



­

Assignment Document:

Core Spring

Version: Core Spring Practice CaseStudy 4/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)

Contents

[Cognizant ID Card Service 2](#_Toc420065332)

[1. Infrastructure Section 2](#_Toc420065333)

[i. Hardware, Software Specification 2](#_Toc420065334)

[2. Case Study Assignments 4](#_Toc420065335)

[3. Instructions 4](#_Toc420065336)

[4. DDL/DML for creating tables and inserting data: 5](#_Toc420065337)

[5. Register for ID Card: 5](#_Toc420065338)

[6. Technical Specifications: 5](#_Toc420065339)

[7. Business Rules& Validations: 7](#_Toc420065340)

[8. Sequence Diagram 9](#_Toc420065341)

[9. View Registered Id Details: 9](#_Toc420065342)

[10. Technical Specifications: 10](#_Toc420065343)

[11. Business Rules & Validations: 10](#_Toc420065344)

[12. Sequence Diagram 11](#_Toc420065345)

[13. Solution: 11](#_Toc420065346)

[14. Evaluation Rubrics 12](#_Toc420065347)

[15. Summary of this Case Study: 12](#_Toc420065348)

# Cognizant ID Card Service

1. Infrastructure Section

i. 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:

Cognizant Facilities Team wants to automate the process of applying for ID card for its employees in case if their id card is damaged or in case of change in their details displayed in ID card. Employees can register through the system and the admin corresponding to the system can take a report of the same.

They would like to develop the below components.

**Service1: Register for ID card**

**Service2: View Registered Id Details**

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

1. Instructions

* Please make sure that JAVA\_HOME is set your JDK installation folder, otherwise please set as following: ***C:\Program Files\Java\jdk1.6.0\_28***
* Download the artifacts (Skeleton) from tool. Unzip the zip files downloaded on your desktop.
* Open Eclipse IDE and import the below 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 the skeleton code 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 these compilation errors would get resolved.
* It contains the partially implemented project which includes Tester class, DetailsVO, IDCardBO, RegistrationVO class etc.
* Refer to IdCard.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 idCard\_config.xml is also created and given.
* Submit the entire project. Do not submit the whole workspace. Steps to submit are:

1. Copy the entire project and 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: **LOCATION**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraint** |
| locationId | NUMBER(2) | PRIMARY KEY |
| locationName | VARCHAR2(10) | NOT NULL |
| adminEmail | VARCHAR2(20) | NOT NULL |
| adminDesk | VARCHAR2(25) | NOT NULL |

* 1. Table Name: **EMPLOYEE**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraint** |
| empNo | NUMBER(6) | PRIMARY KEY |
| empName | VARCHAR2(15) | NOT NULL |
| emergencyNo | VARCHAR2(10) | NOT NULL |
| locationId | NUMBER(2) | FOREIGN KEY |
| registrationDate | DATE | NOT NULL |

The service is used by the system to register employee for ID Card. This service is going to become a part of different other modules in the system so have been asked to develop the Service as an independent module so that it can be plugged into other modules easily.

1. Register for ID Card:

This service accepts the employee details and register for ID Cards. If the above business rules are satisfied, then persist the data and will return an Integer value (registrationId). If any of the business rules is not satisfied, throw an appropriate user defined exception as given in the below table

1. Technical Specifications:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Method Name** | **Input** | **Output** | **Exception** |
| IDCardManager | **registerForId()** | **RegistrationVO registrationVO** | Boolean | InvalidEmpNoException InvalidNameException,  InvalidLocationIdException  InvalidEmergencyNoException,  InvalidEmployeeException  These Exceptions to be caught and thrown back to Tester class |
| IDCardFacade | **registerForId()** | **RegistrationVO registrationVO** | Boolean | InvalidEmpNoException InvalidNameException,  InvalidLocationIdException  InvalidEmergencyNoException,  InvalidEmployeeException These Exceptions to be caught and thrown back to IDCardManager class |
| IDCardBO | **registerForId()**  \*\* check for the following business rules given below | **RegistrationVO registrationVO** | Boolean | InvalidEmpNoException InvalidNameException,  InvalidLocationIdException  InvalidEmergencyNoException,  InvalidEmployeeException These Exceptions to be caught and thrown back to IDCardFacade class |
| IDCardDao interface | **getLocationDetails()** | Integer locationId |  |  |
| **getEmployeeDetails()** | Integer empNo |  |  |
| **registerForId()** | **RegistrationVO registrationVO** |  |  |
| IDCardDaoImpl  Implements IDCardDao | **getLocationDetails()** | Integer locationId | return the corresponding Location object for the locationId if valid, null if invalid |  |
| **getEmployeeDetails()** | Integer empNo | return the corresponding Employee object for the empNo if valid, null if invalid |  |
| **registerForId()** | **RegistrationVO registrationVO** | Return Boolean value in case of successful execution |  |
| IdCard\_config.xml | Contains all the xml configurations related to Service1(**Register for ID Card)** |  |  |  |

