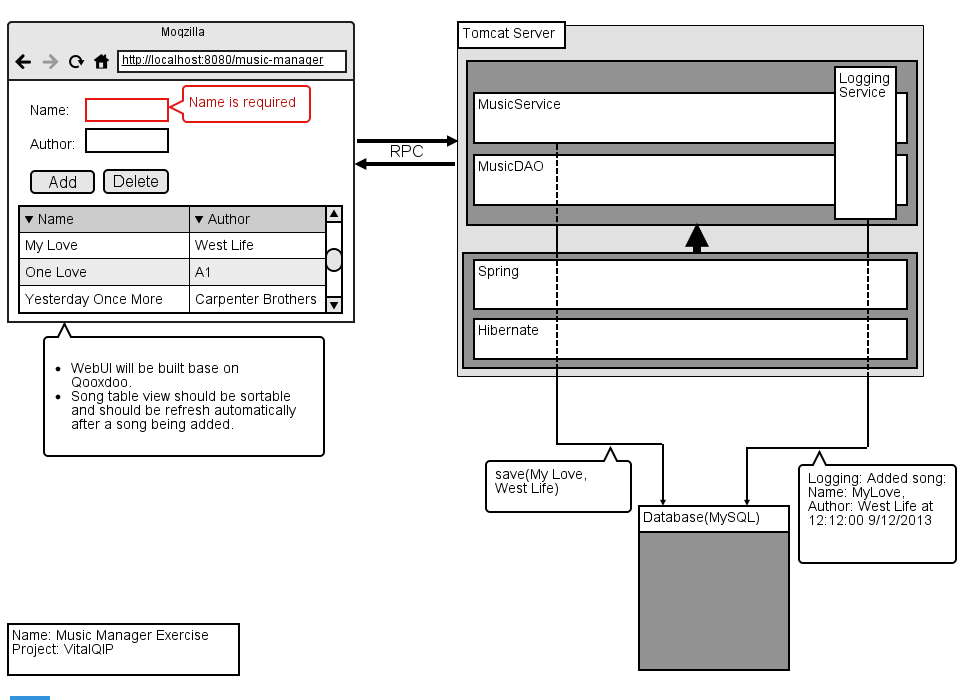
**Music Manager Exercise for New Members**

Hello, welcome to QIP ☺

If you are reading this document, you probably a new member ☺ so, we are here to help you to getting started with QIP. What we are doing here is to provide you a simple exercise that simulates a simple version of QIP. The real QIP is more complicated than this, but it just works as same as this one.

After complete this exercise, you will have knowledge about Qooxdoo, Spring, Hibernate, Java Servlet, and the way to combine them all to have a final web application.   
The exercise is described as below:



The goal of the exercise is to make a music manager web application.  A music manager web application allows you to browse through a music library, add new entries and delete them (so the usual Create, Update and Delete stuffs (CRUD)).

All music data should be stored server side in a relational database.  The client should be a Rich Internet application (RIA). Communication between clients and server should go over RESTful.

On the server side we expect the application to be deployed in a Tomcat server.  The client should always communicate to one dedicated service interface (*MusicService*).  The *MusicService* implementation is a POJO which uses Hibernate to interact with the database (*MusicDAO*). All interactions of user should be logged into relational database via *LoggingService*.

Technology requirements

**Java basic:**

Version 1.8.0\_112

Overview of Java architecture and basic concepts of OOD

Be able to implement Object Oriented based applications

Be able to design and implement effective Java programs

**RESTful:**

Understand REST concept

Be able to use RESTful for Web UI & Server communication.

**Hibernate:**

Version 4.3.10

Overview of Hibernate Architecture

Be able to define configuration data (hibernate.properties, XML mapping, Annotation mapping)

Understand POJO programming model and common Hibernate APIs for database operations

Understand HQL (Hibernate Query Language) and native SQL

**Spring:**

Version 4.1.7

Understand and be able to apply 2 new concepts: IOC and AOP.

