Java Technologies for Web Applications

Lab Guides

Document Code	**25e-BM/HR/HDCV/FSOFT**
Version	**1.1**
Effective Date	**20/11/2012**

Hanoi, 04/2019

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1	25/Jun/2018	Create a new Lab	Create new	DieuNT1	VinhNV
2	01/May/2019	Update Fsoft Template	Update	DieuNT1	VinhNV

Contents

CODE:	NWEB.M.L101
TYPE:	Medium
LOC:	
DURATION:	60 MINUTES

Unit 1 - JSP/Servlet Introduction

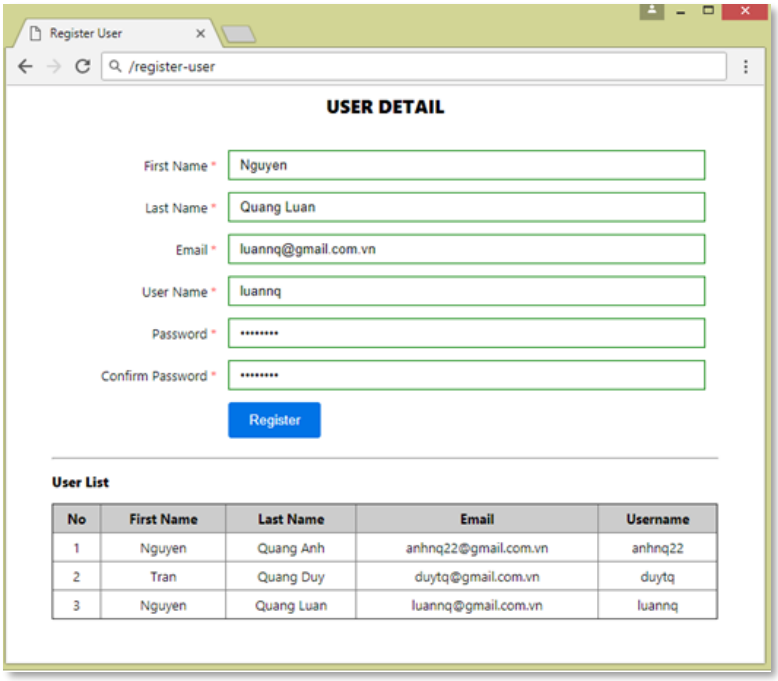
Objectives:

- ✓ Understand the basic concepts of web development technologies with java (JSP / Servlet)
- ✓ Able to write servlets using the Java programming language (Java servlets)
- ✓ Create dynamic HTML content with Servlets and JavaServer Pages, using the Expression Language, and the JSP Standard Tag Library (JSTL)
- ✓ Create robust web applications using MVC architecture, session management, filters, and database integration (JDBC)
- ✓ Make Servlets and JSP work together cleanly
- ✓ Create secure web applications using the features of the Java EE web container

Problem Descriptions:

Học viên được yêu cầu tạo 1 .jsp trang đơn giản với HTML, CSS và JavaScript. Nội dung của trang được miêu tả như hình dưới đây:

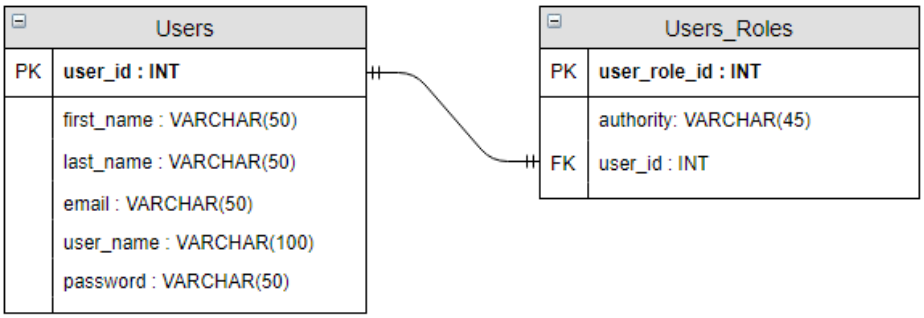
- ✓ *register-user.jsp*



Screen 01_Layout 01

Step 1: Tạo Database

Tạo DB có tên "**JNWEBML101_SMS**" có các bảng và quan hệ như sau:

