Due Date: 29th October @ 23:59

Lab 4 will help you with this project and you might find Using the H2 Console useful.

Overview

This assignment assesses the following learning outcomes:

LO1 Configure a software project to use framework libraries.

LO2 Design scalable enterprise applications with reusable components.

LO3 Implement advanced data access technologies for a given requirement.

This project is designed for you to practise the Spring framework ***without*** Spring Boot/Web.

You must create a scalable application, designed following good design practice to ensure re-usability of components. It will access a relational database.

Develop a **text/console-based** application using **Spring** and **JdbcTemplate**. This is NOT a web-based project.

Use a **Legacy Project** with **Maven** and **XML** configuration.

Use **H2 embedded database** with **SQL** scripts for the schema and initial data. It must be possible to run this “out of the box” with no configuration.

You must create **four unit tests,** at least one of which should use **Mockito** and one of which should conduct a database-query test. These unit tests must be meaningful and commented to describe what they are testing.

You must use Maven (not Gradle).

Provide a **brief document** outlining the high-level design of your system (1 or 2 A4 pages) including but not limited to your database design and class diagrams and the beans which you used.

You must employ a sensible and effective architecture i.e. use of packages, interfaces and service layers.

You may use Lombok to reduce boilerplate code e.g. it will generate your getters/setters/constructors for you, but this is not a requirement.

The Application

You will write an application for a bank.

The bank stores account data i.e. account holder's details and the account's balance and overdraft limit.

An account can have more than one owner i.e. **joint accounts** are allowed.

A customer can have more than one account.

The application must allow an employee to facilitate customers to

- become a bank customer i.e. register with the bank
- create a new account
- add a person to an account (for joint accounts)
- view his/her own accounts
- withdraw money from his/her account, subject to an overdraft limit
- deposit money into his/her account
- transfer money from one account to another
- close an account

Furthermore, the application must allow the bank employee to display the

- total amount of money deposited in the bank
- the number of accounts with deposits over €10,000

Authentication is not required.

Marking

Consult the rubric for a breakdown of the marks.

Submission

Submit via the e-assignment in Canvas.

You will need the following in your **zip archive**:

- A **brief document** outlining the high-level design of your system (2/3 A4 pages) including but not limited to your database design and the beans which you used and a list of issues e.g. functionality not implemented, limitations in code etc.
- Your **Spring project**, including SQL scripts, in .zip or .7z but NOT .rar - just zip up your Spring project folder.

Penalties

Penalties for late submission are applied as per section 4.4.2 of the Exam Regulations (see http://www.mycit.ie/examregulations (Links to an external site.)).

This is an independent project. Do not plagiarize: don't copy code, don't provide code to anyone else, don't work with someone else on the project. Taking someone else's code, renaming classes and variables is not a good strategy and will likely result in a zero grade or worse.

Rubric

Project #1 - Console Spring Project

Project #1 - Console Spring Project

Criteria					Ratings	Pts
This criterion is linked to a learning outcome Project Configuration Have you used the correct architecture including service / business layer and interfaces where appropriate? Have you created packages? Have you used appropriate names for packages?						
10.0 to >7.0 Pts Excellent Proper use of Maven, package structure is consistent and adheres to best practise, files are where they should be.	7.0 to >5.0 Pts Good Configuration is mostly correct - there may be some issues around package structure, files not being where they should.	5.0 to >3.0 Pts Promising Shows promise but far from perfect.	3.0 to >0.0 Pts Weak Some configuration is evident but it may not be poorly constructed e.g. no consistency in naming packages.	0.0 Pts Poor Configuration is not evident.		10.0 pts
This criterion is linked to a learning outcome Spring Bean Configuration Have you created beans in XML or Java? Use of Spring Boot will result in 0 for this section.						
15.0 to >10.5 Pts Excellent All beans properly configured, dependency injection operating as it should using XML or Java configuration.	10.5 to >7.5 Pts Good Beans are mostly correctly configured, most DI is working as expected, though it may not be perfect	7.5 to >4.5 Pts Promising Shows promise but far from perfect.	4.5 to >0.0 Pts Weak Some configuration is evident but it may not be working properly e.g. autowiring may not be working.	0.0 Pts Poor Spring beans not properly configured, not injecting correctly or uses Spring Boot, thus avoiding all bean configuration.		15.0 pts

Criteria					Ratings	Pts
This criterion is linked to a learning outcome Functionality						
30.0 to >21.0 Pts Excellent All, or close to all, functionality implemented correctly.	21.0 to >15.0 Pts Good Some or most of the functionality has been implemented correctly.	15.0 to >10.0 Pts Promising Shows promise but far from perfect. Basic functionality is evident but missing lots.	10.0 to >0.0 Pts Poor A significant amount of functionality not implemented correctly although there is evidence of some functionality.	0.0 Pts Nothing Spring beans not properly configured, not injecting correctly or uses Spring Boot, thus avoiding all bean configuration.		30.0 pts
This criterion is linked to a learning outcome DAO / Repositories						
30.0 to >21.0 Pts Excellent All, or close to all, DAO functionality implemented correctly.	21.0 to >15.0 Pts Good DAO / Repository classes are close to complete and correctly access the database, demonstrating most skills in SQL.	15.0 to >10.0 Pts Promising Shows promise but far from perfect.	10.0 to >0.0 Pts Poor DAO / Repository classes are started but incomplete. Some database access is evident but is limited.	0.0 Pts Nothing No DAO created.		30.0 pts
This criterion is linked to a learning outcome Design Document Should provide a description of the architecture used and of the database design as well as details of what you did not have time to include e.g. functionality/service layer. Approximately pages should be sufficient to describe both but more might be needed.						
10.0 to >7.0 Pts Excellent The design / architecture for the system has been well outlined in a document.	7.0 to >5.0 Pts Good The design / architecture for the system has been outlined in a document, though may not be very thorough.	5.0 to >3.0 Pts Promising Shows promise but far from perfect.	3.0 to >0.0 Pts Poor The design / architecture for the system has been very briefly outlined in a document, though may not be very thorough.	0.0 Pts Nothing No document provided.		10.0 pts
This criterion is linked to a learning outcome Unit Tests						
5.0 to >4.0 Pts Excellent	4.0 to >1.0 Pts OK More tests needed or not very useful tests.		1.0 to >0 Pts No marks			5.0 pts

Total points: 100.0