EECS 221

Fall 2017

MIni Project 2 - Web Application with Database

Assigned on: 10/19/2017

In this mini-project, you will learn how to set up a web server and create a web application that can interact with a database. The project is due on Thursday, November 9, 11:59PM.

Deliverables

A .jsp file or a .py file that is the webpage you write to interact with MySQL.

STEP 1 - Install JDK

If you're using Java, you can type "java –version" in the command line to see if your system already has Java or not. If you don't have JDK or JRE on your computer, go to the JAVA download page: http://www.oracle.com/technetwork/java/javase/downloads/index.html) to download the JDK. Select the correct package for your system and download it. Execute the package to install it.

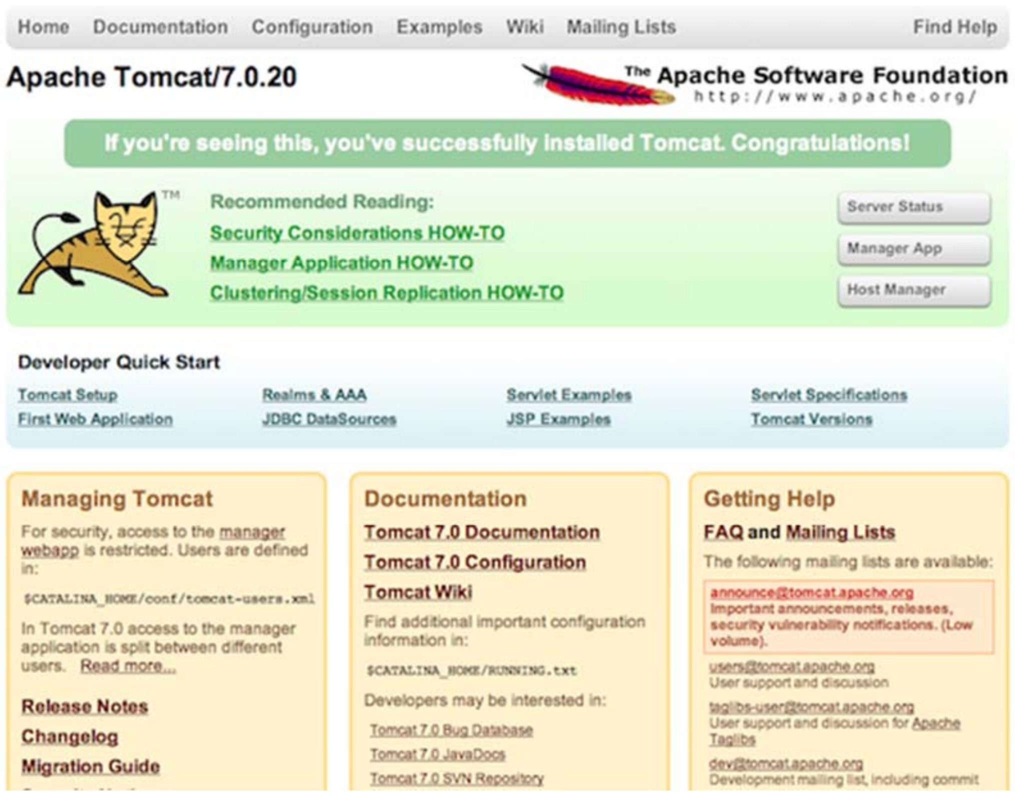
If you're using Python, you need to use python-cgi to work with the Apache CGI server. Please follow the link below for the instructions, and skip STEP 2 & STEP 4: https://docs.google.com/presentation/d/1uSGZqKRmMaAeOuhAdHAy7hwr9NHwXrfOlZGZNrkAVb0/edit#slide=id.p

STEP 2 – Install Apache Tomcat

Go to the Tomcat download page (http://tomcat.apache.org/) to download Tomcat 7.0. Download "tar.gz" under the "core" section. If you are using Windows, you may also choose 32-bit/64-bit Windows Service Installer which will install Tomcat as a service. Instructions for MAC can be found at: https://wolfpaulus.com/mac/tomcat/

