Formation Spring

Horaires:
Lundi: 9h30 - 17h30
Mardi - Vendredi: 9h - 17h

Lien GIT:
https://github.com/MouradDawan75/SpringFormationApi

Mourad MAHRANE: Dév/Formateur Java, .net, Android, Python
e-mail: mmahrane@dawan.fr

Java SE: consoles, graphiques
Java EE: applications client/server
Java ME: remplacée par Android

Spring Galaxy: est un ensemble de framework

Spring core: IOC: Inversion Of Control
> Création d'objets
> Gestion des dépendances

Spring web: api rest - mvc
Spring Data JPA: permet la gestion de la couche persistance des données
Spring Security: sécuriser les apps
Spring Boot:
permet de faciliter la configuration, le déploiement du livrable.
Spring Boot génère un .jar qui embarque le server Tomcat.
Spring Boot utilise des starters (ensemble de dépendances)

Maven: est un outil de build
- Génère la structure du projet
- Gestion des dépendances
- Compilation du code source
- Génération du livrabe (.jar)
- Deployer le livrable
- Générer la doc
- Exécution des tests unitaires et fonctionnels

Environnement:
1- Installer Java 17: https://download.oracle.com/java/17/archive/jdk-17.0.12_windows-x64_bin.exe
Ensuite ajoutez le chemin du dossier bin dans le path de windows:
C:\Program Files\Java\jdk-17\bin
2- Installer un IDE: Spring Tool Suite - IntelliJ:
https://www.jetbrains.com/fr-fr/idea/download/download-thanks.html?platform=windows&code=IIC

********** **Rest API**********

Un web service est une application sans IHM. Expose un ensemble de méthodes accessibles via le web.
2 types de web services:

- SOAP:
  est un protocole d'échange de messages codés en xml.
- REST: Representation State Transfert
  est un style d'architecture.
  Une Api REST utilise uniquement le protocole HTTP.
  Elle n'est pas limitée au contenu JSON: elle peut renvoyer du xml, texte, binaire.....

Une API Rest est un ensemble de ressources (images, article d'un journal...), où chaque ressource possède un ID (URI: Uniform Resource Identifier - appelé aussi End point), une méthode d'accès (GET, POST, DELETE, PUT), et elle peut être publique ou privée.

Toutes ces infos sont fournies dans la doc de l'api.

Lien doc application.properties:

#https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html

********** **Gestion des logs en Java + Gestion des exceptions** **********

permet l'écriture des traces dans un support de persistance (fichiers, BD.....)
Le jdk fournit une api permettant d'écrire des logs: java.ujit.logging
On peut utuliser des librairies externes: Log4j, Logback, Commons-Logging.....

Spring utilise slf4j qui une façade pour pouvoir écrire sur plusieurs systèmes utilisants des bibliothèques différentes.
Spring Boot: utilise Log4j + logback (succésseur de Log4j) car Log4j est vulnerable.
Vocabulaire:

- Logger: objet permet d'écrire des logs
- Appender: support des logs
- Pattern: format du message: %date%message
- Level: niveau de gravité du message: debug,info,warning,error

Mise en place:

- Logback est déjà chargé par le starter de SPring Boot

- - Mettre en place une conf, soit dans application.properties, soit l'externaliser dans un fichier (xml) appelé logback.xml dans le dossier resources
- La gestion des exception peut s'effectuer à n'importe quel endroit du traitement métier en capturant et traitant l'exception avec un try/catch (service, controller...), ou en remontant l'exception (throws) et en définissant un GlobalExceptionHander par type d'exception capturée. Voir le package interceptors.
- ********** **Documentation de l'API** **********
- Open API Specification (Swagger3): est la spécification d'une API

- Elle décrit l'api REST au travers d'un fichier json/yaml qu'on peut visualiser dans Swagger Editor.
- Mise en place: ajout de la dépendance dans pom.xml
- <dependency>
- <groupId>org.springdoc</groupId>
- <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
- <version>2.8.5</version>
- </dependency>
- 2 ends point:
- Pour la doc json:
- server:port/v3/api-docs
- Pour Swagger UI:
    - server:port/swagger-ui.html

Dans application.properties, on a la possibilité de modifier ces 2 end points.
springdoc.api-docs.path=/api-docs

Important: désactiver swagger ui en prod.

springdoc.swagger-ui.enabled=false

********** **Spring Profils** **********
On peut définir plusieurs conf. pour une app. Spring.
- 1 conf pour la prod
-1 conf pour le dév.
-1 conf pour le test
Il suffit de créer un fichier application.properties pour chaque conf
application-prod.properties
application-dev.properties
application-test.properties

Dans application.properties, choisir le profil à activer:
spring.profiles.active=dev

