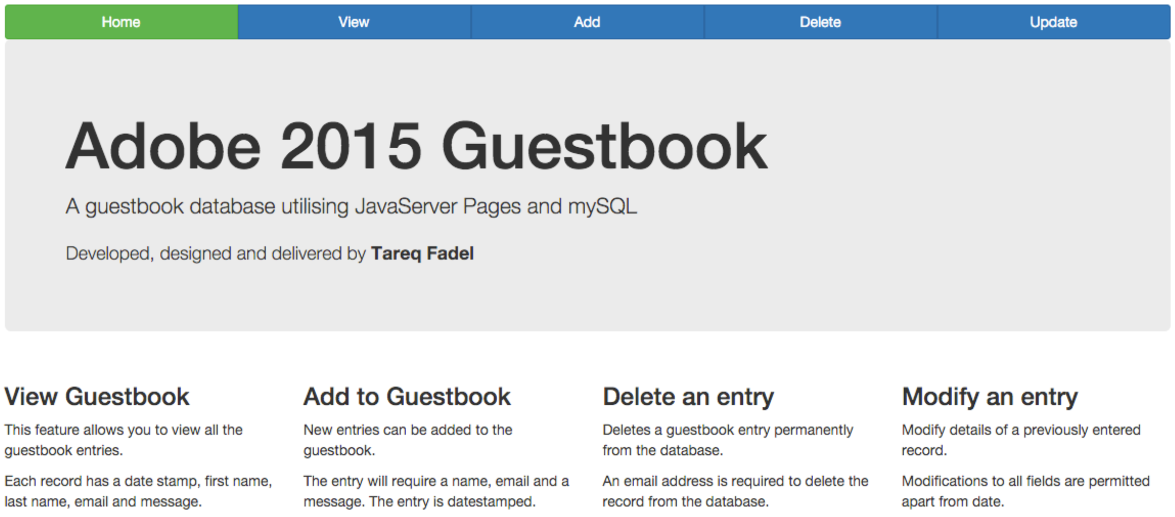
Guestbook Application

Built with JavaServer Pages and MySQL



Adobe 2015 Guestbook

A guestbook database utilising JavaServer Pages and mySQL

Developed, designed and delivered by **Tareq Fadel**

View Guestbook

This feature allows you to view all the guestbook entries.

Each record has a date stamp, first name, last name, email and message.

Add to Guestbook

New entries can be added to the guestbook.

The entry will require a name, email and a message. The entry is datestamped.

Delete an entry

Deletes a guestbook entry permanently from the database.

An email address is required to delete the record from the database.

Modify an entry

Modify details of a previously entered record.

Modifications to all fields are permitted apart from date.

1: Installation

Download and Install the following software

- Java SE Development Kit version 8u51
- NetBeans IDE 8.0.2 Full
- MySQL Community Server 5.6.25
- MySQL Connector/J 5.1.36

Ensure that JAVA & MySQL is set in Environment PATH

2: Database Setup

Login to MySQL database with username root and no password (default) and create Database Guestbook

```
mysql --user=root --password=
  show databases;
  CREATE DATABASE Guestbook;
  USE Guestbook;
```

3: Create Table

Create the table that the guestbook will use.

```
CREATE TABLE Guestbook (
Id INT NOT NULL AUTO_INCREMENT,
LastName VARCHAR(70) NOT NULL,
FirstName VARCHAR(70) NOT NULL,
Email VARCHAR(70) NOT NULL,
Message VARCHAR(140) NOT NULL,
Date DATE NOT NULL, PRIMARY KEY (Id) );
```

4: Add Dummy Data

Using SQL add dummy data to the database.

INSERT INTO Guestbook (LastName,FirstName,Email,Message)

VALUES ('Fadel','Tareq','me@tareq.uk','message', '2015-07-31');

INSERT INTO Guestbook (LastName,FirstName,Email,Message)

VALUES ('Adobe','James','james@adobe.com','another', '2015-07-31');

INSERT INTO Guestbook (LastName,FirstName,Email,Message)

VALUES ('Galaxy','Box','box@adobe.com','world', '2015-08-01');

5: Define a New Project

In Netbeans, define a new project of type **Java Web**, **Web Application**.

Give it a name of **Guestbook** and store it in the default location.

