



­

Assignment Document: Core Spring

Version: Core Spring Practice Case study 6/ASSIGNMENT/1.0

Date: 11-05-2015

Cognizant

500 Glen Pointe Center West

Teaneck, NJ 07666

Ph: 201-801-0233  
[www.cognizant.com](http://www.cognizant.com)

Table of Contents

[Shopper Stop 2](#_Toc452972635)

[1. Infrastructure Section 2](#_Toc452972636)

[I. Hardware, Software Specification 2](#_Toc452972637)

[2. Case Study Assignments 3](#_Toc452972638)

[3. Instructions 4](#_Toc452972639)

[4. Ddl/Dml for Creating Tables and Inserting Data 4](#_Toc452972640)

[5. Purchase Product 5](#_Toc452972641)

[6. Technical Specifications 5](#_Toc452972642)

[7. Business Rules& Validations 7](#_Toc452972643)

[8. Sequence Diagram 10](#_Toc452972644)

[9. View Sales Report 10](#_Toc452972645)

[10. Technical Specifications 11](#_Toc452972646)

[11. Business Rules & Validations 12](#_Toc452972647)

[12. Sequence Diagram 12](#_Toc452972648)

[13. Solution 12](#_Toc452972649)

[14. Evaluation Rubrics 13](#_Toc452972650)

[15. Summary of this Case Study 13](#_Toc452972651)

# Shopper Stop

1. Infrastructure Section
2. Hardware, Software Specification

This section captures the hardware and software specifications for the effective delivery of the course.

1. **Hardware Specification**

|  |  |
| --- | --- |
| **Server Specification** | Windows 7 - 32bit  4GB RAM |
| **Desktop / Client Specification** | Admin Rights |

**b. Software** **Specification**

| **#** | **Name of the software(s) to be installed** | **Version** | **License available in RAMS?**  **(Yes/No)** | **License available in RAMS for onsite?**  **(Yes/No)** | **Description of any patch to be installed** | **Can be used through Tool Wire / SoftGrid?** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | JDK | 7.0 | Yes | Yes |  | Yes |
| 2 | SDE 7.0 | 7.0 | Yes | Yes |  | Yes |
| 3 | Tomcat 6.0/7.0 | 6.0/7.0 | Yes | Yes |  | Yes |
| 4 | MySQL Workbench | 6.0.8 | Yes | Yes |  | Yes |
| 5 | Netbeans 6.9.1- Complete Installation (Including Derby database and Tomcat Server) | 6.9.1 | Yes | Yes |  | Yes |
| 6 | Eclipse | 3.6 | Yes | Yes |  | Yes |

**Instruction for installing the software in the “Tool Wire/SoftGrid” environment *(if the response is ‘Yes’ in the last column in the above table)*:**

| **#** | **Name of the software** | **Instruction** |
| --- | --- | --- |
| 1 | Spring Jars 3.0 | <http://www.java2s.com/Code/Jar/s/Downloadspringweb310RELEASEjar.htm> |
| 2 | Mysql-connector-java-5.1.12-bin | <http://www.java2s.com/Code/Jar/m/Downloadmysqlconnectorjava5123binjar.htm> |
| 3 | spring-hibernate3-2.0.8 | <http://www.java2s.com/Code/Jar/s/Downloadspringhibernate3208jar.htm> |
| 4 | Hibernate 3.5 jars | <http://www.java2s.com/Code/Jar/h/Downloadhibernate353jar.htm> |

**Installation Details:**

Refer the below link for JDK and JRE installation:

<http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html>

Refer the below link for SE download and installation:

<https://gto.cognizant.com/javacoe/SitePages/SDEDownload.aspx>

**Test Cases to check the installed software’s:**



1. Case Study Assignments

Estimated Completion Time: xx Minutes

Objective:

Shoppers Stop wants to enhance its business via online. So it wants to automate the process of purchasing products online for Customers who are interested in online shopping. Customers can access their website and purchase products that are available and the Shoppers Stop admin can view the Sales Report of the different products.

It would like to develop the below components.

