# Start Your Project

Environment setup and template project

Notes on Web DB Programming

# Outline

- Notes on Web DB Programming
- Set up Environment and Introduction of the template project

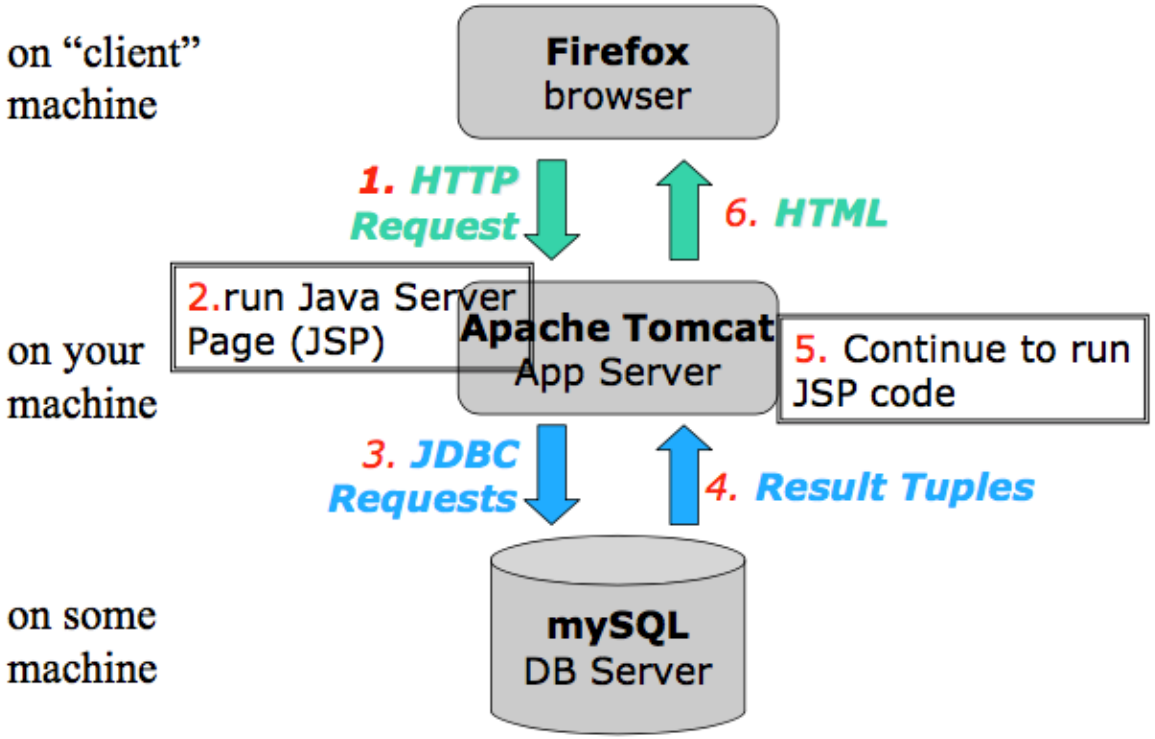
# Notes on Web DB Programming

Based on M.Muscari and UCSD (anon)

# Three-Tier Application

- **Presentation Tier** : user interface to make requests, provide input and see results
- **Middle Tier**: application logic
- **Data Management Tier:** database management

# Three-Tier architecture



# HTTP protocol

- Protocol that allows web servers and clients to **exchange data** over the web.
- It is a **request - response protocol**.
- Clients (web browsers) send requests to web servers
  - GET : ask for a resource
  - POST : send some data (e.g. HTML form)
- Server sends response
  - Status code (200 OK, 404 Not Found!)
- HTTP is a "***stateless***" protocol; each time a client retrieves a Webpage, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request.

# Difference between GET/POST requests

**Anatomy of GET request**


**Anatomy of POST request**

# HTML (Hyper Text Markup Language)

- Standard markup language for creating web pages
  - Language for creating **structured documents**
    - It consists of elements which can be nested
  - The HTML standard specifies a number of universally supported elements (“tags”)
- Web browsers receive HTML documents from a webserver and render them into multimedia web pages.
- HTML is commonly delivered as part of an HTTP response

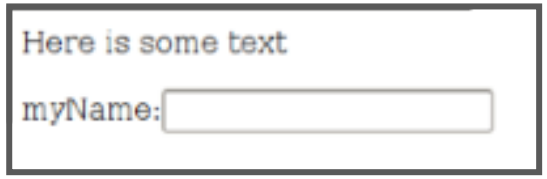
Tutorial: https://www.w3schools.com/html/

# HTML Tags

- Common HTML tags include:
  - ** ** - a logical division (section)
  - ** ** - a paragraph
  - **<table> </table>** -a table of values
    - **<tr> </tr>** - table row
    - **<td> </td>** -table column
  - **<form></form>** - a form enclosing input fields
    - **<input></input>** - an input field

# HTML Example

```
<html>
    <body>
        <div>
            <p>Here is some text</p>
            <form action="submit.jsp" method="post">
                myName: <input name="myInput" type="text"/>
            </form>
        </div>
    </body>
</html>
```



# HTML Table Example

```
<html>
    <body>
        <table border='1'>
            <tr>
                <td>First Name</td>
                <td>Last Name</td>
            </tr>
            <tr>
                <td>Matt</td>
                <td>Muscari</td>
            </tr>
        </table>
    </body>
</html>
```

| First Name | Last Name |
| --- | --- |
| Matt | Muscari |

# JSP (Java Server Pages)

- A technology for building web applications that serve **dynamic content**
- A **JSP page** is a text document that contains two types of text:
  - **static data**, which can be expressed in any text-based format (e.g. HTML)
  - JSP elements, which construct **dynamic content**.
- The dynamic content in a **JSP page** is in specially marked Java code fragments (enclosed between <% and %>).
- To deploy and run JSPs, a compatible web server with a servlet container, such as Apache Tomcat is required.
- When executed, the Java code fragments usually generate additional HTML into the page (in our case either accessing the database or processing parameters passed to HTTP requests)
- At the end, the resulting HTML page is sent to the browser to be displayed.

