ABOUT PACKAGES

PROJECT STRUCTURES

1. **Controller** Package:

The **controller** package contains classes that handle incoming HTTP requests and prepare HTTP responses. Controllers are responsible for handling user input, validating input data, and invoking the appropriate business logic. Controllers are typically annotated with **@Controller** or **@RestController** annotations, which enable Spring to route HTTP requests to the appropriate method.

2. **Service** Package:

The **service** package contains classes that encapsulate business logic and implement use cases. Services typically define operations that can be performed on entities and perform the necessary checks and validations before modifying the data.

The primary responsibility of a service class is to coordinate and orchestrate the work of multiple repository and domain classes to implement a specific use case. A service class is also responsible for providing transaction management and error handling for the use case.

3. **Entity** Package:

The **entity** package contains classes that represent the domain model of the application. These classes are typically annotated with JPA annotations, which define how the entities are mapped to the database. The domain model consists of classes that represent the business entities and their relationships.

For example, if you were building an e-commerce application, you might have entities such as **Product**, **Customer**, **Order**, etc. The **Product** entity would contain information about the products being sold, such as the name, price, and description. The **Customer** entity would contain information about the customers, such as their name, email address, and billing address. The **Order** entity would contain information about the orders placed by the customers, such as the products ordered, the order total, and the shipping address.

4. **Repository** Package:

The **repository** package contains classes that handle data access operations. These classes typically use a persistence framework like JPA to interact with the database. The primary responsibility of a repository class is to provide a high-level, object-oriented interface for interacting with the database.

The repository classes contain methods to perform CRUD operations on the entities, such as **create**, **read**, **update**, and **delete**. The methods can be customized by adding query methods, which allow you to query the database using various criteria.

For example, you might have a **ProductRepository** class that provides methods to retrieve a single product by ID, retrieve all products, create a new product, update an existing product,

and delete a product. The repository can also define custom query methods, such as **findProductsByCategory** to retrieve all products in a specific category.

Overall, organizing your application code into logical modules like these can make your code more modular, easier to maintain, and easier to scale. By separating the responsibilities of different modules, you can make your code more focused and less tightly coupled, which can make it easier to test and refactor.