**Service1: Purchase product**

**Service2: View Sales Report**

**Control flow is Tester --> Manager --> Façade --> BO --> Dao**

1. Instructions

* Please make sure that JAVA\_HOME is set as your JDK installation folder, if not, please set as the following: ***C:\Program Files\Java\jdk1.6.0\_28***
* Download the artifacts (Skeleton) from the tool. Unzip the zip files downloaded on your desktop.
* Open Eclipse IDE and import the project using “**Import -> Existing Projects into workspace**” option. This acts as your code base to begin the case study development.
* Once the skeleton code is imported, some new compilation errors would have appeared in the project. These compilation errors are because of the skeleton code that may contain classes with methods which are expected to return some values, but the return statements are provided.
* Once you build the skeleton code with the necessary code as per you case study requirement and return the expected data type or throw the expected exceptions, the compilation errors would get resolved.
* It contains the partially implemented project which includes, Tester class, DetailsVO, RegistrationVO, HESBO class etc.
* Refer to **HES.sql** file that is present in same folder as your case study document.
* Copy and paste the table creation scripts and then Execute in MySQl command prompt or refer to the “User Manual for Mysql” in the URL provided in the **Instructions** section to run the **.sql** file, before you start the case study.
* Import the project in the workspace and add the necessary jar files in the build path.
* Check all the necessary classes as specified in the question paper. Spring configuration file **hes\_config.xml** is also created and given.
* Submit the entire project, but do not submit the whole workspace.
* Steps to submit are:

1. Copy the entire project. After zipping it, place it into a folder with <your emp no>
2. Submit the folder with <your emp no>.
3. DDL/DML for creating tables and inserting data
4. **Table Description**
   1. Table Name: **Product**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraint** |
| PRODUCTCODE | INT(6) | PRIMARY KEY |
| PRODUCTNAME | VARCHAR(40) | NOT NULL |
| PRODUCTPRICE | DOUBLE | NOT NULL |
| NOOFSTOCK | INT(6) | NOT NULL |
| CATEGORY | VARCHAR(40) | NOT NULL |

* 1. Table Name: **PurchaseOrder**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraint** |
| PURCHASEORDERNO | INT(5) | PRIMARY KEY |
| CUSTOMERNAME | VARCHAR(30) | NOT NULL |
| EMAILID | VARCHAR(30) | NOT NULL |
| PRODUCTCODE | INT(6) | FOREIGN KEY |
| QUANTITY | INT(3) | FOREIGN KEY |
| AMOUNT | DOUBLE | NOT NULL |
| DATEOFPURCHASE | DATE | NOT NULL |

The service is used to automate the process of purchasing products online from Shoppers Stop. This service is going to become a part of different other modules in the system and will be developed as an independent module, so that it can be plugged into other modules easily.

1. Purchase product

This service accepts the customer details and product and quantity to be purchase for online shopping. If the below business rules are satisfied, then persists the data and returns an Integer value (purchaseOrderNo). If any of the business rules is not satisfied, throws an appropriate user defined exception as given in the table below:

1. Technical Specifications

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Method Name** | **Input** | **Output** | **Exception** |
| ShoppingManager | **purchaseProduct()** | **PurchaseOrderVO purchaseOrderVO** | ReturnsInteger value  purchaseOrderNo | InvalidEmpNameException  InvalidEmailException  InvalidPhoneNoException  InvalidUnitCodeException  These exceptions have to be caught and thrown back to Tester class. |
| ShoppingFacade | **purchaseProduct()** | **PurchaseOrderVO purchaseOrderVO** | ReturnsInteger value  purchaseOrderNo | InvalidEmpNameException  InvalidEmailException  InvalidPhoneNoException  InvalidUnitCodeException  These exceptions have to be caught and thrown back to ShoppingManager class. |
| ShoppingBO | **purchaseProduct()**\*\* check for the following business rules given below | **PurchaseOrderVO purchaseOrderVO** | ReturnsInteger value  purchaseOrderNo | InvalidEmpNoException  InvalidNameException  InvalidLocationIdException  InvalidEmergencyNoException  InvalidEmployeeException  These exceptions have to be caught and thrown back to ShoppingFacade class. |
| ShoppingDao interface | **getProductDetails()** | Integer productCode |  |  |
| **purchaseProduct()** | **PurchaseOrderVO purchaseOrderVO** |  |  |
| ShoppingDaoImpl | **getProductDetails()** | Integer productCode | Returns the corresponding Product object for the productCode if valid, null if invalid |  |
| **donateAmount()** | **PurchaseOrderVO purchaseOrderVO** | You need to update the **noOfstock** of **Product** table by deducting the quantity for the productCode which you are purchasing.  Persists the data in the **PurchaseOrder** table and returnsInteger value  **purchaseOrderNo** |  |
| shoppers\_config.xml | Contains all the xml configurations related to Service1 |  |  |  |

1. Business Rules& Validations

In the **ShoppingBO class** check for the following business rules:

|  |  |  |
| --- | --- | --- |
| **Rule**  **No.** | **Rule Description** | **User Define Exception to be thrown** |
| 1. | **customerName** entered should contain only alphabets and spaces and should not contain any digits and special characters | If empName is invalid, throws  **InvalidCustomerNameException.** |
| 2. | **emailId** entered should be valid | If emailId is invalid, throws  **InvalidEmailException.** |
| 3. | **productCode** entered should be valid [ getProductDetails(Integer productCode) method of ShoppingDao. Method returns Product object if productCode is valid , null if valid] | If **locationId** is invalid, throws  **InvalidProductException.** |
| 4. | **quantity** entered should be greater than **noOfstock** for the product | If **phoneNo** is invalid, throws  **InvalidQuantityException.** |

If any of the business rules is not satisfied, throw an appropriate user defined exception as given in the above table.

