PROJET ANNUEL :

Nom et Prénom : Ibrahima Sory DIALLO

Configuration de Thymeleaf et du serveur Tomcat

1. Thymeleaf

Thymeleaf est utilisé comme moteur de templates dans l'interface HTML. Il permet :

- D'afficher dynamiquement les données côté serveur avec th:text, th:if, th:href, etc.
- De gérer des vues conditionnelles (ex. affichage en fonction du rôle).
- De se passer de JSP.

2. Configuration du serveur Tomcat dans IntelliJ

Étapes :

1. **Créer une configuration Run/Debug :**
    - Run > Edit Configurations > + > Tomcat Server > Local
    - Name : Color-Run (Tomcat)
    - Application server : sélectionne ton Tomcat
2. **Onglet Deployment :**
    - + > Artifact > color_run:war exploded
    - Context : color_run_war_exploded (ou ROOT)
3. **Onglet Server :**
    - HTTP port = 8080 (ou celui souhaité)
4. **Lancement :**
    - Clique sur (Run)
    - URL : http://localhost:8080/color_run_war_exploded

Intégration de Stripe

Objectif

Ajouter le paiement Stripe lors de l'inscription à une course.

Étapes

1. **Ajouter la dépendance Stripe Java dans pom.xml :**

```
<dependency>
  <groupId>com.stripe</groupId>
  <artifactId>stripe-java</artifactId>
  <version>24.5.0</version>
</dependency>
```

Intégration de Stripe

Objectif

Ajouter le paiement Stripe lors de l'inscription à une course.

Étapes

1. **Ajouter la dépendance Stripe Java dans pom.xml :**

```
<dependency>
  <groupId>com.stripe</groupId>
  <artifactId>stripe-java</artifactId>
  <version>24.5.0</version>
</dependency>
```

Cacher les clés API Stripe dans un fichier .json ou .txt (ne jamais les exposer dans le code)

3. **Créer un servlet /create-payment-intent** :
    - Ce servlet retourne un client_secret après avoir créé un PaymentIntent via l'API Stripe.
4. **Côté client :**

- Appel fetch('/create-payment-intent') en JS
- Traitement de la carte avec Stripe.js

5. **Carte de test :**
    - Numéro : 4242 4242 4242 4242
    - Expiration : toute valide
    - CVC : 123

Tests unitaires et base de données

Base H2 pour les tests

Utilisation de la base H2 pour les tests unitaires :

- Légère, embarquée, simple à configurer
- JDBC URL : jdbc:h2:~/color_run_db;AUTO_SERVER=TRUE

Commande pour lancer H2 :

java -jar "C:\Program Files (x86)\H2\bin\h2-2.3.232.jar"

Sécurité : Gestion des mots de passe

Pourquoi utiliser BCrypt ?

- Résistant aux attaques brute-force (coût ajustable)
- Génère un sel unique automatiquement
- Bien intégré en Java (ex. via at.favre.lib:bcrypt)

Processus d'inscription :

1. Le **Servlet** transmet le mot de passe en clair à la couche service.
2. La **couche service** appelle PasswordUtil.hash(password) pour le crypter.
3. La **couche repository** stocke le hash en base.

Autres recommandations

- Créer un package exception pour gérer les erreurs personnalisées.
- Lors de l'inscription à une **course**, ajouter une adresse email (utile pour Stripe ou confirmations).
- Structure MVC : Servlet (contrôleur) → Service → DAO → BDD (H2 ou autre)