The following steps are based on "tar.gz" on Windows 7. Extract the file to wherever you like. Go to System Properties->Advanced system settings -> Environment Variables. If you are using JRE, add JRE_HOME to the path you installed the JRE (e.g., JRE_HOME=C:\Program Files (x86)\Java\jre7). If you are using JDK, add JAVA_HOME to the path you installed the JDK (e.g., JAVA_HOME = C:\Program Files\Java\jdk1.7.0_02). If you cannot find the place to set the environment variables, you may also edit setclasspath.bat in the /bin folder under your Tomcat; and add "set JRE_HOME=C:\Program Files (x86)\Java\jre7" at the beginning of the file.

Double click startup.bat in the /bin folder to start the Tomcat. If everything goes well, your will see "Server startup in xxxx ms". Open your browser and type in "http://localhost:8080/". You will see the following page if your Tomcat was installed successfully:



If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations HOW-TO
Manager Application HOW-TO
Clustering/Session Replication HOW-TO

Developer Quick Start

Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Servlet Examples
JSP Examples
Servlet Specifications
Tomcat Versions

Managing Tomcat

For security, access to the manager webapp is restricted. Users are defined in:

```
$CATALINA_HOME/conf/tomcat-users.xml
```

In Tomcat 7.0 access to the manager application is split between different users. Read more...

Release Notes
Changelog
Migration Guide

Documentation

Tomcat 7.0 Documentation
Tomcat 7.0 Configuration
Tomcat Wiki

Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:

Tomcat 7.0 Bug Database
Tomcat 7.0 JavaDocs
Tomcat 7.0 SVN Repository

Getting Help

FAQ and Mailing Lists

The following mailing lists are available:

announce@tomcat.apache.org
Important announcements, releases, security vulnerability notifications. (Low volume).

users@tomcat.apache.org
User support and discussion

taglibs-user@tomcat.apache.org
User support and discussion for Apache Taglibs

dev@tomcat.apache.org
Development mailing list, including commit

STEP 3 - Create a Database and Tables
The schemas below are assumed for the data in a gallery. There are a total of 4 tables. The name of the database should be "gallery".

gallery (gallery_id, name, description)
primary key(gallery_id)

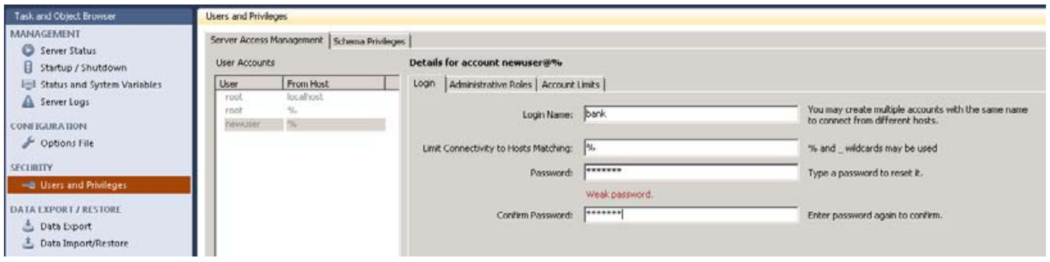
image (image_id, title, link, gallery_id, artist_id, detail_id)
primary key(image_id)

artist (artist_id, name, birth_year, country, description)
primary key(artist_id)
detail (detail_id, image_id, year, type, width, height, location, description)
primary key(detail_id)

This is a gallery database and you are going to write a management system. Each gallery contains a number of artworks which are stored as images. Each tuple in the image table stores the title of an artwork, the link of the picture of the artwork, and the ids that can connect the artwork to the gallery, the artist and details.

Each tuple in the artist table stores the details of an artist. Each tuple in the detail table stores the details of the artwork, including the year of creation, type of the artwork (e.g., painting), the current location of the artwork (to be simple, just country), and descriptions.

Now we are going to create a user account that can access this schema. Open your MySQL Workbench, click on the management tag on the left. Select "User and Privileges" and add a new account with login name "gallery" and password "eecs118".

