**Spring MVC Integration**

Spring MVC (Model-View-Controller) is a framework provided by Spring for building web applications. It facilitates the development of web applications by providing the MVC architecture.

1. **WebAppConfig.java:**

*This class is annotated with @Configuration, indicating that it is a configuration class.*

*@ComponentScan is used to specify the base package where Spring should scan for components.*

*@EnableWebMvc enables Spring MVC support in the application.*

*It configures Thymeleaf as the view resolver to resolve views from templates.*

*addResourceHandlers method configures static resource handling for CSS files.*

*dataSource and sessionFactory methods configure the data source and session factory for Hibernate integration respectively.*

*transactionManager method configures the Hibernate transaction manager.*

1. **WebAppInitializer.java:**

*Extends AbstractAnnotationConfigDispatcherServletInitializer to initialize the Servlet context with Java-based configuration.*

*getRootConfigClasses and getServletConfigClasses methods specify the root and servlet configuration classes respectively.*

*getServletMappings method specifies the Servlet mappings.*

1. **WebSecurityConfig.java:**

*Configures security filters for URL patterns.*

*Implements Filter interface to intercept and process HTTP requests.*

*Defines allowed paths that can be accessed without authentication.*

*Redirects unauthorized requests to the login page.*

1. **MainController.java:**

*Annotated with @Controller, indicating that it's a controller component.*

*Handles HTTP requests and returns appropriate views.*

*Defines methods to handle requests for login, registration, product management, cart management, orders, and logout.*

*Uses @Autowired to inject dependencies of UserService, ProductService, and OrderService.*

**Hibernate Integration**

Hibernate is an object-relational mapping (ORM) framework that provides a mapping from Java classes to database tables and vice versa, along with data query and retrieval facilities.

1. **WebAppConfig.java (Continued):**

*Configures Hibernate properties such as dialect, show SQL, and auto DDL.*

*Defines dataSource bean to specify database connection details.*

*Configures LocalSessionFactoryBean to create a session factory for Hibernate.*

*HibernateTransactionManager is configured to manage transactions in Hibernate.*

1. **Entities**:

*Classes annotated with @Entity represent database tables.*

*Class properties represent table columns, and relationships between entities are defined using annotations such as @ManyToOne, @OneToMany, etc.*

1. **DAO Interfaces and Implementations:**

*DAO (Data Access Object) interfaces define methods for CRUD (Create, Read, Update, Delete) operations.*

*Implementations of DAO interfaces interact with the database using Hibernate APIs.*

1. **Service Interfaces and Implementations**:

*Service interfaces define business logic methods.*

*Service implementations provide the actual implementation of business logic by interacting with DAOs.*

*Service layer acts as a bridge between controllers and DAOs, encapsulating business logic.*