Core JAVA Mini Project Application

Description: The Objective of this assignment is to create a small working application by using various possible OOPs concepts and common JAVA API. This will also help to understand how various concepts can be glued together to build an Application.

Document Control

Version	1.0
Created by	
Updated by	
Reviewed by	
Date	June 2016

Table of Contents

1. Pre-Requisites

- Knowledge of Core JAVA Programming.

2. Hardware-Software Requirements

The learner will require a computer (PC / Laptop) running Windows 7 and above with at least 4 GB of RAM. Additionally, the following software (free for training and learning purpose) is also required:-

- JDK 1.7 or above
- Eclipse Mars or Spring tool suits(STS)

In order to view videos mentioned in this GLP the learner needs to submit a request at this link to get credentials for Pluralsight.com

3. Mini Project Specification

This Project is about creating a Console Application which tracks the details of the trading partners. The Trading Partners are Customers and Suppliers that the current business organization transacts with. The Application allows to Add and View the details of the existing Customers as well as the Suppliers.

The Application is created with 3 Layers:

1st Layer as ApplicationEntities which has the entity classes compiled along with trivial validation logic.

2nd Layer ApplicationDAL which encapsulated Data Access Logic

3rd Layer MyConsoleApplication Project which is an executable which references the above 2 Layers and provides console UI to the End User to perform tasks.

I. Create a package as ApplicationEntities
   a. Add a class "TradingPartner" with TradingPartnerId, TradingPartnerName, City properties.
   b. Add 2 derived classes "Customer" and "Supplier".
   c. Additional Properties: For Customer class: CreditLimit, emailId and for Supplier: CreditBalance, PanNo
   d. Property Validation Requirements:
      i. TradingPartnerId: Mandatory and Non Negative
      ii. TradingPartnerName: Mandatory and Minimum 5 characters in length
      iii. City: Mandatory and Minimum 3 characters in length
      iv. CreditLimit: Not Beyond 50,000
      v. emailId: proper format
      vi. CreditBalance: Not Beyond 1,50,000
      vii. PanNo: Proper Format
   e. Declare function in TradingPartner class:
      i. public String[] Validate(): Do first 3 property validations. It will return error messages as per validation failures in form of string array.
   f. Declare abstract function in TradingPartner class:
      i. public abstract void SaveToFile(string filepath):
   g. Override the virtual Validate function in derived concrete classes to add validation logic.
   h. Override the abstract SaveToFile function in derived concrete classes to serialize the data:
      i. Customer: Binary Serialize

ii. Supplier: XML Serialize

II. Create 2 tables in the SQL database with columns matching with the properties available in the Customer and Supplier Tables.

III. Create a ApplicationDAL Project

a. Add a class DALService.

i. Add static functions as below:

1. Public void SaveCustomerDetails(Customer cust): which saves the customer details to the database table.
2. Public void SaveSupplierDetails(Supplier supp): which saves the Supplier details to the database table.
3. Public List<Customer> GetAllCustomers(): Gets and returns all the Customer details from the database as a generic collection instance.
4. Public List<Supplier> GetAllSuppliers(): Gets and returns all the Supplier details from the database as a generic collection instance.
5. Public Customer GetCustomerById(int customerid): Gets and returns the Customer details from the database based on id
6. Public Supplier GetSupplierById(int supplierid): Gets and returns the Supplier details from the database based on id

IV. Create Console Project MyConsoleApplication

a. Show a console menu to user using loop as per options below and perform as per the user selection of menu until user selects exit menu option. Them Main() function will print the below console menu and accept the option input from end user. Accordngly create separate static functions which Main() will call depending on user selection. These static functions will interact with user validate entered details and communicate with DAL layer as per need.

Add Customer : If this menu is selected App should accept all details pertaining to customer validate it and then add the details to database

Add Supplier: If this menu is selected App should accept all details pertaining to supplier validate it and then add the details to database

Show All Customers : If this menu is selected App should Retrieve and display all Customer Details

Show All Suppliers: : If this menu is selected App should Retrieve and display all Supplier Details

Export a Customer: If this menu is selected App should ask for customer id and export it to the file.

Export a Supplier: If this menu is selected App should ask for supplier id and export it to the file

Exit: If this menu is selected then exit the Application

V. Execute and Test the Application for the functionality

<u>Possible flow of Logic for **Add Customer** Menu option</u>

I. Once user selects “Add Customer” Call a static function “AddCustomerConsoleScreen()”

II. This function will ask End User about new customer details as per properties in the Customer Class.

III. Encapsulate all the collected information in Customer class instance.

IV. Call Validate function on the Customer instance
   a. If validation fails flash message and re accept data else call the SaveCustomerDetails() function on the DAL Layer and based on success/failure flash message to the end user.

Similarly implement other functionality

Handle the following Change Request:

The Application needs to no have additional following features:

A. Update the Existing Customer Details
B. Update the Existing Supplier Details
C. Export All Customer Details to an XML File.

Accordingly Make changes to the Application Layers/Database.