**Общо Описание:**

**UserController** е контролер, отговорен за обработката на заявки свързани с потребителския профил, включително вход в системата, регистрация, управление на профил, адреси и поръчки.

**Функционалности и Операции:**

1. **home()**:
   * Път: GET **/index**
   * Описание: Визуализира началната страница на магазина.
   * Връща: Страницата **index**.
2. **login(Model model)**:
   * Път: GET **/login**
   * Описание: Показва формата за вход.
   * Връща: Страницата **account/login.html**.
3. **showRegistrationForm(Model model)**:
   * Път: GET **/signup**
   * Описание: Подготвя формата за регистрация на нови потребители.
   * Връща: Страницата **account/signup.html**.
4. **registration(@Validated UserDto userDto, BindingResult result, Model model)**:
   * Път: POST **/signup/save**
   * Описание: Обработва регистрацията на нов потребител, проверява за съществуващи емейли и при успешна регистрация пренасочва към страницата за успешна регистрация.
   * Връща: Пренасочва към **/signup?success** при успех или връща обратно към **account/signup.html** при грешки.
5. **users(Model model)**:
   * Път: GET **/users**
   * Описание: Показва списък с всички потребители в системата.
   * Връща: Страницата **users**.
6. **myProfile(Model model, Authentication authentication)**:
   * Път: GET **/account**
   * Описание: Показва профилната страница на текущия потребител.
   * Връща: Страницата **account/account.html**.
7. **myOrders(Model model, Authentication authentication)**:
   * Път: GET **/account/order**
   * Описание: Показва поръчките на текущия потребител.
   * Връща: Страницата **account/orders.html**.
8. **myAddress(Model model, Authentication authentication)**:
   * Път: GET **/account/address**
   * Описание: Показва адресите на текущия потребител.
   * Връща: Страницата **account/address.html**.
9. **add(Model model)**:
   * Път: GET **/account/addressAdd**
   * Описание: Подготвя форма за добавяне на нов адрес.
   * Връща: Страницата **account/addressAdd.html**.
10. **addPost(@Validated Address address, BindingResult bindingResult, Model model, Authentication authentication)**:
    * Път: POST **/account/addressAddUpdate**
    * Описание: Добавя нов адрес към профила на потребителя и актуализира данните.
    * Връща: Страницата **account/address.html**.
11. **delete(@Validated Address address, BindingResult bindingResult, Model model, Authentication authentication)**:
    * Път: POST **/account/addressDelete**
    * Описание: Изтрива адрес от профила на потребителя.
    * Връща: Страницата **account/address.html**.

**Зависимости:**

Контролерът използва **UserService** за управление на данните за потребителите и техните адреси. Сервисът се инжектира чрез Spring's autowiring.

Тази документация представя основните функции, които **UserController** предоставя, улес

**StoreController** е контролер, използван в Spring MVC архитектурата, за обработка на клиентски заявки свързани с продуктите в магазина. Осигурява функционалност за преглед на продукти и детайли за отделни продукти.

**Функционалности и Операции:**

1. **view(Model model)**:
   * Път: GET **/store**
   * Описание: Показва страница с всички налични продукти, извиква информация за продуктите чрез **ProductService**.
   * Връща: Визуализира страницата **store.html**, където са изброени продуктите.
2. **articleDetail(@PathParam("id") Long id, Model model)**:
   * Път: Request **/productDetails**
   * Описание: Осигурява детайлна информация за конкретен продукт, идентифициран чрез подаденото ID. Използва **ProductService** за извличане на продукта.
   * Връща: Страницата **productDetails**, която показва детайлите на избрания продукт.

**Логиране:**

Класът използва **Logger** от **SLF4J** за записване на информацията, когато потребителят достъпва определени функционалности, например прегледа на магазина.

**Коментари:**

Въпреки че методът **details** за показване на детайлите на продукта е коментиран и заменен с **articleDetail**, важно е да се отбележи, че подходът за обработка на заявки може да варира в зависимост от специфичните нужди на проекта и начина на обработка на потребителски данни и съобщения за грешки.

