Grocery Example
Notes
Rob Lucas

Table of Contents

- Table of Contents
- Overview
- Architecture
  - Technology Versions
  - REST API Layer
  - Service Layer
  - Persistence Layer
  - Database Schema
  - POJOs
- Running the App
- Test the App
  - Testing with Postman
- Generate War
- API Doco

Overview

This documents include notes relating to the Grocery Example project put together for Landmark Information Group.

Architecture

Technology Versions

- Jersey - 2.14
- Jackson - 1.9.13
- Spring/POJOs - 4.1.4
- JPA2/Hibernate - 4.2.7
- Jetty
- MySQL - 5.6.24
- Maven 3

REST API Layer

REST API Facade layer uses Jersey, a JAX-RS reference implementation.

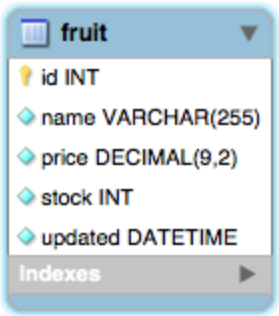
Service Layer

Service Layer contains business logic and is implemented using Spring.

Persistence Layer

JPA2/Hibernate using JPA2 EntityManager with Hibernate. MySQL database. SQL script to generate database can be found in grocery.sql.

Database Schema



POJOs

Two representations of fruit are used:

FruitEntity.java - JPA annotated class used in the DB and business layers
Fruit.java - JAXB/JSON annotated classed used in facade and business layer

Running the App

- Clone/import project from GIT
- Import the grocery.sql file into MySQL.
- Add user to database and give acces to grocery database (grocery_user/grocery_password)
- Go to project folder and excute:

- > mvn clean install jetty:run -Djetty.port=8888 -DskipTests=true
- App will be available at http://locahost:8888/grocery

Test the App

Run unit and integration tests.

> mvn clean install verify -Djetty.port=8888

Testing with Postman

You can use the Postman Chrome plugin to test the API's in this application. You can import the src/test/resources/postman/grocery.json file into Postman and this contains a list of Postman test set to make calls to localhost:8888.

Generate War

> mvn install

API Doco

Rough API documentation - to be fleshed out and improved.

/grocery/fruit/load
POST

Add fruit to the store. Remove any existing fruit first.

Example JSON

```
[{
	"name" : "banana",
	"price" : 0.29,
	"stock" : 20
}, {
	"name" : "melon",
	"price" : 1.01,
	"stock" : 3
}]
```

/grocery/fruit/
GET

Get all fruit.

PARAMS
orderByUpdateDate = ASC or DESC

/grocery/fruit/{id}
POST

Update the values of a fruit (e.g. price) where {id} is the database ID of the fruit.

Example JSON

```
{
  "id": "3",
  "price": "3.4"
}
```

/grocery/fruit/fruitByName
GET

Get fruit by name

PARAMS
name = name of fruit

/grocery/fruit/{id}
DELETE

Delete the fruit with the specified ID

/grocery/fruit
DELETE

Delete all fruit

/grocery/fruit
POST

Add one fruit

Example JSON

```
{
	"name": "peach",
	"price": "2.30",
	"stock": "5"
}
```

/grocery/fruit/{id}
GET

Get the fruit with the specified ID.

Response Format

```
{
	"id": "3",
	"name": "peach",
	"price": "2.30",
	"stock": "5"
	"updated": "2015-04-26T16:44:07.00+0100"
}
```