

Institut National des Postes et Télécommunications

Rapport de Travaux Pratiques

TP1 : Les Servlets HTTP et Architecture JEE

Réalisé par :

Youssef Dadda

Akram El Mamoun

Asmaa Bagri

20 novembre 2025

Table des matières

1	Introduction	2
2	Mise en place et Configuration	2
3	Extension 1 : Technologie JSP (Vue)	2
4	Extension 2 : Filtrage et Sécurité	3
5	Extension 3 : Persistance avec JDBC et Docker	4
5.1	Architecture Mise en Place	4
5.2	Implémentation et Résultats	5
6	Conclusion	6

1 Introduction

L'objectif de ce TP est de manipuler le protocole HTTP en utilisant la plateforme JEE, spécifiquement les Servlets. Nous avons commencé par la configuration de l'environnement (Eclipse, Tomcat), pour ensuite développer une application web dynamique. Enfin, nous avons réalisé plusieurs extensions techniques majeures : la séparation Vue/Contrôleur avec JSP, la sécurisation via des Filtres, et la persistance des données via JDBC et Docker.

2 Mise en place et Configuration

La première étape a consisté à configurer un projet "Dynamic Web Project" sous Eclipse avec le serveur Apache Tomcat 9.0. Nous avons rencontré et résolu des problèmes liés au *Deployment Assembly* pour assurer que les classes Java soient bien déployées dans le dossier WEB-INF/classes.

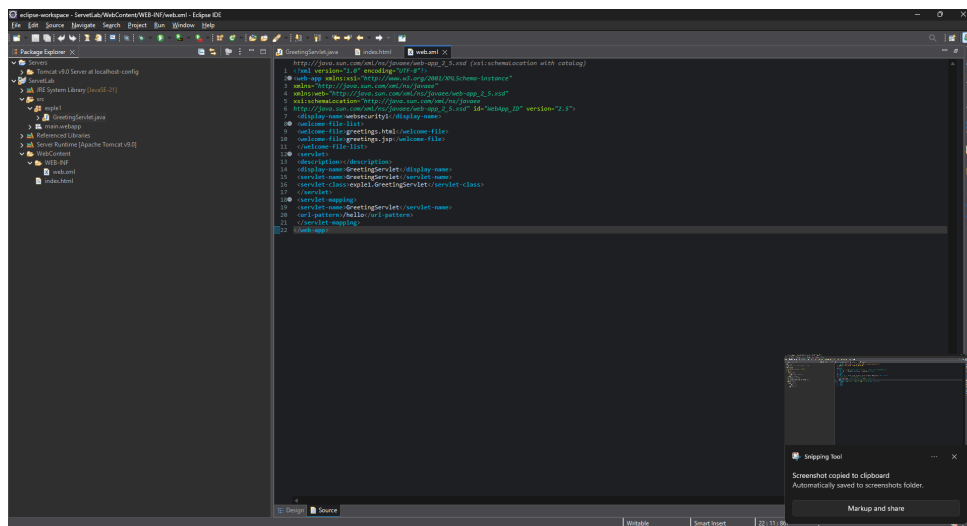


FIGURE 1 – Configuration du descripteur de déploiement web.xml (Version 2.5)

3 Extension 1 : Technologie JSP (Vue)

Pour respecter le modèle MVC (Modèle-Vue-Contrôleur), nous avons remplacé la génération de HTML via `out.println` dans la Servlet par une redirection vers une page JSP.

Nous avons dû assurer la correspondance exacte des clés entre `request.setAttribute` dans la Servlet et `request.getAttribute` dans la JSP. La figure ci-dessous montre une phase de débogage où les clés ne correspondaient pas (valeur null), corrigée par la suite.

