**Документация на класа User**

**Общ преглед**

Класът **User** е същност, която съхранява информация за потребители в база данни. Той включва данни като име, фамилия, електронна поща и парола. Класът също така реализира интерфейса **UserDetails**, което го прави съвместим с механизмите за сигурност на Spring, позволявайки управление на ролите на потребителя и адресите им.

**Анотации и импортиране**

* **@Entity**: Означава, че класът е същност в контекста на JPA.
* **@Table(name = "user")**: Задава името на таблицата в базата данни.
* **@Id** и **@GeneratedValue(strategy = GenerationType.IDENTITY)**: Идентификационното поле на същността и стратегията за автоматично генериране на стойности.
* **@Column**: Определя свойства на колоните, като **nullable=false** указва, че полето не може да бъде празно, и **unique=true**, което указва, че стойността трябва да бъде уникална.
* **@OneToMany**: Дефинира връзки "един към много" с други същности.

**Методи и функции**

* Конструктори: Конструктор по подразбиране за JPA.
* Гетъри и сетъри за всички полета: Функции за достъп и модификация на свойствата на същността.
* **addAddress** и **removeAddress**: Методи за добавяне и премахване на адреси към потребителя.
* **getAuthorities()**: Връща колекция от роли на потребителя, като всяка роля е обект от тип **GrantedAuthority**.
* Предефинирани методи от интерфейса **UserDetails**: Функции, които определят дали акаунтът е изтекъл, заключен, удостоверенията са валидни и дали акаунтът е активен.

**Пример за използване**

**Документация на класа Role**

**Общ преглед**

Класът **Role** представлява роля в система, обикновено използвана за управление на достъпа и привилегиите. Всяка роля може да има множество привилегии, свързани с нея, и е част от взаимоотношенията на същности, което позволява управление на потребителските права в рамките на системата.

**Анотации и импортиране**

* **@Entity**: Означава, че класът е същност, която се съхранява в базата данни.
* **@Table(name = "role")**: Определя името на таблицата в базата данни, където се съхраняват данните за ролите.
* **@Id**: Указва, че полето е идентификатор за същността.
* **@GeneratedValue(strategy = GenerationType.IDENTITY)**: Конфигурация за автоматично генериране на идентификатора, използвайки базата данни за увеличаване на стойността.
* **@Column(unique = true)**: Задава, че колоната **name** трябва да съдържа уникални стойности.
* **@OneToMany**: Определя отношение "един към много" между ролята и привилегиите, като уточнява, че една роля може да има множество привилегии.

**Конструктори**

* Конструктор по подразбиране (**public Role()**): Използва се за създаване на празен обект **Role**.

**Описание на полетата**

* **Long id**: Уникален идентификатор на ролята.
* **String name**: Име на ролята.
* **List<Privilege> userRoles**: Списък на привилегиите, асоциирани с тази роля.

**Методи**

* **getId()**, **setId(Long id)**: Гетър и сетър за **id**.
* **getName()**, **setName(String name)**: Гетър и сетър за **name**.
* **toString()**: Предефиниран метод, който връща текстово представяне на обекта **Role**.

**Документация на класа Product**

**1. Общ преглед на класа**

Класът **Product** представлява модел на продукт в система за управление на инвентар. Той съдържа основни свойства като идентификатор, заглавие, количество наличност и цена. Този клас се използва за съхранение на информация за продукти в база данни.

**2. Анотации и импортиране**

* **@Entity**: Означава, че класът е същност, която трябва да бъде съхранявана в базата данни.
* **@Id**: Означава, че полето следващо тази анотация е идентификатор на същността.
* **@GeneratedValue(strategy = GenerationType.AUTO)**: Определя, че стойността на идентификатора трябва да бъде генерирана автоматично.

**3. Описание на конструкторите**

* Конструктор по подразбиране (**public Product()**): Използва се за създаване на празен обект на **Product**.
* Параметризиран конструктор (**public Product(Long id, String title, int stock, double price)**): Използва се за създаване на обект с конкретни начални стойности за **id**, **title**, **stock** и **price**.

**4. Описание на полетата**

* **Long id**: Уникален идентификатор на продукта.
* **String title**: Заглавие на продукта.
* **int stock**: Количество наличност на продукта.
* **double price**: Цена на продукта.

**5. Резюме на методите**

* **getId()**, **setId(Long id)**: Гетър и сетър за полето **id**.
* **getTitle()**, **setTitle(String title)**: Гетър и сетър за полето **title**.
* **getStock()**, **setStock(int stock)**: Гетър и сетър за полето **stock**.
* **getPrice()**, **setPrice(double price)**: Гетър и сетър за полето **price**.
* **toString()**: Предефиниран метод, който връща текстово представяне на обекта **Product**.

**Общо Описание:**

Класът **Privilege** е JPA ентити, който описва свързаността между потребителите и техните роли, което определя достъпа им до различни ресурси в системата. Той служи като връзка в многозначната връзка между **User** и **Role**.

**Свойства:**

* **id (Long)**: Уникален идентификатор за всяка привилегия.
* **user (User)**: Потребителят, към когото принадлежи привилегията.
* **role (Role)**: Ролята, асоциирана с привилегията.

