.3 Résumé

Le fichier `web.xml` relie la présentation (URL) et la logique Java : à chaque requête sur `/hello`, le conteneur Tomcat instancie ou réutilise `GreetingServlet` pour produire la réponse.

## 5 Déploiement et tests sur Tomcat 9

### 5.1 Build et déploiement

Le projet est empaqueté en `war` via Maven puis déployé sur Tomcat 9 :
— configuration d'un *Run/Debug Configuration* « Tomcat Server »,
— ajout de l'artifact « Web Application : Exploded »,
— application context : `/HttpServletLab-TP1` (par exemple),
— démarrage du serveur depuis l'IDE.
Une fois Tomcat lancé :
— si l'application est déployée comme `ROOT.war` : `http://localhost:8080/`,
— sinon (cas Maven standard) : `http://localhost:8080/HttpServletLab-TP1/`.

### 5.2 Test de la page `greetings.html`

On ouvre l'URL complète de la page d'accueil, par exemple :
— `http://localhost:8080/HttpServletLab-TP1/greetings.html`
La page affiche le formulaire de saisie du nom.

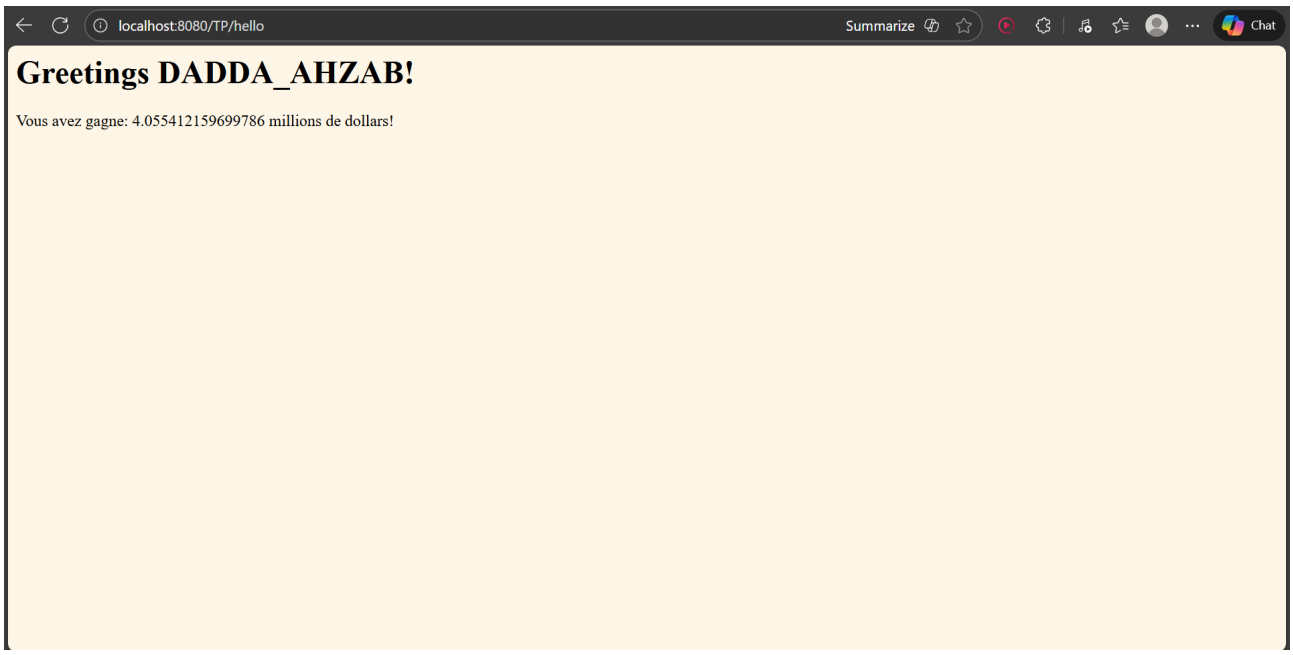


FIGURE 3 – Formulaire greetings.html chargé depuis Tomcat

### 5.3 Test de la servlet `/hello`

Après avoir saisi un nom et cliqué sur « Submit Query », le navigateur envoie une requête `POST` vers `/hello`. Tomcat appelle alors `GreetingServlet.doPost()`, qui délègue à `doGet()`.

La réponse contient :

— un titre « Greetings NOM_EN_MAJUSCULES »,
— un montant de gain aléatoire entre 0 et 10 millions.

## 6 Extensions possibles

Le sujet du TP propose plusieurs extensions pour approfondir les servlets JEE : :contentReference[oaicite :4]index=4

— connecter l'application à une base de données MySQL via JDBC pour stocker et consulter les gains,
— sécuriser l'accès à la servlet en fonction du rôle « tomcat »,
— utiliser des filtres (`Filter`) pour bloquer les noms présents dans une *blacklist* configurée dans l'application,
— remplacer la page HTML par une page JSP,
— implémenter d'autres méthodes de `HttpServlet` (gestion des erreurs, etc.),
— configurer des pages d'erreur dédiées (codes 4xx, 5xx) dans `web.xml`.

## Conclusion générale

Ce TP1 a permis de mettre en place une première application web Java EE basée sur les servlets HTTP. On a couvert :

— la structure d'un projet web JEE (HTML, servlets, `web.xml`),
— le cycle de vie d'une servlet et le mapping d'URL,
— le déploiement d'une application sur Tomcat 9,
— le test de la chaîne complète : formulaire → servlet → réponse HTML dynamique.

Cette base sera réutilisée et enrichie dans les TP suivants (sécurité, JSP, bases de données, WebRTC, etc.).