**Présentation du framework Spring**

Préambule sur l’abstraction (5mn) {

* L’homme fait abstraction de toute sa connaissance au quotidien : homo sapiens sapiens
* Les langages ont différents lvl de complexité

}

Spring ? (5mn) {

A ne pas confondre avec Maven , qui est un gestionnaire de build et de dépendances.

Les caractéristiques principales de Spring sont :

Core technologies: dependency injection, events, resources, i18n, validation, data binding, type conversion, SpEL, AOP.

Testing: mock objects, TestContext framework, Spring MVC Test

Data Access: transactions, DAO support, JDBC, ORM, Marshalling XML.

Spring MVC and Spring WebFlux web frameworks.

Integration: remoting, JMS, JCA, JMX, email, tasks, scheduling, cache.

Languages: Kotlin, Groovy, dynamic languages.

Il facilite le travail et évite d’écrire du code grâce a différentes annotations.

Il permet d’accéder à de multiples starter qui configures différents paramètres pour nous.

Il embarque un serveur tomcat pour déployer une app d’un click

}

Live coding avec exemples (20mn) {

* **Spring Iniializer**

*Liste des annotations utilisées :*

@Configuration : Permet de choisir quelles dépendances vont êtres chargées en d’une manière générale de configurer son app

@Builder : Permet de construire ses objets sans a faire new, et gagner en lisibilité sur des gros objets

@Data : Par lombok, construit les getter setter hash code tostring et equals.

@Entity : Sert pour la persistance en ORM avec JPA

@AllArgsConstructor : construit un constructeur avec tous les paramètres

@NoArgsConstructor : la même mais sans paramètres.

@Id : au dessus de la clef primaire pour spécifier la PK

@GeneratedValue : Avec le @Id pour définir la manière dont va être généré l’ID

@Transient : sert a indiquer que la valeur ne sera pas persisté en base

@Table : a indiquer au dessus de la classe pour personnaliser le nom en base

@Column : permet d’ajouter des infos comme le nom de la table, la nullabilité

@JoinColumn : indique la classe a target pour le FK

@JoinTable : Same avec la table

@CrossOrigin : Accepte les requêtes de tous lieux

@RestController : Indique qu’il s’agit un controller rest

@Service : Indique qu’il s’agit d’une classe de service envoyé vers le controller

@Autowired : sert a injecter un repo dans le controller sans avoir a l’instancier

@RequestMapping : indique la valeur de ‘URL après le localhost :XXX/

@OneToMany @ManyToMany @OneToOne @ ManyToOne : sert pour les différents types de relations entres les tables.

@PutMapping, GetMapping, PostMapping, DeleteMapping pour indiquer le verbe de la requête

@Slf4j : Permet d’ajouter des logs

}