# JSP Syntax

- **Comment**
  <%-- Comment--%>
- **Expression**
  <%= Java expression %>
  **Expression** tag evaluates the **expression** placed in it, converts the result into String and send the result back to the client through response object.
  **e.g. Today is <%= new Date().toString(); %> **

- **Scriplet**
  <% java code fragment%>
  **e.g. <% person.getFirstName();%>**

- **Include**
  <jsp:include page="relativeURL"/>

# JSP Implicit Objects

| Object | Class |
| --- | --- |
| request | **HttpServletRequest** |
| response | **HttpServletResponse** |
| session | **HttpSession** |
| out | **Writer** |

# request

- <%@ page language=“java” contentType=“text/html”%>
  <html>
  <head>
  <title>RequestExamplePage</title>
- </head>
  <body>
  <%
  // Get the User's Name from the request
  out.println("Hello: " + request.getParameter(”myInput") + "");
  %>
  </body>

</html>

# session

- <%@ page language="java" contentType="text/html"%>

```
<html>
    <head>
        <title>SessionExamplePage</title>
    </head>
    <body>
        <%
        HttpSession session = request.getSession(); //create a session object
        // Try and get the current count from the session
        Integer count = (Integer)session.getAttribute("COUNT");
        // If COUNT is not found, create it and add it to the session
        if ( count == null ) {
            count = new Integer(1);
            session.setAttribute("COUNT", count);
        } else {
            count = new Integer(count.intValue() + 1);
            session.setAttribute("COUNT", count);
        }
        // Print the number of times the user has visited the site
        out.println("<b>Hello you have visited this site: " + count + " times. </b>");
        %>
    </body> </html>
```

# Java Database Connectivity (JDBC)

- An interface to communicate with a relational database
  - Allows database agnostic Java code
  - Treat database tables/rows/columns as Java objects
- JDBC driver
  - An implementation of the JDBC interface
  - Communicates with a particular database

# JDBC steps

1. Connect to database
2. Query database (or insert/update/delete)
3. Process results
4. Close connection to database

# 1. Connect to database

- Load JDBC driver
  - Class.forName("com.mysql.jdbc.Driver").newInstance();
  - Make connection
    - Connection conn = DriverManager.getConnection(**url**);
- URL
  - Format: “jdbc:mysql//<*hostname*>:<port>/<*databaseName*>”
  - jdbc:mysql://localhost:3036/BarBeerDrinkerSample

# 2. Query database

- Create statement
  - Statement stmt = conn.createStatement();
    - stmt object sends SQL commands to database
  - Methods
    - executeQuery() for SELECT statements
    - executeUpdate() for INSERT, UPDATE, DELETE, statements
- Send SQL statements
  - stmt.executeQuery("SELECT ...");
  - stmt.executeUpdate("INSERT ...");

# 2. Query database

- Prepared Statements
  - If you want to execute dynamic or parameterized SQL queries, use a “PreparedStatement” object instead of a statement.

```
PreparedStatement updateStud=conn.prepareStatement("UPDATE Student SET
fname=? WHERE lastname LIKE?");

updateStud.setString(1,"John");
updateStud.setString(2,"Doe");
updateStud.executeUpdate();
```

# 3. Process results

- Result of a SELECT statement (rows/columns) returned as a ResultSet object
  - ResultSet rs = stmt.executeQuery("SELECT drinker,beer from LIKES");
- Step through each row in the result
  - rs.next()
- Get column values in a row
  - String userid = rs.getString(“drinker”);
  - int type = rs.getInt(“type”);

# 3. Process results

➢Add a row to the users table

- String str = "INSERT INTO LIKES VALUES('Bob', ‘Corona')”;

➢Returns number of rows in table

- int rows = stmt.executeUpdate(str);

# 4. Close connection to database

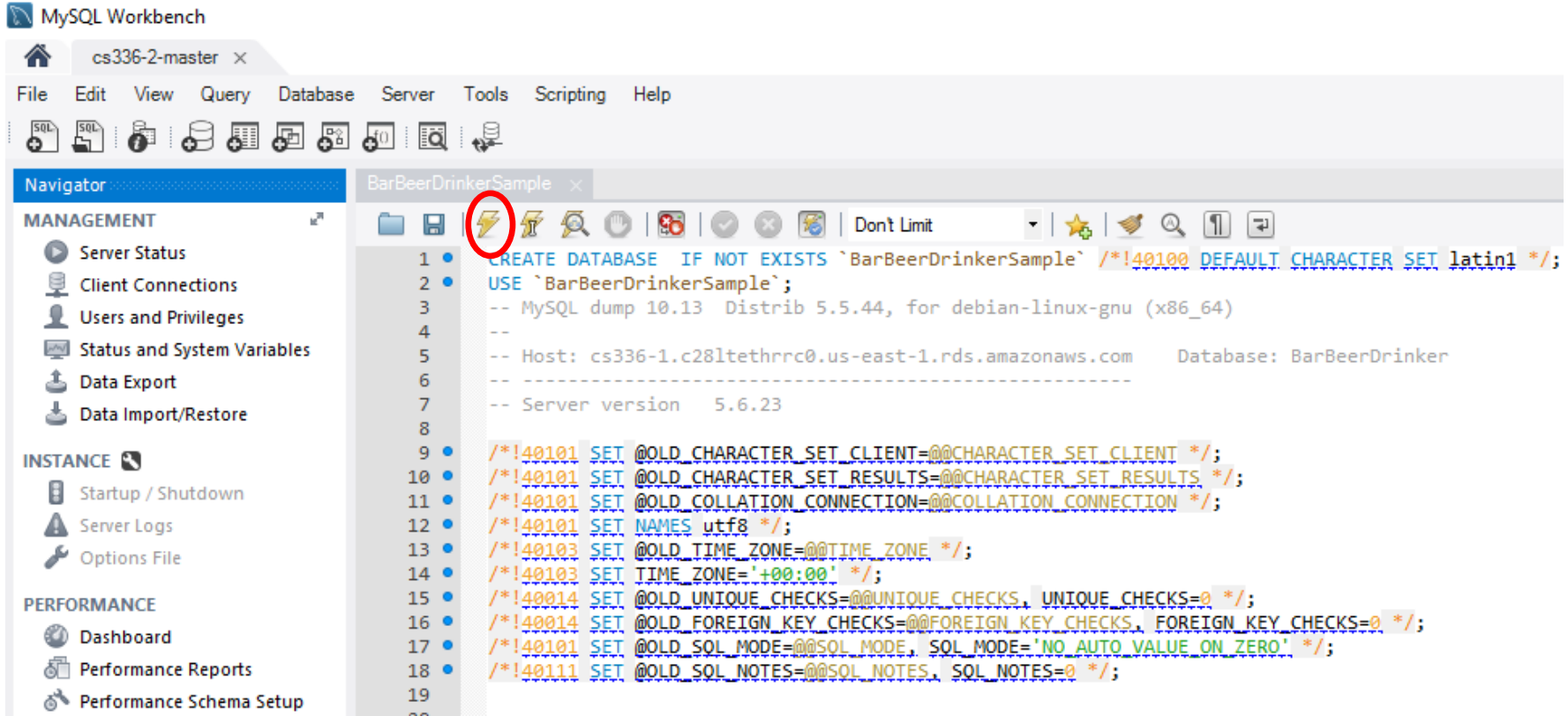
- Close the ResultSet object
  - rs.close();
- Close the Statement object
  - stmt.close();
- Close the connection
  - conn.close();