Use **Java EE 7 Web** as the Java EE version.

6: Add Java Libs

Before we start we need to add the downloaded MySQL Connector to the build. In Project properties… > Libraries > Compile > Add Jar…

then choose the mysql-connector-java.jar file previously downloaded.

7: Create Connection to DB

Ensure the MySQL database is running, in system preferences, then in NetBeans go to the **Services** tab and choose the **Guestbook** database from the MySQL list and right click and choose **Connect..** Choose default parameters.

This will produce a content uri of type jdbc:mysql://localhost….

We will use this later

8: File Creation

The project had created a index.html, create 4 other files view.jsp, add.jsp delete.jsp and update.jsp

9: Index.html – Home Page

In the index.html create links to the 4 other jsp pages as follows.

```
<a href="index.html">Home</a>
<a href="view.jsp">View</a>
<a href="add.jsp">Add</a>
<a href="delete.jsp">Delete</a>
<a href="update.jsp">Update</a>
```

10: Imports & Namespaces

For each of the JSP pages ensure you add the imports and namespaces.

```
<!--Imports-->
<%@page import="java.util.*,java.io.*,java.sql.*" %>
<%@page import="javax.servlet.http.*,javax.servlet.*" %>

<!--Namespace-->
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
```

11: setDataSource

For each of the JSP pages we will be connecting to the database add the following command to establish a connection.

```
<!--Connect to Database-->
<sql:setDataSource var="snapshot" user="root" password=""
driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/Guestbook?zeroDateTimeBehavior=conve
rtToNull" />
```

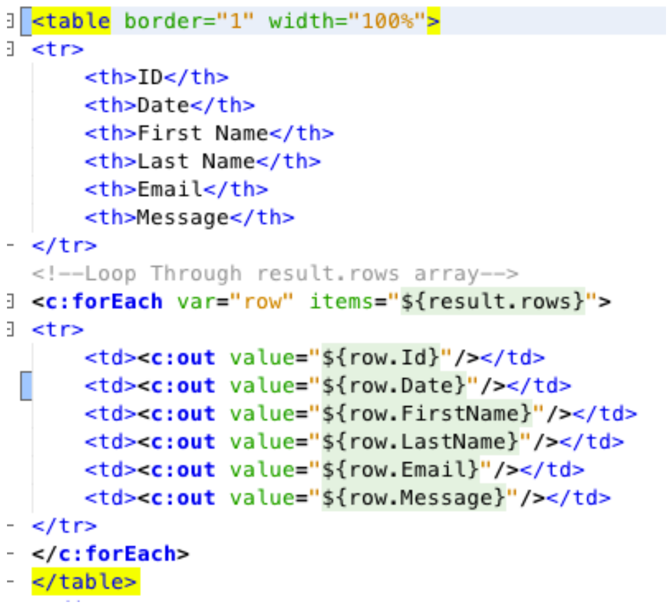
12: View Page

Query all fields and records in the database to view

<sql:query dataSource="${snapshot}" var="result"**>**
    SELECT * **FROM** Guestbook;
</sql:query>

13: View - Display Table

Add the HTML table to display the data by using the JSP forEach command to loop through the data.



```
<table border="1" width="100%">
<tr>
        <th>ID</th>
        <th>Date</th>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Email</th>
        <th>Message</th>
</tr>
<!--Loop Through result.rows array-->
<c:forEach var="row" items="${result.rows}">
<tr>
        <td><c:out value="${row.Id}"/></td>
        <td><c:out value="${row.Date}"/></td>
        <td><c:out value="${row.FirstName}"/></td>
        <td><c:out value="${row.LastName}"/></td>
        <td><c:out value="${row.Email}"/></td>
        <td><c:out value="${row.Message}"/></td>
</tr>
</c:forEach>
</table>
```

14: Add.jsp – Define Declaration

<%! String date = "";   *// date display for DB*
   java.util.Date dNow; *// for Now date*
%>

Add the following declarations for the add to work with the date manipulation. As the date will be pulled dynamically from the server and added to the database on record creation and for display on the HTML page.