**JUnit:**

Version 4.12

Overview of JUnit

Be able to implement a JUnit class for verification the implementation class.

**Qooxdoo:**

Version 5.0.2

Overview of Qooxdoo.

Understand Qooxdoo’s architecture and be able to implement some common controls.

**Tomcat Server:**

Version 7.0

Be able to configure (stop, start, …etc) the server.

Deploy/Undeploy a web application on the server.

**Ant:**

Version 1.6.5

Be able to build web project with Ant.

Be able to define a simple target.

**MySQL DBMS**

Where should I start?  
*Below is just a suggestion for easier for you to investigate and finish the exercise. But you are free to have your own way ☺*

*The idea here is to separate the server side and the Web UI side, so you should implement them separately. Once you finished implementation for both server side and Web UI side, you will use RESTful to tight them together.*

1. Implement Web UI

We will use Qooxdoo to build Web UI, hence I suggest you to read “*Bachu R.K., Raffi M. - qooxdoo. Beginner’s Guide – 2011*”. This book describes very details on how to build Web UI with Qooxdoo, it’s very easy to understand and follow. I think just need 2 or 3 days to complete Web UI part.

After finish this part, please make that you understand and finish follow things:

1. Build/Run a simple web application with Qooxdoo.
2. Create music manager Web UI (see the mockup in the first page.)
3. Implement server side:

For the server side, you will need to build *MusicService*, *LoggingService* and *MusicDAO*, *MusicService* and *LoggingService* will use *MusicDAO* which uses Hibernate to interact with relational database (MySQL).

To make the components in server side more loosely couple, we will apply Spring IOC and AOP concepts.

To implement the server side, you need to know Hibernate, Spring, Java Servlet. So, for more easier for you to finish the server side, I suggest you to follow below steps:

* 1. Learn Hibernate

We have slides, books and a hello hibernate example for you. Please check them out.

My suggestion here is that you should take an over view of Hibernate through slides, run the given example, then come back to read more details about Hibernate. Below are some questions that you should answer within this part:

* + 1. What is Hibernate?
    2. What its benefit?
    3. How to do a One-to-One, One-to-Many mapping? When do you need them?
    4. How to do an inheritance mapping?
    5. How to use HQL? (Let’s do some examples)

After finish this part, you should achieve below items:

* + 1. Created model/DAO for the exercise (*Song/MusicDAO*)
    2. Having a test class to prove that your DAO works
  1. Learn Spring

Same as Hibernate, we have slides, examples and books that packaged along with this document. Let’s check them out.

Below are some questions that you should answer within this part:

* + 1. What is Spring?
    2. Why do you need Spring?
    3. What is IOC?
    4. What is AOP? What can you do with it?
    5. What does Spring support?

After finish this part, you should achieve below items:

* + 1. Finish implementation for *MusicService* (apply IOC)
    2. Finish implementation for *LoggingService* (apply AOP)
    3. Finish integrating Hibernate with Spring.
  1. Test your server
     1. Now you already had both Services and DAO, it’s time to create a class to test if the service is OK. The server side should save/update/delete song successful at this step.
  2. Java Servlet.
     1. Create a simple Java Servlet application
     2. Apply it into the exercise

After finish investigating Java Servlet, please ensure you understand below items:

* + - 1. servlet-mapping
      2. servlet
      3. listener
      4. context-param
      5. Why do you need web.xml? When is it deployed?
  1. Deploy the server
     1. How to load Spring applicationContext.xml on Tomcat automatically?

1. Tight the Web UI and server side together:
   1. Now everything is OK. You only need to find a way to call the services in your servlet. Let’s use RESTful to address this. Read <http://www.qooxdoo.org/5.0.2/pages/communication/rest.html>

Reference resources:

1. Google ☺
2. All documents that need for your references are packaged with this document. Please make sure that you read them carefully.

---------------------------------------------------- GOOD LUCK--------------------------------------------------