Cognizant Academy

truYum

Spring Core Specification Document

Version 1.0

	Prepared By / Last Updated By	Reviewed By	Approved By
Name	Chandrasekaran Janardhanan	Vimalathithan Krishnan	Ramadevanahalli Lingachar, Shashidhara Murthy
Role	Learning Solution Designer	Learning Solution Architect	Learning Solution Lead
Signature			
Date			

Table of Contents

1.0 Introduction

1.1 Purpose of this document

The purpose of this document is to define the Java class related implementation for truYum project.

1.2 Definitions & Acronyms

Definition / Acronym	Description

1.3 Project Overview

Refer truYum-use-case-specification.docx for understanding the functionality and features.

1.4 Scope

Creation of model and data access object classes for truYum application

1.5 Intended Audience

- Product Owner
- Scrum Master
- Application Architect
- Project Manager
- Test Manager
- Development Team
- Testing Team

1.6 Hardware and Software Requirement

1. Hardware Requirement:
    a. Developer PC with 4GB Ram
2. Software Requirement
    a. Git

b. JDK 1.8

c. Eclipse IDE for Enterprise Java Developers 2019-03 R

1.7 Eclipse Project Configuration

Earlier we created the classes in Eclipse project named “truYum”. We will create a new project named “truyum-spring”, which will contain Spring related configuration and bean loading. Find below the steps to create this new project. All the specification defined in this document needs to be implemented in this new Eclipse project.

1. In Windows File Explorer go to the Eclipse Workspace folder. Enter "cmd" in the address bar which will open the command prompt in that folder.

2. In the command prompt, execute the following maven command which creates a quickstart project:

```
mvn archetype:generate -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -
Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=125546 -
Dhttp.proxyPassword=nov2019$ -DgroupId=com.cognizant.truyum -DartifactId=truyum-spring -
DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4 -
DinteractiveMode=false
```

3. Import the newly created maven project into Eclipse following the steps below:

   a. Open Eclipse

   b. File > Import... > Maven > Existing Maven Projects

   c. Select the "truyum-spring" folder of the newly created maven project

   d. Click Finish

4. The above steps creates a new project in Eclipse with name "truyum-spring"

5. Open pom.xml of this new project and include the below dependency inside the <dependencies> tag.

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>5.2.3.RELEASE</version>
</dependency>
```

6. If required change the version of spring to the latest available version.

7. Go back to command prompt. Use the command "cd truyum-spring" to change to the maven project folder, then execute the below command to download dependencies and compile the code.