15: add.jsp – Format the date for DB

```
<% dNow = new java.util.Date();
   SimpleDateFormat dBInput = new SimpleDateFormat ("yyyy-MM-dd");
   date = dBInput.format(dNow);

%>
```

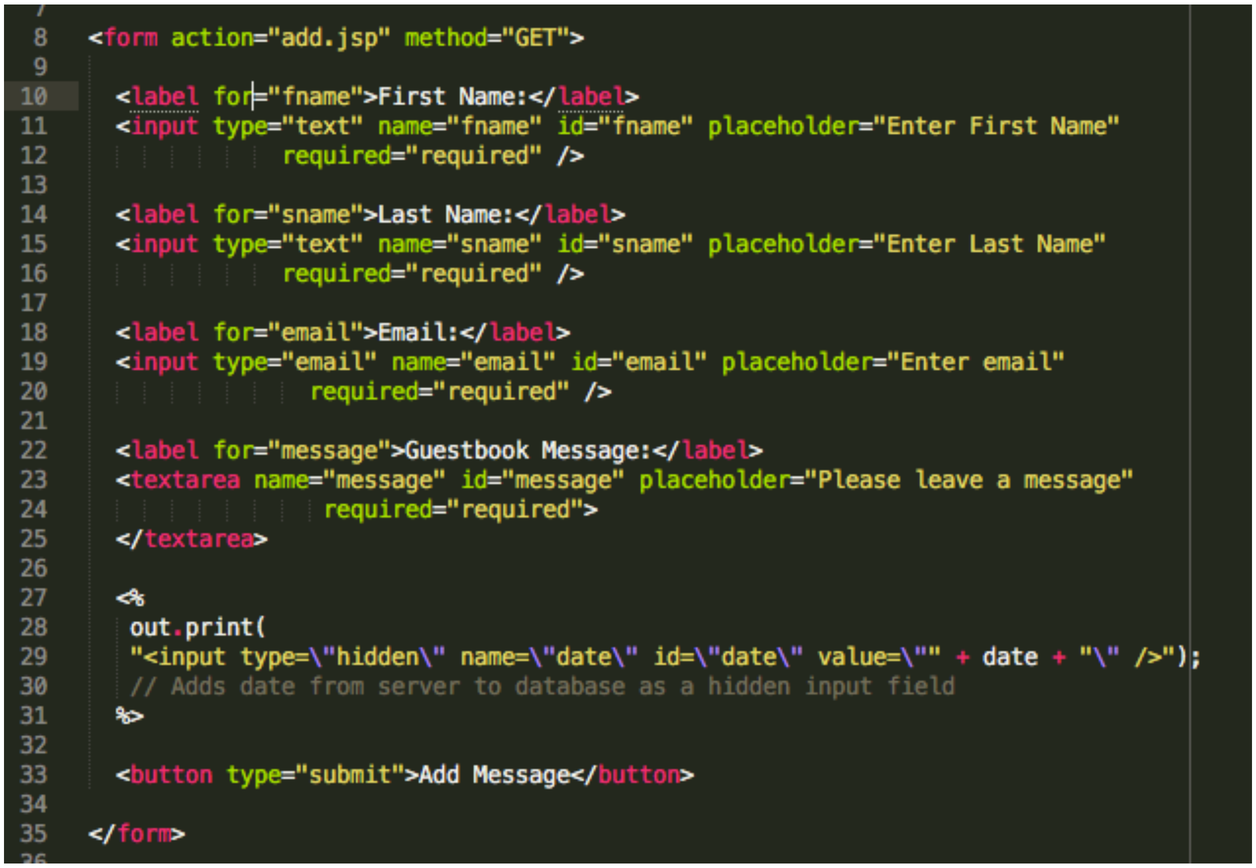
The Java code above extracts the date from the server and formats in MySQL date format ready for input.

16: add.jsp – Format the date for HTML

```
<p>A new record entry to the guestbook - <b>
<%
  SimpleDateFormat dateHtml = new SimpleDateFormat ("d MMMMM,
yyyy");
  String dateHtmlStr = dateHtml.format(dNow); out.print(dateHtmlStr);

%>
</b></p>
```

The above code formats the date to a readable format for display.

