**Документация на класа OrderItemService**

**Общ преглед**

Класът **OrderItemService** предоставя бизнес логика за управление на поръчкови артикули (**OrderItem**) в системата. Той използва **OrderItemRepository** за взаимодействие с базата данни и предоставя различни методи за достъп, добавяне, премахване и обновяване на поръчкови артикули.

**Зависимости**

* **OrderItemRepository**: Инжектиран чрез анотацията **@Autowired**, този репозиторий осигурява необходимите операции за взаимодействие с базата данни за обекти от тип **OrderItem**.

**Методи**

1. **getOrderItems()**: Връща списък на всички поръчкови артикули. Използва **findAll()** метода от репозитория.
2. **getOrderItem(Long id)**: Връща поръчков артикул по зададен идентификационен номер. Използва **findById(id)** и връща резултат, ако съществува такъв.
3. **getOrderItemsByOrderId(Long id)**: Връща всички поръчкови артикули, които са свързани с конкретна поръчка. Филтрира списъка от всички артикули по ID на поръчката.
4. **addOrder(OrderItem p)**: Добавя нов поръчков артикул в системата. Използва **save(p)** метода от репозитория.
5. **removeOrder(Long id)**: Премахва поръчков артикул по зададен ID. Използва **deleteById(id)**.
6. **updateOrder(Long id, OrderItem item)**: Обновява съществуващ поръчков артикул. Задава новото ID на артикула и го запазва отново в базата данни.

**Документация на класа OrderService**

**Общ преглед**

Класът **OrderService** предоставя бизнес логика за управление на поръчки (**Order**) в системата. Той използва **OrderRepository** за взаимодействие с базата данни и предоставя различни методи за достъп, добавяне, премахване и обновяване на поръчки.

**Зависимости**

* **OrderRepository**: Инжектиран чрез анотацията **@Autowired**, този репозиторий осигурява необходимите операции за взаимодействие с базата данни за обекти от тип **Order**.

**Методи**

1. **getOrders()**: Връща списък на всички поръчки. Използва **findAll()** метода от репозитория.
2. **getOrder(Long id)**: Връща поръчка по зададен идентификационен номер. Използва **findById(id)** и връща резултат, ако съществува такъв.
3. **addOrder(Order order)**: Добавя нова поръчка в системата. Използва **save(order)** метода от репозитория.
4. **removeOrder(Long id)**: Премахва поръчка по зададен ID. Използва **deleteById(id)**.
5. **updateOrder(Long id, Order order)**: Обновява съществуваща поръчка. Задава новото ID на поръчката и го запазва отново в базата данни.

**Документация на класа ProductService**

**Общ преглед**

Класът **ProductService** предоставя бизнес логика за управление на продукти (**Product**) в системата. Този сервизен клас използва **ProductRepository** за взаимодействие с базата данни и предлага методи за достъп, добавяне, премахване и обновяване на продуктови записи.

**Зависимости**

* **ProductRepository**: Инжектиран чрез анотацията **@Autowired**, този репозиторий предоставя основните операции за работа с базата данни за обекти от тип **Product**.

**Методи**

1. **getProducts()**: Връща списък на всички налични продукти. Използва метода **findAll()** от репозитория.
2. **getProduct(Long id)**: Връща продукт по зададен идентификационен номер. Използва **findById(id)** и връща резултат, ако продуктът съществува.
3. **addProduct(Product p)**: Добавя нов продукт в системата. Използва **save(p)** метода от репозитория.
4. **removeProduct(Long id)**: Премахва продукт по зададен ID. Използва **deleteById(id)**.
5. **updateProduct(Long id, Product p)**: Обновява съществуващ продукт. Задава новото ID на продукта и го запазва отново в базата данни.

**Документация на класа UserService**

**Общ преглед**

Класът **UserService** предоставя функционалности за управление на потребителски профили и автентикация в системата. Той реализира интерфейса **UserDetailsService** на Spring Security, което позволява настраиване на потребителската автентикация чрез потребителски данни.

**Зависимости**

* **UserRepository**: За достъп до потребителските данни от базата данни.
* **RoleRepository**: За управление на ролите на потребителите.
* **PasswordEncoder**: За кодиране на паролите на потребителите преди тяхното съхранение в базата данни.

**Основни Методи и Функции**

1. **findUserByEmail(String email)**: Търси потребител по електронна поща.
2. **findAllUsersDto()**: Връща списък от потребителски DTOs, които са облекчени версии на потребителските обекти.
3. **getUsers()**: Връща списък на всички потребители.
4. **getUser(Long id)**: Връща потребител по ID.
5. **createUser(...)**: Създава нов потребител със зададени атрибути и роли.
6. **save(User user)**: Запазва потребител в базата данни.
7. **loadUserByUsername(String email)**: Предоставя детайли за потребителя на Spring Security, базирани на електронна поща.

**Документация на класа OnlineStoreApplication**

**Общ преглед**

Класът **OnlineStoreApplication** служи като начална точка за Spring Boot приложение, което управлява онлайн магазин. Той стартира приложението и инициализира всички компоненти, необходими за функционирането му.

**Анотации**

* **@SpringBootApplication**: Основна анотация на Spring Boot, която обединява **@Configuration**, **@EnableAutoConfiguration**, и **@ComponentScan**. Тази анотация указва на Spring къде да търси компоненти, конфигурации и автоматично да конфигурира приложението.

**Основен метод**

* **public static void main(String[] args)**: Този метод е точката на вход за Java приложението. Използва **SpringApplication.run(OnlineStoreApplication.class, args);** за стартиране на Spring контекста и всички свързани услуги.

**окументация на класа OnlineStoreAppStartup**

**Общ преглед**

Класът **OnlineStoreAppStartup** имплементира интерфейса **CommandLineRunner**, който позволява изпълнението на код веднага след стартирането на Spring Boot приложението. Този клас се използва за начална инициализация на приложението, като например създаването на начални потребителски акаунти.

**Анотации**

* **@Component**: Означава, че класът е компонент на Spring и трябва да бъде автоматично открит и регистриран от Spring контейнера.
* **@Autowired**: Автоматично инжектиране на зависимостта **UserService**, която управлява логиката за потребители.

**Методи**

* **run(String... args)**: Основният метод, който се изпълнява при стартиране. В този метод, се създава администраторски акаунт и се извежда списък на всички потребители.