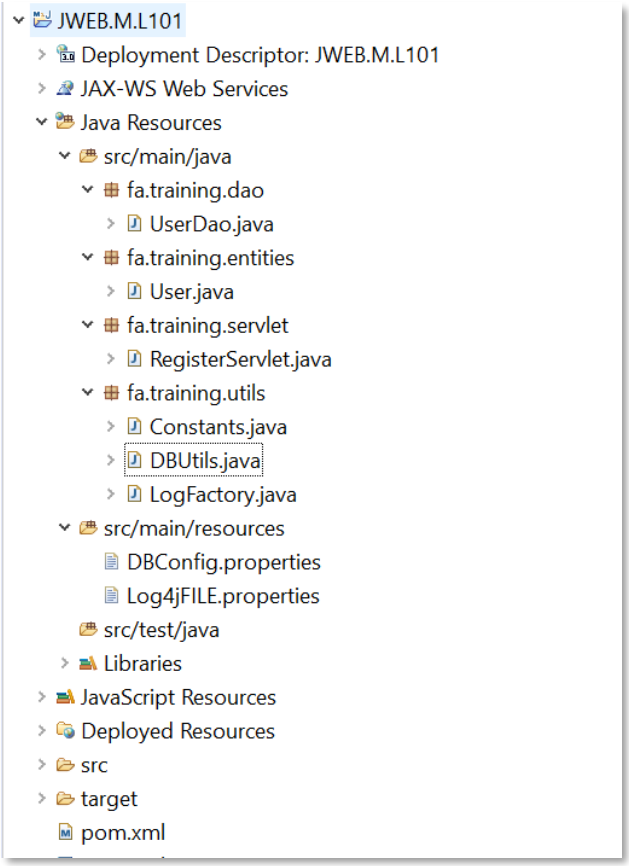


Tạo stored procedure "**usp_registerUser**" như sau:

```
CREATE PROC [dbo].[usp_registerUser]
@firstName VARCHAR(50),
@lastName VARCHAR(50),
@email VARCHAR(100),
@userName VARCHAR(50),
@password VARCHAR(50)
AS
BEGIN
    INSERT INTO Users VALUES (@firstName, @lastName, @email, @userName, @password)
END
```

Step 2: Tạo maven project

Tạo 1 maven project với tên **"JWEB_M_L101"** có cấu trúc thư mục như sau:



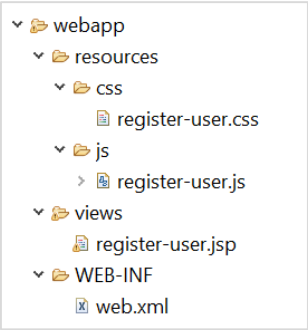
File **pom.xml**

```
<dependencies>
        <dependency>
                <groupId>javax.servlet</groupId>
                <artifactId>javax.servlet-api</artifactId>
                <version>3.1.0</version>
        </dependency>

        <dependency>
                <groupId>com.microsoft.sqlserver</groupId>
                <artifactId>mssql-jdbc</artifactId>
                <version>7.0.0.jre8</version>
        </dependency>

        <dependency>
                <groupId>log4j</groupId>
                <artifactId>log4j</artifactId>
                <version>1.2.17</version>
        </dependency>
</dependencies>
<build>
        <finalName>NWEB_M_L101</finalName>
        <plugins>
                <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-compiler-plugin</artifactId>
                        <version>3.2</version>
                        <configuration>
                                <source>1.8</source>
                                <target>1.8</target>
                        </configuration>
                </plugin>
        </plugins>
</build>
```

Step3: Validate

Để validate dữ liệu và thêm dữ liệu vừa nhập vào bảng User List bên dưới, ta sẽ sử dụng JavaScript hoặc jQuery.

Tạo file JS, **register-user.js**. Cấu trúc thư mục **webapp** như sau:

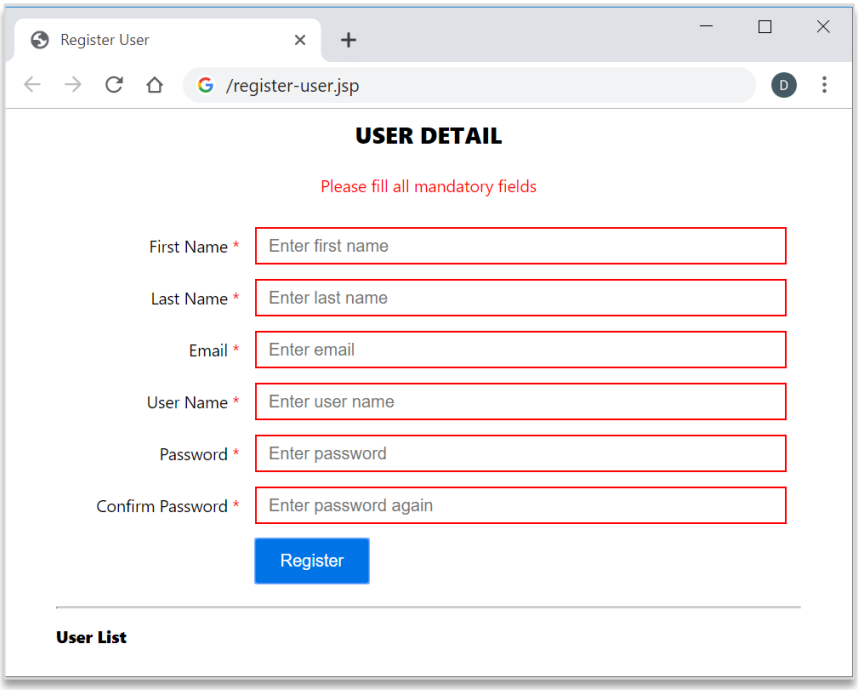



```
/*
    This function called when "Register" button clicked.
*/
function validateRegister() {
    // Get the value that user enters at the form
    var firstNameElement = document.getElementById("firstName");
    var lastNameElement = document.getElementById("lastName");
    var emailElement = document.getElementById("email");
    var userNameElement = document.getElementById("userName");
    var passwordElement = document.getElementById("password");
```