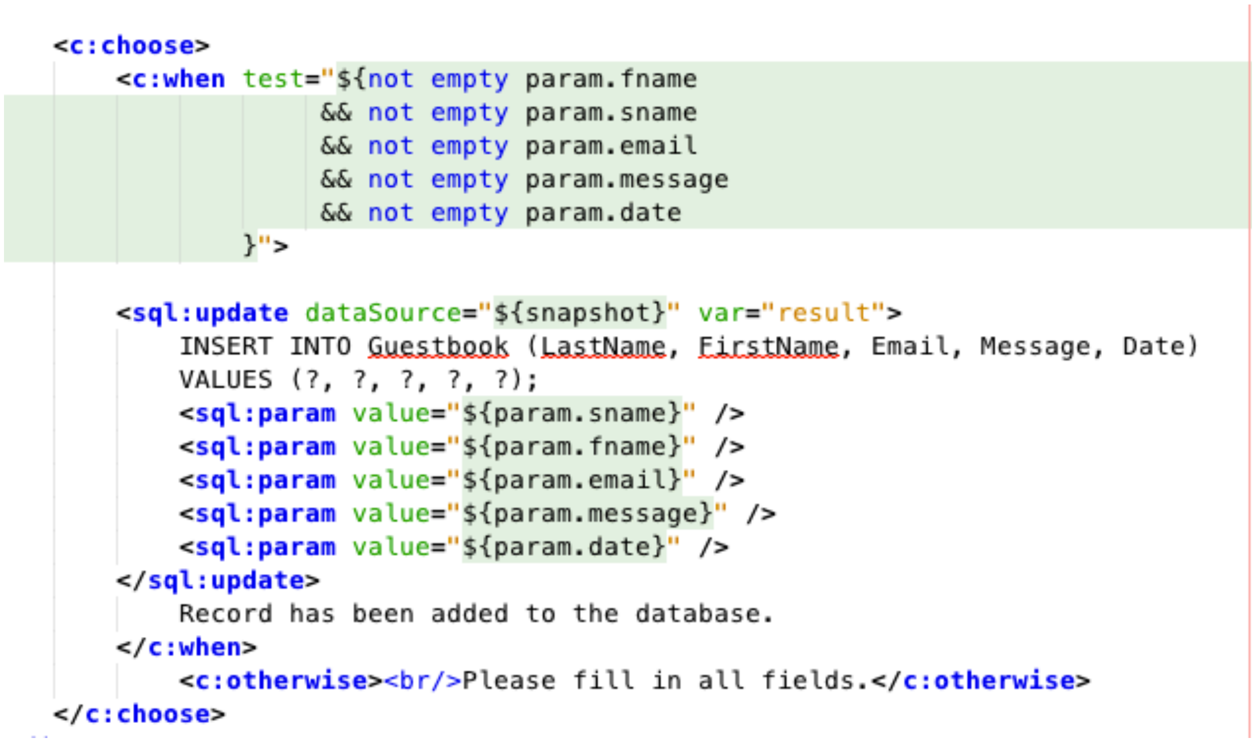
17: add.jsp – Input Form for Add Data



```
<form action="add.jsp" method="GET">

  <label for="fname">First Name:</label>
  <input type="text" name="fname" id="fname" placeholder="Enter First Name"
              required="required" />

  <label for="sname">Last Name:</label>
  <input type="text" name="sname" id="sname" placeholder="Enter Last Name"
              required="required" />

  <label for="email">Email:</label>
  <input type="email" name="email" id="email" placeholder="Enter email"
                required="required" />

  <label for="message">Guestbook Message:</label>
  <textarea name="message" id="message" placeholder="Please leave a message"
                 required="required">
  </textarea>

  <%
   out.print(
   "<input type=\"hidden\" name=\"date\" id=\"date\" value=\"" + date + "\" />");
   // Adds date from server to database as a hidden input field
  %>

  <button type="submit">Add Message</button>

</form>
```

Use the above code to provide the input form fields allowing the user to add the data on the web page. The Java code at the end is used for adding the current date dynamically for the record upon creation.

