HCMUTE

LẬP TRÌNH WEB

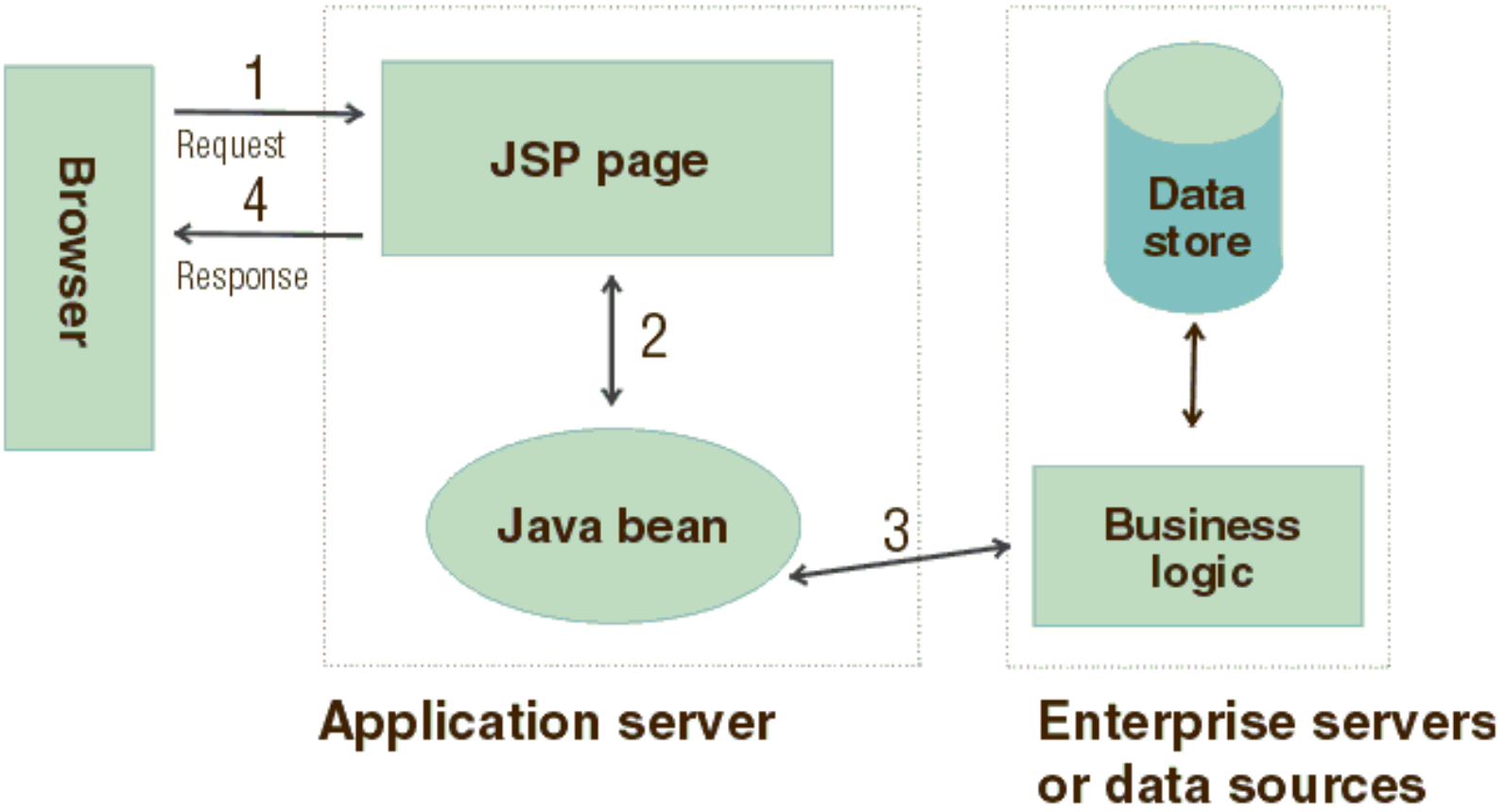
Lập trình JAVA JSP

Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM

JSP là gì?

- **JSP(JavaServer Pages)** là một công nghệ để phát triển các trang web động.
- **JSP** là một kiểu Java Servlet được thiết kế để tạo ra giao diện người dùng cho một ứng dụng Java web.
- Sử dụng **JSP** thu thập dữ liệu đầu vào từ người dùng thông qua các Form của trang web, trình bày các bản ghi từ một cơ sở dữ liệu hoặc một nguồn khác.
- Các thẻ **JSP** có thể được sử dụng cho nhiều mục đích khác nhau, chẳng hạn như truy xuất thông tin từ cơ sở dữ liệu hoặc đăng ký mới, truy cập các thành phần **JavaBeans**, kiểm soát giữa các trang và chia sẻ thông tin giữa các request, các trang.

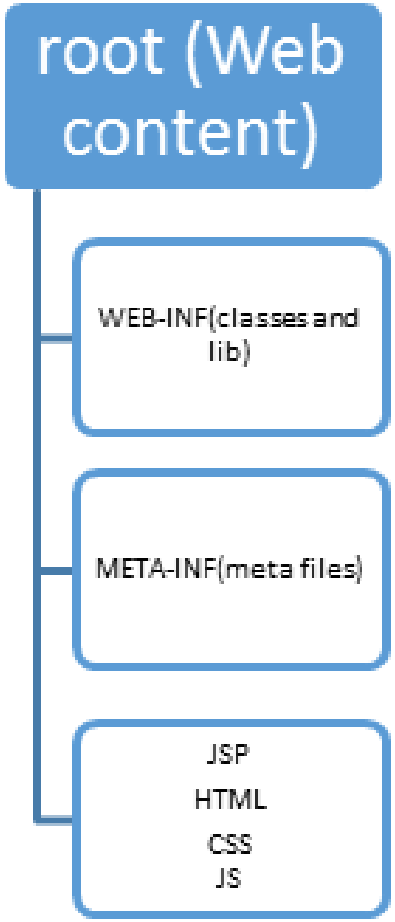
JSP là gì?


Browser
1
Request
4
Response
JSP page
2
Java bean
3
Data store
Business logic
Application server
Enterprise servers or data sources

Tại sao phải dùng JSP?