********** **Gestion des upload et download** **********
Activer Lombock:
ajouter dans le pom.xml:

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.36</version>
    <scope>provided</scope>
</dependency>
```

Installer le plugin Lombock dans IntelliJ
Doc: https://objectcomputing.com/resources/publications/sett/january-2010-reducing-boilerplate-code-with-

project-lombok

Voir package controllers: UploadDownloadCOntroller

********** **Spring Data JPA** **********

Spring Data JPA est l'implémentation de Spring du JPA/Hibernate.
Il propose un ensemble d'annotations pour mapper les Entity, les relations et l'héritage.
Il porpose un ensemble de repository génériques qui contiennent les méthodes classiques implementées qu'on peut compléter via les finders méthodes de Spring, ou via des commandes JP-QL ou via des commandes Sql natives.

>>>>>>> **Annotations de mapping d'entity:**

Commandes SQL LMD: Langage de manipulation des données (SELECT,DELETE,UPDATE,INSERT)
Commandes SQL LDD: Langage de définition de données (structure): CREATE,DROP,ALTER
@Entity
@Id @GeneratedValue(strategy)
@Version: versionner les modifications
@Column
@Enumerated
@ElementCollection
@Transient

>>>>>>>> **Annotations de mapping des relations:**

ManyToOne - OneToMany:
Voir Product - Category

ManyToMany:
Voir product - Supplier

OneToOne:
3 représentations possibles:

> 1 seule table:

- 1 Player est encadré par 1 Manager

@Entity Player(id,name) @Embedable Manager(id,name)

- @Embedded private Manager manger
- **> 2 tables: association par clé primaire**
- 2 tables séparées: @OneToOne que d'un côté
- Voir Player1 ---- Manager1
- **> 2 tables: association par clés étrangères**
- OneToOne dans les 2 entity avec 1 seul mappedBy
- Voir Player2 --- Manager2
- >>>>>>> **Annotations de mapping de l'héritage:**
- **- Single Table:**
- L'héritage est représenté par une seule table en BD (uniquement pour la classe mère)

- Une colonne contiendra un identifiant pour déterminer le type de classe fille (@DescriminatorColumn)
- **- Joined:**
- L'héritage est représenté par une jointure entre la table de l'entity mère et les entity filles
- Classes: Vehicule - Voiture - Moto
- **-Table Per Classe:**
- L'héritage est représenté par une table pour chaque classe fille (pas de table pour la classe mère)
- Plus complexe à gérer car l'id ne peut pas être en auto incrément dans la classe mère.
- Option la moins utilisée en pratique
- Classes: Employe - Ingenieur - Technicien
- >>>>>>>>>>>> **Jpa Repository:**
- Il suffit de créer une interface qui hérite de JpaRepository<Type_Objet, Type_Id> qui propose des méthodes déjà implémentées qu'on compléter via les finders méthodes ou via les commandes JP-QL ou les commandes Sql natives.
- Finder Method doc:
- https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html
- Voir ProductRepository
- ********** **Génération du Token** **********
- **Dépendances a ajouter:**
- Spring Security Crypto: pour le cryptage du password:
- **Jwt dépendances:**
    - <dependency>
    - <groupId>io.jsonwebtoken</groupId>
    - <artifactId>jjwt-api</artifactId>
    - <version>0.12.6</version>
    - </dependency>
    - <!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-impl -->
    - <dependency>
    - <groupId>io.jsonwebtoken</groupId>
    - <artifactId>jjwt-impl</artifactId>
    - <version>0.12.6</version>
    - <scope>runtime</scope>
    - </dependency>
    - <dependency>
    - <groupId>io.jsonwebtoken</groupId>
    - <artifactId>jjwt-jackson</artifactId>
    - <version>0.12.6</version>
    - <scope>runtime</scope>
    - </dependency>
    - Classe JwtService fournie par Spring Security. Utiliser une clé dans application.properties pour la signature du Token (clé de 56 caractéres au minimum)
    - **Ajouter un intercepteur pour la vérification du Token:**
    - Voir TokenInterceptor: l'ajouter dans conf. du module MVC -> voir la classe principale
- ********** **Audit des tables** **********

- Mise en place d'un système d'historisation des données permettant de conserver l'historique des modifications sur les données.
- > 2 solutions possibles:
- >>>>> **Solution1: Utiliser Hibernate Envers:**
- Etape1: ajouter la dépendance dans le pom.xml
- Etape2:
- Pour toute Entity annotée avec @Audited, il créera une table d'audit nom_entity_aud et une table globale revinfo qui conserve un historique des révision avec un timestemp
- Avantage: simplicité de mise en place
- Inconvénient: multitude de tables à maintenir + peu de personnalisations
- >>>>> **Solution2: Utiliser les intercepteurs de Spring Data JPA**
- Mise en place:
- > Dans la classe principale, utiliser @EnableJpaAuditing pour activer les JPa interceptors
- > Dans chaque Entit, utilisez les @PrePersisit, @PreRemove, @PostPersist.....
- Pour une solution optimale, utilisez une classe mère (BaseEntity) qui représente l'intercepteur global, via l'éannotation @EntityListener: @CreatedDate, @LastModifiedDate
- Possibilité de combiner les 2 solutions
- ********** **Mode Asynchrone** **********
- Etape1: définir un ThreadPool dans une classe de conf. void AsyncConf
- Etape2: activer le mode asynchrone via l'annotation @EnableAsync soir dans la classe principae, soit dans une classe de conf.
- ********** **Planification de tâches**********
- Nécessite que le mode asynchrone soit activé (définir un threadpool)
- Activer l'exécution de t^ches planifiées via l'annotation @EnableScheduling
- Définir des tâches planifiées: voir packages: mytasks
- ********** **Spring MVC** **********
- Lien GIT:
- https://github.com/MouradDawan75/springmvc
- Lien Git Workspace:
- https://github.com/MouradDawan75/springworkspace