18: add.js Populate the DB with the data

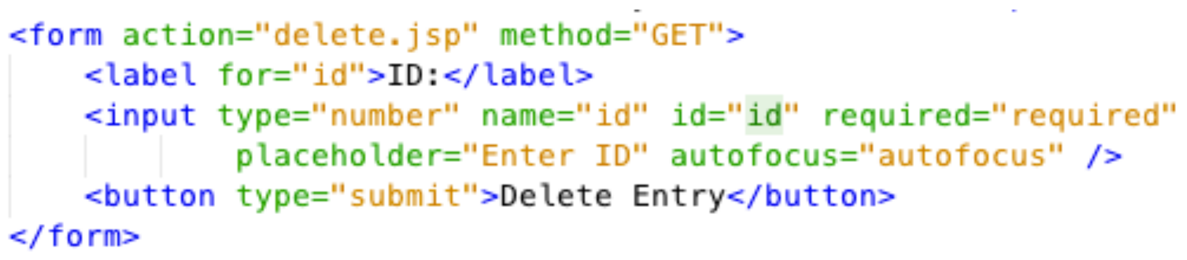


```
<c:choose>
    <c:when test="${not empty param.fname
                && not empty param.sname
                && not empty param.email
                && not empty param.message
                && not empty param.date
            }">

    <sql:update dataSource="${snapshot}" var="result">
        INSERT INTO Guestbook (LastName, FirstName, Email, Message, Date)
        VALUES (?, ?, ?, ?, ?);
        <sql:param value="${param.sname}" />
        <sql:param value="${param.fname}" />
        <sql:param value="${param.email}" />
        <sql:param value="${param.message}" />
        <sql:param value="${param.date}" />
    </sql:update>
        Record has been added to the database.
    </c:when>
        <c:otherwise><br/>Please fill in all fields.</c:otherwise>
</c:choose>
```

Using JSP's choose function, ensuring that none of the fields are empty then we can proceed with the SQL update by using the param values from the form.

Show a Record has been added to the database upon completion.

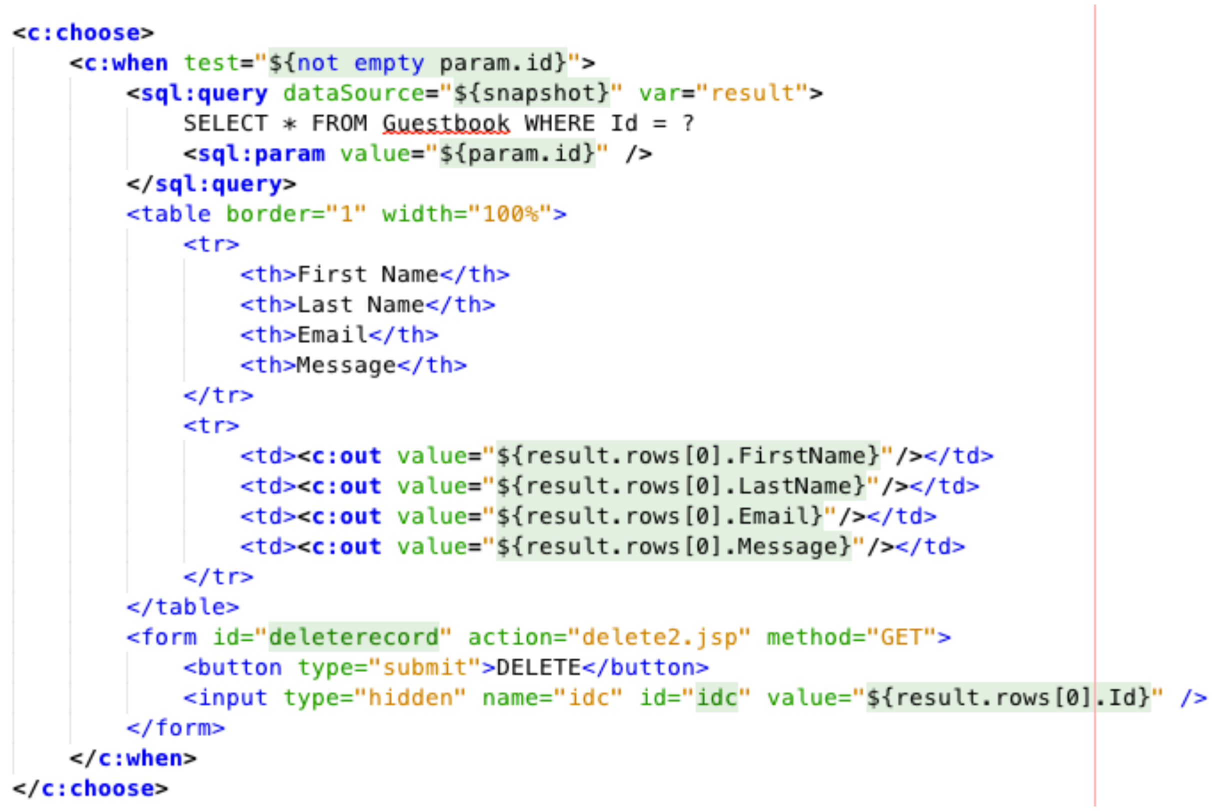
19: delete.jsp Delete a Record



```
<form action="delete.jsp" method="GET">
    <label for="id">ID:</label>
    <input type="number" name="id" id="id" required="required"
            placeholder="Enter ID" autofocus="autofocus" />
    <button type="submit">Delete Entry</button>
</form>
```