- Hiệu suất tốt hơn vì JSP cho phép nhúng các thành phần động trong các trang HTML thay vì có các tệp CGI riêng biệt.
- JSP luôn được biên soạn trước khi chúng được xử lý bởi máy chủ không giống như CGI / Perl, yêu cầu máy chủ tải một trình thông dịch và tập lệnh đích mỗi khi trang được request.
- Giống như Servlet, JSP cũng có quyền truy cập vào tất cả các Enterprise Java APIs, bao gồm JDBC, JNDI, EJB, JAXP vv
- Các trang JSP có thể được sử dụng kết hợp với các servlet xử lý logic nghiệp vụ, model được hỗ trợ bởi Java servlet.
- JSP là một phần của Java EE, một nền tảng hoàn chỉnh cho các ứng dụng enterprise.

Cấu trúc thư mục của JSP



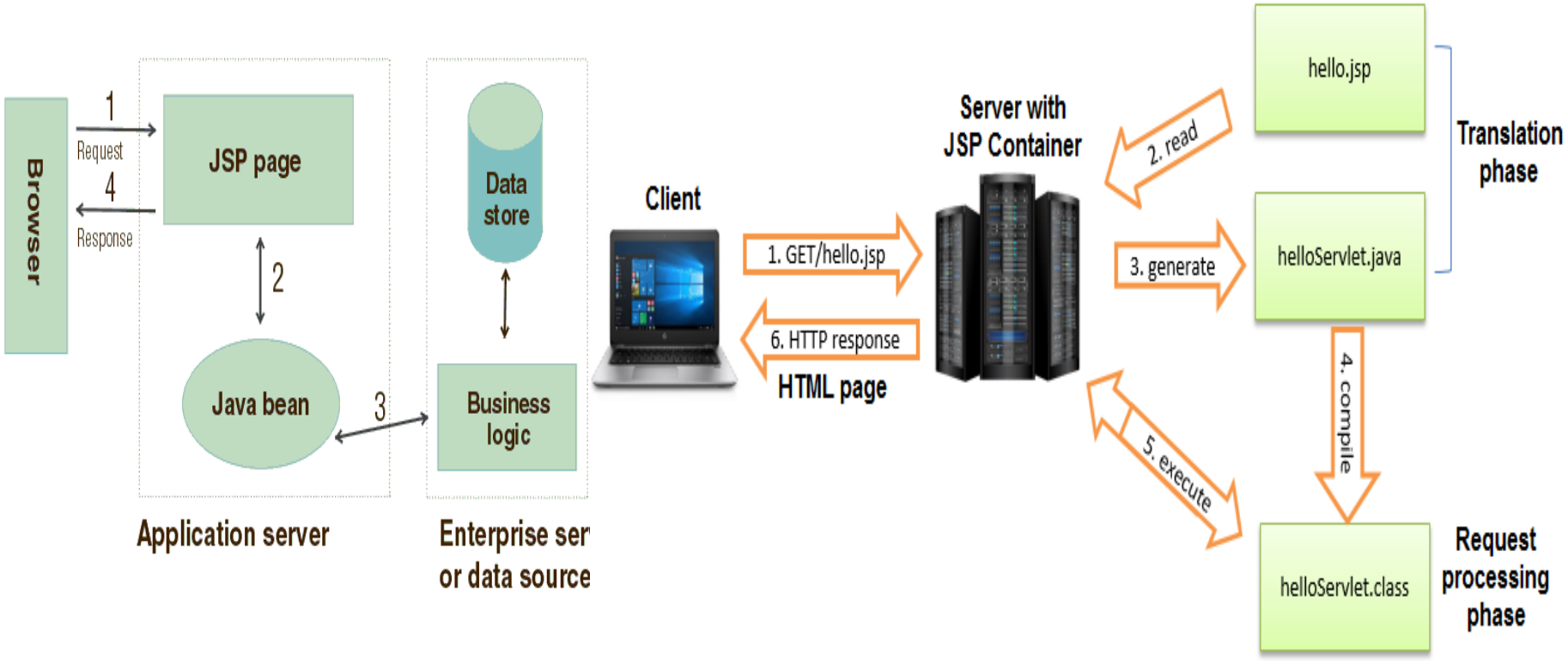
Thiết lập môi trường

- Cài Tomcat 8
- Cài JRE 1.8
- Eclipse

Ví dụ trang JSP: index.jsp


```
<%@ page language="java" contentType="text/html; charset=UTF-8 pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title></title>
</head>
<body>
  <p> This is the body of the jsp page. </p>
</body>
</html>
```