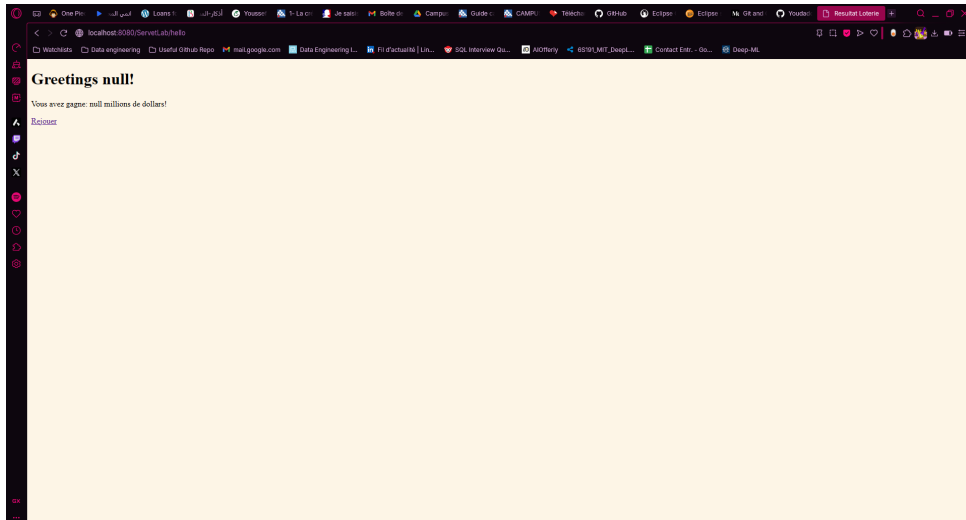


FIGURE 2 – Débogage de la transmission des attributs vers la JSP

Une fois corrigé, l’affichage dynamique fonctionne parfaitement :

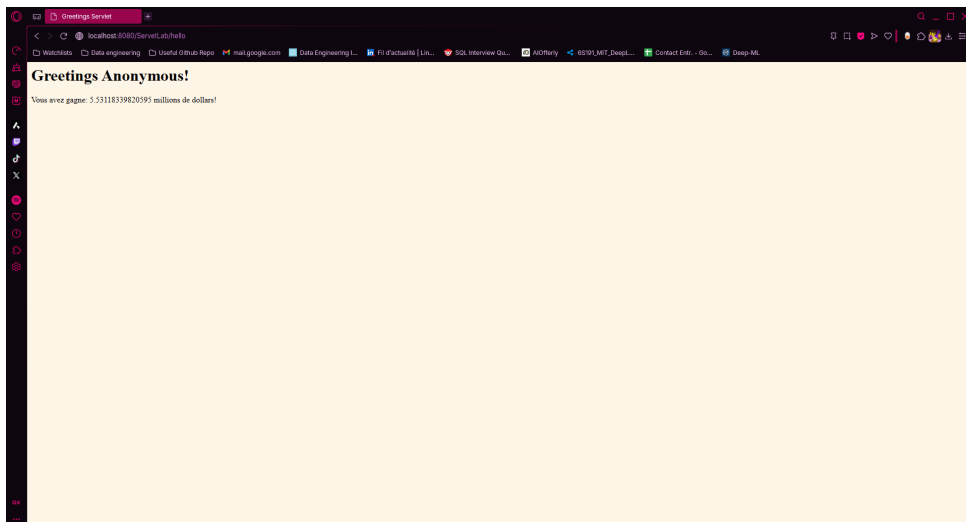


FIGURE 3 – Affichage réussi du résultat via la JSP

4 Extension 2 : Filtrage et Sécurité

Nous avons implémenté un `javax.servlet.Filter` pour intercepter les requêtes avant qu’elles n’atteignent la Servlet. L’objectif était de mettre en place une "Blacklist". Si un utilisateur tente de se connecter avec le nom "hacker", l’accès lui est refusé.

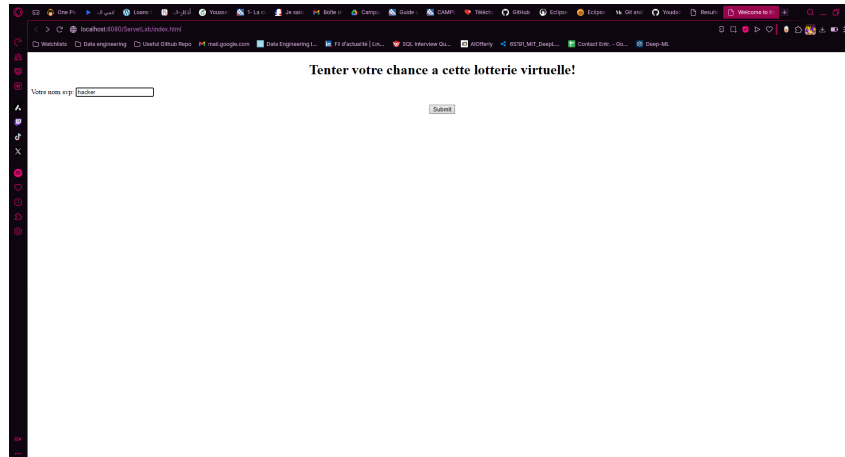


FIGURE 4 – Tentative d'intrusion avec le nom "hacker"