Add a form that will allow the user to enter the record ID of the record they wish to delete.

20: delete.jsp collect record.ID from the form

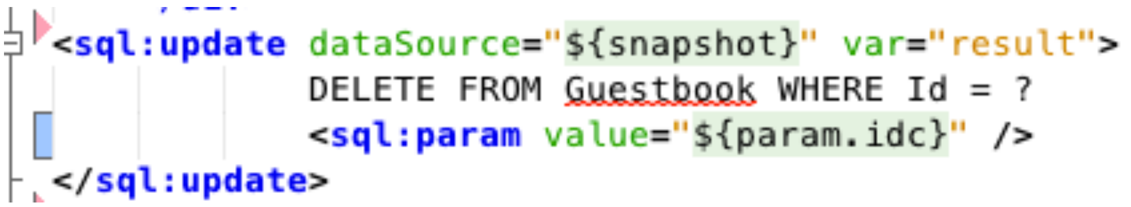


```
<c:choose>
    <c:when test="${not empty param.id}">
        <sql:query dataSource="${snapshot}" var="result">
            SELECT * FROM Guestbook WHERE Id = ?
            <sql:param value="${param.id}" />
        </sql:query>
        <table border="1" width="100%">
            <tr>
                <th>First Name</th>
                <th>Last Name</th>
                <th>Email</th>
                <th>Message</th>
            </tr>
            <tr>
                <td><c:out value="${result.rows[0].FirstName}"/></td>
                <td><c:out value="${result.rows[0].LastName}"/></td>
                <td><c:out value="${result.rows[0].Email}"/></td>
                <td><c:out value="${result.rows[0].Message}"/></td>
            </tr>
        </table>
        <form id="deleterecord" action="delete2.jsp" method="GET">
            <button type="submit">DELETE</button>
            <input type="hidden" name="idc" id="idc" value="${result.rows[0].Id}" />
        </form>
    </c:when>
</c:choose>
```

Using the previous form get the id pass this through to the database using the sql:query to select all records from the database that match the given ID. If found, then display the record in a HTML table to the user to confirm deletion of record.

21: delete2.jsp DELETE using SQL

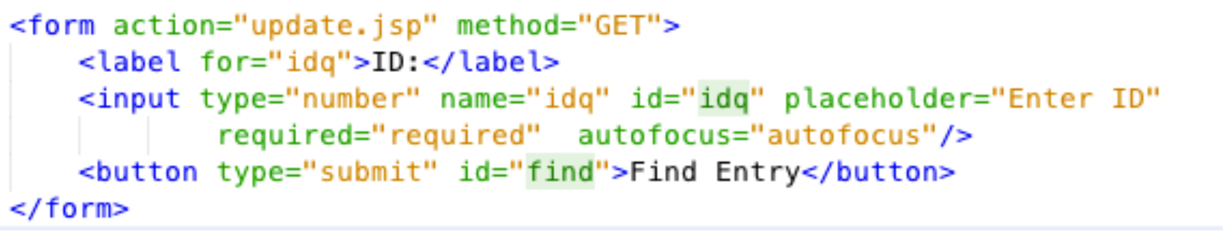
To handle the deletion of the record from the database, the previous form for deletion confirmation has an action of delete2.jsp. Create a delete2.jsp ensuring imports, namespaces and datasource is set then add the following code for the SQL



```
<sql:update dataSource="${snapshot}" var="result">
            DELETE FROM Guestbook WHERE Id = ?
            <sql:param value="${param.idc}" />
</sql:update>
```

