Rapport de TP

TP Servlet et JDBC
Application “Loterie”

Filière

- SCIENCES DE DONNEES -

Réalisé par :

Hatim Haddioui

Octobre 2025

SOMMAIRE

INTRODUCTION

L'objectif de ce TP est de se familiariser avec la plateforme **Java EE (Jakarta EE)** et plus particulièrement avec la création et l'exécution de **servlets**.
Nous allons créer une petite application web simulant un jeu de loterie. Lorsqu'un utilisateur saisit son nom, un gain aléatoire est généré, enregistré dans une base de données MySQL, puis affiché à l'écran avec les cinq derniers résultats.

OBJECTIFS DU TP

- Manipuler le cycle de vie d'une servlet (doGet, doPost).
- Apprendre à interagir avec une base de données MySQL via JDBC.
- Générer dynamiquement du contenu HTML depuis une servlet.
- Gérer les exceptions (erreur 500) dans une application web.
- Comprendre le déploiement sous **Apache Tomcat**.

ENVIRONNEMENT DE DEVELOPPEMENT

❑ Outil / Technologie

➢ IDE : Eclipse for Enterprise Java Developers

Eclipse for Enterprise Java Developers est un environnement de développement intégré (IDE) libre et open source, conçu pour créer des applications Java et Web. Il inclut des outils pour Java EE (Jakarta EE), JavaServer Pages (JSP), Servlets, Maven, Git et bien d'autres outils de productivité logicielle.

➢ Serveur : Apache Tomcat 9.0

Apache Tomcat 10.1.x est un serveur open source d'applications Web Java développé par la Fondation Apache. Il implémente les spécifications Jakarta Servlet 6.0, JSP 3.1, EL 5.0 et WebSocket 2.1 de Jakarta EE 10. Il permet d'exécuter des applications Java Web (fichiers .war) et requiert Java 11 ou une version supérieure.

➢ Langage : Java 23

Java 23 est une version LTS (Long-Term Support) du langage Java publiée par Oracle. Elle fournit des améliorations de performance, de sécurité et de syntaxe (comme les classes scellées et les améliorations du pattern matching), garantissant la compatibilité pour les applications d'entreprise.

➢ Base de données : MySQL

MySQL est un système de gestion de base de données relationnelle open source. Il offre de hautes performances, la compatibilité avec SQL standard, le support de transactions ACID et des fonctionnalités avancées comme la réplication et les index JSON.

- **PhpMyAdmin**

PhpMyAdmin est une application web libre et open source écrite en PHP, utilisée pour administrer des bases de données MySQL ou MariaDB directement depuis un navigateur web.

- **Pilote JDBC: mysql-connector-j-8.x.jar**

MySQL-connector-j est un pilote JDBC officiel fourni par Oracle qui permet à une application Java de se connecter à une base de données MySQL. Il est compatible avec MySQL et intègre les nouvelles fonctionnalités de performance et de sécurité du protocole MySQL.

STRUCTURE DU PROJET

Voici la description des principaux éléments du projet :

Élément	Description
src/main/java/exple1/GreetingServlet.java	Servlet principale qui gère la logique de la loterie. Elle récupère le nom du joueur, calcule un gain aléatoire, insère le résultat dans la base de données et affiche les 5 derniers gagnants.
src/main/java/exple1/BlacklistFilter.java	Filtre permettant de bloquer certains utilisateurs (par exemple, ceux dont le nom est dans une liste noire). Il intercepte les requêtes avant qu'elles atteignent la servlet.
src/main/webapp/greetings.jsp	Page d'accueil (welcome page) contenant le formulaire de saisie du nom pour participer à la loterie.
src/main/webapp/404.html	Page affichée lorsque la ressource demandée n'existe pas (erreur HTTP 404).
src/main/webapp/500.html	Page affichée lorsqu'une erreur interne du serveur se produit (erreur HTTP 500).
src/main/webapp/WEB-INF/web.xml	Fichier de configuration principal du projet web. Il définit les servlets, les filtres, les pages d'erreurs et la page d'accueil.
src/main/webapp/WEB-INF/lib/mysql-connector-j-8.0.xx.jar	Bibliothèque nécessaire pour connecter la servlet à la base de données MySQL via JDBC.

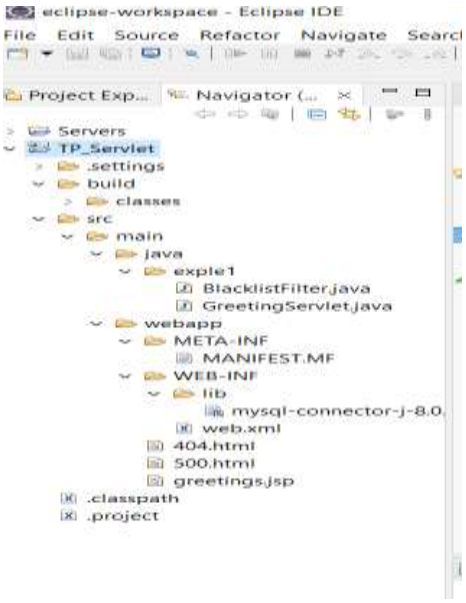

Figure 1:structure du projet

CODE COMPLET

❑ Code de la Servlet

➢ Classe GreetingServlet.java

Cette servlet traite les requêtes de l'utilisateur.