**Общо Описание:**

**ProductController** е контролер в рамките на Spring MVC, който обработва заявки, свързани с управлението на продукти в административния панел. Той предоставя функционалност за преглед, добавяне, редактиране, обновяване и изтриване на продукти.

**Функционалности и Техните Операции:**

1. **view(Model model)**:
   * Път: GET **/admin/productView**
   * Описание: Показва списък на всички продукти, налични в системата.
   * Връща: Страницата **admin/productView.html** за преглед на продукти.
2. **add(Model model)**:
   * Път: GET **/admin/productAdd**
   * Описание: Подготвя форма за добавяне на нов продукт.
   * Връща: Страницата **admin/productAdd.html** за добавяне на продукт.
3. **addPost(@Validated Product product, BindingResult bindingResult, Model model)**:
   * Път: POST **/admin/productAddUpdate**
   * Описание: Обработва заявката за добавяне на нов продукт в системата и обновява списъка на продукти.
   * Връща: Пренасочва към страницата за преглед на продукти.
4. **edit(@Validated Product product, BindingResult bindingResult, Model model)**:
   * Път: POST **/admin/productEdit**
   * Описание: Подготвя форма за редактиране на съществуващ продукт.
   * Връща: Страницата **admin/productEdit.html** за редакция на продукт.
5. **editPost(@Validated Product product, BindingResult bindingResult, Model model)**:
   * Път: POST **/admin/productUpdate**
   * Описание: Обновява детайлите на даден продукт и ги записва в системата.
   * Връща: Пренасочва към страницата за преглед на продукти.
6. **delete(@Validated Product product, BindingResult bindingResult, Model model)**:
   * Път: POST **/admin/productDelete**
   * Описание: Изтрива продукт по неговото ID и обновява списъка на продукти.
   * Връща: Пренасочва към страницата за преглед на продукти.

**Общо Описание:**

Класът **OrderController** представлява контролер в рамките на Spring MVC, който обработва заявки свързани с управлението на поръчки в административния панел. Контролерът предоставя функционалност за преглед, добавяне, редактиране, обновяване и изтриване на поръчки.

**Методи и Тяхната Функционалност:**

1. **view(Model model)**:
   * Път: GET **/admin/orderView**
   * Описание: Показва списък на всички поръчки.
   * Връща: Името на HTML страницата за преглед на поръчки (**admin/orderView.html**).
2. **add(Model model)**:
   * Път: GET **/admin/orderAdd**
   * Описание: Подготвя форма за добавяне на нова поръчка.
   * Връща: Името на HTML страницата за добавяне на продукт (**admin/productAdd.html**).
3. **addPost(@Validated Order order, BindingResult bindingResult, Model model)**:
   * Път: POST **/admin/orderAddUpdate**
   * Описание: Обработва заявката за добавяне или актуализиране на поръчка.
   * Връща: Пренасочва към страницата за преглед на поръчки.
4. **edit(@Validated Order order, BindingResult bindingResult, Model model)**:
   * Път: POST **/admin/orderEdit**
   * Описание: Подготвя форма за редактиране на съществуваща поръчка.
   * Връща: Името на HTML страницата за редактиране на поръчка (**admin/orderEdit.html**).
5. **editPost(@Validated Order order, BindingResult bindingResult, Model model)**:
   * Път: POST **/admin/orderUpdate**
   * Описание: Обновява детайлите на дадена поръчка и я записва.
   * Връща: Пренасочва към страницата за преглед на поръчки.
6. **delete(@Validated Order order, BindingResult bindingResult, Model model)**:
   * Път: POST **/admin/orderDelete**
   * Описание: Изтрива поръчка по нейното ID.
   * Връща: Пренасочва към страницата за преглед на поръчки.
7. **details(@Validated Order order, BindingResult bindingResult, Model model)**:
   * Път: POST **/admin/orderDetails**
   * Описание: Показва детайли за поръчка, включително списък на артикулите в поръчката.
   * Връща: Името на HTML страницата за детайли на поръчка (**admin/orderDetails.html**).