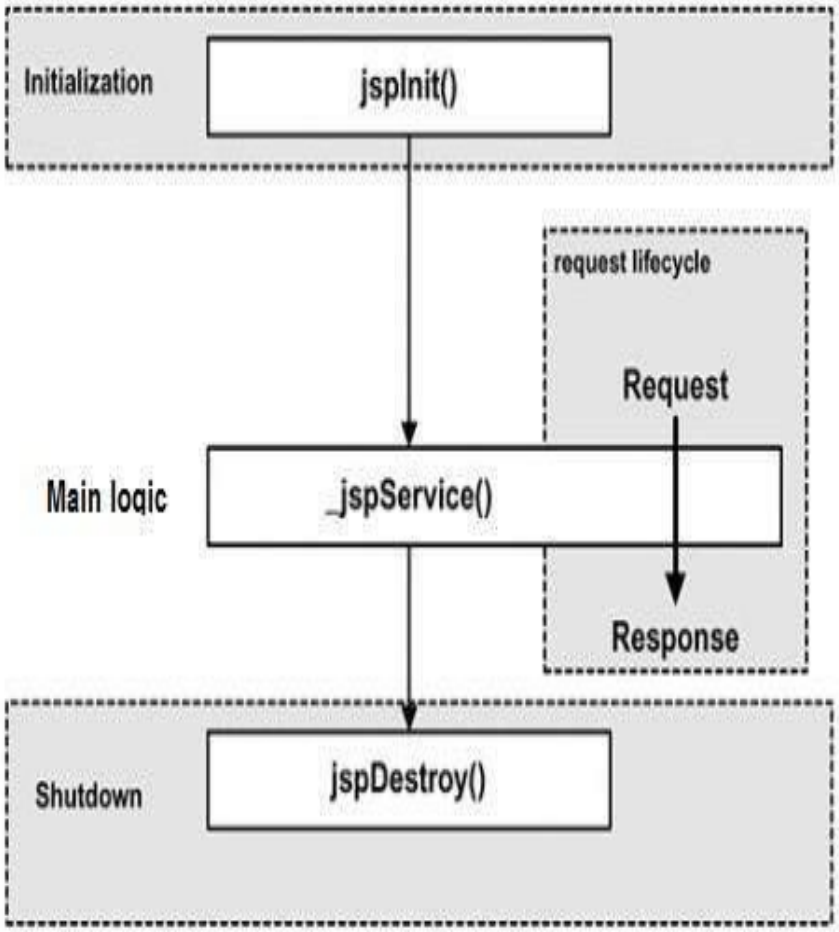
Kiến trúc JSP



Vòng đời JSP

- **Vòng đời của JSP** là quá trình từ khi nó được tạo ra cho đến khi nó bị hủy. Vòng đời của JSP cũng tương tự như một vòng đời của servlet với việc bổ sung thêm một bước biên dịch một JSP thành servlet.
- Việc hiểu về vòng đời của JSP chính là chìa khóa để hiểu về bản chất của JSP hoạt động như thế nào.
- Vòng đời của JSP bao gồm **4 pha** sau:
  - Biên dịch (Compilation).
  - Khởi tạo (Initialization).
  - Thực thi (Execution).
  - Hủy (Cleanup).

Vòng đời JSP



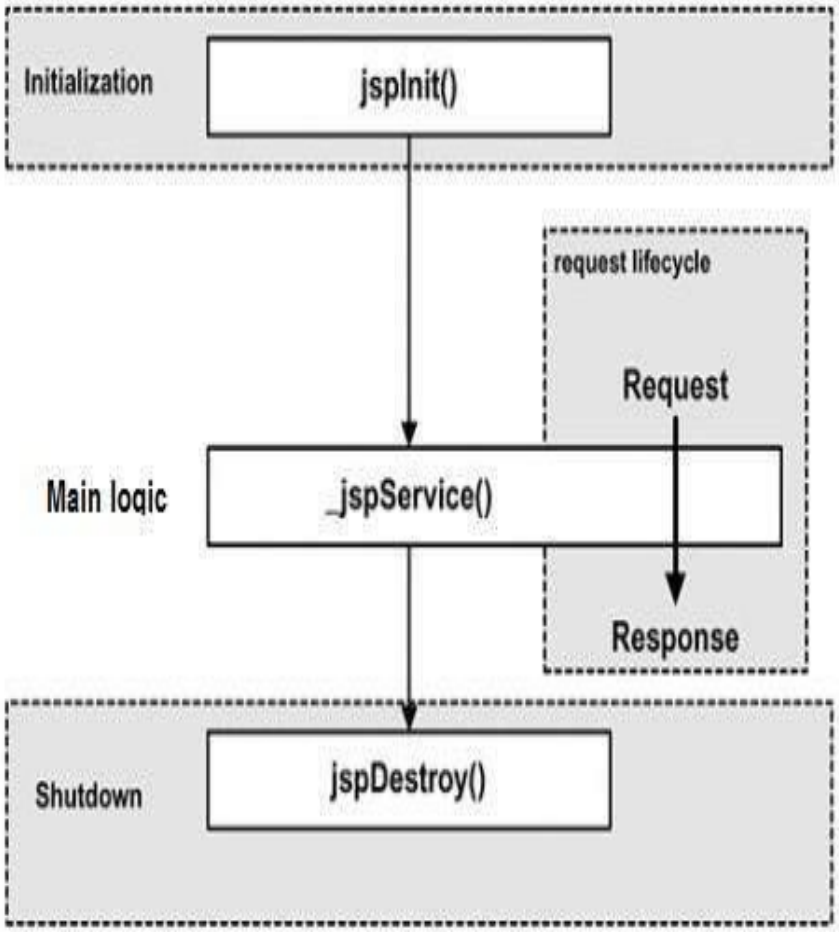
1. Biên dịch JSP

Khi trình duyệt yêu cầu một JSP, công cụ JSP đầu tiên sẽ kiểm tra xem liệu nó cần phải biên dịch trang này hay không. Nếu trang chưa bao giờ được biên dịch, hoặc nếu JSP đã được sửa đổi kể từ khi nó được biên dịch lần cuối.

Quá trình biên dịch bao gồm ba bước:

- **Phân tích cú pháp JSP.**
- **Chuyển JSP thành một servlet.**
- **Biên dịch servlet.**