# Needed tools and installation

- **JRE, IDE** (JAVA, Eclipse for EE developers)
- **MySQL**
- **Apache Tomcat** (or any web server)
  - You will install it locally in your computer for development purposes.
- **JDBC**

# The Template Project

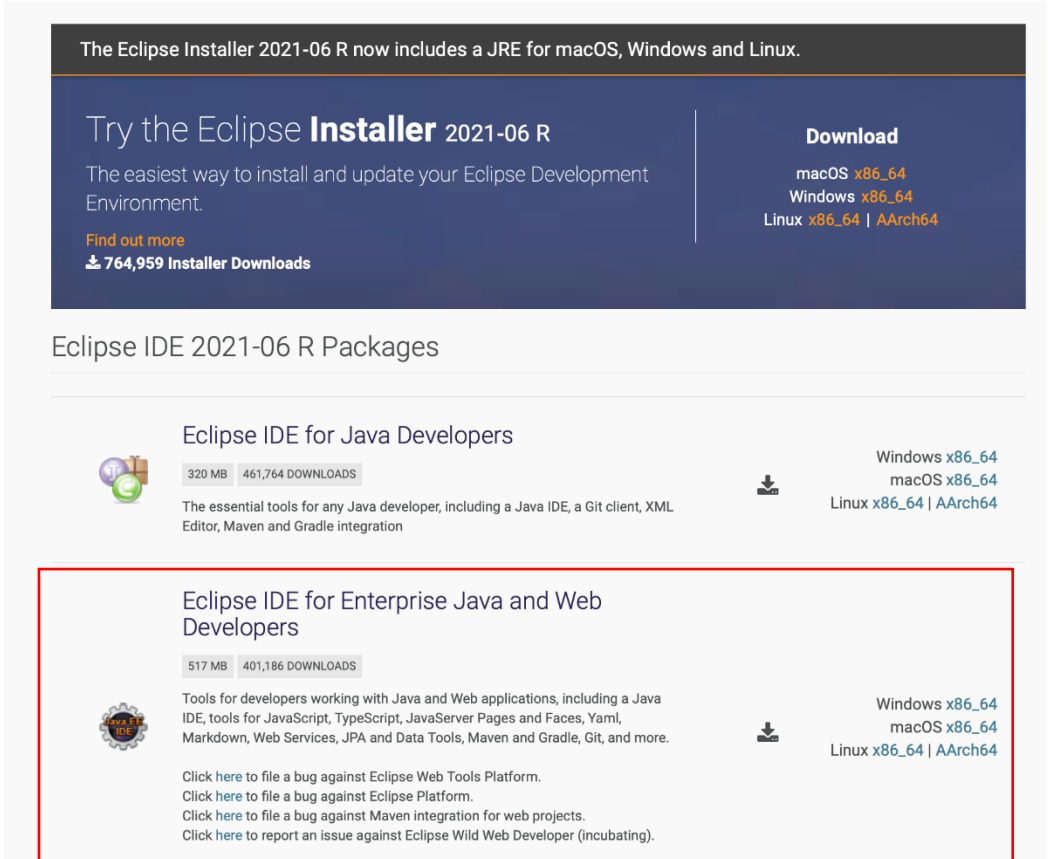
**0.** import schema **BarBeerDrinkerSample** in your DB instance using the provided script ”BarBeerDrinkerSample.sql”. Open the script and run it in your MySqlWorkbench. (File->Open SQL script)



