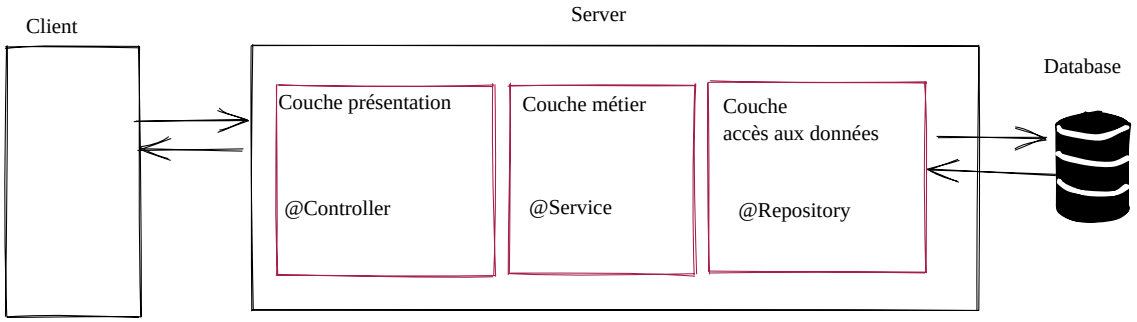
Fiche MEMO SpringBoot

▼ Architecture d'une application avec SpringBoot



▼ Création d'un projet

- Aller sur : https://start.spring.io/
- Choisir les paramètres du projet
- Ajouter au moins la dépendance: Spring Web
- Cliquer sur le bouton Generate pour télécharger le fichier zip du projet
- Dézipper
- Dans IntelliJ Community :

  Menu : File > New > Project from existing source > import project from external model > Maven

▼ (Optionnel) Configuration

▼ Changer le numéro de port de Tomcat

`src/main/resources/application.properties`

```
server.port=9090
```

▼ Création d'un Composant Spring de type `Service` (couche métier)

```
@Service
public class CrmService {
```

▼ Création d'un Composant Spring de type `Controller` (couche présentation)

```
@RestController
@RequestMapping("api")
public class ClientController {
```

```
    @GetMapping("clients")
    public List<Client> getAll() {
    }

    @PostMapping("clients")
    public void createClient(@RequestBody Client client){
    }

    @GetMapping("clients/{id}")
    public Client findClientById(@PathVariable("id") Long id){
    }

    @DeleteMapping("clients/{id}")
    public void delete(@PathVariable("id") Long id) {
    }

    @PutMapping("clients/{id}")
    public void update(@PathVariable("id") Long id, @RequestBody Client client){
    }
}
```

▼ `Injection` de dépendance avec Spring

```
public class ClientController {

  @Autowired
  CrmService crmService;
```

▼ Configuration de Spring `JPA` (Couche données)

▼ PostgreSQL

▼ Maven

`pom.xml`

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <scope>runtime</scope>
</dependency>
```

▼ Properties

`src/main/resources/application.properties`

```
spring.datasource.url=jdbc:postgresql://localhost:5432/poe20221107_demo_personnes
spring.datasource.username=postgres
spring.datasource.password=mypassword
```

▼ MySQL

▼ Maven

`pom.xml`

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
</dependency>
```

▼ Properties

`src/main/resources/application.properties`

```
spring.datasource.url=jdbc:mysql://localhost:3306/poe_ang_v1?useSSL=false&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=mypassword
```

▼ (Optionnel) si **BUG avec annotation @Column dans SpringBoot+JPA : Column name annotation ignored**

```
# BUG avec annotation @Column dans SpringBoot+JPA : Column name annotation ignored
# Sinon erreur : unknown column in field list
# Explication du Pb : dans Mysql les noms des tables sont écrits en camelCase(Java) au lieu de snakecase(SQL)
spring.jpa.hibernate.naming.implicit-strategy=org.hibernate.boot.model.naming.ImplicitNamingStrategyLegacyJpaImpl
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

▼ Conversion d'un POJO en Entity JPA

```
@Entity
@Table(name="clients")
public class Client {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

▼ Création d'un DAO correspondant à un Entity

→ Penser à vérifier les types de l'Entity et de l'Id

```
@Repository
public interface ClientRepository extends JpaRepository<Client, Long> {

}
```

▼ (Optionnel) Configurations Hibernate

`src/main/resources/application.properties`

```
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.show-sql=true
```

▼ Operations dans le Repository

▼ CRUD operations

findAll()
count()
existsById()
save()
findById()
findAllById()
deleteAll()
deleteById()
saveAll()

▼ Custom finders

▼ findAllBy

```
public interface TrackRepository extends JpaRepository<Track, Long> {
```

```
    public List<Track> findAllByTitle(String title);
    public List<Track> findAllByAlbum(String album);
    public List<Track> findAllByArtist(String artist);
}
```

▼ And / Or

```
public List<Track> findAllByArtistAndTitle(String artist, String title);
```

▼ Divers...

→ PagingAndSortingRepository
→ ignoreCase
→ long countByLastname(String lastname);
→ long deleteByLastname(String lastname);
→ List<User> removeByLastname(String lastname);
→ Query keywords : Between, LessThan, GreaterThan, Like, ...

▼ Documentation

https://docs.spring.io/spring-data/data-jpa/docs/current/reference/html/#jpa.query-methods.query-creation

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/

▼ Ajouter `Swagger` pour générer la documentation de l'API web

▼ Spring v3

▼ Install

▼ Ajouter une dépendance Maven

`pom.xml`

```
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
  <version>2.0.2</version>
</dependency>
```

▼ Redémarrer le serveur

▼ Consulter la documentation de votre API

http://localhost:8080/swagger-ui/index.html

▼ Spring v2

▼ Install

▼ Ajouter une dépendance Maven

`pom.xml`

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.6.8</version>
</dependency>
```

▼ Redémarrer le serveur

▼ Consulter la documentation de votre API

http://localhost:8080/swagger-ui.html