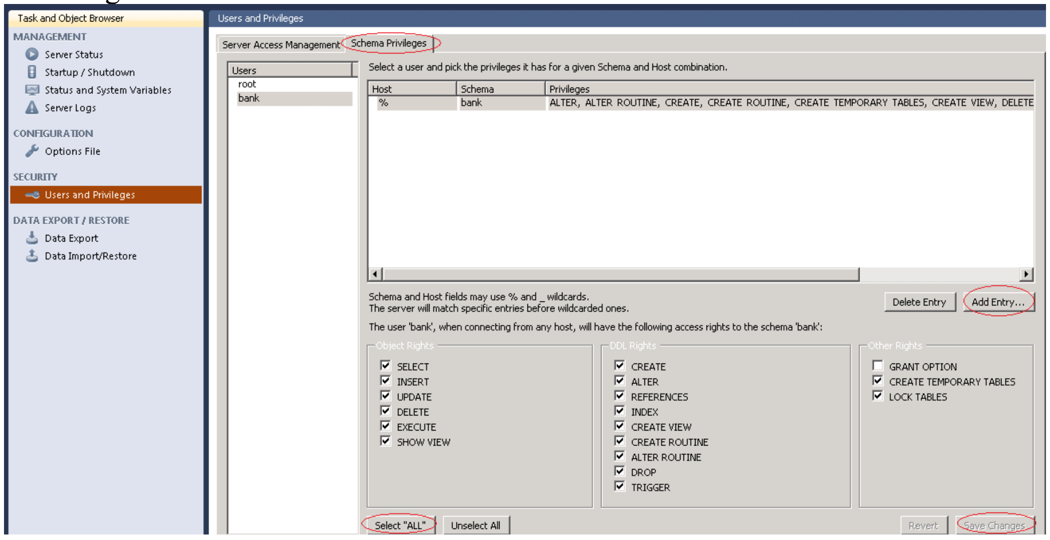


User	From Host
root	localhost
root	%
newuser	%

Details for account newuser@%

Login Name: bank — You may create multiple accounts with the same name to connect from different hosts.
Limit Connectivity to Hosts Matching: % — % and _ wildcards may be used
Password: ******** — Type a password to reset it.
Weak password.
Confirm Password: ******** — Enter password again to confirm.

Click on the "Schema Privileges" tag; click on the user "gallery" and then "Add Entry…"; and select the schema "gallery". Click on the "Select All" button and save the changes.



Select a user and pick the privileges it has for a given Schema and Host combination.

Host	Schema	Privileges
%	bank	ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE VIEW, DELETE

Schema and Host fields may use % and _ wildcards.
The server will match specific entries before wildcarded ones.
The user 'bank', when connecting from any host, will have the following access rights to the schema 'bank':

Object Rights: [✓] SELECT, [✓] INSERT, [✓] UPDATE, [✓] DELETE, [✓] EXECUTE, [✓] SHOW VIEW
DDL Rights: [✓] CREATE, [✓] ALTER, [✓] REFERENCES, [✓] INDEX, [✓] CREATE VIEW, [✓] CREATE ROUTINE, [✓] ALTER ROUTINE, [✓] DROP, [✓] TRIGGER
Other Rights: [ ] GRANT OPTION, [✓] CREATE TEMPORARY TABLES, [✓] LOCK TABLES

Details of the tables

```
CREATE TABLE `artist` (
  `artist_id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) DEFAULT NULL,
  `birth_year` int(11) DEFAULT NULL,
  `country` varchar(45) DEFAULT NULL,
  `description` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`artist_id`)
)

CREATE TABLE `detail` (
  `detail_id` int(11) NOT NULL AUTO_INCREMENT,
  `image_id` int(11) DEFAULT NULL,
  `year` int(11) DEFAULT NULL,
  `type` varchar(45) DEFAULT NULL,
  `width` int(11) DEFAULT NULL,
  `height` int(11) DEFAULT NULL,
  `location` varchar(45) DEFAULT NULL,
  `description` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`detail_id`)
)

CREATE TABLE `gallery` (
  `gallery_id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) DEFAULT NULL,
  `description` varchar(2000) DEFAULT NULL,
  PRIMARY KEY (`gallery_id`)
)

CREATE TABLE `image` (
  `image_id` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(45) DEFAULT NULL,
  `link` varchar(200) DEFAULT NULL,
  `gallery_id` int(11) DEFAULT NULL,
  `artist_id` int(11) DEFAULT NULL,
  `detail_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`image_id`)
)
```

