Interface Spec and Design

Table of Contents

[1 Overview 3](#_Toc448903961)

[2 Design Concept 3](#_Toc448903962)

[3 Technical Details 5](#_Toc448903963)

[4 Sample Request 5](#_Toc448903964)

# Overview

This is a basic pizza ordering application which has exposed functionality through REST APIs. It supports some basic operations for seeing catalog, calculate total amount, confirming order from user’s point of view. Store manager will be able to view current inventory items. Store manager can update an item’s quantity and add new items as well. Store manager will also be able to add promotions.

Below is a list of available APIs with brief details.

**End-Point 🡪 /api/pizzastore**

|  |  |  |  |
| --- | --- | --- | --- |
| **API Path** | **Permission** | **Operation** | **Description** |
| /orders | ROLE\_USER,  ROLE\_ADMIN | GET | This will return a list of available items including base, toppings, spread etc. |
| /orders/showTotalAmt | ROLE\_USER,  ROLE\_ADMIN | POST | It will calculate and return totalAmt for given List of items and promotion if any. |
| /orders/confirmOrder | ROLE\_USER,  ROLE\_ADMIN | POST | It submits request and returns totalAmt and order confirmation number. |
| /inventories | ROLE\_ADMIN | GET | Returns all List of Items, available in inventory with their current count and price. |
| /inventories/{itemId} | ROLE\_ADMIN | GET | Returns Item for request itemId. |
| /inventories/addItem | ROLE\_ADMIN | POST | It allows store manager to add new items in catalog. |
| /inventories/udpateItem | ROLE\_ADMIN | PUT | It allows store manager to update quantity of any given item in inventory. |
| /promos/addPromo | ROLE\_ADMIN | POST | Store manager can add new promo using this API call. |
| /promos/{promocode} | ROLE\_ADMIN | GET | Return all details of requested promocode like promo Amt, description etc. |

# Design Concept

This application is broadly divided into three major modules (packages).

Interface

DB

Entity

Service

Interface (package com.craftexercise.interfac)–

This package includes all request and response objects including DemoController. Idea is anything exposed to outer world should be package here. This package also include a DemoConverter which converts all interface objects into internal domain/entity representing bean objects.

DemoController is a place where each incoming request will land first. Before moving ahead Spring security will perform authentication and authorization activity to make decision if request user is allowed or not.

Each request is annotated with JSR-303 validation, appropriately.

Service (package com.craftexercise.service)–

Service module/package is keep all business related logic which be core part of application. Service layer will perform business validation and any conditional checks. Once all business check are completed successfully calls will move to Entity/Repository to perform DB operations.

Entity/Repository (package com.craftexercise.entity) –

This part is only to interact with Database. This will be CRUD operation source. Spring’s jdbcTemplate is being used here.

Database

To keep it simple for now, in-memory database HSQLDB is used. For now there are three tables created for basic activities. Table details are as follows -

|  |  |  |
| --- | --- | --- |
| **ITEM** | | |
| **ColumnName** | **DataType** | **Description** |
| ID | INTEGER | Primary Key, using HSQLDB’s feature to generate new keys. |
| NAME | VARCHAR(50) | Name of item e.g. BaseLarge, Onion, checken |
| PRICE | DOUBLE | Price for each item |
| QUANTITY | INTEGER | Current count of items in inventory |
| CATEGORY | Varchar(100) | Type of item like base, toppings etc. |

|  |  |  |
| --- | --- | --- |
| **PROMO** | | |
| **ColumnName** | **DataType** | **Description** |
| ID | INTEGER | Primary Key, using HSQLDB’s feature to generate new keys. |
| PROMO\_CODE | VARCHAR(50) | Promocode to be used by users. |
| PROMO\_AMT | DOUBLE | Price of PROMO |
| NOTES | Varchar(50) | Any comments for this promo. |

|  |  |  |
| --- | --- | --- |
| **ORDER** | | |
| **ColumnName** | **DataType** | **Description** |
| ID | INTEGER | Primary Key, using HSQLDB’s feature to generate new keys. |
| ITEM\_IDS | VARCHAR(150) | Comma separate list of item Ids for each order. |
| PROMO\_ID | INTEGER | Promo\_id used for this order |
| PRICE | DOUBLE | Total Amount for this order |
| ORDER\_NUM | INTEGER | Order number to be shared with customer. |

# Technical Details

This application is developed in Java. Below are some key tools to mentions -

**Maven build** – To build and package maven is being used. We can use below commands to build and deploy.

From folder exercise. Type below commands to build and deploy.

