­



Assignment Document: Core Spring

Version: Core Spring Practice Case study 3/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

[Cognizant Network and System (NSS) 2](#_Toc452803001)

[1. Infrastructure Section 2](#_Toc452803002)

[i. Hardware, Software Specification 2](#_Toc452803003)

[2. Case Study Assignments 4](#_Toc452803004)

[3. Instructions 4](#_Toc452803005)

[4. DDL/DML for creating tables and inserting data 5](#_Toc452803006)

[5. Register For Port Number 5](#_Toc452803007)

[6. Technical Specifications 5](#_Toc452803008)

[7. Business Rules& Validations 7](#_Toc452803009)

[8. Sequence Diagram 9](#_Toc452803010)

[9. View Port Details 9](#_Toc452803011)

[10. Technical Specifications 10](#_Toc452803012)

[11. Business Rules & Validations 11](#_Toc452803013)

[12. Sequence Diagram 11](#_Toc452803014)

[13. Solution 11](#_Toc452803015)

[14. Evaluation Rubrics 12](#_Toc452803016)

[15. Summary of this Case Study 12](#_Toc452803017)

# Cognizant Network and System (NSS)

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 |

1. **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:**



1. Case Study Assignments

Estimated Completion Time: xx Minutes

Objective:

Cognizant Network and System Service(NSS) team allocate a system whenever a new employee join the organization, or an employee get transferred from one location to another, or shifting from one building to another building within the same location. The NSS team needs to allocate a system and register a port number for the employee. It also needs to view how many port numbers are available for a particular building. It wants to automate the process of registering ports and viewing the available ports. It would like to develop the below components:

**Service1: Register for Port Number**

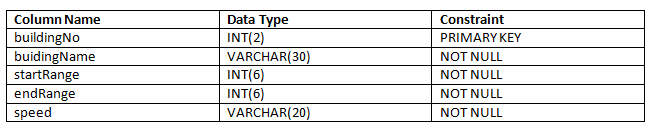
**Service2: View Port Details**

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

1. Instructions

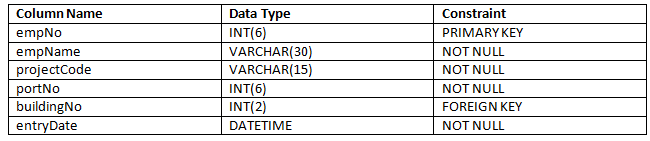
* Please make sure that JAVA\_HOME is set as your JDK installation folder, if not, please set as the following: ***C:\Program Files\Java\jdk1.6.0\_28***
* Download the artifacts (Skeleton) from the tool. Unzip the zip files downloaded on your desktop.
* Open Eclipse IDE and import the project using “**Import -> 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: **Building**

****

**P.S: The CHAR column courseOffered denotes M.E/M.TECH(M) or PhD(P).**

* 1. Table Name: **Employee**



Please run the below .sql file before you start the case study. Refer the “User Manual for Mysql” in the URL provided in the Instructions section.

<Refer to NSSPort.sql file that is present in same folder as your case study document is>

The service is used by the system to register employee for PortNo. 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. Register For Port Number:

This service accepts the employee details and registers them using the Port Number. If the above business rules are satisfied, then persist the data and will return a Boolean value. 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** |
| NSSPortManager | registerPort**()** | **EmployeeVO employeeVO** | Boolean value(True or False) | InvalidEmpNoException InvalidNameException  InvalidBuildingNoException  InvalidPortNoException  InvalidRegistrationException  These exceptions have to be caught and thrown back to Tester class. |
| NSSPortFacade | registerPort**()** | **EmployeeVO employeeVO** | Boolean value(True or False) | InvalidEmpNoException InvalidNameException  InvalidBuildingNoException  InvalidPortNoException  InvalidRegistrationException  These exceptions have to be caught and thrown back to NSSManager class. |
| NSSPortBO | registerPort**()**  \*\* check for the following business rules given below | **EmployeeVO employeeVO** | Boolean value(True or False) | InvalidEmpNoException InvalidNameException  InvalidBuildingNoException  InvalidPortNoException  InvalidRegistrationException  These exceptions have to be caught and thrown back to NSSFacade class. |
| NSSPortDao interface | **getBuildingDetails ()** | **Integer buildingNo** |  |  |
| **getEmployeeDetails ()**: | **Integer empNo** |  |  |
| registerPort**()** | **EmployeeVO employeeVO** |  |  |
| NSSPortDAOImpl  Implements NSSPortDao | **getBuildingDetails ()** | **Integer buildingNo** | return the corresponding **Building** object for the **buildingNo** if valid**,** null if invalid |  |
| **getEmployeeDetails ()**: | **Integer empNo** | return the corresponding **Employee** object for the **empNo** if valid,null if invalid |  |
| registerPort**()** | **EmployeeVO employeeVO** | Boolean value(True or False) |  |
| nss\_config.xml | Contains all the xml configurations related to Service1(**Register For Port Number)** |  |  |  |