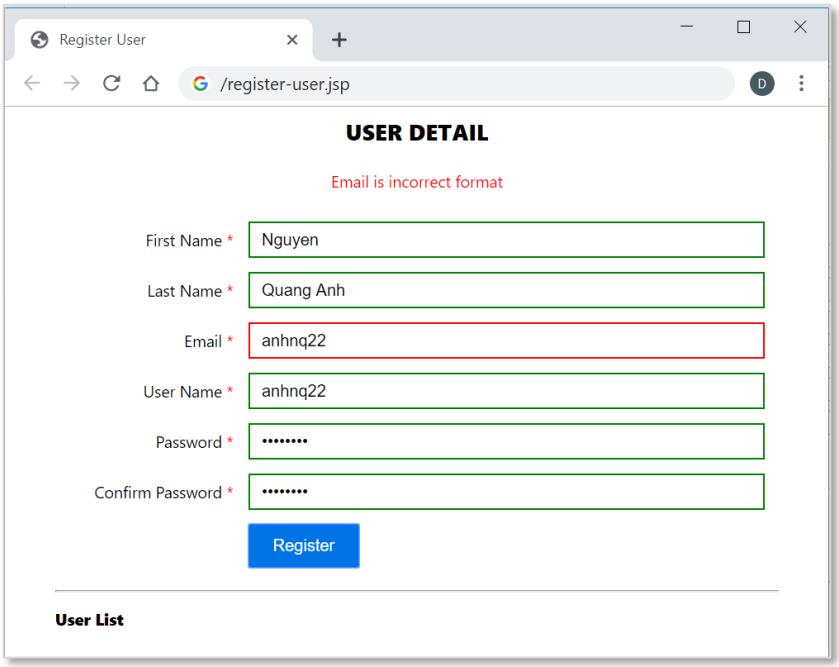
```
11.     var confirmPasswordElement =  document.getElementById("confirmPassword");
12.
13.     // variable to check valid input
14.     var status = false;
15.
16.     var message = "Please fill all mandatory fields";
17.
18.     setBorderColor(firstNameElement);
19.     setBorderColor(lastNameElement);
20.     setBorderColor(emailElement);
21.     setBorderColor(userNameElement);
22.     setBorderColor(passwordElement);
23.     setBorderColor(confirmPasswordElement);
24.
25.     var email = emailElement.value;
26.
27.     if (email != "" && !validateEmail(email)) {
28.         message = "Email is incorrect format";
29.         emailElement.style.borderColor = "red";
30.     } else if (passwordElement.value != confirmPasswordElement.value) {
31.         message = "Confirm password is not match with password";
32.         confirmPasswordElement.style.borderColor = "red";
33.     }
34.
35.     // count number of input tags
36.     var numberOfInput = document.getElementsByTagName("input").length;
37.     var countNumberValidInput = 0;
38.     for (var j = 0; j < numberOfInput; j++) {
39.         // check all input are valid
40.         if (document.getElementsByTagName("input")[j].
41.                                     style.borderColor == "green") {
42.             countNumberValidInput++;
43.         }
44.     }
45.     // if all input are valid, set ok = true
46.     if (countNumberValidInput == numberOfInput) {
47.         message = "";
48.         status = true;
49.     }
50.
51.     document.getElementById("error").innerHTML = message;
52.     // if statuc -> call method showUserRegisted()
53.     if(status) {
54.         showUserRegisted();
55.     }
56. }
57.
58. /*
59.  This function to create a header row for an existed table and
60.  append data to it.
```

```
61. */
62.
63. function showUserRegisted() {
64.     // get element tbody of table with id = tbl-result
65.     var table = document.getElementById("tbl-result").
66.                                 getElementsByTagName("tbody")[0];
67.     var index = table.rows.length;
68.     // if number rows of table == 0, insert th into thead of table
69.     if (table.rows.length == 0) {
70.         var thead = document.getElementById("tbl-result").
71.                                         getElementsByTagName("thead")[0];
72.         var row = thead.insertRow(0);
73.         row.insertCell(0).outerHTML = "<th>No</th>";
74.         row.insertCell(1).outerHTML = "<th>First Name</th>";
75.         row.insertCell(2).outerHTML = "<th>Last Name</th>";
76.         row.insertCell(3).outerHTML = "<th>Email</th>";
77.         row.insertCell(4).outerHTML = "<th>Username</th>";
78.     }
79.     // insert user registed rows into table result
80.     var row = table.insertRow(table.rows.length);
81.     row.insertCell(0).innerHTML = ++index;
82.     row.insertCell(1).innerHTML = document.getElementById("firstName").value;
83.     row.insertCell(2).innerHTML =
84.                         document.getElementsByClassName("lastName")[0].value;
85.     row.insertCell(3).innerHTML = document.getElementsByTagName("input")[2].value;
86.     row.insertCell(4).innerHTML = document.getElementById("userName").value;
87. }
88. /*
89.     Check valid email.
90. */
91. function validateEmail(email) {
92.   var re =
    /^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-
    9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-
    Z]{2,}))$/;
93.   return re.test(email);
94. }
95. /*
96.     Change border to an element.
97. */
98. function setBorderColor(element) {
99.     if (element.value == "") {
100.                element.style.borderColor = "red";
101.            } else {
102.                element.style.borderColor = "green";
103.            }
104.        }
```