Vòng đời JSP

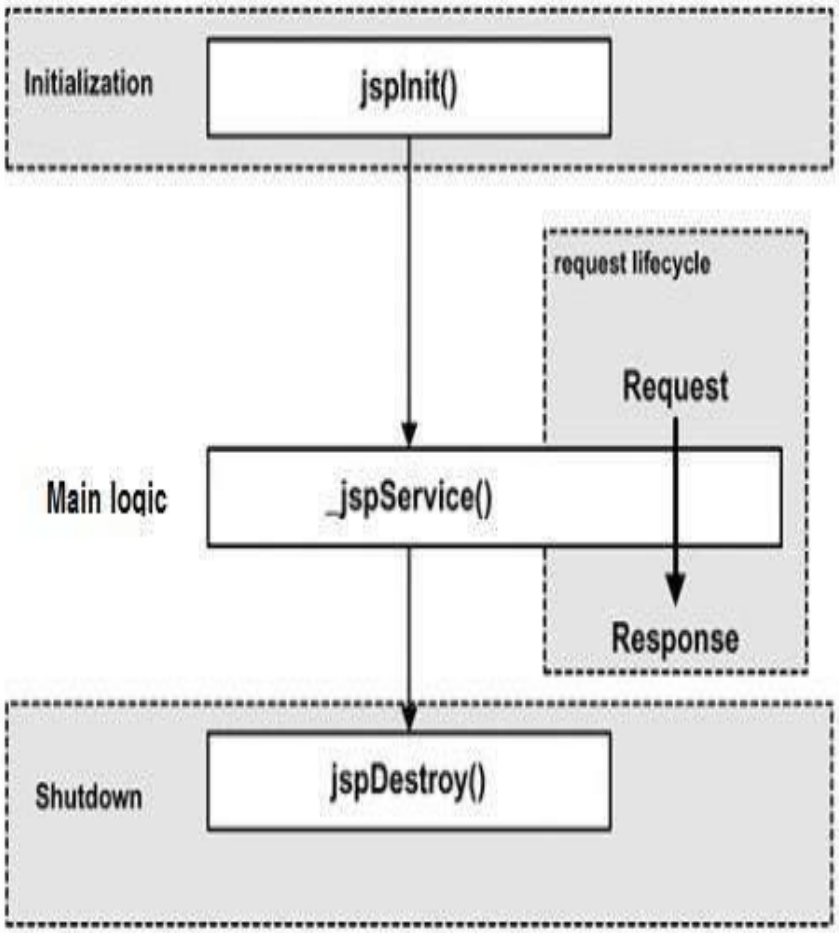


2. Khởi tạo JSP

Khi một container tải một JSP nó gọi phương thức **jspInit()** trước khi phục vụ các yêu cầu. Nếu bạn cần thực hiện khởi tạo JSP theo cách của bạn, hãy ghi đè phương thức jspInit().

```
public void jspInit() {
  // Initialization code...
}
```

Vòng đời JSP



3. Thực thi JSP

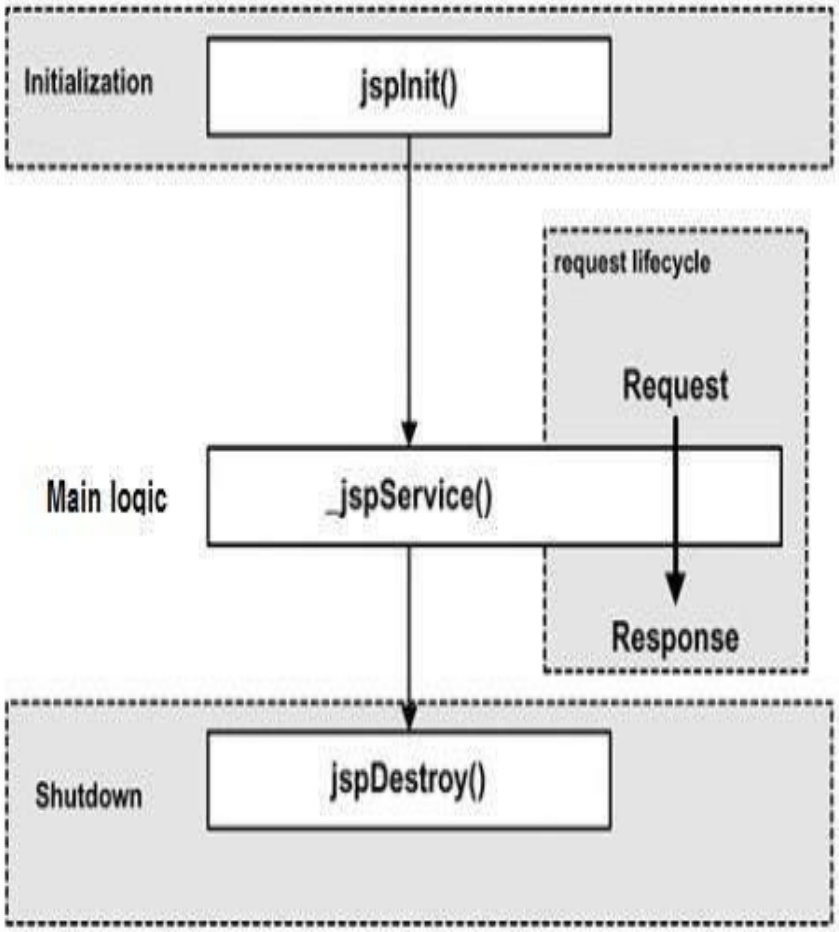
Giai đoạn này của vòng đời JSP đại diện cho tất cả các tương tác với các yêu cầu cho đến khi JSP bị hủy.

Bất cứ khi nào trình duyệt yêu cầu một JSP và trang đã được nạp và khởi tạo, công cụ JSP sẽ gọi phương thức **_jspService()**.

Phương thức _jspService() có một **HttpServletRequest** và một **HttpServletResponse** làm các tham số của nó như sau:

```
void _jspService(HttpServletRequest request,
HttpServletResponse response) {
   // Service handling code...
}
```

Vòng đời JSP



4. Hủy JSP

Giai đoạn hủy của chu kỳ sống JSP biểu thị khi một JSP không được tiếp tục sử dụng bởi một container.

Phương thức **jspDestroy()** tương đương phương thức hủy đối với servlet. Ghi đè phương thức jspDestroy() khi bạn cần thực hiện bất kỳ công việc dọn dẹp nào, chẳng hạn như giải phóng kết nối cơ sở dữ liệu hoặc đóng các tệp.

Phương thức jspDestroy() có dạng sau:

```
public void jspDestroy() {
   // Your cleanup code goes here.
}
```

Cú pháp JSP

- Các thành phần của JSP:
  - JSP Scriptlet.
  - JSP Declaration.
  - JSP Expression.
  - JSP Comment
  - JSP Directive.
  - JSP Action.

Cú pháp JSP

1. JSP Scriptlet

- Với Scriptlet bạn có thể viết bất kỳ câu lệnh java nào, như khai báo biến, khai báo phương thức, khởi tạo biến, in dữ liệu ra trình duyệt.
- Cú pháp của JSP Scriptlet:

```
<% java code %>
```

```
<% String message = "Hello JSP!"; %>
<% out.print(message); %>
```

Cú pháp JSP

Ví dụ: JSP Scriptlet

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title> Cú pháp JSP</title>
</head>
<body>
  <p>
    <% String message = "Hello JSP!"; %>
    <% out.print(message); %>
  </p>
</body>
</html>
```

Cú pháp JSP

2. JSP Declaration

- Với JSP Declaration bạn có thể khai báo biến và phương thức bằng java code bên trong tệp JSP. Bạn phải khai báo biến và phương thức trước khi sử dụng nó trong tệp JSP.
- Cú pháp của JSP Declaration:

<%! declaration; [declaration;]+ ... %>

  - Ví dụ:

```
<%! List<String> list = new ArrayList<String>();%>
<%! int i = 0;%>
<%! int a, b, c;%>
```

Cú pháp JSP

3. JSP Expression

- JSP Expression - biểu thức JSP được sử dụng để in một chuỗi ký tự.
- Cú pháp của JSP Expression:

<%= expression %>