No foreign keys or triggers are used in the schema.

You will have to maintain the data consistency by your application. Make sure that the names of the schema, tables, attributes and the account you create are exactly the same as what is in the descriptions.

STEP 4 – Design and Implement the Web Application
We introduce how to set up and create a web application that can interact with MySQL here. To connect MySQL from Java, you have to use the JDBC driver from MySQL. The MySQL JDBC driver is called “MySQL Connector/J”. You should be able to find the latest MySQL JDBC driver on this page: http://dev.mysql.com/downloads/connector/j/5.0.html

Download the file, extract it, and put the mysql-connector-java-5.0.8-bin.jar file to the /lib directory under Tomcat. Restart Tomcat.

Now you can start to write your web application. Create a folder called “gallery” under the /webapps directory of Tomcat. Create a file named “index.jsp” under the gallery folder. This index.jsp will be your web page, in which you can write standard HTML code as in a normal web page. In addition you can add Java code, quoted by “<%” and “%>”. You can visit your page on http://127.0.0.1:8080/gallery/index.jsp

We provide sample code and you can visit http://128.195.204.85:8080/bank/index.jsp to see how it works. You can then start to modify the code to do this project and provide the requested functions described at the end of this document.

- Explanation of the Sample Code -
In the beginning of the file, import the library you need:

```
<%@ page contentType="text/html;charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import="java.util.*"%>
<%@ page import="java.io.*" %>
<%@ page import="java.sql.*"%>
<% String funcID = request.getParameter("funcID");
   String name = request.getParameter("name");
   ….                           %>
```

request.getParameter() can get the attributes submitted by the form you define:

```
<form method="post">
        <input name="funcID" type="hidden" value="2">
                Customer name: <input name="name" type="text">
                Street: <input name="street" type="text">
                City: <input name="city" type="text">
        <input type="submit" value="Add"/>
</form>
```

To Load the SQL driver:

```
try {
        Class.forName("com.mysql.jdbc.Driver").newInstance();
}
catch(Exception e) {
out.println("can't load mysql driver");
out.println(e.toString());
}
```

To Create a Connection with MySQL Server for the database "bank":

```
String url="jdbc:mysql://127.0.0.1:3306/bank";
                                String id="bank";
                                String pwd="eecs118";
                                Connection con=
DriverManager.getConnection(url,id,pwd);
```

In the url part, 127.0.0.1 is the address of the server (127.0.0.1 means the localhost which is your own computer; if you want to connect to other computers just change the address), and "bank" is the database name. The program uses user id "bank" and password "eecs118" to login. When you write program for this project, you should use "eecs118" as the login and "gallery" as the password.

To Perform SQL Queries:

```
stmt = con.createStatement();
String sql="SELECT * FROM customer";
rs=stmt.executeQuery(sql);
…
while (rs.next()) {
        out.println("<tr>");
        out.println("<td>"+rs.getString("customer_id")+"</td>");
        out.println("<td>"+rs.getString("customer_name")+"</td>");
        out.println("<td>"+rs.getString("customer_street")+"</td>");
        out.println("<td>"+rs.getString("customer_city")+"</td>");
        out.println("</tr>");
}
```

- Use executeQuery() to query the data from the table.
- Use rs.getString("Attribute") to get the value of the attribute from a tuple.
- Out.println() can write the content back to the webpage HTML.

