

Mini-Projet (3h) – Programmation Web côté Serveur (Java EE)

Intitulé du mini-projet

Application Web Java EE : Gestion d'accès utilisateur (Authentification) Durée: 3 heures

Objectifs pédagogiques

À l'issue de ce mini-projet, l'apprenant sera capable de :

- ✓ Utiliser Git et GitHub pour le travail collaboratif
- ✓ Créer et configurer des Servlets
- ✓ Utiliser des JSP avec séparation des rôles
- ✓ Implémenter le modèle MVC
- ✓ Accéder à une base de données avec JDBC
- ✓ Mettre en place une authentification utilisateur
- ✓ Créer et utiliser une librairie de tags JSP (Custom Tags)

Technologies et outils

- ✓ Java EE (Servlets, JSP)
- ✓ Apache Tomcat
- ✓ JDBC (MySQL ou MariaDB)
- ✓ Git & GitHub
- ✓ IDE : NetBeans / IntelliJ / Eclipse
- ✓ Base de données : users_db

Contexte du projet

Vous devez développer une application web de connexion utilisateur :

Un utilisateur saisit son login et mot de passe

Le système vérifie les informations dans la base de données

Si les informations sont correctes :

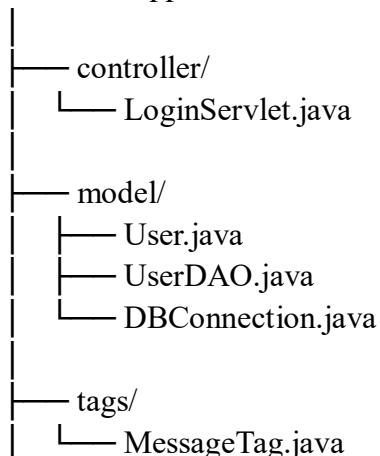
L'utilisateur est redirigé vers une page dashboard

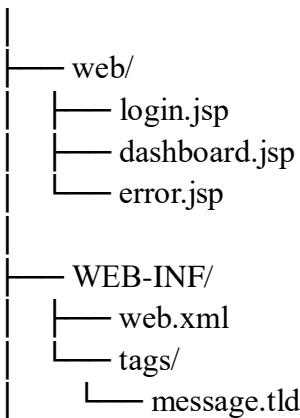
Sinon :

Un message d'erreur s'affiche

Structure du projet (MVC)

MiniAuthApp/





ÉNONCÉ DE L'EXERCICE

Partie 1 – Git & GitHub (20 min)

1. Créer un dépôt Git local nommé MiniAuthApp
2. Ajouter le projet Java EE
3. Effectuer :


```
git init
      git add .
      git commit -m "Initial commit"
```
4. Créer un dépôt GitHub et pousser le projet

Partie 2 – Base de données & JDBC (30 min)

Créer la base de données suivante :

```

CREATE DATABASE users_db;
USE users_db;

CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50),
  password VARCHAR(50)
);

INSERT INTO users(username, password)
VALUES ('admin', '1234');
  
```

Partie 3 – Modèle (MVC) : JDBC & DAO (30 min)

Créer la classe `DBConnection`

Créer la classe `User`

Créer la classe `UserDAO` avec une méthode :

`public boolean authenticate(String username, String password)`

Partie 4 – Servlet (30 min)

Créer LoginServlet

Récupérer les données du formulaire

Appeler UserDAO

Rediriger vers :

dashboard.jsp si succès

login.jsp avec message d'erreur sinon

Partie 5 – JSP (20 min)

Créer login.jsp

Créer dashboard.jsp

Utiliser EL (`${ }`) pour afficher les messages

Partie 6 – Crédation d'une librairie de TAGs (20 min)

Créer un tag personnalisé pour afficher les messages :

`<mytags:message text="Erreur d'authentification"/>`

Partie 7 – Authentification & Session (20 min)

Créer une session utilisateur

Empêcher l'accès au dashboard sans authentification

ÉTAPES BIEN EXPLIQUÉES

1. Connexion JDBC

```
public class DBConnection {  
    public static Connection getConnection() throws Exception {  
        Class.forName("com.mysql.cj.jdbc.Driver");  
        return DriverManager.getConnection(  
            "jdbc:mysql://localhost:3306/users_db",  
            "root", ""  
        );  
    }  
}
```

2. Classe User

```
public class User {  
    private String username;  
    private String password;  
  
    // getters et setters
```

```
}
```

3. UserDao (Authentification)

```
public boolean authenticate(String username, String password) {  
    try (Connection con = DBConnection.getConnection()) {  
        PreparedStatement ps = con.prepareStatement(  
            "SELECT * FROM users WHERE username=? AND password=?"  
        );  
        ps.setString(1, username);  
        ps.setString(2, password);  
  
        ResultSet rs = ps.executeQuery();  
        return rs.next();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    return false;  
}
```

4. Servlet (Contrôleur MVC)

```
protected void doPost(HttpServletRequest req, HttpServletResponse resp)  
    throws ServletException, IOException {  
  
    String user = req.getParameter("username");  
    String pass = req.getParameter("password");  
  
    UserDao dao = new UserDao();  
  
    if (dao.authenticate(user, pass)) {  
        HttpSession session = req.getSession();  
        session.setAttribute("user", user);  
        resp.sendRedirect("dashboard.jsp");  
    } else {  
        req.setAttribute("error", "Login ou mot de passe incorrect");  
        req.getRequestDispatcher("login.jsp").forward(req, resp);  
    }  
}
```

5. JSP – login.jsp

```
<form action="LoginServlet" method="post">  
    Login : <input type="text" name="username"><br>  
    Password : <input type="password" name="password"><br>  
    <input type="submit" value="Connexion">  
</form>  
  
<mytags:message text="${error}" />
```

6. Tag personnalisé

Classe du tag

```
public class MessageTag extends SimpleTagSupport {  
    private String text;  
  
    public void setText(String text) {  
        this.text = text;  
    }  
  
    public void doTag() throws IOException {  
        if (text != null) {  
            getJspContext().getOut().print("<p style='color:red'>" + text + "</p>");  
        }  
    }  
}
```

□ 7. Sécurité (Session)

```
<%if (session.getAttribute("user") == null) {  
    response.sendRedirect("login.jsp");  
}  
%>
```

□ Critères d'évaluation (suggestion)

Critère	Points
Git & GitHub	2
JDBC & DAO	3
MVC respecté	3
Servlets	3
JSP & EL	3
Tags personnalisés	3
Authentification & session	3
Total	20 pts