1. Business Rules& Validations:

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

|  |  |  |
| --- | --- | --- |
| **Rule**  **No.** | **Rule Description** | **User Define Exception to be thrown** |
| 1. | **empNo** entered should not be negative .it can be minimum value of 3 and maximum value of 7 | If empNo is invalid  **InvalidEmpNoException** |
| 2. | **empName** entered should contain only alphabets and spaces and should not contain any digits and special characters | If empName is invalid  **InvalidNameException** |
| 3. | **locationId** entered should be valid [ getLocationDetails(Integer locationId) method of IDCardDAO. Method returns Location object if locationId is valid , null if valid] | If **locationId** is invalid  **InvalidLocationIdException** |
| 4 | **empNo** entered should be valid [ getRegistrationDetails(Integer empNo) method of IDCardDAO. Method returns Employee object if empNo is valid , null if valid] | If **empNo** is invalid  **InvalidEmployeeException** |
| 5 | **emergencyNo** entered should contain only 10-digit numbers | If **emergencyNo** is invalid  **InvalidEmergencyNoException** |

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

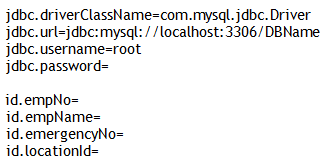
**Note: The RegistrationVO, Location and Employee class will be provided with the required getter and setter methods which should not be modified. No changes should be done to the skeletons provided (Especially the name and method signature)**

**Limitations and Constraints**

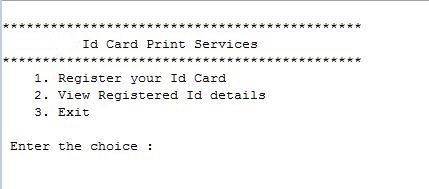
1. All the above mentioned java classes to be declared as beans in spring configuration file.
2. IDCardFacade should be integrated IDCardManager using Setter Based Dependency Injection.
3. IDCardBO should be integrated with IDCardFacade using Constructor based dependency injection
4. IDCardDao should be integrated with IDCardBO using Setter Based Dependency
5. Use ONLY IdCard\_config.xml for all configurations related to Service1 (**Register for ID card).**
6. Read the **input data** and **Data Source** details from the **constants.properties**
7. 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 register IDCard**