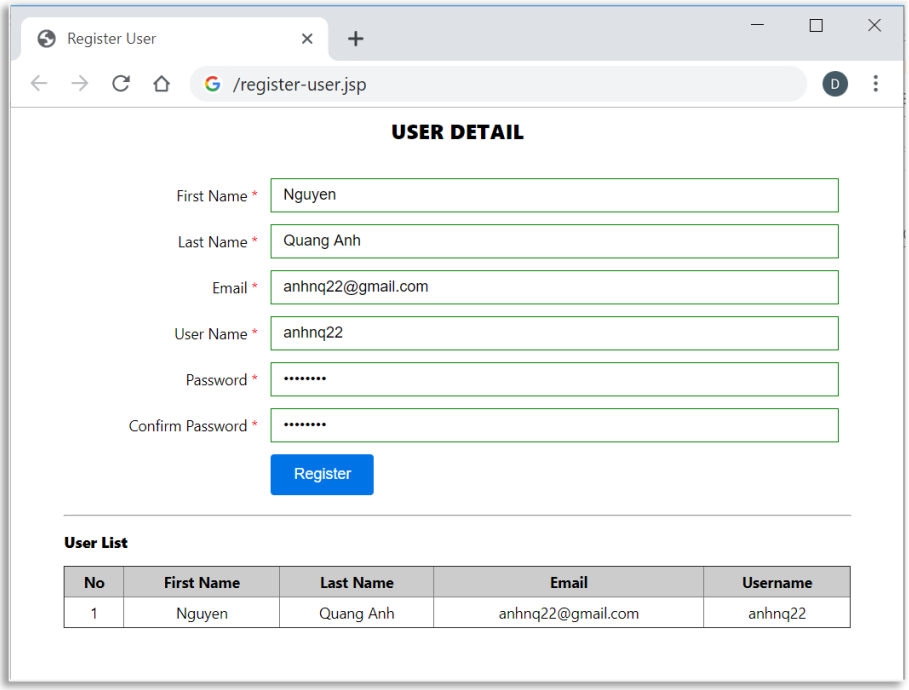
Layout màn hình khi validate:



Screen 01_Layout 02



Screen 01_Layout 03



Screen 01_Layout 04

Step4: Tạo servlet class và cấu hình

Tạo lớp **User** trong package **fa.training.entities** như sau:

```
User.java
9   public class User {
10    private String firstName;
11    private String lastName;
12    private String email;
13    private String userName;
14    private String password;
15
16⊕   public User(String firstName, String lastName, String email, String userName,
25
26⊕   public User(String userName, String password) {
31
32⊕   public User(String firstName, String lastName, String email,
40
41⊕   public User() {
44
45⊕   public String getFirstName() {
48
49⊕   public void setFirstName(String firstName) {
52
53⊕   public String getLastName() {
56
57⊕   public void setLastName(String lastName) {
60
61⊕   public String getEmail() {
64
65⊕   public void setEmail(String email) {
68
69⊕   public String getUserName() {
72
73⊕   public void setUserName(String userName) {
76
77⊕   public String getPassword() {
80
81⊕   public void setPassword(String password) {
```