```
mvn clean compile -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -
Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456
```

8. The above command should create a new folder named "target" in "truyum-spring" folder. Within "target" folder classes folder will contain the compiled class files.

9. Open Eclipse > Right click on "truyum-spring" > Maven > Update Project. This will update the maven project and synchronize the depdendencies. This will resolve the errors in this Eclipse project.

10. Copy files from earlier "truYum" java project into the new "truyum-spring" project. Final project structure in truyum-spring project should look like this:

    Note:

    1. truYum Java project code base is present in the sharepoint link provided.

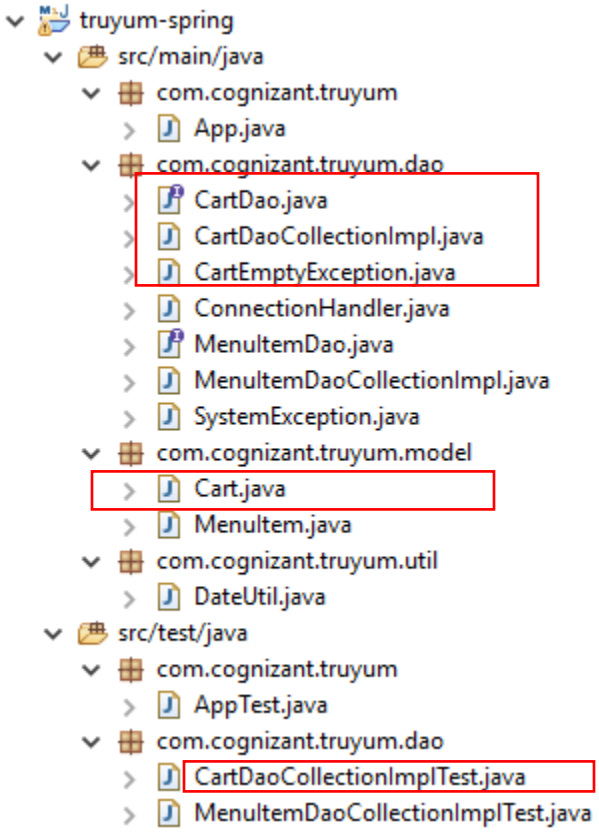
    2. Below mentioned use cases only needs to be worked on.Classes highlighletd below needs to be deleted.

        a. View Menu Item List Admin (TYUC001)

        b. View Menu Item List Customer (TYUC002)

        c. Edit Menu Item(TYUC003)

        d.



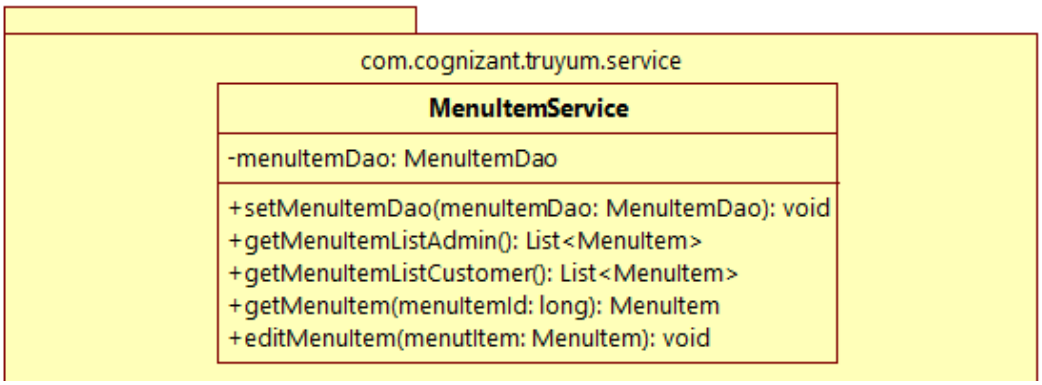
2.0 Class Diagram

We will introduce the Service layer in this project, which acts as a façade for data access. A new

package needs to be created for service. The objective of having this layer is to introduce business logic. Any application or client will be accessingly only the service layer methods and not the dao layer methods.

2.1 Service package

Find below the classes that will be created as part of service layer.



3.0 Design for View Menu Item List Admin (TYUC001)

As part of this use case we will implement the following aspects:

1. Menu Items list is currently hard coded in MenuItemDaoCollectionImpl. This data should be loaded from spring configuration xml file (spring-config.xml)
2. Create a new package for service with new class MenuItemService and implement the method to get the menu items from Dao.
3. Autowire MenuItemService and MenuItemDao.
4. Unit Test the new method in MenuItemService.

3.1 src\spring-config.xml

Create a new spring file named "spring-config.xml" in src folder. We will use this configuration file to hold the menu item list static data. Currently MenuItemDaoCollectionImpl holds hard coded data of menu items. Now this data will be loaded from this spring configuration file. Include the following aspects in this xml file.

1. Create 5 beans in spring-config.xml file with id as "sandwich", "burger", "pizza", "fries", "brownie" representing each menu item.
    a. Use <property> tag to set the MenuItem bean attributes.
    b. Use SimpleDateFormat to create Date instance for dateOfLaunch attribute. Refer example here (use the SimpleDateFormat and factory-bean example).
2. Create array list of MenuItems based on the instructions provided below.
    a. Create <bean> tag with id as "menuItems" and class as fully qualified name of ArrayList.
    b. Include <constructor-arg> tag within the above <bean> tag.
    c. Include <list> table within the above <constructor-arg> tag.
    d. Include each menu item using <ref> tag. One example menu item reference xml entry provided below:

        <ref bean="sandwich"/>

        NOTE: In the above code "sandwich" refers to the bean id provided earlier.
3. Refer diagram below for actual data for menu items:

3.1 MenuItem.java

Include a empty parameter constructor without any implementation so that it enables spring to create instance of MenuItem.

3.2 MenuItemDaoCollectionImpl.java

Implement following modifications, so that it can be loaded from spring xml configuration file.

1. Remove static for menuItemList member variable and make it an instance variable

2. Remove the existing constructor

3.3 MenuItemService.java

Class for managing data of menu items using Java Collections Framework.

