

***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 |
|  | 25/Jun/2018 | Create a new Lab | Create new | DieuNT1 | VinhNV |
|  | 01/May/2019 | Update Fsoft Template | Update | DieuNT1 | VinhNV |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Contents

[Unit 1 - JSP/Servlet Introduction 4](#_Toc20499480)

[Objectives: 4](#_Toc20499481)

[Problem Descriptions: 4](#_Toc20499482)

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

# Unit 1 - JSP/Servlet Introduction

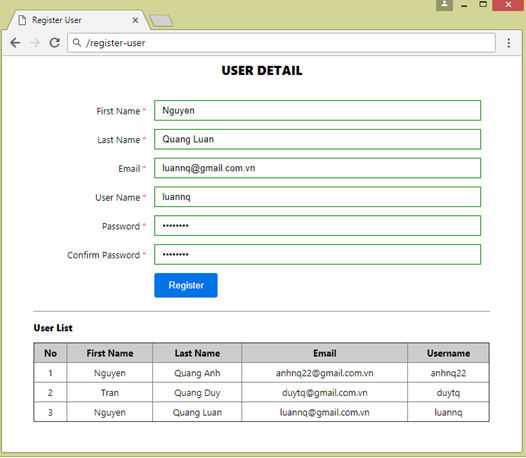
Objectives:

* 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:

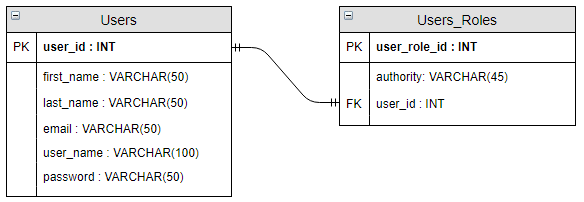
* *register-user.jsp*



*Screen 01\_Layout 01*

**Step 1: Tạo Database**

Tạo DB có tên “**JNWEBML101\_SMS**” có các bảng và quan hệ như sau:

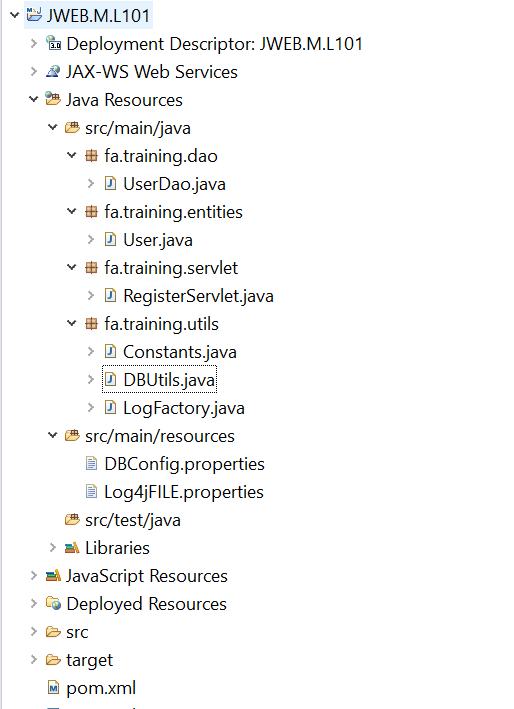


Tạo stored procedure “**usp\_registerUser**” như sau:

1. CREATE PROC [dbo].[usp\_registerUser]
2. @firstName VARCHAR(50),
3. @lastName VARCHAR(50),
4. @email VARCHAR(100),
5. @userName VARCHAR(50),
6. @password VARCHAR(50)
7. AS
8. BEGIN
9. INSERT INTO Users VALUES (@firstName, @lastName, @email, @userName, @password)
10. END

**Step 2: Tạo maven project**

Tạo 1 maven project với tên **“JWEB\_M\_L101”** có cấu trúc thư mục như sau:



