

Exercise 03 - Handling New Product Events

Required Services

The following services are involved and have to be started before the final exercise validation:

- NotificationSrv (<http://localhost:8010>)
- ProductSrv (<http://localhost:8050>)
- WarehouseSrv (<http://localhost:8070>)

Description

The product management team has decided to establish a new follow-up process when a new product has been added to the DB of the [ProductSrv](#). To make their job easier and reduce manual efforts, several automatic actions have to be performed when a new product has been successfully created via [POST /products](#) ([experiment.webshop.products.resources.ProductResource](#)).

Tasks

The following extensions to the [createProduct\(\)](#) method ([experiment.webshop.products.resources.ProductResource](#)) have to be added as a follow up to a successful finish:

1. **NotificationSrv: Add the product to the internal new product DB.** The [NotificationSrv](#) has an internal DB with new products. Products can be added by invoking [POST /new-products](#) ([experiment.webshop.notifications.resources.NotificationResource](#)). The payload for this method is the newly created [experiment.webshop.products.api.Product](#) instance that is returned from the [storeProduct\(\)](#) method of the [experiment.webshop.products.db.ProductRepository](#). Use the provided Jersey [restClient](#) instance for this. You can copy and adapt one of the existing invocation examples (e.g. the [OrderSrv](#)'s marketing mail request from exercise 1, task 4).
2. **NotificationSrv: Notify the sales department about the new product.** The [NotificationSrv](#) provides functionality for this via [POST /product-mails](#) ([experiment.webshop.notifications.resources.NotificationResource](#)). The payload for this method is an instance of [experiment.webshop.products.api.NewProductMailRequest](#). Use the provided Jersey [restClient](#) instance for this, in the same fashion as for task 1. Below is an exemplary payload ([product](#) will of course be the newly created product):

```
{
  "type": "NEW_PRODUCT_MAIL",
  "product": {
    "id": 1,
    "name": "NewTestProduct"
    "categoryId": 1,
    "price": 9.99
  }
}
```

3. **WarehouseSrv: Stock-up on 10 copies of the newly created product.** As a start, the `WarehouseSrv` needs to have 10 copies of the new product available for purchase. This stock-up process can be initiated by invoking `PUT /products/{id}/availability?amount=10` (`experiment.webshop.warehouse.resources.WarehouseResource`). Use the provided Jersey `restClient` instance for this, in the same fashion as for task 1 and 2. Since there is no payload (only the URL parameter `amount`), you need to use an empty string payload as a workaround like so:

```
Invocation.Builder request = restClient.target(url).request();
BaseResponse response = request.put(Entity.json(""), BaseResponse.class)
```

Validation

When you are finished with all tasks, make sure all required services (see [Required Services](#)) and the exercise validation UI is up and running (if not, execute `exercise-validation/build-and-run-validation-ui.bat`) and then navigate to `http://localhost:5001`. Click on **Exercise 03** and then on **Start Validation**. If every check is successful (`status: true`), pause your stopwatch and notify an experiment admin to write down your time.