```
<%!
        int factorial(int n)
        {
        if (n == 0)
            return 1;
          return n*factorial(n-1);
        }
    %>
       <%= "Factorial of 5 is:"+factorial(5) %>
```

Cú pháp JSP


```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title> Cu phap JSP</title>
</head>
<body>
  <p>Today's date: <%= (new java.util.Date()).toLocaleString()%></p>
</body>
</html>
```

Cú pháp JSP

- JSP Comments: chú thích trong jsp
  - <% -- JSP Comment --%>

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>JSP Comments</title>
</head>
<body>
<%-- Comments section --%>
<% out.println("This is comments example"); %>

</body>
</html>
```

Example


```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>JSP Example</title>
</head>
<body>
<%-- This is a JSP example with scriplets, comments , expressions --%>
<% out.println("This is guru JSP Example"); %>
<% out.println("The number is "); %>
<%! int num12 = 12; int num32 = 12; %>
<%= num12*num32 %>
Today's date: <%= (new java.util.Date()).toLocaleString()%>
</body>
</html>
```

4. JSP Directives

- Một chỉ thị JSP ảnh hưởng đến cấu trúc tổng thể của lớp Servlet. Nó thường có dạng sau:

<%@ directive attribute="value" %>

No.	Chỉ thị và Mô tả
1	<%@ page ...%> Định nghĩa thuộc tính page-dependent, chẳng hạn như import package, trang lỗi và yêu cầu bộ đệm.
2	<%@ include ...%> Bao gồm tập tin trong giai đoạn biên dịch.
3	<%@ taglib ...%> Khai báo một tag library, chứa các hành động tùy chỉnh, được sử dụng trong trang.

Cú pháp JSP

4. JSP Directives

- Một chỉ thị JSP ảnh hưởng đến cấu trúc tổng thể của lớp Servlet. Nó thường có dạng sau: <%@ directive attribute="value" %>

No.	Chỉ thị và Mô tả
1	<%@ page ...%> Định nghĩa thuộc tính page-dependent, chẳng hạn như import package, trang lỗi và yêu cầu bộ đệm. Language, Extends, Import, contentType, info, session, isThreadSafe, autoflush, buffer, IsErrorPage, pageEncoding, errorPage, isELIgonored
2	<%@ include ...%> Bao gồm tập tin trong giai đoạn biên dịch. <%@ include file="directive_header_jsp3.jsp" %>
3	<%@ taglib ...%> Khai báo một tag library, chứa các hành động tùy chỉnh, được sử dụng trong trang. <%@ taglib prefix="gtag" uri="http://java.sun.com/jsp/jstl/core" %> <gtag:hello/>

Cú pháp JSP

Ví dụ

```
<%@ page import="java.util.List"%>
<%@ page language="java" contentType="text/html; charset=UTF-8 pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title> Cú pháp JSP directive </title>
</head>
<body>
  <p>
    <%! List<String> list; %>
  </p>
</body>
</html>
```

Đây là ví dụ sử dụng jsp directive để khai báo việc import interface **java.util.List** vào tệp JSP.

5. JSP Action

- JSP action sử dụng **cấu trúc XML** để điều khiển hành vi của servlet engine. Bạn có thể chèn một file, tái sử dụng JavaBeans component, chuyển tiếp người dùng đến một trang khác hoặc tạo ra HTML cho Java plugin.
- Cú pháp của JSP Action:

```
<jsp:action_name attribute="value" />
```

5. JSP Action

No.	Cú pháp & Mục đích
1	jsp:include Thêm nội dung được include tại thời điểm trang được request.
2	jsp:useBean Tìm hoặc khởi tạo một JavaBean.
3	jsp:setProperty Thiết lập thuộc tính của một JavaBean.
4	jsp:getProperty Chèn thuộc tính của một JavaBean vào đầu ra.
5	jsp:forward Chuyển tiếp người yêu cầu tới một trang mới.

5. JSP Action

No.	Cú pháp & Mục đích
6	jsp:plugin Tạo mã trình duyệt cụ thể mà làm cho một thẻ OBJECT hoặc EMBED cho Java plugin.
7	jsp:element Định nghĩa các phần tử XML động.
8	jsp:attribute Định nghĩa thuộc tính của phần tử XML được tự động định nghĩa.
9	jsp:body Định nghĩa phần thân của phần tử XML được tự động định nghĩa.
10	jsp:text Được sử dụng để viết văn bản mẫu trong các trang JSP và tài liệu.

Tạo ứng dụng JSP trên Eclipse

- Tạo Java dynamic web project trong Eclipse
- Cấu hình trang index trong web.xml
  - Cấu hình trang index.jsp làm trang chủ trong file **WebContent/WEB-INF/web.xml** như sau:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID"
version="3.1">
  <display-name>jsp-hello-world</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