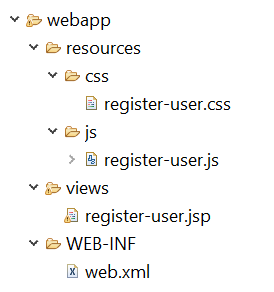
File **pom.xml**

1. <dependencies>
2. <dependency>
3. <groupId>javax.servlet</groupId>
4. <artifactId>javax.servlet-api</artifactId>
5. <version>3.1.0</version>
6. </dependency>
8. <dependency>
9. <groupId>com.microsoft.sqlserver</groupId>
10. <artifactId>mssql-jdbc</artifactId>
11. <version>7.0.0.jre8</version>
12. </dependency>
13. <dependency>
14. <groupId>log4j</groupId>
15. <artifactId>log4j</artifactId>
16. <version>1.2.17</version>
17. </dependency>
18. </dependencies>
19. <build>
20. <finalName>NWEB\_M\_L101</finalName>
21. <plugins>
22. <plugin>
23. <groupId>org.apache.maven.plugins</groupId>
24. <artifactId>maven-compiler-plugin</artifactId>
25. <version>3.2</version>
26. <configuration>
27. <source>1.8</source>
28. <target>1.8</target>
29. </configuration>
30. </plugin>
31. </plugins>
32. </build>

**Step3: Validate**

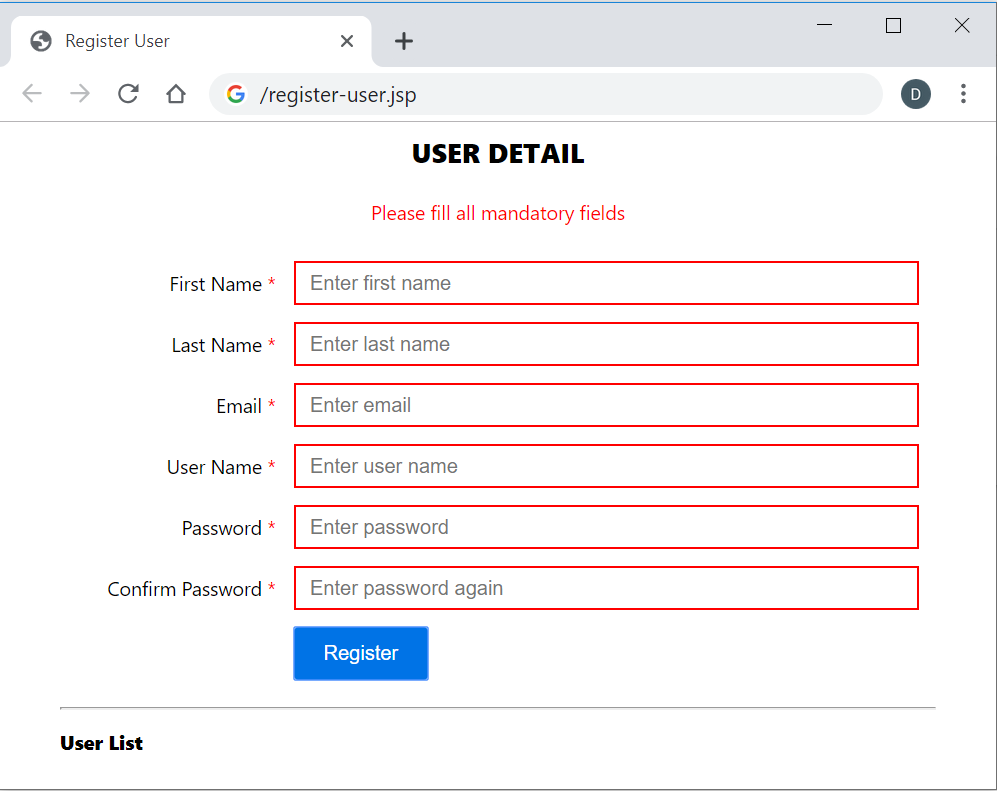
Để validate dữ liệu và thêm dữ liệu vừa nhập vào bảng User List bên dưới, ta sẽ sử dụng JavaScript hoặc jQuery.

