Shantanu KAMBLE

EPITA | Msc spring 2018

Email: kambleshan04@gmail.com

Quiz Management

Technical specification Documentation

Table of Contents

[**1.** **Subject Description** 2](#_Toc530775886)

[**2.** **Subject Analysis** 3](#_Toc530775887)

[**2.1** **Major Features** 3](#_Toc530775888)

[**2.2** **Application Feasibility** 3](#_Toc530775889)

[**2.3** **Data Description** 3](#_Toc530775890)

[**2.4** **Expected Results** 4](#_Toc530775891)

[**2.5** **Scope of Application** 4](#_Toc530775892)

[**3.** **Conception** 5](#_Toc530775893)

[**3.1** **Chosen Algorithm** 5](#_Toc530775894)

[**3.2** **Data Structures** 5](#_Toc530775895)

[**3.3** **Global Application Flow** 5](#_Toc530775896)

[**3.4** **Global Schema and Major Features Schema** 6](#_Toc530775897)

[**4.** **Web Operations Description** 7](#_Toc530775898)

[**4.1** **Creating an MCQ** 7](#_Toc530775899)

[**4.2** **Updating an MCQ** 7](#_Toc530775900)

[**4.3** **Searching an MCQ** 7](#_Toc530775901)

[**4.4** **Deleting an MCQ** 7](#_Toc530775902)

[**5.** **Configuration Instructions** 8](#_Toc530775903)

[**5.1** **Development Tools** 8](#_Toc530775904)

[**5.2** **Schema and Database Configuration** 8](#_Toc530775905)

[**6.** **Commented Screenshots** 9](#_Toc530775906)

[**7.** **Bibliography** 9](#_Toc530775907)

## **Subject Description**

The quiz management project implements basic features of CRUD system. It deals with managing the multiple-choice questions. The application exposes the RESTful APIs for creating, searching, updating, deleting and displaying of MCQs.

The project is implemented using the following technologies: Java, Maven, JAX-RS and Hibernate.

The application consists of total three projects.

* quiz-core-sk (project containing core functionality)
* quiz-restful (project containing the REST services)
* quiz-web (project to display the User Interface)

## **Subject Analysis**

### **Major Features**

The major advantage of this project is that it is reusable and configurable. This project can be used to cater the needs of other projects that involve database interaction. Some of the features of this project can be listed below:

* RESTful APIs
* Creating an MCQ
* Updating an MCQ
* Searching an MCQ
* Deleting an MCQ
* View all MCQs

### **Application Feasibility**

* The REST APIs offer following advantages:
  + Separation between the client and the server
  + Visibility, reliability and scalability
  + It is independent of the type of platform
* The application is developed using Java, Derby database, Maven, JAX-RS, Hibernate which are easily available to all.
* This application follows a strategic approach, easy configuration and self-explanatory error messages.

### **Data Description**

The data stored by the application is related to multiple choice questions i.e. questions and choices. The structure of data stored in the database is as below.

**QUESTION table**

|  |  |  |  |
| --- | --- | --- | --- |
| Column name | Data type | Length | Description |
| ID | Integer |  | This Id is a unique identifier for a question |
| QUESTIONLABEL | Varchar | 255 | This is the label name for a question |

**MCQCHOICE table**

|  |  |  |  |
| --- | --- | --- | --- |
| Column name | Data type | Length | Description |
| ID | Integer |  | This Id is a unique identifier for MCQ choice |
| QUESTION\_ID | Integer |  | This is the id of the question to which the choice is linked. |
| CHOICELABEL | Varchar | 255 | This is the label name for a choice |
| VALID | Boolean |  | This specifies whether the choice is valid or not. |

### **Expected Results**

Quiz management:

* The application presents the user with options for View-All questions and choices, Create an MCQ, Update an MCQ, Search for an MCQ and delete an MCQ.
* The application displays specific error messages if the user does any mistake while performing the operations.

### **Scope of Application**

The primary focus of this application is on MCQ management. So, it does not consider the user management functionality. As the identities are stored in a database, the user needs to install a database to use this application.

Limitation:

* Lack of organizing sample evaluation of the quiz. Only MCQ’s are composed not any other like open questions
* Lack of identity management of the users like students, teachers, etc.

Evolution:

* Storing the user identities and managing them.
* Encryption of the passwords stored in database.
* Concept of users and admins can be integrated by adding functionality to assign roles to the users based on the operation.

## **Conception**

### **Chosen Algorithm**

Here we can discuss about the design of the application.

* The application exposes REST services/operations for consumption.
* The application follows a factory method design pattern. The advantage of using this design is that we can have multiple implementations.

For example: It consists of a base class named “GenericDAO” (inside “quiz-core-sk” project).