# The Template Project

**1.** Download Eclipse IDE for **Java EE** Developers

https://eclipse.org/downloads/eclipse-packages/



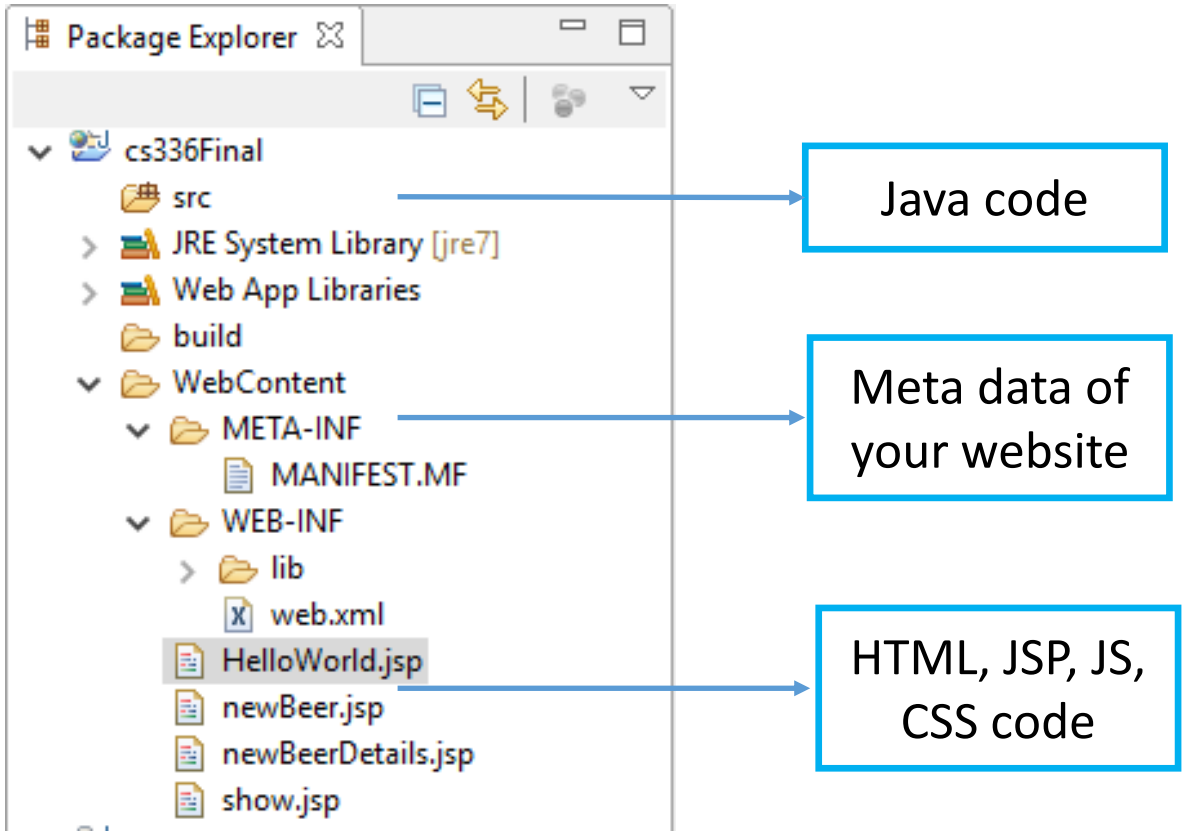
# The Template Project

**2.** Open eclipse and import the template project (cs336Sample.war)

File – Import – Web – WAR file

# The Template Project

**3.** Structure of the template



# The Template Project

**4.** Set your Tomcat server in eclipse

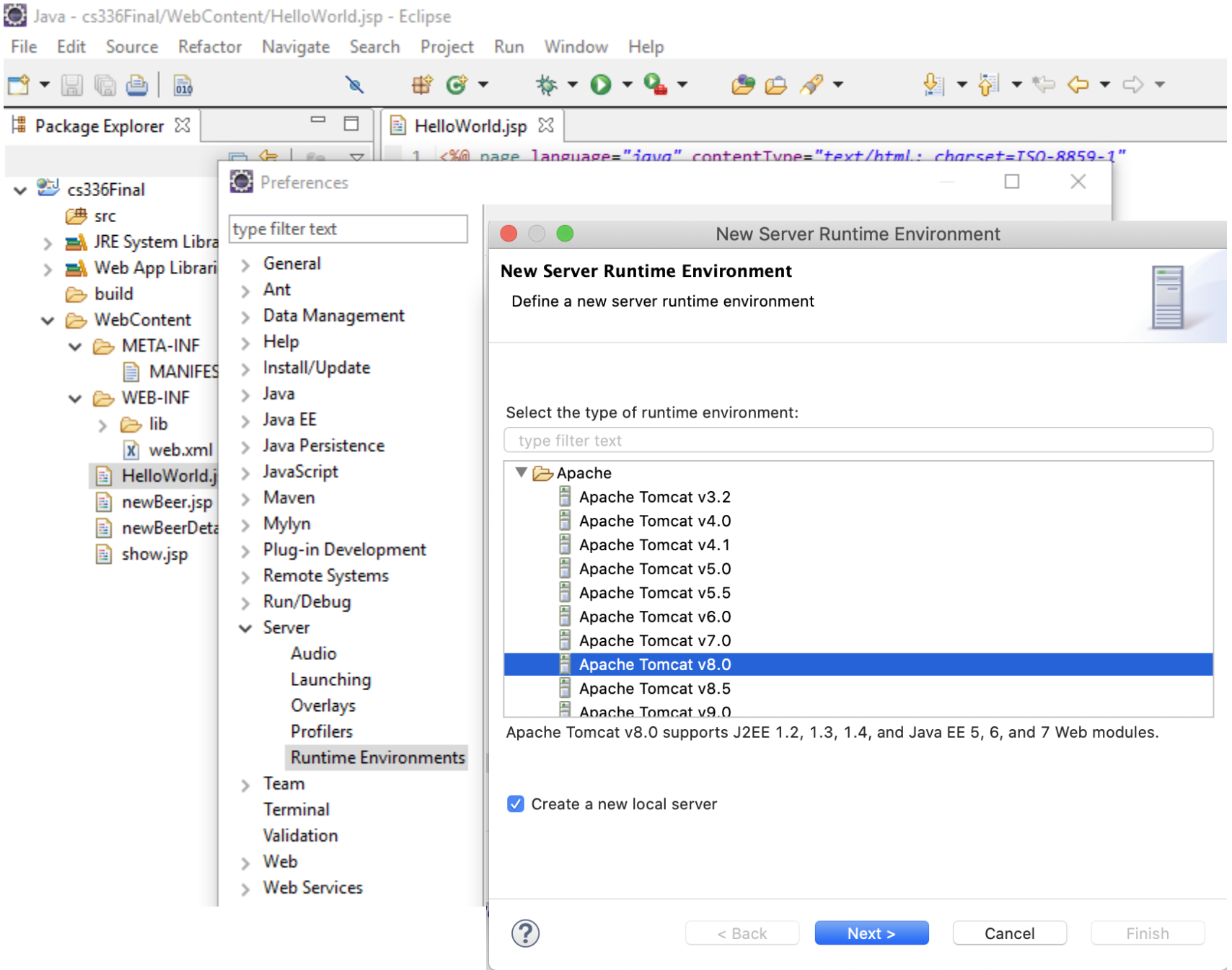
- If you don’t have tomcat yet go to: https://tomcat.apache.org/download-80.cgi