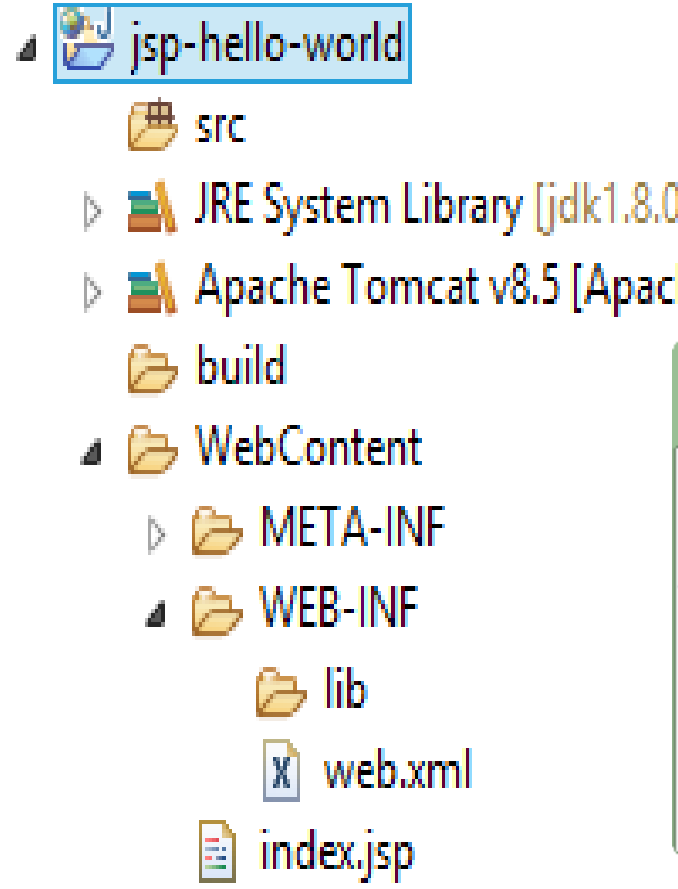
Tạo ứng dụng JSP trên Eclipse

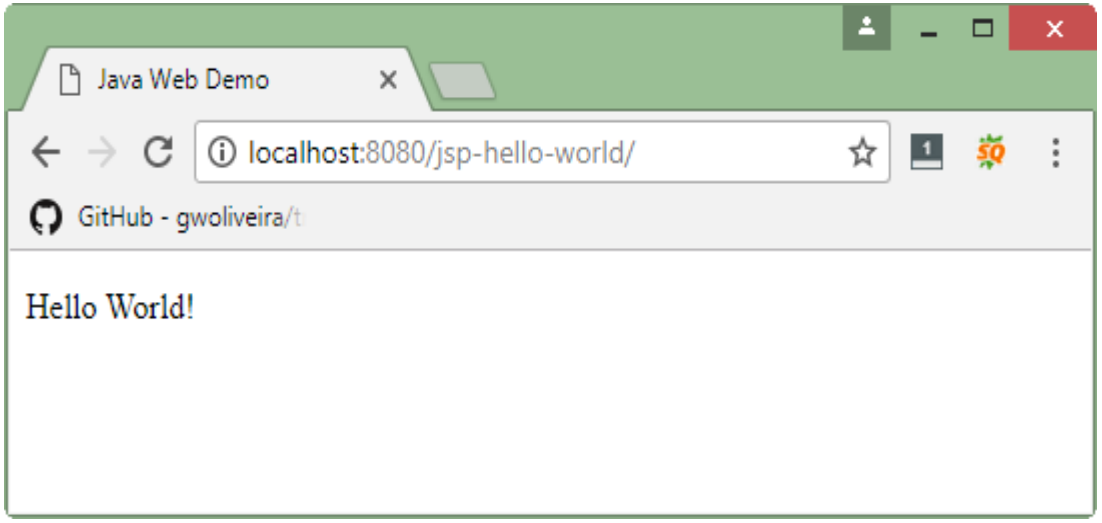
- Tạo các trang JSP
  - Tất cả các file JSP trong dự án java web được tạo trong thư mục **WebContent**.
  - Tạo file index.jsp trong thư mục WebContent:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Java Web Demo</title>
</head>
<body>
<%! String message = "Hello World!"; %>
<p> <%= message %> </p>
</body>
</html>
```

Tạo ứng dụng JSP trên Eclipse

- Cấu trúc của project

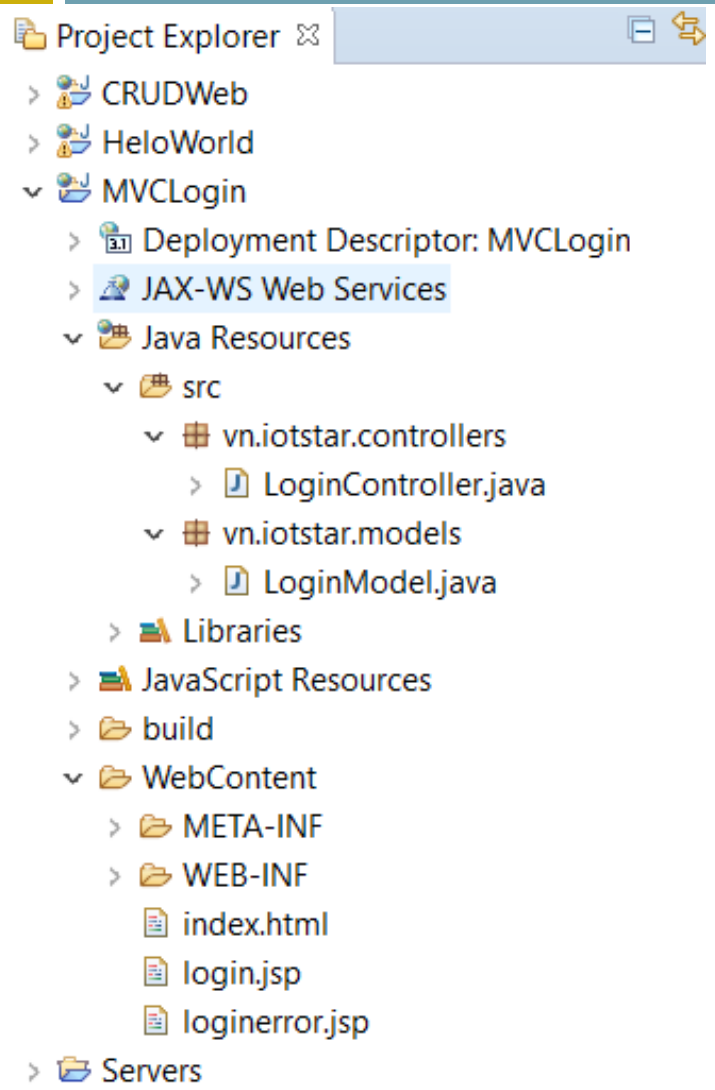




MVC trong JSP

- **MVC** là viết tắt của **Model View Controller** là một mẫu thiết kế phần mềm được chia thành 3 phần riêng biệt đó là xử lý nghiệp vụ, xử lý giao diện và dữ liệu.
- Ví dụ về MVC trong JSP
  - Trong ví dụ này chúng ta sử dụng servlet như một controller, jsp như một thành phần view và lớp java bean như một model.

MVC trong JSP



Trong đó:

- **index.jsp** là trang thu thập thông tin người dùng.
- **ControllerServlet.java** một servlet được cài đặt như một controller.
- **login-success.jsp** và **login-error.jsp** là các view.
- **web.xml** là file mapping servlet.

LoginModel.java (Models)

```
public class LoginBean {
    private String name,
password;

    public String getName()
{
        return name;
    }

    public void
setName(String name) {
        this.name = name;
    }
    public String getPassword() {
        return password;
    }

    public void setPassword(String password)
{
        this.password = password;
    }

    public boolean validate() {
        if ("admin".equals(password)) {
            return true;
        } else {
            return false;
        }
    }
}
```

MVC trong JSP

LoginController.java