1. Business Rules& Validations:

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

|  |  |  |
| --- | --- | --- |
| **Rule**  **No.** | **Rule Description** | **User Define Exception to be thrown** |
| 1. | **EmpNo** entered should not be negative. It can be a minimum of 1 digit and maximum of 6 digits. | If empNo is invalid, throws  **InvalidEmpNoException.** |
| 2. | **empName** entered should contain only alphabets and spaces and should not contain any digits and special characters. | If empName is invalid, throws  **InvalidNameException.** |
| 3. | **BuildingNo** entered should be valid [use **getBuildingDetails (Integer buildingNo)** method of **NSSPortDao**. Methods return **Building object** if **buildingNo** is valid, null if invalid]. | If buildingNo is invalid, throws  **InvalidBuildingNoException.** |
| 4 | **portNo** entered should be within the **startRange** and the **endRange** of the **Building table.** | If portNo is invalid, throws **InvalidPortNoException.** |
| 5 | **EmpNo** entered should be valid [use **getEmployeeDetails (Integer empNo)** method of **NSSPortDao**. Methods return not null if empNo already exists, null if not exists]. | If empNo is invalid, throws  **InvalidRegistrationException.** |

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

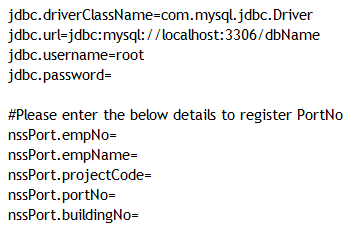
**Note:** The EmployeeVO, Building, Employee class will be provided with the required getter and setter methods which should not be modified. No changes should be done to the skeletons provided (Especially the name and method signature).

**Limitations and Constraints**

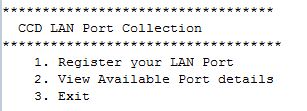
1. All the java classes to be declared as beans in spring configuration file nss\_config.xml.
2. NSSPortFacade should be integrated **NSSPortManager** using setter-based dependency injection.
3. NSSPortBO should be integrated with **NSSPortFacade** using constructor-based dependency injection.
4. **NSSPortDao** should be integrated with **NSSPortBO** using setter-based dependency injection.
5. Use ONLY nss\_config.xml for all configurations related to Service1(**Register For Port Number)**
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 portNo.

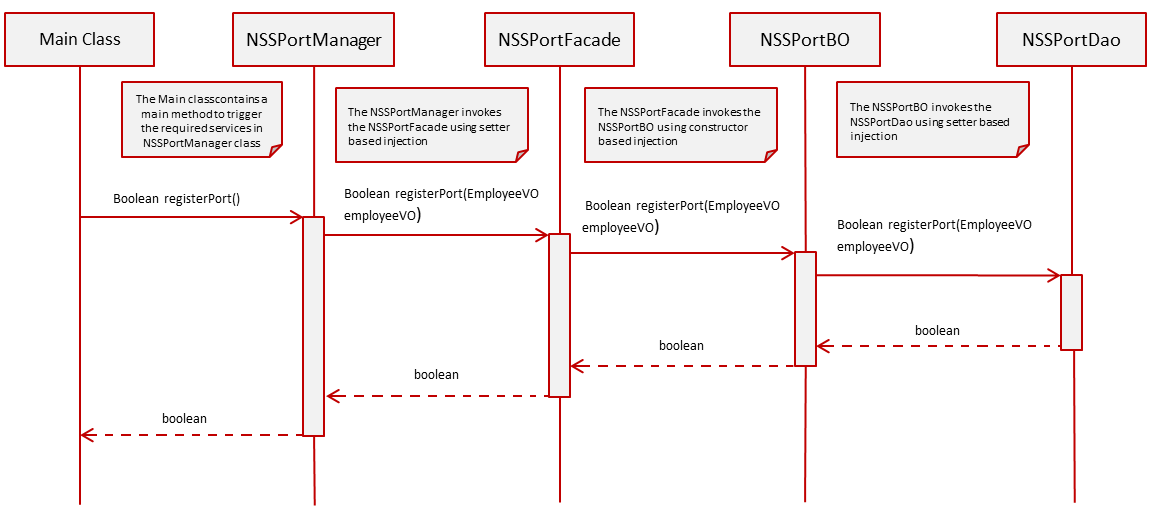


Output





1. Sequence Diagram

****

1. View Port Details:

* When the user selects Option 2**, viewPortDetails () method of Tester** is invoked.
* The NSS admin will input the **buildingNo.**
* The **ViewPortDetails** menu in the **viewPortDetails()** method of Tester class looks like below:

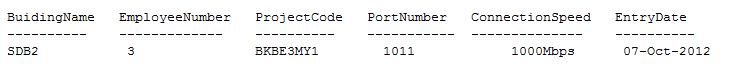


Control flow is **Tester --> Manager --> Façade --> BO --> Dao**

* In the **Tester** Class,
* Invoke **viewPortDetails (Integer buildingNo)** from Tester toManager 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 entryDate in proper Date Format (dd-MMM-yyyy), use **DateUtility.getStringFromDate (Date date)** to convert entryDate from Date to String**]**

**OUTPUT**



1. Technical Specifications:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Method Name** | **Input** | **Output** | **Exception** |
| NSSPortManager | **viewPortDetails()** | Integer buildingNo | List<DetailsVO> detailVoList | NoRecordException  This exception has to be caught and thrown back to Tester class. |
| NSSPortFacade | **viewPortDetails()** | Integer buildingNo | List<DetailsVO> detailVoList | NoRecordException  This exception has to be caught and thrown back to NSSPortManager class |
| NSSPortBO | **viewPortDetails()** | Integer buildingNo | List<DetailsVO> detailVoList | NoRecordException  This exception has to be caught and thrown back to NSSPortFacade class |
| NSSPortDao  interface | **viewPortDetails()** | Integer buildingNo | List<DetailsVO> detailVoList |  |
| NSSPortDaoImpl  Implements NSSPortDao | **viewPortDetails()** | Integer buildingNo | List<DetailsVO> detailVoList |  |
| nss\_config.xml | 1. Contains all the xml configurations related to Service2 (**ViewPort Details)** |  |  |  |

1. Business Rules & Validations:

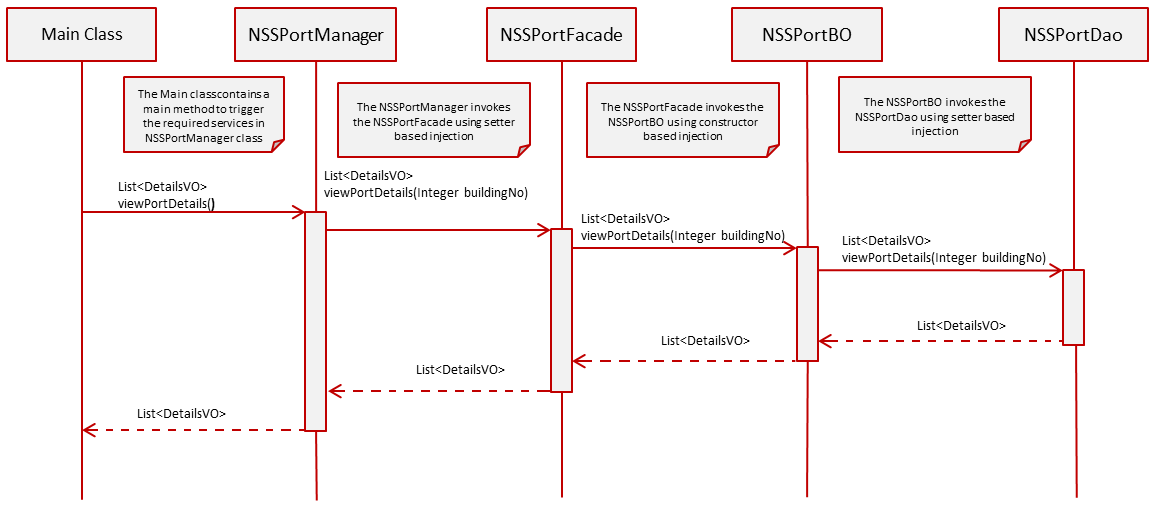
In the **NSSPortBO class** check for the following business rules given below

|  |  |  |
| --- | --- | --- |
| **Rule**  **No.** | **Rule Description** | **User Define Exception to be thrown** |
| 1. | List<DetailsVO> returned by the method viewPortDetails (**Integer buildingNo**) of NSSPortDao 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
* PropertyPlaceholderConfigurer