**Конструктори:**

* **Privilege()**: Празен конструктор, използван от JPA.
* **Privilege(User user, Role role)**: Конструктор, който инициализира потребител и роля.

**Методи:**

* **Гетъри и сетъри за всички полета**: Предоставят методи за достъп и модификация на свойствата на привилегията.
* **toString()**: Предоставя текстово представяне на привилегията, включващо идентификатора и името на ролята.

**Използване:**

Моделът **Privilege** се използва в системата за управление на потребителски акаунти, позволявайки на администраторите да назначават и управляват различни роли на потребителите. Това включва контрол на достъпа до различни части на системата в зависимост от ролята на потребителя.

Този модел е критичен за сигурността и ефективното управление на достъпа в контекста на онлайн магазини, осигурявайки, че потребителите имат подходящите права в зависимост от тяхната роля.

Начало на формуляра

**Общо Описание:**

Класът **OrderItem** е JPA ентити, който представя единичен артикул в поръчка. Той съхранява информация за количеството, свързания продукт, и общата стойност на артикула в контекста на поръчка.

**Свойства:**

* **id (Long)**: Уникален идентификатор за всеки артикул в поръчка.
* **quantity (int)**: Количество на продукта в поръчката.
* **total (double)**: Общата стойност за този артикул, базирана на количеството и цената на продукта.
* **product (Product)**: Продуктът, който е част от поръчката.
* **order (Order)**: Поръчката, към която принадлежи артикулът.

**Конструктори:**

* **OrderItem()**: Празен конструктор, използван от JPA.

**Методи:**

* **Гетъри и сетъри за всички полета**: Позволяват достъп и модификация на свойствата на артикула в поръчката.
* **toString()**: Връща текстово представяне на артикула, включващо всички полета.

**Използване:**

Моделът **OrderItem** се използва за подробно проследяване на всеки продукт, който е част от поръчка. Това включва управление на складови наличности, калкулации за стойността на поръчки и анализ на продажбите. Помага на търговци и администратори да оценяват и управляват продуктовия поток и наличности.

Този модел е особено важен за онлайн магазините, които трябва да управляват сложни поръчки, включващи множество продукти и различни типове транзакции.

**Общо Описание:**

Класът **Order** е JPA ентити, който представя поръчка в базата данни. Този модел се използва за съхранение и обработка на информация за поръчките на потребителите, включително статуса, датата, общата сума и свързването с потребител и адрес.

**Свойства:**

* **id (Long)**: Уникален идентификатор за всяка поръчка. Генерира се автоматично.
* **status (String)**: Статусът на поръчката, например "обработва се", "изпратена", "доставена".
* **date (Date)**: Датата на създаване на поръчката.
* **total (Double)**: Общата стойност на поръчката.
* **user (User)**: Потребителят, който е направил поръчката.
* **address (Address)**: Адресът за доставка на поръчката.

**Конструктори:**

* **Order()**: Празен конструктор, използван от JPA.

**Методи:**

* **Гетъри и сетъри за всички полета**: Предоставят методи за достъп и модификация на свойствата на поръчката.
* **toString()**: Предоставя текстово представяне на поръчката, което включва всички нейни полета.

**Използване:**

Моделът **Order** се използва в системата за управление на процеса на поръчка от началото до края, включително проследяването на статуса и управлението на плащанията и доставките. Потребители могат да преглеждат историята на своите поръчки, а администраторите да управляват поръчките в системата.

Този модел е критичен за функционирането на онлайн търговията, тъй като улеснява организацията и ефективността на обработката на поръчки, обеспечавайки гладко потребителско изживяване.

Начало на формуляра

Документация за класа **Address**, който представлява модел в проект за онлайн магазин и се използва за управление на адресите на потребителите:

**Общо Описание:**

Класът **Address** е JPA ентити, който представя адрес на потребител в базата данни. Този модел се използва за съхранение и обработка на адресна информация на потребителите в системата.

**Свойства:**

* **id (Long)**: Уникален идентификатор за всеки адрес. Генерира се автоматично.
* **label (String)**: Опционален етикет за адреса, например "дом" или "работа".
* **country (String)**: Страната, в която се намира адреса.
* **city (String)**: Градът на адреса.
* **street (String)**: Улицата на адреса.
* **zip (String)**: Пощенският код на адреса.
* **userId (Long)**: Идентификатор на потребителя, към когото принадлежи адресът.

**Конструктори:**

* **Address()**: Празен конструктор, използван от JPA.
* **Address(Long id, String label, String country, String city, String street, String zip, Long userId)**: Конструктор за инициализация на всички полета на адреса.

**Методи:**

* **Гетъри и сетъри за всички полета**: Използват се за достъп и модификация на свойствата на адреса.
* **toString()**: Предоставя текстово представяне на адреса, което включва всички негови полета.

**Използване:**

Моделът **Address** се използва в различни части на системата за управление на потребителски адреси. Помага за свързване на адресите със съответните потребители и улеснява обработката на доставки и поръчки в контекста на електронна търговия.

Този модел е критичен за управлението на адресната информация в системата и е интегриран в различни процеси свързани с поръчки и доставки.

Начало на формуляра