```
public class LoginController extends HttpServlet {
    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String name = request.getParameter("name");
        String password = request.getParameter("password");

        LoginBean bean = new LoginBean();
        bean.setName(name);
        bean.setPassword(password);
        request.setAttribute("bean", bean);

        boolean status = bean.validate();
```

MVC trong JSP

LoginController.java

```
if (status) {
            RequestDispatcher rd = request
                    .getRequestDispatcher("login-success.jsp");
            rd.forward(request, response);
        } else {
            RequestDispatcher rd = request
                    .getRequestDispatcher("login-error.jsp");
            rd.forward(request, response);
        }
    }
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse
resp)
            throws ServletException, IOException {
        doPost(req, resp);
    }
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    id="WebApp_ID" version="3.1">
    <display-name>jsp-mvc-example</display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
    <servlet>
        <servlet-name>LoginController</servlet-name>
        <servlet-class>vn.iotstar.LoginController</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginController</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>
</web-app>
```

MVC trong JSP

index.jsp

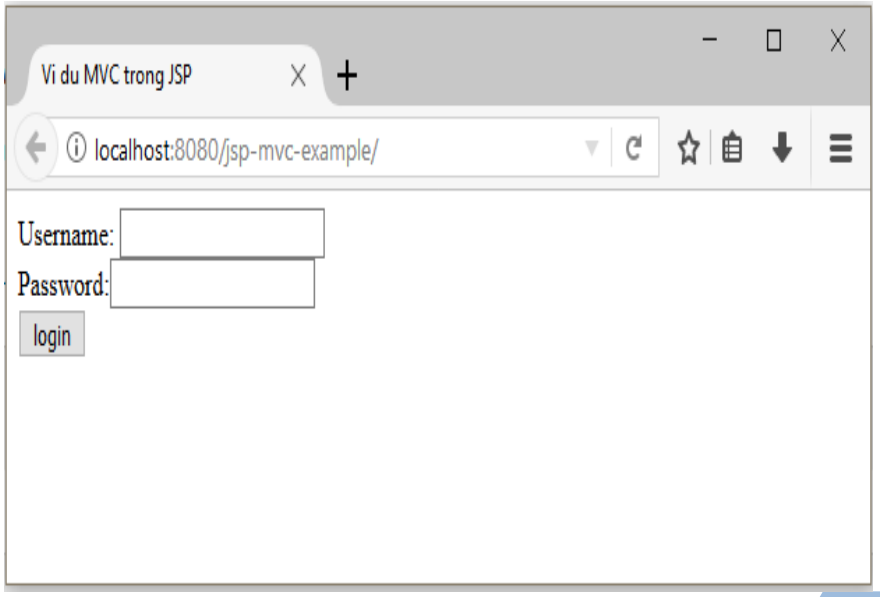
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Vi du MVC trong JSP</title>
</head>
<body>
  <form action="login" method="post">
    Username: <input type="text" name="name"><br>
    Password:<input type="password" name="password"><br>
    <input type="submit" value="login">
  </form>
</body>
</html>
```

login.jsp (Views)

```
<%@page import="vn.iotstar.models.LoginModel"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Login Success</title>
</head>
<body>
  <p>You are successfully logged in!</p>
  <%
      LoginBean bean = (LoginBean) request.getAttribute("bean");
      out.print("Welcome, " + bean.getName());
  %>
</body>
</html>
```

loginerror.jsp (Views)

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Login Error</title>
</head>
<body>
  <p>username or password is incorrect!</p>
  <%@ include file="index.jsp"%>
</body>
</html>
```



Đối tượng session trong JSP

- **Đối tượng session trong JSP** là một đối tượng ẩn của **javax.servlet.http.HttpSession**.
- Đối tượng session được sử dụng để theo dõi phiên của các request của client.
- Đối tượng này để set, get hoặc remove thuộc tính hoặc để lấy thông tin về phiên làm việc.

Đối tượng session trong JSP

- index.jsp

```
<form action="welcome.jsp">
  <input type="text" name="username">
  <input type="submit" value="Submit">
</form>
```

welcome.jsp

```
<%
      String name = request.getParameter("username");
      out.print("Welcome " + name);

      session.setAttribute("user", name);
  %>
  <a href="second.jsp">second jsp page</a>
```

second.jsp

```
<%
      String name = (String)
session.getAttribute("user");
      out.print("Hello " + name);
 %>
```

Đối tượng request trong JSP

- **Đối tượng request trong JSP** là một đối tượng ẩn của **javax.servlet.http.HttpServletRequest**. Mỗi lần client request một trang thì JSP engine tạo ra một đối tượng mới để đại diện cho request đó.
- Đối tượng request cung cấp các phương thức để lấy thông tin HTTP header bao gồm Form data, cookie, HTTP method, vv.
- Nó có thể được sử dụng để get, set và remove các thuộc tính trong phạm vi jsp request.

Đối tượng request trong JSP

- index.jsp

```
<form action="welcome.jsp">
  <input type="text" name="username">
  <input type="submit" value="Submit">
</form>
```

welcome.jsp

```
<%
        String name = request.getParameter("username");
        out.print("Welcome " + name);

        session.setAttribute("user", name);
    %>
    <a href="second.jsp">second jsp page</a>
```

Đối tượng response trong JSP

- **Đối tượng response trong JSP** là một đối tượng ẩn của **javax.servlet.http.HttpServletResponse**. Cũng như việc server tạo ra các đối tượng resquest, nó cũng tạo ra một đối tượng response để đại diện cho các phản hồi cho client.
- Đối tượng response cũng định nghĩa các giao diện để tạo ra HTTP header mới. Thông qua đối tượng này, lập trình viên JSP có thể thêm cookie mới hoặc date stamps, HTTP status code, chuyển hướng, vv.

Đối tượng response trong JSP


```
<form action="welcome.jsp">
  <input type="text" name="username">
  <input type="submit" value="Submit">
</form>
```

Welcome.jsp