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

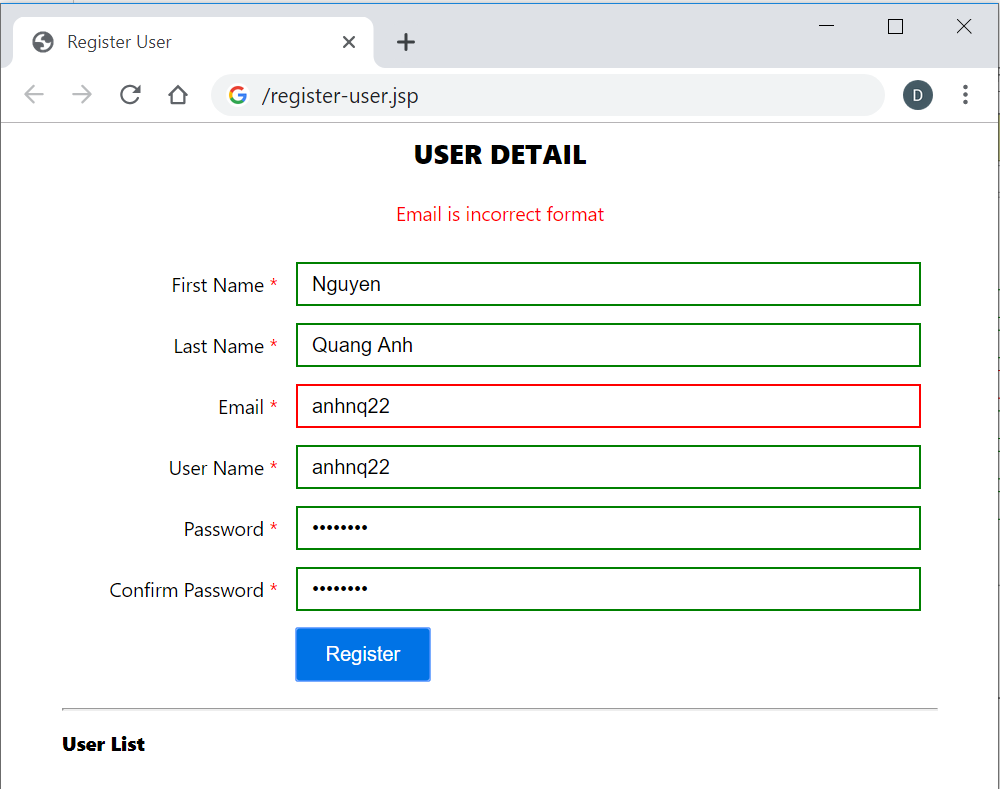


1. /\*
2. This function called when "Register" button clicked.
3. \*/
4. function validateRegister() {
5. // Get the value that user enters at the form
6. var firstNameElement = document.getElementById("firstName");
7. var lastNameElement = document.getElementById("lastName");
8. var emailElement = document.getElementById("email");
9. var userNameElement = document.getElementById("userName");
10. var passwordElement = document.getElementById("password");
11. var confirmPasswordElement = document.getElementById("confirmPassword");
13. // variable to check valid input
14. var status = false;
16. var message = "Please fill all mandatory fields";
18. setBorderColor(firstNameElement);
19. setBorderColor(lastNameElement);
20. setBorderColor(emailElement);
21. setBorderColor(userNameElement);
22. setBorderColor(passwordElement);
23. setBorderColor(confirmPasswordElement);
25. var email = emailElement.value;
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. }
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. }
51. document.getElementById("error").innerHTML = message;
52. // if statuc -> call method showUserRegisted()
53. if(status) {
54. showUserRegisted();
55. }
56. }
57. /\*
58. This function to create a header row for an existed table and
59. append data to it.
60. \*/
61. function showUserRegisted() {
62. // get element tbody of table with id = tbl-result
63. var table = document.getElementById("tbl-result").
64. getElementsByTagName("tbody")[0];
65. var index = table.rows.length;
66. // if number rows of table == 0, insert th into thead of table
67. if (table.rows.length == 0) {
68. var thead = document.getElementById("tbl-result").
69. getElementsByTagName("thead")[0];
70. var row = thead.insertRow(0);
71. row.insertCell(0).outerHTML = "<th>No</th>";
72. row.insertCell(1).outerHTML = "<th>First Name</th>";
73. row.insertCell(2).outerHTML = "<th>Last Name</th>";
74. row.insertCell(3).outerHTML = "<th>Email</th>";
75. row.insertCell(4).outerHTML = "<th>Username</th>";
76. }
77. // insert user registed rows into table result
78. var row = table.insertRow(table.rows.length);
79. row.insertCell(0).innerHTML = ++index;
80. row.insertCell(1).innerHTML = document.getElementById("firstName").value;
81. row.insertCell(2).innerHTML =
82. document.getElementsByClassName("lastName")[0].value;
83. row.insertCell(3).innerHTML = document.getElementsByTagName("input")[2].value;
84. row.insertCell(4).innerHTML = document.getElementById("userName").value;
85. }
86. /\*
87. Check valid email.
88. \*/
89. function validateEmail(email) {
90. 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,}))$/;
91. return re.test(email);
92. }
93. /\*
94. Change border to an element.
95. \*/
96. function setBorderColor(element) {
97. if (element.value == "") {
98. element.style.borderColor = "red";
99. } else {
100. element.style.borderColor = "green";
101. }
102. }