✓ Tạo một Servlet có tên **RegisterServlet** trong package **fa.training.servlet** và override phương thức **doPost()** như sau:

```
@WebServlet(urlPatterns = "/register")
public class RegisterServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    @Override
    protected void doPost(HttpServletRequest request,
       HttpServletResponse response) throws ServletException, IOException {
        // Get data from the request using request.getParameter()
        String firstName = request.getParameter("firstName");
        String lastName = request.getParameter("lastName");
        String email = request.getParameter("email");
        String userName = request.getParameter("userName");
        String password = request.getParameter("password");

        // Set data for the user
        User user = new User(firstName, lastName, email, userName, password);
        try {
            UserDao userDao = new UserDao();
            // Call registerUser() method to insert user into DB
            if (userDao.registerUser(user)) {
                //    Send a attribute name as "userRegister"
                      to register-user-process.jsp page
                request.setAttribute("userRegister", user);
                // Forward to register-user-process.jsp page
                request.getRequestDispatcher("/views/login.jsp").
                                                    forward(request, response);
            } else {
                // send a attribute name as "message" to register-user.jsp page
                request.setAttribute("message", Constants.REGISTER_FAIL_MESSAGE);
                // forward to register-user.jsp page
                request.getRequestDispatcher("/views/register-user.jsp").
                                                    forward(request, response);
            }

        } catch (ClassNotFoundException | SQLException e) {
            // log error if exception occurs
            LogFactory.getLogger().
                            error("An exception occurs while register user");
        }
    }
}
```

✓ **Cấu hình servlet để mapping với request,** thay đổi **action** của form register thành:

```
<form action="<%=request.getContextPath()%>/register" method="post"
                        onsubmit="return validateRegister()" name="frm-register">
```

Có 2 cách cấu hình servlet để mapping với request tương ứng:

<u>Cách 1</u>: Cấu hình trong file **web.xml bằng cách thêm** đoạn code sau vào file **web.xml**

```
<servlet>
  <servlet-name>RegisterServlet</servlet-name>
  <servlet-class>com.fa.servlet.RegisterServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>RegisterServlet</servlet-name>
  <url-pattern>/register</url-pattern>
</servlet-mapping>
```

Cách 2: Sử dụng Annotation **@WebServlet**

Thêm Annotation **@WebServlet** trước class **RegisterServlet**

```
@WebServlet(urlPatterns = "/register", description = "This is RegisterServlet")
public class RegisterServlet extends HttpServlet {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

}
```

Step5: Xử lý DAO

Tạo method **registerUser(User user)** trong class **UserDao** để xử lý insert user vừa đăng ký vào bảng **Users** trong database.

```
86. package fa.training.dao;
87.
88. import java.io.IOException;
89. import java.sql.CallableStatement;
90. import java.sql.Connection;
91. import java.sql.ResultSet;
92. import java.sql.SQLException;
93.
94. import fa.training.entities.User;
95. import fa.training.utils.DBUtils;
96.
97. /**
98.  * The class contains methods to update and retrieve data from database
99.  *
100.        * @author FA
101.        *
102.        */
103.       public class UserDao {
104.
105.           /**
106.            * The method to insert a new user into database.
107.            *
108.            * @param user an user object.
109.            * @return true if register successfully.
110.            * @throws SQLException
111.            * @throws IOException
112.            * @throws ClassNotFoundException
113.            */
114.           public boolean registerUser(User user)
115.                     throws ClassNotFoundException, IOException, SQLException {
116.               Connection connection = null;
117.               try {
118.                   connection = DBUtils.getConnection();
119.                   CallableStatement callableStatement =
120.                      connection.prepareCall("{call usp_registerUser(?,?,?,?,?)}");
121.                   int param = 0;
122.                   callableStatement.setString(++param, user.getFirstName());
123.                   callableStatement.setString(++param, user.getLastName());
124.                   callableStatement.setString(++param, user.getEmail());
125.                   callableStatement.setString(++param, user.getUserName());
126.                   callableStatement.setString(++param, user.getPassword());
127.                   int result = callableStatement.executeUpdate();
128.
129.                   if (result > 0) {
130.                       return true;
```

```
                    }
                    return false;

                } finally {
                    DBUtils.closeConnection(connection);
                }

            }
        }
```

-- THE END --