NOTES

The annotation @Document from the package

***org.springframework.data.mongodb.core.mapping.Document***

Is used in **Spring Data MongoDB** to map a Java class to a MongoDB collection. It indicates that a particular class is intended to represent a MongoDB document and will be stored as a record (or "document") in a MongoDB collection.

**What are Annotations?**

In Java, **annotations** are special metadata (information) attached to classes, methods, fields, or other elements of your code. They provide instructions to the compiler, frameworks, or libraries about how to handle the annotated code. Annotations do not affect the logic of the program directly but can influence how the code is processed at runtime or compile-time.

**@Id**

* **From**: org.springframework.data.annotation.Id
* **Purpose**: This annotation marks the objectId field as the **primary key** of the MongoDB document. MongoDB will store this field as the \_id field in the document, which uniquely identifies each document in the movies collection.
  + In MongoDB, each document is required to have an \_id field, and @Id maps the objectId field to that \_id.

**@DocumentReference**

* **From**: org.springframework.data.mongodb.core.mapping.DocumentReference
* **Purpose**: This annotation is used to create a **reference** between documents in MongoDB. Instead of embedding Review objects directly inside the Movie document, @DocumentReference tells Spring Data MongoDB to store only the **references (ObjectIds)** of the related Review documents in the movies collection.
  + This is similar to foreign key relationships in relational databases.
  + The field reviewIds will hold references (ObjectIds) to Review documents, and the actual Review documents will be stored in a separate MongoDB collection.

**Lombok** is a popular Java library that helps reduce boilerplate code by automatically generating commonly used methods such as **getters**, **setters**, **constructors**, **toString()**, **hashCode()**, and **equals()** methods. It uses annotations to simplify the creation of Java classes, improving code readability and reducing the need to manually write repetitive code.

**@NoArgsConstructor**

* **From**: lombok.NoArgsConstructor
* **Purpose**: This **Lombok annotation** generates a **no-argument constructor**. It's useful when frameworks like Spring or Hibernate need to create objects through reflection (which requires a default constructor).

**@AllArgsConstructor**

* **From**: lombok.AllArgsConstructor
* **Purpose**: Another **Lombok annotation** that generates a constructor with **parameters for all fields** in the class. This way, you don't need to write a constructor that takes all the properties of Movie.