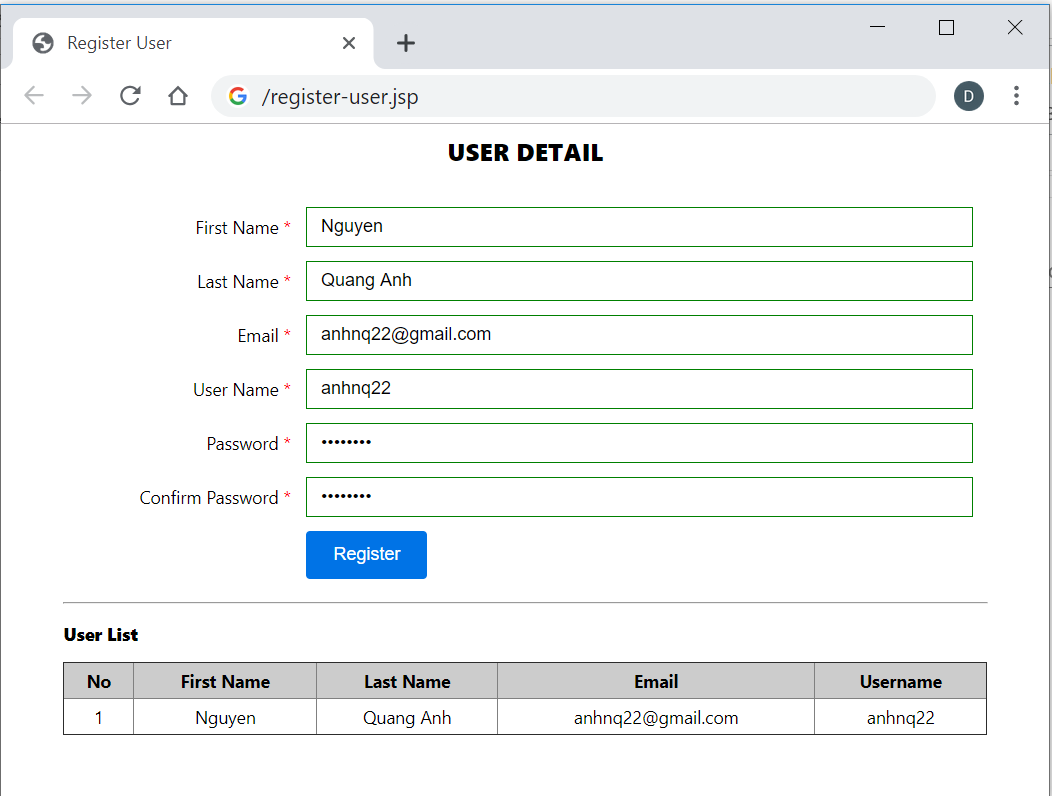
***Layout màn hình khi validate:***



*Screen 01\_Layout 02*



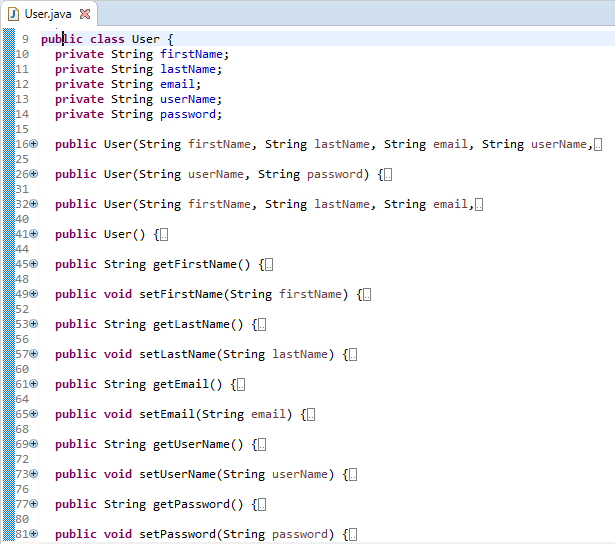
*Screen 01\_Layout 03*



*Screen 01\_Layout 04*

**Step4: Tạo servlet class và cấu hình**

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



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

1. @WebServlet(urlPatterns = "/register")
2. **public** **class** RegisterServlet **extends** HttpServlet {
3. **private** **static** **final** **long** ***serialVersionUID*** = 1L;
4. @Override
5. **protected** **void** doPost(HttpServletRequest request,
6. HttpServletResponse response) **throws** ServletException, IOException {
7. // Get data from the request using request.getParameter()
8. String firstName = request.getParameter("firstName");
9. String lastName = request.getParameter("lastName");
10. String email = request.getParameter("email");
11. String userName = request.getParameter("userName");
12. String password = request.getParameter("password");
14. // Set data for the user
15. User user = **new** User(firstName, lastName, email, userName, password);
16. **try** {
17. UserDao userDao = **new** UserDao();
18. // Call registerUser() method to insert user into DB
19. **if** (userDao.registerUser(user)) {
20. // Send a attribute name as "userRegister"
21. to register-user-process.jsp page
22. request.setAttribute("userRegister", user);
23. // Forward to register-user-process.jsp page
24. request.getRequestDispatcher("/views/login.jsp").
25. forward(request, response);
26. } **else** {
27. // send a attribute name as "message" to register-user.jsp page
28. request.setAttribute("message", Constants.***REGISTER\_FAIL\_MESSAGE***);
29. // forward to register-user.jsp page
30. request.getRequestDispatcher("/views/register-user.jsp").
31. forward(request, response);
32. }
33. } **catch** (ClassNotFoundException | SQLException e) {
34. // log error if exception occurs
35. LogFactory.*getLogger*().
36. error("An exception occurs while register user");
37. }
38. }
39. }

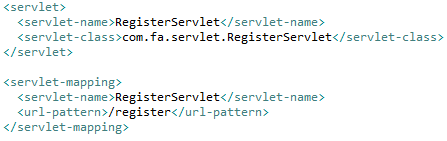