/>**mvn clean package**

This will clear and package new jar.

/>**java -jar target/exercise-0.0.1-SNAPSHOT.jar**

This will deploy application tomcat.

**Spring Boot** – To start application standalone. This will include web-container library as well to deploy web application. This also helps to avoid XML configuration and allows annotation for all config related activities.

**Spring Security** – Used basic features to perform user validation check.

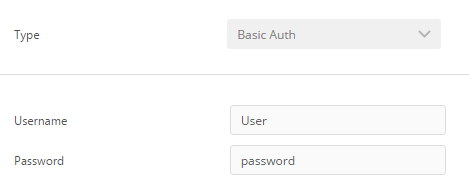
**HSQLDB** – In memory DB for initial DB related activities. Script files have been included in src/main/resources folder. On application startup these scripts will run.

**TestNG** & **Mockito** – Used these libraries to write and run unit test cases.

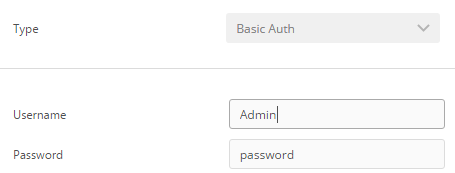
# Sample Request

**Chrome Postman** – Using Postman to call REST APIs. Under Documents folder DemoApplication.json.postman\_collection is collection of sample requests. Collection is accessiblehere - <https://github.com/mdsworld/CraftExercise/blob/master/Documents/DemoApplication.json.postman_collection>

To call APIs as **Customer/User** below details should be in use. User has ROLE\_USER permission. Please use “**Basic Auth**” from drop-down and provide these details-



To call as **store manager**, below should be in use. Admin has ROLE\_ADMIN permission. Please use “**Basic Auth**” from drop-down and provide these details-



Here are some sample request with end points –

1. **getCatalog items – call type GET**

http://<hostname>:8080/api/pizzastore/orders

1. **showTotalAmt for selected items – call type POST**

http://<hostname>:8080/api/pizzastore/orders/showTotalAmt

|  |
| --- |
| {  "itemIds":[  101,  102,  100  ],  "promoCode":"PROMO02"  } |

1. **confirmOrder for selected items – call type POST**

http://<hostname>:8080/api/pizzastore/orders/confirmOrder

|  |
| --- |
| {  "itemIds":[  101,  102,  100  ],  "promoCode":"PROMO02"  } |

1. **List all inventory Items – call type GET**

http://<hostname>:8080/api/pizzastore/inventories

1. **Check a particular inventory items – call type GET**

http://<hostname>:8080/api/pizzastore/inventories/101

1. **Add new item in inventory – call type POST**

http://<hostname>:8080/api/pizzastore/inventories/addItem

|  |
| --- |
| {  "name": "SunDriedTomato",  "price": 1.4,  "category": "Topping",  "quantity": 30  } |

1. **Update/Reset quantity for inventory item – call type PUT**

http://<hostname>:8080/api/pizzastore/inventories/updateItem

|  |
| --- |
| {  "itemId": 100,  "quantity": 3  } |

1. **Add new promo**

http://<hostname>:8080/api/pizzastore/promos/addPromo

|  |
| --- |
| {  "promoCode": "PROMO05",  "promoAmt": 3.5,  "description": "Save $3.5 now"  } |

1. **Check a promo details**

http://<hostname>:8080/api/pizzastore/promos/PROMO02