1. Define the @Service annotation at class level with name “menuItemService”. The definition of the name will help to load the class in JUnit testing.

2. Create an instance variable for MenuItemDao

3. Autowire MenuItemDao with MenuItemDaoCollectionImpl using annotation

   HINT: Use Interface Injection technique to map instance variable MenuItemDao to MenuItemDaoCollectionImpl.

getMenuItemListAdmin(): List<MenuItem>

This method returns the list of menu items that will be displayed in the MenuItem listing screen for Admin.

1. Invoke getMenuItemListAdmin() method in menuItemDao and return the same.

4.0 Design for View Menu Item List Customer (TYUC002)

4.1 MenuItemService.java

getMenuItemListCustomer() : List<MenuItem>

Invoke getMenuItemListCustomer() from MenuItemDao and return the same.

5.0 Design for Edit Menu Item (TYUC003)

5.1 MenuItemService.java

1. Add method `modifyMenuItem(menuItem: MenuItem): void` which invokes the respective dao method.

2. Add method `getMenuItem(menuItemId: long): MenuItem` which invokes the respective dao method.

6.0 Standards and Guidelines

6.1 Java

1. Ensure that the class names, method names and variable names are followed exactly as specified in the class diagram
2. Ensure that access modifier are in line with the class diagram specification
3. Naming standards to be followed:
    a. Variable
        i. Should be in mixed case with the first letter lowercase and with the first letter of each internal word capitalized (Example: firstName, dateOfBirth)
        ii. Variable names should be short, but meaningful
        iii. Variable name defined should indicate the purpose to a casual observer
        iv. Single character variable names should be avoided except for temporary variables
        v. Temporary variables include i, j, k and m
    b. Class
        i. Class name should be a noun
        ii. Class name should be in mixed case with the first letter uppercase and with the first letter of each internal word capitalized
        iii. Must use whole words and should not have acronyms or abbreviations

        Examples: Employee, TaxCalculator
    c. Method
        i. Method names should be verbs
        ii. Method names should be in mixed case with the first letter lowercase and with the first letter of each internal word capitalized

        Example: changeGear(), calculateBalance()
4. Code Formatting
    a. Class Structure
        i. Place the elements of a class in the following order:
            1. Static variables
            2. Instance variables
            3. Constructors
            4. Methods and Getter/Setters
            5. hashCode(), equals(), toString,
    b. Spacing
        i. A space before and after an operator is required

ii. A space before curly braces is required
iii. A space after a comma is required
iv. A space after semicolon in for loop is required
v. A single line space after a method is required

c. Curly braces position
i. Opening curly braces should be in the same line
ii. Closing curly braces should always be in a new line

d. Tab spacing
i. Use 4 spaces instead of tab character
ii. Increase a tab character in the lines after opening curly braces
iii. Reduce a tab character on the of closing curly braces
iv. Include one more tab in the wrapped line

e. Line Width
i. Width of a line should not exceed 100 characters

7.0 Change Log

<table>
<tr><th></th><th colspan="4">Changes Made</th></tr>
<tr><td>V1.0.0</td><td colspan="4">Initial baseline created on <dd-Mon-yy> by <Name of Author></td></tr>
<tr><td rowspan="3">Vx.y.z</td><td colspan="4"><Please refer the configuration control tool / change item status form if the details of changes are maintained separately. If not, the template given below needs to be followed></td></tr>
<tr><td>Section No.</td><td>Changed By</td><td>Effective Date</td><td>Changes Effected</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>