**Note:** The PurchaseOrderVO, Product, PurchaseOrder class will be provided with the required getter and setter methods which should not be modified. No changes should be done to the skeletons provided (Especially the name and method signature).

**Limitations and Constraints**

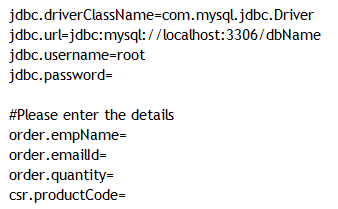
1. All the java classes to be declared as beans in spring configuration file **shopping\_config.xml.**
2. **ShoppingFacade** should be integrated **ShoppingManager** using Setter Based Dependency Injection.
3. **ShoppingBO** should be integrated with **ShoppingFacade** using Constructor based dependency injection.
4. **ShoppingDao** should be integrated with **ShoppingBO** using Setter Based Dependency
5. Use ONLY **shopping\_config.xml** for all configurations related to Service1.
6. Create two properties files messages\_en\_US.properties, messages\_fr.properties for English and French locale.
7. Read the **input data** and **Data Source** details from the **constants.properties.**
8. Enable transactions by configuring transaction manager.

**<SQL Connection details will be provided to the associate as below>**

Create a **constant.property** file with the following details. Please set the value for DBName, password and also set the values to purchase product.

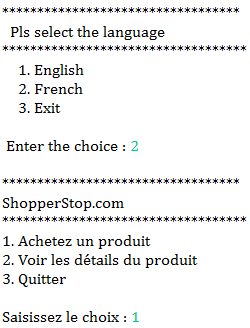
Please enter the below details:

**Note:** User inputs will be entered in the **constant.property file**

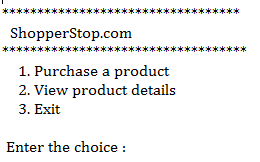


**Output**

French locale

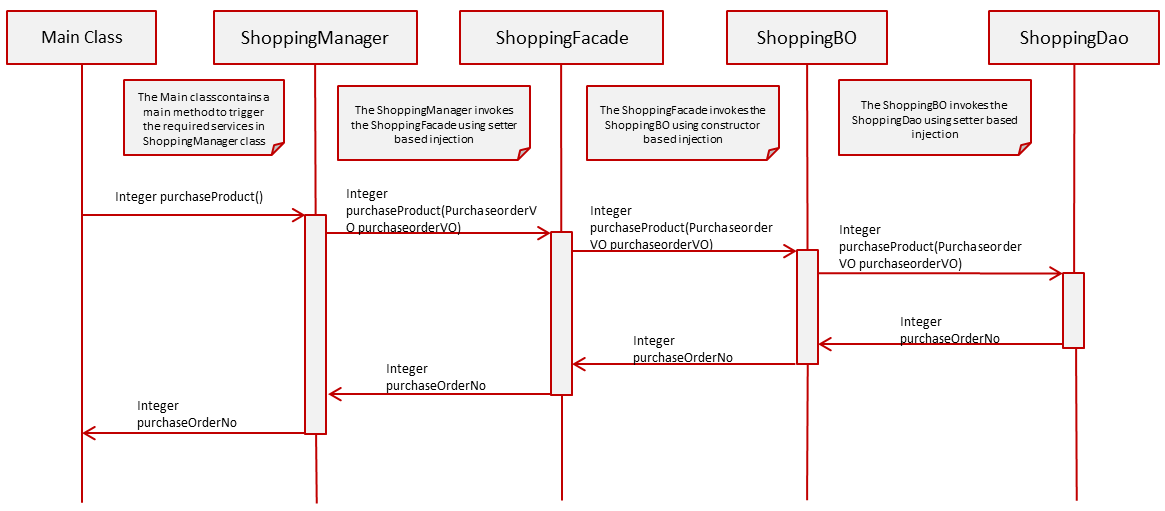


English locale



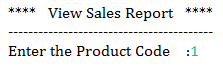


1. Sequence Diagram

****

1. View Sales Report

* When the user selects Option 2**,** viewSalesReport() **method of Tester** is invoked.
* The Shopping admin will input the **productCode.**
* The View Sales Report menu in the viewSalesReport()method of Tester class looks like below:



Control flow is **Tester --> Manager --> Façade -->BO -->Dao**

* In the **Tester** Class,
* Invoke ViewSalesReport (Integer productCode)from Tester to **ShoppingManager** following the Control flow mentioned above. This method returns a list of type **DetailsVO** objects.
* Write code to display the output in the below format by using the returned values if the business rules are satisfied, otherwise display the appropriate exception message.