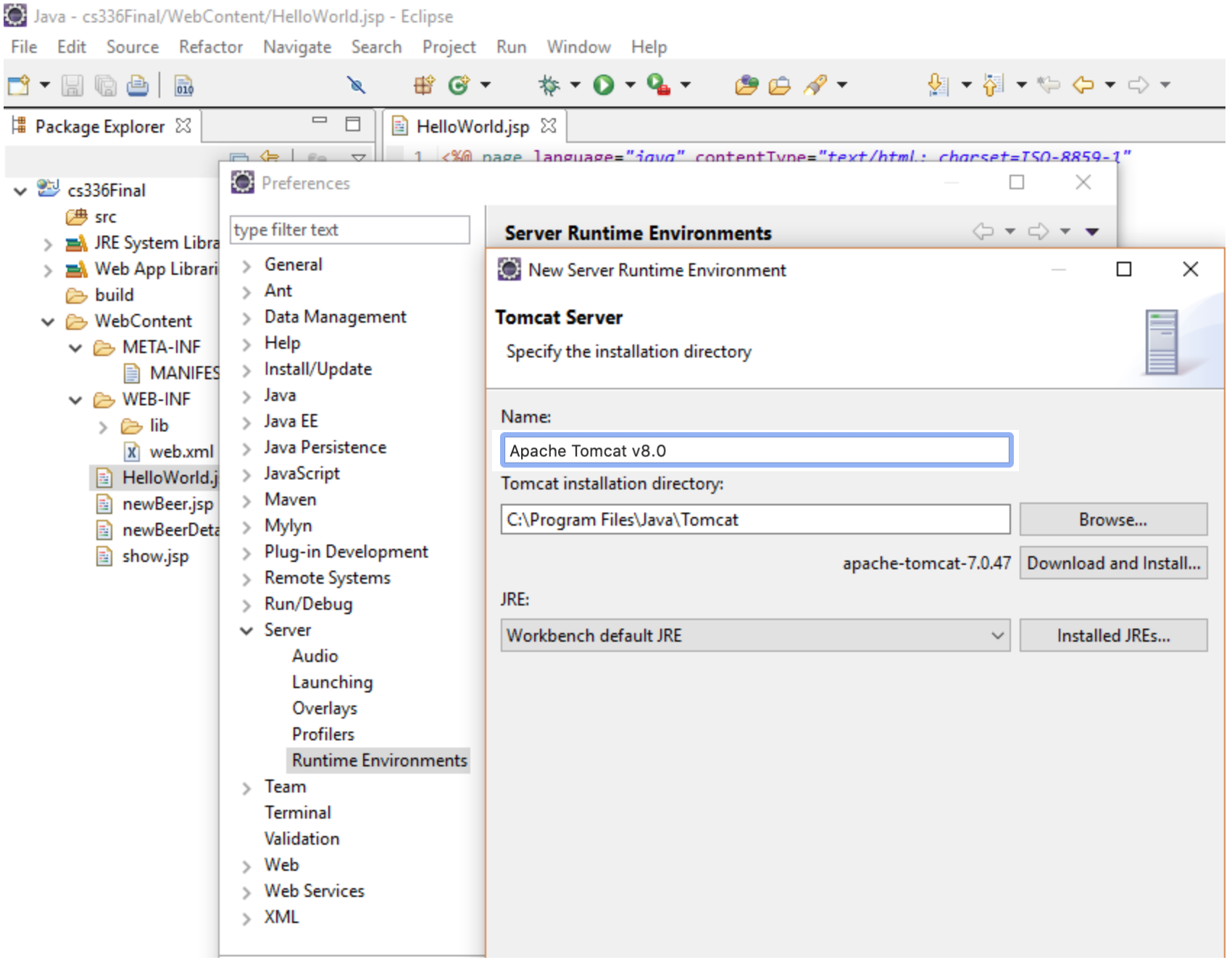
and download the binary distribution for your OS.

- After go back to eclipse:

Windows - Preference - Server - Runtime Environment - Add - Apache Tomcat v8.0 **or**

Eclipse- Preferences - Server - Runtime Environments - Add - Apache Tomcat v8.0


Java - cs336Final/WebContent/HelloWorld.jsp - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
HelloWorld.jsp
cs336Final
src
JRE System Libra
Web App Librari
build
WebContent
META-INF
MANIFES
WEB-INF
lib
web.xml
HelloWorld.j
newBeer.jsp
newBeerDeta
show.jsp
Preferences
type filter text
General
Ant
Data Management
Help
Install/Update
Java
Java EE
Java Persistence
JavaScript
Maven
Mylyn
Plug-in Development
Remote Systems
Run/Debug
Server
Audio
Launching
Overlays
Profilers
Runtime Environments
Team
Terminal
Validation
Web
Web Services
New Server Runtime Environment
New Server Runtime Environment
Define a new server runtime environment
Select the type of runtime environment:
type filter text
Apache
Apache Tomcat v3.2
Apache Tomcat v4.0
Apache Tomcat v4.1
Apache Tomcat v5.0
Apache Tomcat v5.5
Apache Tomcat v6.0
Apache Tomcat v7.0
Apache Tomcat v8.0
Apache Tomcat v8.5
Apache Tomcat v9.0
Apache Tomcat v8.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, and 7 Web modules.
Create a new local server
< Back
Next >
Cancel
Finish


Java - cs336Final/WebContent/HelloWorld.jsp - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
HelloWorld.jsp
cs336Final
src
JRE System Libra
Web App Librari
build
WebContent
META-INF
MANIFES
WEB-INF
lib
web.xml
HelloWorld.j
newBeer.jsp
newBeerDeta
show.jsp
Preferences
type filter text
General
Ant
Data Management
Help
Install/Update
Java
Java EE
Java Persistence
JavaScript
Maven
Mylyn
Plug-in Development
Remote Systems
Run/Debug
Server
Audio
Launching
Overlays
Profilers
Runtime Environments
Team
Terminal
Validation
Web
Web Services
XML
Server Runtime Environments
New Server Runtime Environment
Tomcat Server
Specify the installation directory
Name:
Apache Tomcat v8.0
Tomcat installation directory:
C:\Program Files\Java\Tomcat
Browse...
apache-tomcat-7.0.47
Download and Install...
JRE:
Workbench default JRE
Installed JREs...