To Add a Tuple into a Table:

```
pstmt = con.prepareStatement("insert into customer values
(default,?,?,?)",Statement.RETURN_GENERATED_KEYS);
pstmt.clearParameters();
pstmt.setString(1, name);
pstmt.setString(2, street);
pstmt.setString(3, city);
pstmt.executeUpdate();
rs=pstmt.getGeneratedKeys();
while (rs.next()) {
        out.println("Successfully added. Customer_ID:"+rs.getInt(1));
}
```

Note that:
- PreparedStatements can use variables and are more efficient.
- "default" means to use the default value for customer_ID, which should be auto incremented.
- "Statement.RETURN_GENERATED_KEYS" means you want to know the auto-generated customer_ID after insertion.
- Use executeUpdate() to perform any modification to the data, e.g., insert, delete.
- Use re.getInt(1) to obtain the first attribute from the result (with type Integer), which should be the customer_ID generated.

STEP 5 – Requirements

In this project, you are required to write the following functions in your web application:

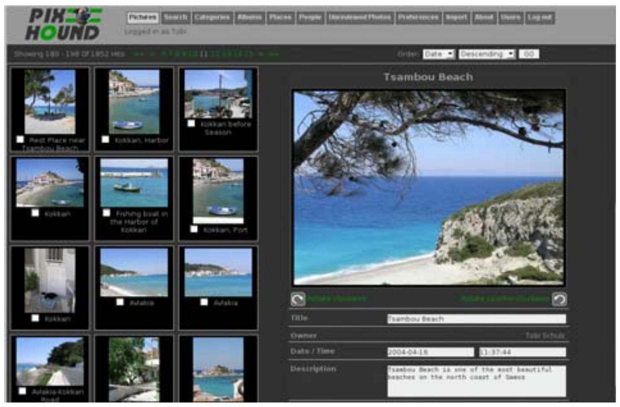
1. List all the galleries (including name and descriptions).
2. List all the images and the number of images in a gallery (including title and link).
3. List the details of a given image (This function should be a link from the result of function 2. It should show the picture, the artist name, and all the details of the artwork.)
4. List the details of an artist. (This function should be a link from the result of function 3.)
5. Create a new gallery.
6. Create a new artist.
7. Add a new image to a gallery. To simplify, you can just input the link of the picture instead of actually uploading it. This function should also input the artist_id, and input all the details of the image.
8. Delete an image from the gallery. You should also delete the details of the

image but not the artist.

9. Modify the details of an image (including title and link).
10. Modify the details of an artist.
11. Modify the title and description of a gallery.
12. Find the images by type ("Find" means to list all the results.)
13. Find the images by a given range of creation year.
14. Find the images by artist name.
15. Find the images by location.
16. Find the artists by country.
17. Find the artists by birth year.

Extra Credits:

(A) Assemble the above functions and design a modern album/gallery interface. For example, when listing the gallery, also show the first picture of that gallery, and you can just click on it to go into the gallery. When in the gallery, you can see the thumbnails of all the images. The search functions can be merged in a user-friendly search interface. Following picture is a good example:



(B) When adding an image, you can actually upload the image instead of just a url link. You should put the uploaded images under the /image subfolder, and, if necessary, write descriptions to explain how to make this function work if your program needs some other tools.

Please turn in your JSP file or Python file in the EEE dropbox under mini-project 2. The name of the file should be "xxxxxxxx_index.jsp"or "xxxxxxxx_index.py", xxxxxxxx being your student id. If you implement extra credits and have more than one file to turn in, you should archive the files into "mini-project2-xxxxxxxx.zip", xxxxxxxx being your student id.

Procedure of our testing

We will put your web application on our own server. Make sure to use a relative URL for any link of local resources. The test will start with an empty database. Then we will try to add new galleries, new artists, and new images. If your application doesn't support uploading images, we will use some existing external URLs of images.

We will only input normal data and will not try to input a value that may exceed the column size. However, if you implement data validation that will be great, as it can be a part of extra credits in your UI.

After adding the items, we will then test the search functions, modification functions and delete functions. If you have any special requirements for your application to run correctly, please contact the TA.