* The child classes (for example: class QuestionDAO) implement the search method which is implemented using a LIKE clause. This feature helps to compare a part of search string entered by the user, to search for a particular thing. This makes the search easier and flexible.
* The application is made configurable by keeping the database properties in applicationContext.xml file.
* The application uses log4j library for logging purpose. The logging configuration is stored in the file named “log4j2.properties”.

### **Data Structures**

Data structures used in this application are,

* LIST: which is a collection of elements. List is used in the project to store the collection of questions and choices.
* MAP: which uses the question as the key and the related choices as a List
* String is also a widely used data structure in this application.

### **Global Application Flow**

It is a three-tier application in which the user requests are captured by the launcher with help of the Console operations. The Configuration service provides the required properties to the factory. The DAO is responsible for data retrieval and its manipulation. The DAO interacts with the database and sends back the required information. The retrieved information is then displayed to the user.

**USER**

User Interface

RESTful resource

Service

DAO

Database

### **Global Schema and Major Features Schema**

The application has two tables, QUESTION and MCQCHOICE. The QUESTION table contains information related to the questions inserted in the database and the MCQCHOICE table contains information related to the choices like id, choice label, question-id and valid flag.

**DAOs**

**Create**

**Search**

**Update**

**Delete**

**QUESTION**

**MCQCHOICE**

## **Web Operations Description**

### **Creating an MCQ**

* This operation is responsible for creating an MCQ and storing it in the database. It takes the question label and choices (choice label, valid) as input from the user.

### **Updating an MCQ**

* This operation is responsible for updating an MCQ and storing it in the database. It takes the question label and choices (choice label, valid) as input from the user.

### **Searching an MCQ**

* This operation is responsible for searching an identity from the database. It takes the question label as input from the user for search criteria.

### **Deleting an MCQ**

* This operation is responsible for deleting an MCQ from the database. It takes the id of the question as input from the user.

## **Configuration Instructions**

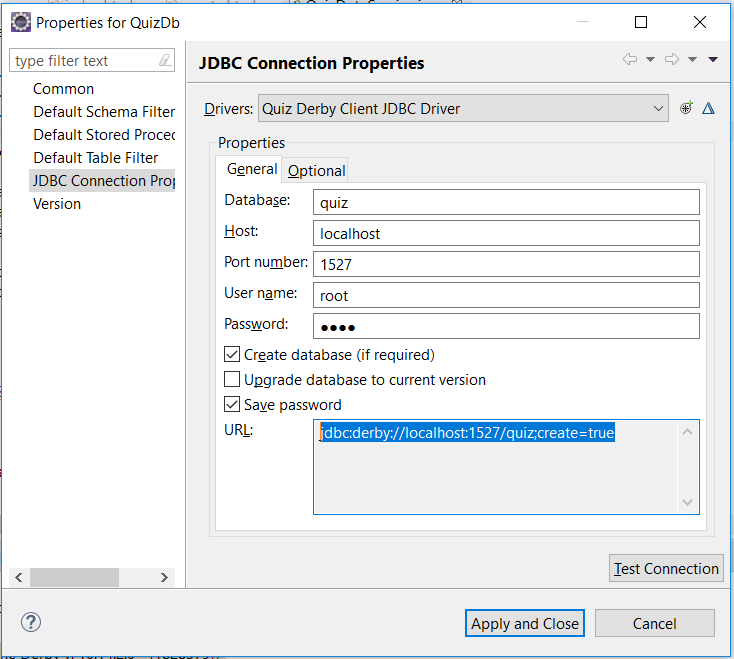
### **Development Tools**

* Eclipse IDE (Java EE IDE for Web Developers, version: Oxygen)
* Derby database (10.14.2)
* Java JDK 8
* Log4j libraries for logging
  1. Log4j-api-2.11.0
  2. Log4j-core-2.11.0

### **Schema and Database Configuration**

* The application mainly uses two database tables: “QUESTION” for user management and “MCQCHOICE” for managing the identities.
* Following are the database properties:

|  |  |
| --- | --- |
| Database properties | Values |
| Host | localhost |
| Port | 1527 |
| Username | root |
| Password | root |
| URL | jdbc:derby://localhost:1527/quiz;create=true |



## **Commented Screenshots**

These screenshots are covered as part of user guide.

## **Bibliography**

* Object Oriented concepts and project scope: <http://thomas-broussard.fr/>
* For programming:

<https://stackoverflow.com>

<https://www.w3schools.com/>

[http://cxf.apache.org/docs/jax-rs-client-api.html](https://www.google.com/url?q=http://cxf.apache.org/docs/jax-rs-client-api.html&sa=D&source=hangouts&ust=1543096993212000&usg=AFQjCNGO8U0fkKK3ux059bx6DIOXs4X9bg)

<https://mvnrepository.com/> (Maven repository)