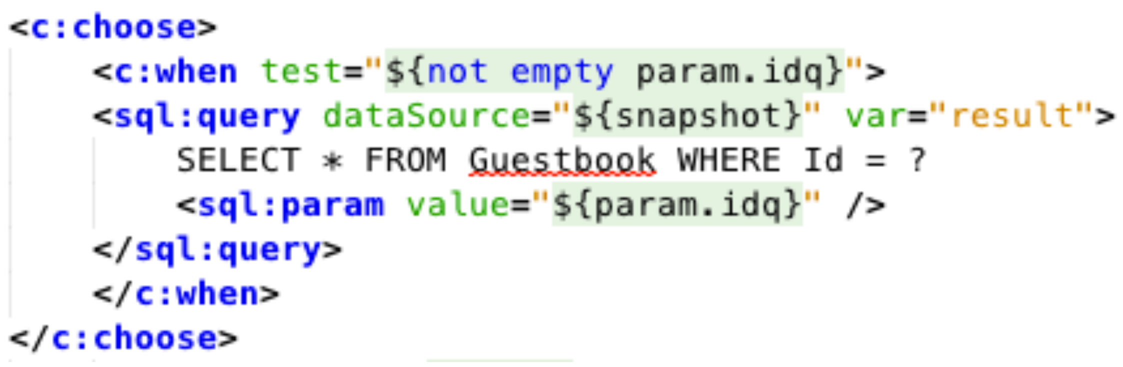
22: Update.jsp get ID Form

In this page, we need a form to ask the user for the ID of the record they wish to update



```
<form action="update.jsp" method="GET">
    <label for="idq">ID:</label>
    <input type="number" name="idq" id="idq" placeholder="Enter ID"
            required="required"  autofocus="autofocus"/>
    <button type="submit" id="find">Find Entry</button>
</form>
```

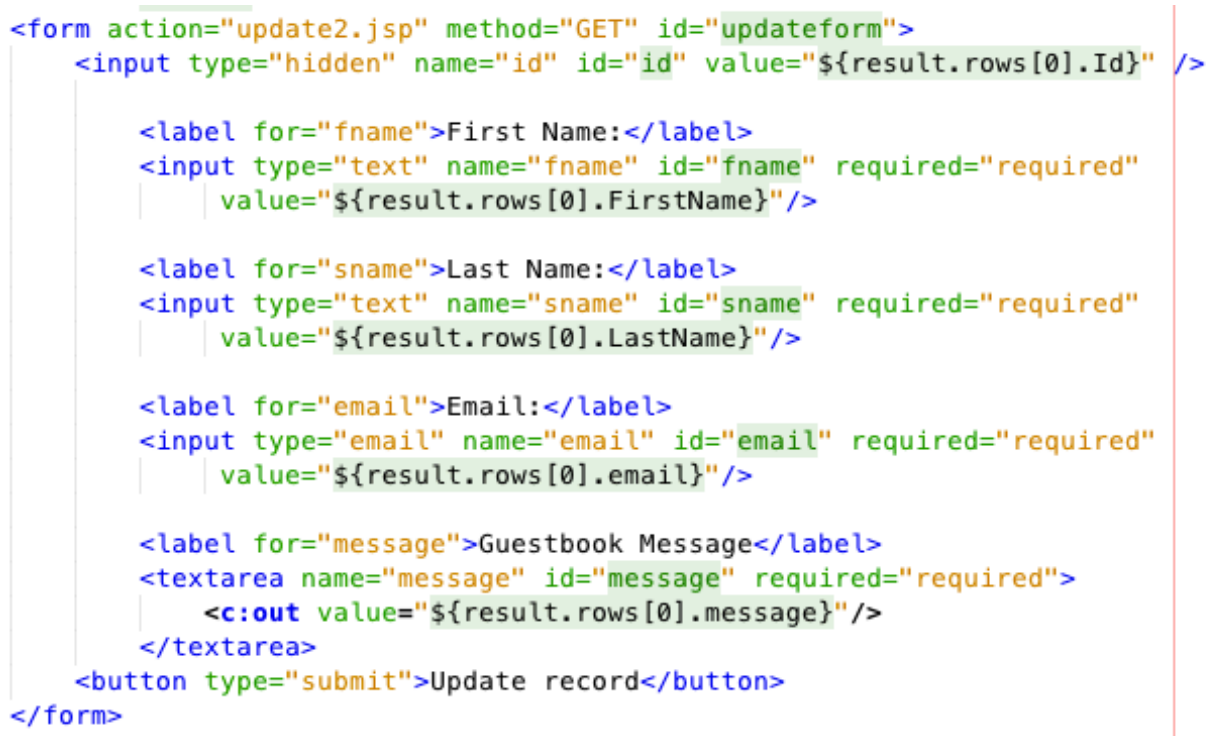
23: Update.jsp – Send ID to DB for Query



```
<c:choose>
    <c:when test="${not empty param.idq}">
    <sql:query dataSource="${snapshot}" var="result">
        SELECT * FROM Guestbook WHERE Id = ?
        <sql:param value="${param.idq}" />
    </sql:query>
    </c:when>
</c:choose>
```