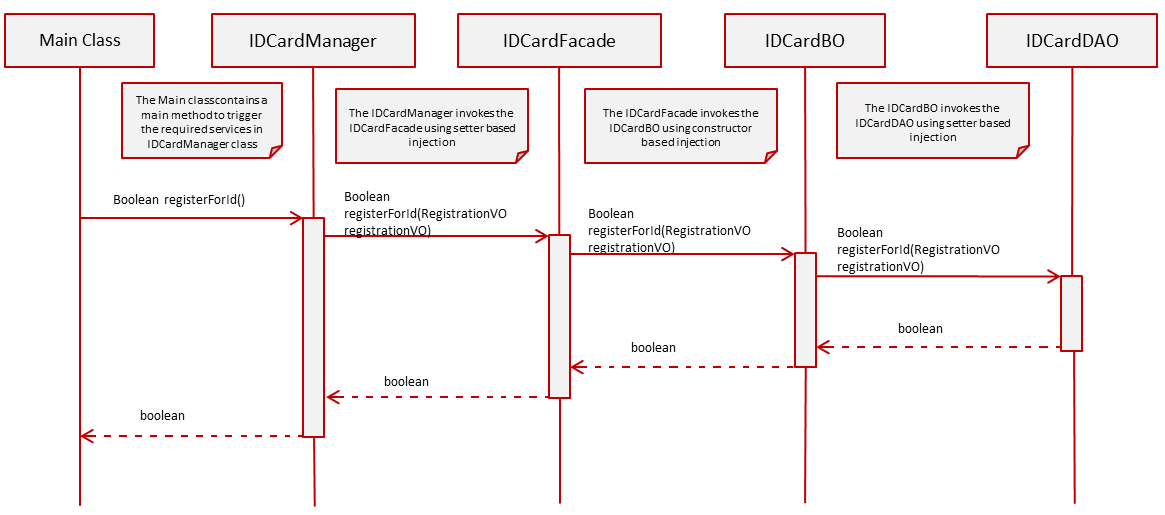


Output



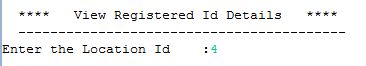


1. Sequence Diagram

****

1. View Registered Id Details:

* When the user selects Option 2**, viewRegisteredDetails () method of Tester** is invoked.
* The ID card admin will input the **locationId.**
* The **viewRegisteredDetails** menu in the **viewRegisteredDetails()** method of Tester class looks like below

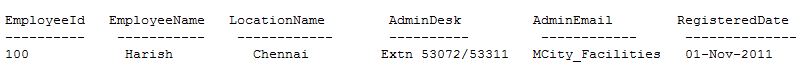


Control flow is **Tester -> Manager -> Façade->BO->Dao.**

* In the **Tester** Class,
* Invoke **viewRegisteredDetails (Integer locationId)** from Tester to **IDCardManager** 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 registeredDate in proper Date Format (dd-MMM-yyyy),

Use **DateUtility.getStringFromDate (Date date)** to convert registeredDate from Date to String.**]**

**OUTPUT**

1. Technical Specifications:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Method Name** | **Input** | **Output** | **Exception** |
| IDCardManager | viewRegisteredDetails () | Integer locationId | List<DetailsVO> detailVoList | NoRecordException,  This Exception to be caught and thrown back to Tester class |
| IDCardFacade | viewRegisteredDetails () | Integer locationId | List<DetailsVO> detailVoList | NoRecordException,  This Exception to be caught and thrown back to IDCardManager class |
| IDCardBO | viewRegisteredDetails () | Integer locationId | List<DetailsVO> detailVoList | NoRecordException,  This exception to be caught and thrown back to IDCardFacade class |
| IDCardDao interface | viewRegisteredDetails () | Integer locationId | List<DetailsVO> detailVoList |  |
| IDCardDaoImpl  Implements IDCardDao | viewRegisteredDetails () | Integer locationId | List<DetailsVO> detailVoList |  |
| IdCard\_config.xml | Contains all the xml configurations related to Service2(**View Registered Id Details)** |  |  |  |

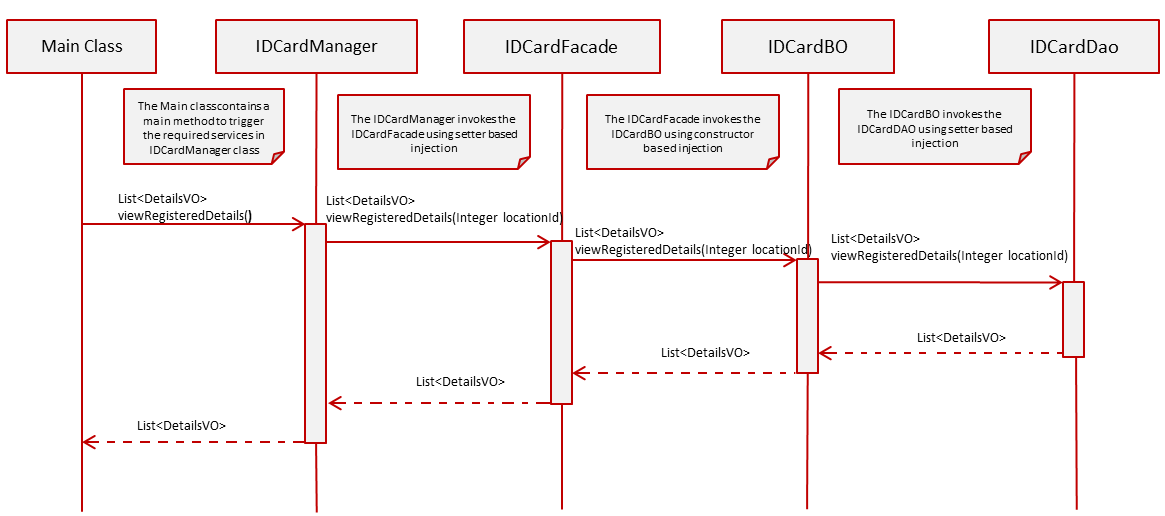
1. Business Rules & Validations:

In the **IDCardBO 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 **viewRegisteredDetails** () of IDCardDao 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