**[Hint:** While displaying, to display dateOfPurchase in proper Date Format (dd-MMM-yyyy), use **DateUtility.getStringFromDate (Date date)** to convert dateOfPurchase from Date to String**]**

**OUTPUT**



1. Technical Specifications

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Method Name** | **Input** | **Output** | **Exception** |
| ShoppingManager | **viewSalesReport()** | Integer productCode | List<DetailsVO> | NoRecordException  This exception has to be caught and thrown back to Tester class. |
| ShoppingFacade | **viewSalesReport()** | Integer productCode | List<DetailsVO> | NoRecordException  This exception has to be caught and thrown back to ShoppingManager class. |
| ShoppingBO | **viewSalesReport()** | Integer productCode | List<DetailsVO> | NoRecordException  This exception has to be caught and thrown back to ShoppingFacade class. |
| ShoppingDao interface | **viewSalesReport()** | Integer productCode | List<DetailsVO> |  |
| ShoppingDaoImpl | **viewSalesReport()** | Integer productCode | List<DetailsVO> |  |
| shoppers\_config.xml | Contains all the xml configurations related to Service1 |  |  |  |

1. Business Rules & Validations

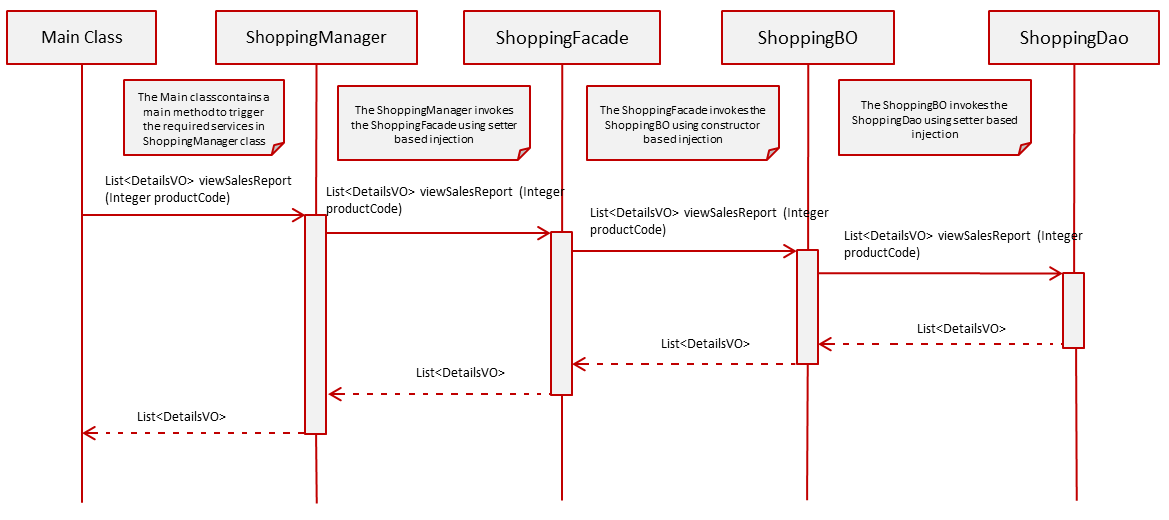
In the **ShoppingBO class** check for the following business rules given below

|  |  |  |
| --- | --- | --- |
| **Rule**  **No.** | **Rule Description** | **User Define Exception to be thrown** |
| 1. | List<DetailsVO> returned by the method **viewSalesReport** () of ShoppingDao class should not be empty. | If list is empty  **NoRecordException** |

If any of the business rules is not satisfied, throw an appropriate user defined exception as given in the above table.

**Note:** The DetailsVO class will be provided with the required getter and setter methods which should not be modified. No changes should be done to the skeletons provided (Especially the name and method signature).

1. Sequence Diagram

****

1. Solution



1. Evaluation Rubrics

|  |  |
| --- | --- |
| Parameters | Weightage |
| 1. Completeness |  |
| 1. Accuracy |  |
| 1. Clarity of understanding |  |
| 1. Presentation |  |
| Total |  |

1. Summary of this Case Study

You have just learnt:

* Spring Frame Work and IOC Container
* Spring Dependency Injection
* Spring JDBC Framework
* Spring Transaction Management
* ResourceBundleMessageSource