# The Template Project

**5.** Run the project based on Tomcat 8

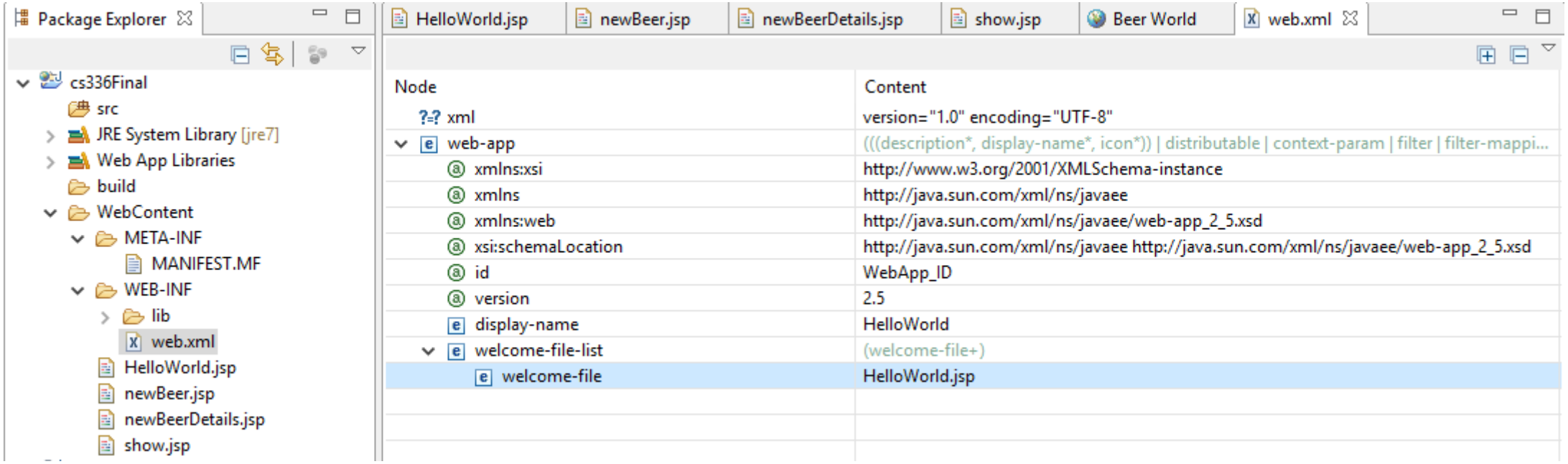
Right click on the project - Run as - Run on Server - Apache – Tomcat8

# The Template Project

- Now you can see your project home page, HelloWorld.jsp page.

# The Template Project

**6.** The home page is set in web.xml, you can set your own page if you want.

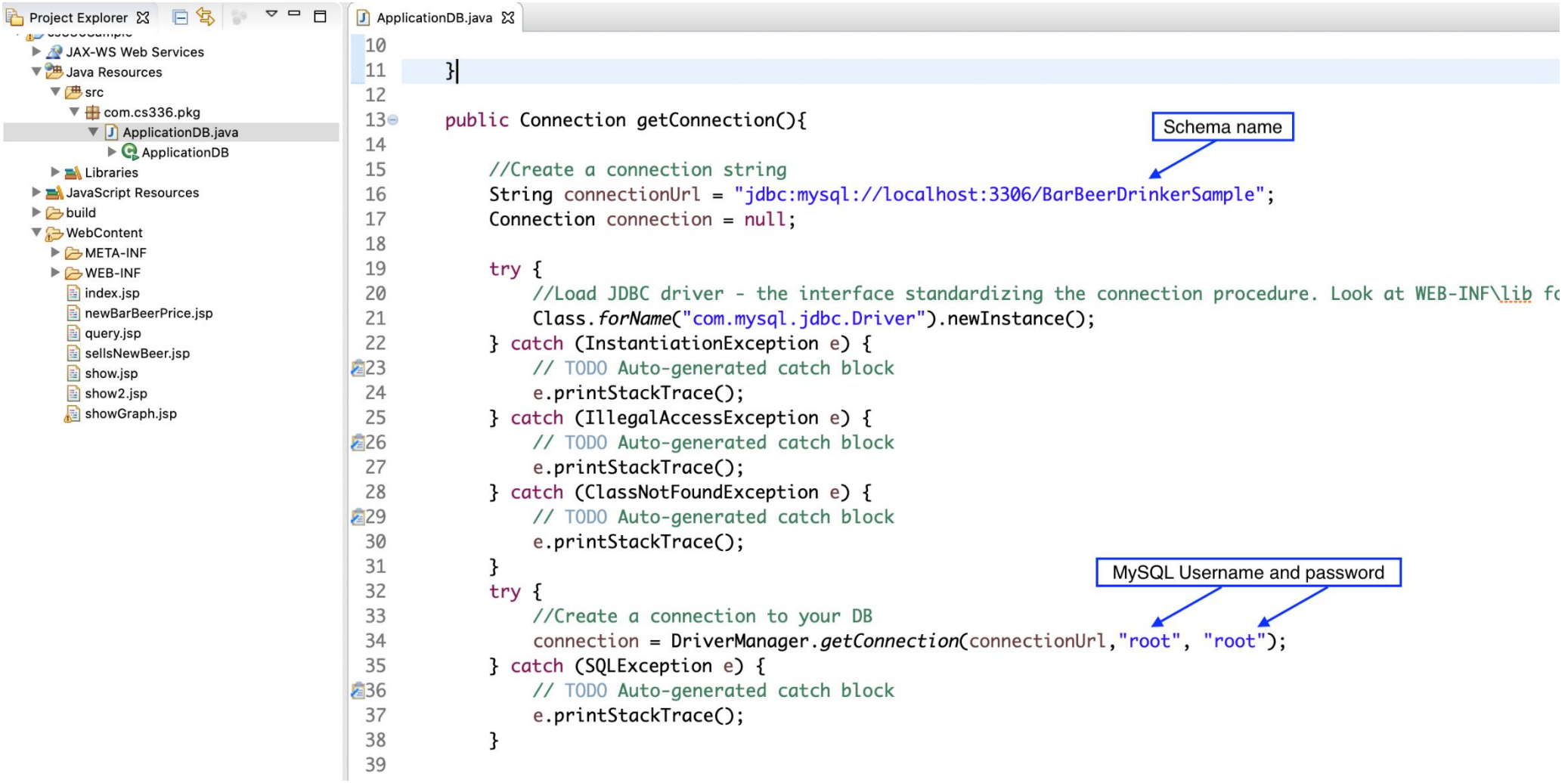


# The Template Project

**7.** Connect to your own db instance in Project

- In order to interact with db instance (add, delete, update, select), you need to set your own database address in the project.
- At the same time, the database username and password are both essential.
- Replace the database information with your own database information as follows.