* **Cấu hình servlet để mapping với request, t**hay đổi **action** của form register thành:

<form action=*"*<%=request.getContextPath()%>*/register"* method=*"post"*

onsubmit="return validateRegister()" name=*"frm-register"*>

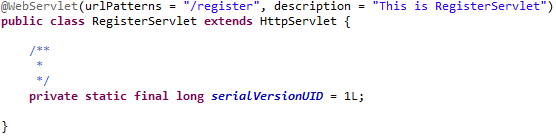
Có 2 cách cấu hình servlet để mapping với request tương ứng:

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



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

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



**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.

1. package fa.training.dao;
2. import java.io.IOException;
3. import java.sql.CallableStatement;
4. import java.sql.Connection;
5. import java.sql.ResultSet;
6. import java.sql.SQLException;
7. import fa.training.entities.User;
8. import fa.training.utils.DBUtils;
9. /\*\*
10. \* The class contains methods to update and retrieve data from database
11. \*
12. \* @author FA
13. \*
14. \*/
15. public class **UserDao** {
16. /\*\*
17. \* The method to insert a new user into database.
18. \*
19. \* @param user an user object.
20. \* @return true if register successfully.
21. \* @throws SQLException
22. \* @throws IOException
23. \* @throws ClassNotFoundException
24. \*/
25. public boolean **registerUser**(User user)
26. throws ClassNotFoundException, IOException, SQLException {
27. Connection connection = null;
28. try {
29. connection = DBUtils.getConnection();
30. CallableStatement callableStatement =
31. connection.prepareCall("{call **usp\_registerUser**(?,?,?,?,?)}");
32. int param = 0;
33. callableStatement.setString(++param, user.getFirstName());
34. callableStatement.setString(++param, user.getLastName());
35. callableStatement.setString(++param, user.getEmail());
36. callableStatement.setString(++param, user.getUserName());
37. callableStatement.setString(++param, user.getPassword());
38. int result = callableStatement.executeUpdate();
39. if (result > 0) {
40. return true;
41. }
42. return false;
43. } finally {
44. DBUtils.closeConnection(connection);
45. }
46. }
47. }

**-- THE END --**