Elle récupère le nom saisi, calcule un gain aléatoire, enregistre les données dans une base MySQL via JDBC et affiche les résultats récents.

```
package exple1;
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class GreetingServlet extends HttpServlet {

   private final String URL = "jdbc:mysql://localhost:3306/loterie";
   private final String USER = "root";
   private final String PASSWORD = "";

   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();

      String votreNom = request.getParameter("nom");
      String nomPrenom = (votreNom != null) ? votreNom.toUpperCase() :
"ANONYMOUS";
      double gain = Math.random() * 10;

      try {
         Class.forName("com.mysql.cj.jdbc.Driver");
         Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

         String sql = "INSERT INTO gains (nom, montant) VALUES (?, ?)";
         PreparedStatement ps = conn.prepareStatement(sql);
         ps.setString(1, nomPrenom);
         ps.setDouble(2, gain);
         ps.executeUpdate();
```

```
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT nom, montant, date_jeu FROM
gains ORDER BY date_jeu DESC LIMIT 5");
            out.println("<html><head><title>Résultat Loterie</title>");
            out.println("<style>");
            out.println("body { font-family: Arial, sans-serif; background-color: #f9f9f9;
text-align: center; }");
            out.println("h1 { color: #2e8b57; }");
            out.println("p { font-size: 18px; }");
            out.println("ul { list-style-type: none; padding: 0; }");
            out.println("li { margin: 5px; padding: 8px; border-radius: 5px; }");
            out.println("</style>");
            out.println("</head><body>");

            out.println("<h1>Bonjour " + nomPrenom + " !</h1>");
            out.println("<p>Vous avez gagné : <strong>" + String.format("%.2f", gain) +
"</strong> millions de dollars !</p>");

            out.println("<h3>Derniers résultats :</h3>");
            out.println("<ul>");
            while (rs.next()) {
               out.println("<li>" + rs.getString("nom") + " - " + String.format("%.2f",
rs.getDouble("montant"))
                     + " millions (" + rs.getTimestamp("date_jeu") + ")</li>");
            }
            out.println("</ul>");

            out.println("</body></html>");

            rs.close();
            stmt.close();
            ps.close();
            conn.close();

        } catch (Exception e) {
            throw new ServletException(e);
        }
    }

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

- **Classe BlacklistFilter.java**

Ce filtre intercepte les requêtes avant qu'elles n'arrivent à la servlet.

Il vérifie si le nom de l'utilisateur figure dans une liste noire (blacklist).

Si le nom est bloqué, la requête est refusée ; sinon, elle est transmise à la GreetingServlet.

```
package exple1;
import java.io.IOException;
import javax.servlet.*;
import javax.servlet.http.*;

public class BlacklistFilter implements Filter {
   private String[] blacklist = {"BADNAME1", "BADNAME2"};

   public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
         throws IOException, ServletException {
      String nom = request.getParameter("nom");
      if (nom != null) {
         for (String bad : blacklist) {
            if (nom.equalsIgnoreCase(bad)) {
               response.getWriter().println("Accès refusé : nom dans la liste noire !");
               return;
            }
         }
      }
      chain.doFilter(request, response);
   }

   public void init(FilterConfig config) throws ServletException {}
   public void destroy() {}
}
```

- **Page JSP (greetings.jsp)**

La page JSP est utilisée pour afficher le formulaire d'entrée et les résultats de manière dynamique.

- **Page 404.html**

Cette page est affichée lorsque la ressource demandée n'existe pas (erreur **HTTP 404**).

- **Page 500.html**

La page 500.html est utilisée pour gérer les erreurs internes du serveur (erreur **HTTP 500**).

RESULTATS OBTENUS

❑ Page d'accueil

Formulaire de saisie du nom

L'utilisateur accède à la page d'accueil greetings.jsp et saisit son nom dans le champ prévu à cet effet.

Figure 2:Page d'accueil

❑ Page de résultat :

Résultat (message et liste des gains)

Affiche le résultat (nom et gain) ainsi que l'historique des derniers joueurs.

Figure 3:Page de résultat

❑ Page d'erreur :

404.html

Page statique affichée lorsqu'un utilisateur tente d'accéder à une ressource inexistante.

Elle améliore l'expérience utilisateur en remplaçant l'erreur brute par un message lisible.

Figure 4:Page 404

❑ Page d'erreur :

505.html

Page d'erreur interne affichée lorsqu'une exception survient côté serveur (par exemple, problème de base de données).
Elle signale que le serveur a rencontré une erreur tout en conservant une présentation simple et claire.

Figure 5:Page 500

Filtre de noms interdit :

Non dans la liste noire

« Accès refusé : nom dans la liste noire » s'affiche lorsqu'un utilisateur utilise un nom interdit pour des raisons de sécurité ou de prévention d'abus.

Figure 6:Accès refusé

CONCLUSION

Ce TP nous a permis de découvrir et de mettre en pratique les concepts fondamentaux du ***développement web en JEE****, notamment : les* ***servlets HTTP****, l'utilisation d'un* ***formulaire HTML****, le* ***traitement des requêtes GET et POST****, ainsi que l'affichage dynamique des résultats.*

Nous avons également exploré l'extension des fonctionnalités grâce à :

- ***JDBC et MySQL*** *pour stocker et gérer les données de manière persistante.*
- ***Filtres*** *pour sécuriser l'accès et contrôler l'utilisation de l'application (par exemple, bloquer certains noms via une blacklist).*
- *La* ***gestion des rôles et des erreurs*** *pour améliorer la sécurité et la robustesse de l'application.*

Ce TP a donc permis de comprendre le rôle de chaque couche dans une application JEE :

- *La* ***couche présentation*** *(HTML, JSP, servlets) pour interagir avec l'utilisateur.*
- *La* ***couche métier*** *pour la logique applicative.*
- *La* ***couche de persistance*** *via JDBC et bases de données.*

En conclusion, cette activité pratique illustre comment développer une ***application web complète et sécurisée****, tout en appliquant les bonnes pratiques de séparation des responsabilités et de contrôle d'accès dans la plateforme JEE.*