Using the sql:query, the collected ID is passed back to the Guestbook for collection. The previous form will reload itself and pass back the parameters through the url.

24: update.jsp – Provide form for user to modify record

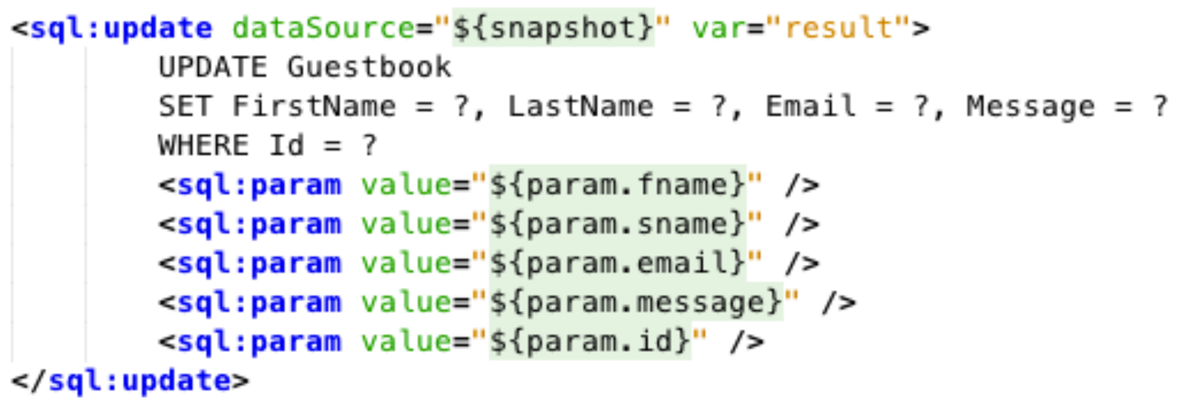


```
<form action="update2.jsp" method="GET" id="updateform">
    <input type="hidden" name="id" id="id" value="${result.rows[0].Id}" />

        <label for="fname">First Name:</label>
        <input type="text" name="fname" id="fname" required="required"
            value="${result.rows[0].FirstName}"/>

        <label for="sname">Last Name:</label>
        <input type="text" name="sname" id="sname" required="required"
            value="${result.rows[0].LastName}"/>

        <label for="email">Email:</label>
        <input type="email" name="email" id="email" required="required"
            value="${result.rows[0].email}"/>

        <label for="message">Guestbook Message</label>
        <textarea name="message" id="message" required="required">
            <c:out value="${result.rows[0].message}"/>
        </textarea>
    <button type="submit">Update record</button>
</form>
```

A form is added to update.jsp to dynamically populate the values of the form from the database by using the c:out function of JSP and displaying the first record in the results.

When the data has been modified the data will be passed back to the database for processing. This is handled in update2.jsp

25: Update2.jsp – Update DB with new data from User



```
<sql:update dataSource="${snapshot}" var="result">
        UPDATE Guestbook
        SET FirstName = ?, LastName = ?, Email = ?, Message = ?
        WHERE Id = ?
        <sql:param value="${param.fname}" />
        <sql:param value="${param.sname}" />
        <sql:param value="${param.email}" />
        <sql:param value="${param.message}" />
        <sql:param value="${param.id}" />
</sql:update>
```

As before, ensure, namespaces, imports and datasource are set in this file then include a sql:update to collect the parameters from the previous and update all records based on the ID