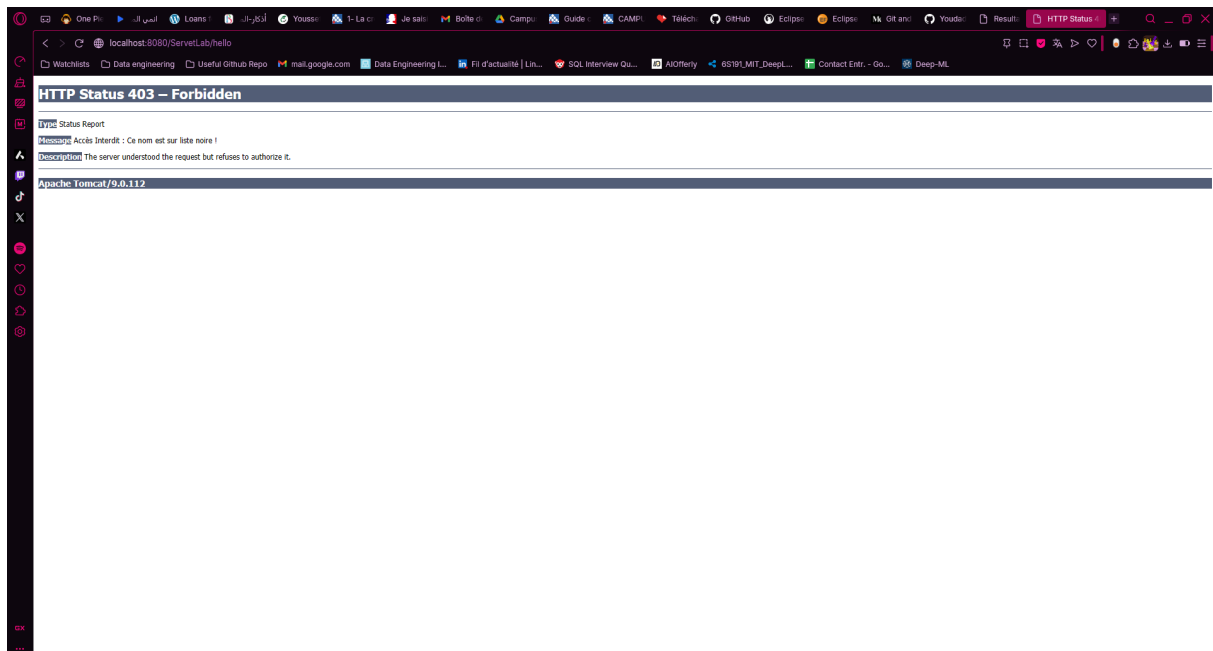


FIGURE 5 – Interception par le Filtre et renvoi d'une erreur 403 Forbidden

5 Extension 3 : Persistance avec JDBC et Docker

Dans le cadre de notre spécialisation Data Engineering, nous avons choisi d'utiliser **PostgreSQL** conteneurisé via **Docker** au lieu d'une installation locale MySQL standard.

5.1 Architecture Mise en Place

- **SGBD** : PostgreSQL (Image Docker officielle)
- **Connecteur** : Pilote JDBC PostgreSQL ajouté au WEB-INF/lib
- **Code** : Utilisation de `PreparedStatement` pour éviter les injections SQL.