# The Template Project

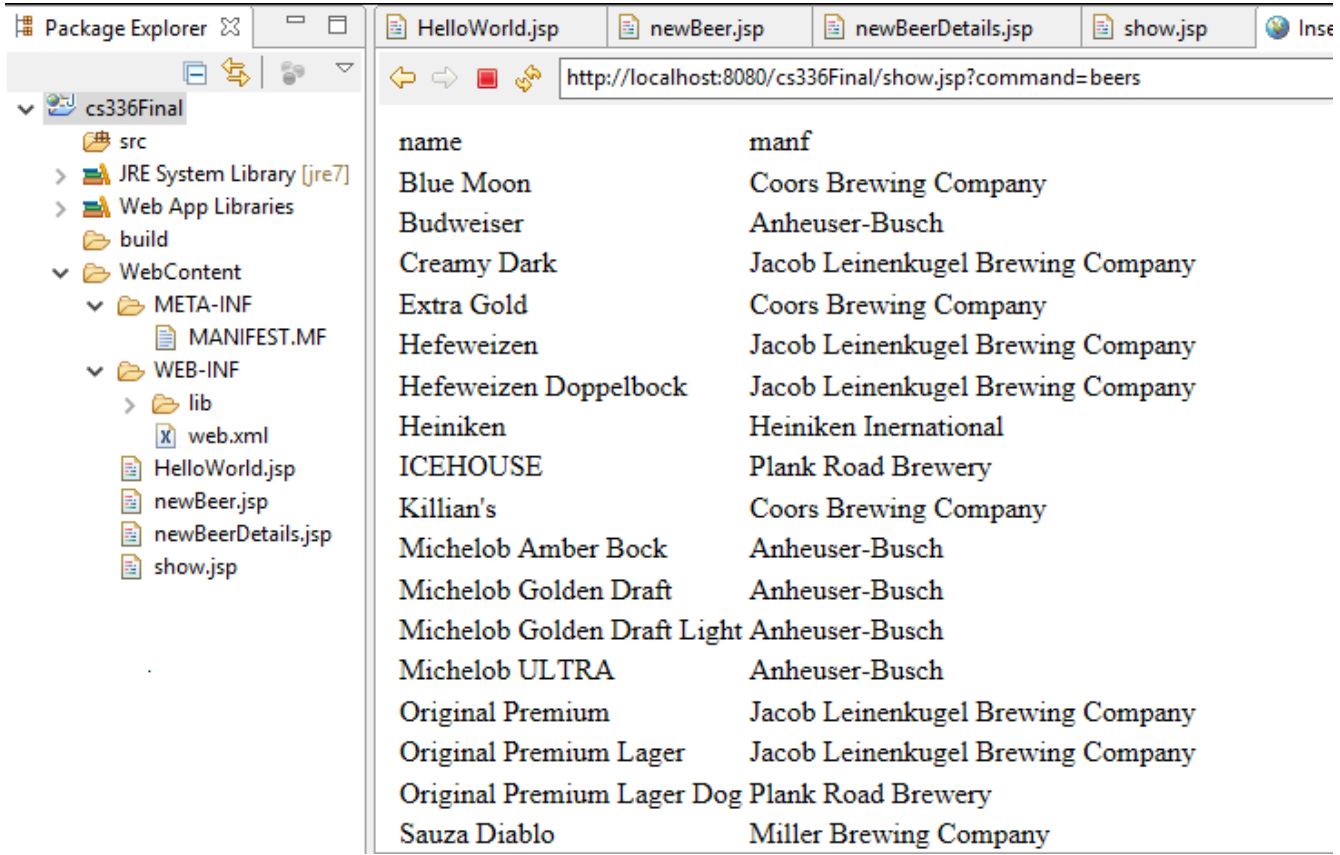



```
10
11      }
12
13      public Connection getConnection(){
14
15          //Create a connection string
16          String connectionUrl = "jdbc:mysql://localhost:3306/BarBeerDrinkerSample";
17          Connection connection = null;
18
19          try {
20              //Load JDBC driver - the interface standardizing the connection procedure. Look at WEB-INF\lib fo
21              Class.forName("com.mysql.jdbc.Driver").newInstance();
22          } catch (InstantiationException e) {
23              // TODO Auto-generated catch block
24              e.printStackTrace();
25          } catch (IllegalAccessException e) {
26              // TODO Auto-generated catch block
27              e.printStackTrace();
28          } catch (ClassNotFoundException e) {
29              // TODO Auto-generated catch block
30              e.printStackTrace();
31          }
32          try {
33              //Create a connection to your DB
34              connection = DriverManager.getConnection(connectionUrl,"root", "root");
35          } catch (SQLException e) {
36              // TODO Auto-generated catch block
37              e.printStackTrace();
38          }
39
```