5.2 Implémentation et Résultats

Le code de la Servlet a été modifié pour ouvrir une connexion JDBC, insérer les données (Nom et Gain), puis fermer la connexion avant de rediriger vers la JSP.

```

14 // 1. LIREUNE MÉTHODE
15 String votreNom = request.getParameter("nom");
16 String nomPrenom = "Anonymous";
17
18 // Gestion de la session par défaut si le nom n'est pas vide
19 if (votreNom != null && !votreNom.isEmpty()) {
20     nomPrenom = votreNom.toUpperCase();
21 }
22
23 // Calcul du gain aléatoire
24 double gain = Math.random() * 10;
25
26
27 // --- 2. PERSISTENCE (PostgreSQL) ---
28 Connection conn = null;
29 PreparedStatement ps = null;
30
31 try {
32     // A. Charger le driver PostgreSQL
33     Class.forName("org.postgresql.Driver");
34
35     // B. Se connecter à la base de données
36     // Remarque : remplacez ci-dessous votre_nom_de_passer par vos vrais identifiants
37     String dbUser = "postgres";
38
39     // C. Utilisez le mot de passe que vous avez défini dans le fichier de configuration
40     String dbPassword = "root";
41
42     // D. URL, remplacez localhost par le nom de votre serveur PostgreSQL
43     String url = "jdbc:postgresql://localhost:5432/loterie_db";
44
45     conn = DriverManager.getConnection(url, dbUser, dbPassword);
46
47     // E. Préparez la requête SQL
48     // Note : Remplacez ci-dessous les valeurs "nom_joueur" et "montant_gain" par les données
49     String sql = "INSERT INTO historique (nom_joueur, montant_gain) VALUES (?, ?)";
50     ps = conn.prepareStatement(sql);
51
52     // F. Remplissez les paramètres (?)
53     ps.setString(1, nomPrenom);
54     ps.setDouble(2, gain);
55
56     // G. Exécutez l'insertion
57     ps.executeUpdate();
58     System.out.println("Succès : Données enregistrées dans PostgreSQL pour " + nomPrenom);
59
60 } catch (ClassNotFoundException e) {
61     e.printStackTrace();
62     System.err.println("Erreur : Driver PostgreSQL introuvable dans WEB-INF/lib");
63 } catch (SQLException e) {
64     e.printStackTrace();
65     System.err.println("Erreur SQL : Vérifiez l'URL, le user/password ou si la table existe.");
66 } finally {
67     // H. Fermez proprement les ressources
68     try { if (ps != null) ps.close(); } catch (SQLException e) {}
69     try { if (conn != null) conn.close(); } catch (SQLException e) {}
70 }
71
72

```

Console logs (Tomcat v9.0 Server at localhost):

```

Nov 29, 2025 3:16:48 PM org.apache.catalina.core.AprLifecycleListener lifecycleEvent
INFO: Configuring a JMX-enabled connector (FailFast, suspendOnFailure)
Nov 29, 2025 3:16:48 PM org.apache.catalina.core.AprLifecycleListener lifecycleEvent
INFO: Connector is initialized with success (OpenSSL 3.0.14 4 Jun 2024)
Nov 29, 2025 3:16:48 PM org.apache.coyote.AbstractProtocol init
INFO: Initializing the protocol handler [http-nio-8080]
Nov 29, 2025 3:16:48 PM org.apache.catalina.startup.Catalina load
INFO: Initialization of the server is complete [2007] milliseconds
Nov 29, 2025 3:16:48 PM org.apache.catalina.core.StandardService startInternal
INFO: Starting the service [localhost]
Nov 29, 2025 3:16:48 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting the engine [localhost]
Nov 29, 2025 3:16:48 PM org.apache.catalina.core.StandardHost startInternal
INFO: Starting the host [localhost]
Nov 29, 2025 3:16:48 PM org.apache.catalina.core.StandardContext startInternal
INFO: Starting the context [/]
Nov 29, 2025 3:16:48 PM org.apache.catalina.startup.Catalina start
INFO: The startup of the server is complete [2007] milliseconds
Succès : Données enregistrées dans PostgreSQL pour YOUSSEF

```

FIGURE 6 – Code JDBC final et logs console confirmant l'insertion

Nous avons vérifié la persistance des données directement à l'intérieur du conteneur Docker via la commande psql.

```

CREATE TABLE
loterie_db=# select * from historique;
 id | nom_joueur | montant_gain | date_jeu
-----+-----+-----+-----
(0 rows)

loterie_db=# select * from historique;
 id | nom_joueur | montant_gain | date_jeu
-----+-----+-----+-----
(0 rows)

loterie_db=# exit

C:\Users\user>docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                    NAMES
c41824e21000   postgres  "docker-entrypoint.s..." 2 minutes ago  Up 2 minutes  0.0.0.0:5432->5432/tcp   postgres-lab

C:\Users\user>docker exec -it postgres-lab psql -U postgres
psql (18.1 (Debian 18.1-1.pgdg13+2))
Type "help" for help.

postgres=# \c loterie_db
You are now connected to database "loterie_db" as user "postgres".
loterie_db=# select * from historique;
 id | nom_joueur | montant_gain | date_jeu
-----+-----+-----+-----
 1 | YOUSSEF    | 0.4755605788104722 | 2025-11-20 15:17:00.972571
(1 row)

loterie_db=#

```

FIGURE 7 – Vérification des données dans PostgreSQL via CLI Docker

6 Conclusion

Ce TP nous a permis de maîtriser les fondamentaux du développement JEE. Nous avons appris à configurer un serveur d'application, à séparer la logique métier de la présentation (JSP), à sécuriser l'application via des filtres, et enfin à intégrer une base de données relationnelle moderne (PostgreSQL) dans un environnement conteneurisé, simulant une architecture de production réelle.