# The Template Project

**8.** Let's have a beer

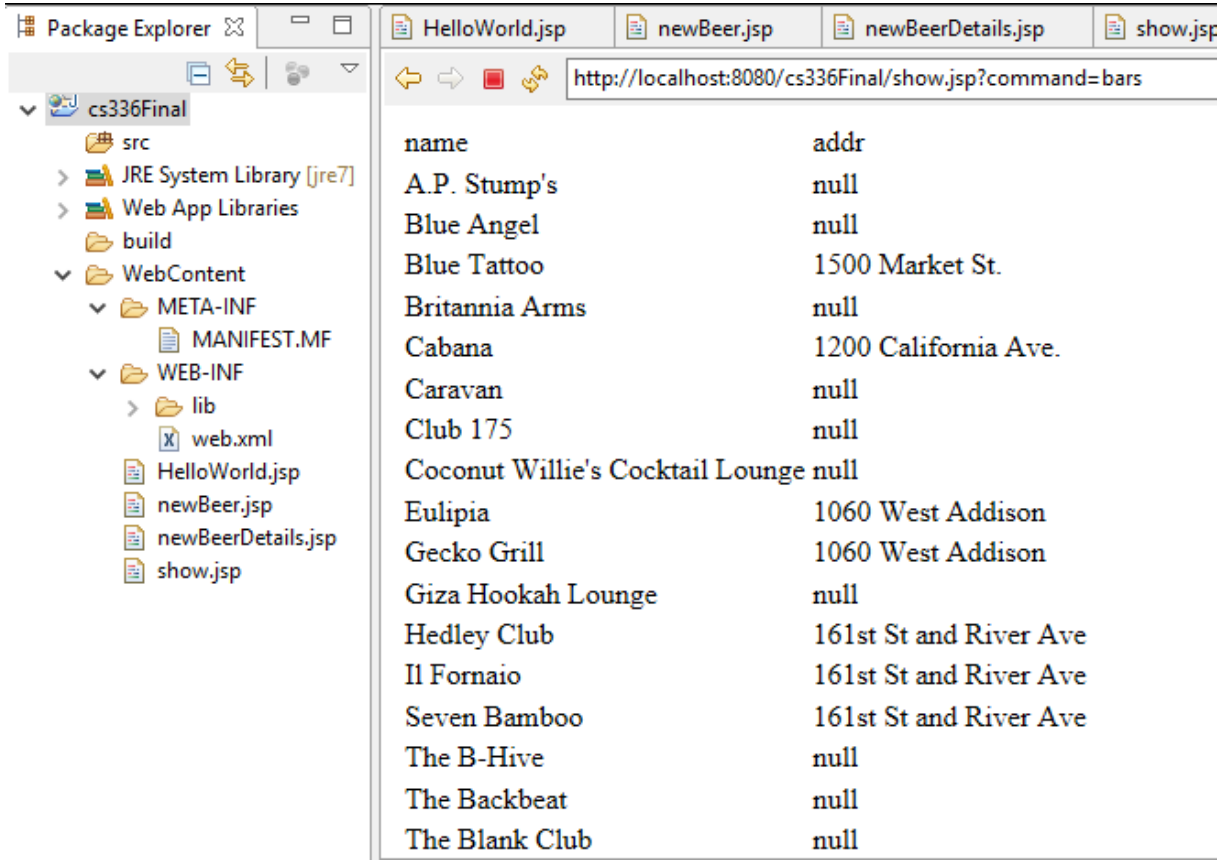
Select the radio button and then click submit below it



# The Template Project

**9.** Let's go to a bar

Select the radio button and then click submit below it



# The Template Project

**10.** Insert a tuple into sells table

Input bar name, beer name and cost, then click submit.

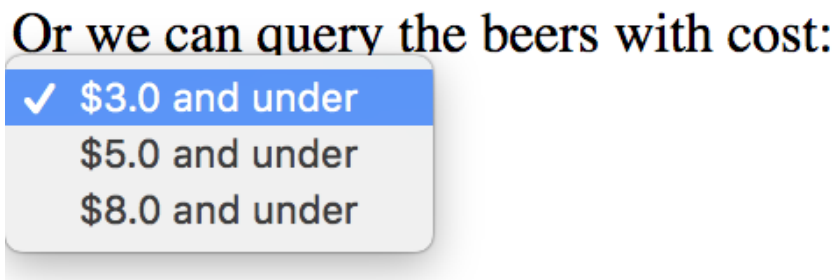
You can find a new record inserted into your database after submitting this form.

-**NOTE**: since you insert a tuple in sells table which has FKs in the bar and beer table, make sure the beer and bar you insert already exist in these two tables.

# The Template Project

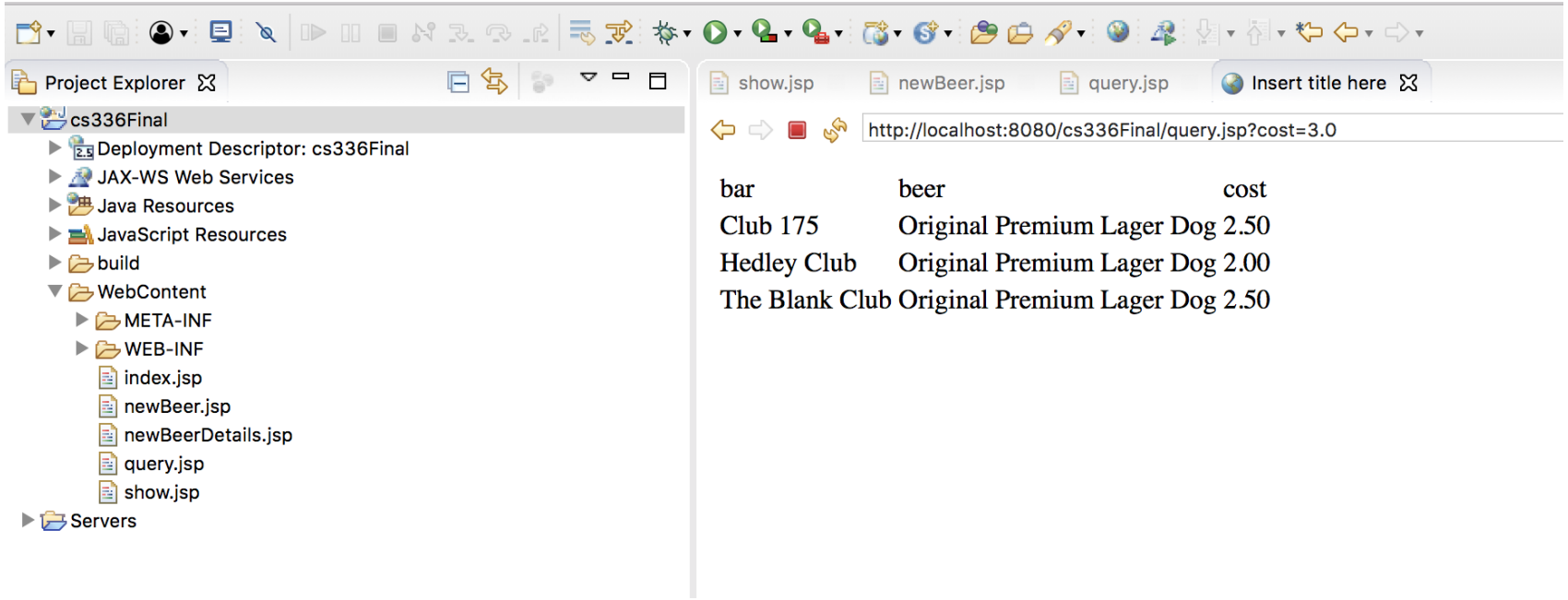
**11.** Query the beers with cost

Choose one option from the dropdown menu, then click submit.



# The Template Project

- Query the beers with cost <= 3



# The Template Project

- Query the beers with cost<= 5