```
<%
    response.sendRedirect("http://www.google.com");
%>
```

Đối tượng pageContext trong JSP

- **Đối tượng pageContext trong JSP** là một đối tượng ẩn của javax.servlet.jsp.PageContext. Đối tượng pageContext được sử dụng để đại diện cho toàn bộ trang JSP. Nó có thể được sử dụng để get, set hoặc remove thuộc tính từ một trong các phạm vi sau:
  - page
  - request
  - session
  - application
- Lớp **PageContext** định nghĩa một số trường, bao gồm định nghĩa bốn phạm vi **PAGE_SCOPE**, **REQUEST_SCOPE**, **SESSION_SCOPE** và **APPLICATION_SCOPE**. Nó cũng hỗ trợ hơn 40 phương thức, khoảng một nửa trong số đó được kế thừa từ lớp **javax.servlet.jsp.JspContext**.
- Một trong những phương thức quan trọng là **removeAttribute**. Phương thức này chấp nhận một hoặc hai đối số. Ví dụ, **pageContext.removeAttribute ("attrName")** loại bỏ các thuộc tính từ tất cả các phạm vi, trong khi mã sau chỉ loại bỏ nó từ phạm vi trang:

```
pageContext.removeAttribute("attrName", PAGE_SCOPE);
```

Đối tượng pageContext trong JSP

- Index.jsp

```
<form action="welcome.jsp">
  <input type="text" name="username">
  <input type="submit" value="Submit">
</form>
```

Welcome.jsp

```
<%
      String name =
request.getParameter("username");
      out.print("Welcome " +
name);


pageContext.setAttribute("user",
name, PageContext.SESSION_SCOPE);
  %>
  <a href="second.jsp">second jsp
page</a>
```

Second.jsp

```
<%
      String name = (String)
pageContext.getAttribute("user",

PageContext.SESSION_SCOPE);
      out.print("Hello " + name);
  %>
```

Đối tượng page trong JSP

- **Đối tượng page trong JSP** là một tham chiếu thể hiện của một page.
- Đối tượng page là tương đương vơi đối tượng **this**.
- Để sử dụng đối tượng page nó phải được ép về kiểu Servlet. ví dụ:

```
<%
(HttpServlet)page.log("message");
%>
```

Nó ít được sử dụng vì bạn có thể sử dụng trực tiếp đối tượng this trong JSP, ví dụ:

```
<% this.log("message"); %>
```

Đối tượng config trong JSP

- **Đối tượng config trong JSP** là đối tượng ẩn của **javax.servlet.ServletConfig**.
- Đối tượng này cho phép lập trình viên JSP truy cập vào các tham số khởi tạo Servlet hoặc JSP engine như đường dẫn hoặc vị trí tập tin, vv.
- Ví dụ phương thức sau trả về tên servlet, là chuỗi chứa trong phần tử **<servlet-name>** được định nghĩa trong file **WEB-INF\web.xml**.

```
config.getServletName();
```

Đối tượng config trong JSP


```
<form action="welcome.jsp">
  <input type="text" name="username">
  <input type="submit" value="Submit">
</form>
```

```
<servlet>
    <servlet-name>welcome</servlet-name>
    <jsp-file>/welcome.jsp</jsp-file>
    <init-param>
        <param-name>dname</param-name>
        <param-value>com.mysql.jdbc.Driver</param-value>
    </init-param>
</servlet>
<servlet-mapping>
    <servlet-name>welcome</servlet-name>
    <url-pattern>/welcome</url-pattern>
</servlet-mapping>
```

Welcome.jsp

```
<%
    out.print("Welcome " + request.getParameter("username"));
    String driver = config.getInitParameter("dname");
    String servlet = config.getServletName();
    out.print("<br> Driver name is = " + driver);
    out.print("<br> Servlet name is = " + servlet);
%>
```

JSP Expression Language (EL)

- JSP Syntax of Expression Language (EL)
- JSP If-else
- JSP Switch
- JSP For loop
- JSP While loop
- JSP Operators

JSP Expression Language (EL)

- Ngôn ngữ biểu thức (EL) là cơ chế đơn giản hóa khả năng truy cập của dữ liệu được lưu trữ trong thành phần bean Java và các đối tượng khác như yêu cầu, phiên và ứng dụng, v.v. Có nhiều toán tử trong JSP được sử dụng trong EL như toán tử số học và logic để thực hiện một biểu hiện. Nó đã được giới thiệu trong JSP 2.0
- **Cú pháp** :${biểu thức}

JSP Expression Language (EL)

HCMUTE

- **JSP If-else**

```
If (điều kiện kiểm tra)
{
// Khối câu lệnh
}
else
{
// Khối câu lệnh
 }
```

```
<%! int month=5; %>
<% if(month==2){ %>
<a>Its February</a>
<% }else{ %>
<a>Any month other than February</a>
<%} %>
```

JSP Expression Language (EL)

HCMUTE

- **JSP Switch**

```
switch (operator)
{
Case 1: Block of statements
break;
Case 2: Block of statements
break;
case n: Block of statements
 break;
default: Block of statements
break;
}
```

```
<%! int week=2; %>
<% switch(week){
case 0: out.println("Sunday");
break;
case 1: out.println("Monday");
break;
case 2: out.println("Tuesday");
break;
case 3:out.println("wednesday");
break;
case 4:
out.println("Thursday");
break;
case 5:
out.println("Friday");
break;
default:
out.println("Saturday");
}
%>
```

JSP Expression Language (EL)

- **JSP For loop**

```
For(inti=0;i<n;i++)
{
   //block of statements
}
```

```
<%! int num=5; %>
<% out.println("Numbers are:");
for(int i=0;i<num;i++)
{
out.println(i);
}%>
```

JSP Expression Language (EL)

- **JSP While loop**

```
While(i<n)
{
   //Block of
   statements
}
```

```
<%! int day=2; int i=1; %>
<% while(day>=i){
    if(day==i){
        out.println("Its Monday");
        break;}
     i++;}
%>
```

JSP Expression Language (EL)

JSP Operators

.	Truy cập thuộc tính bean hoặc mục nhập Bản đồ
[]	Truy cập một mảng hoặc phần tử Danh sách
()	Nhóm một biểu thức con để thay đổi thứ tự đánh giá
+	Thêm vào
-	Phép trừ hoặc phủ định một giá trị
*	Phép nhân
/ or div	Phân công
% or mod	Modulo (phần còn lại)
== or eq	Kiểm tra sự bình đẳng
! = or ne	Kiểm tra sự bất bình đẳng
< or lt	Kiểm tra ít hơn
> or gt	Kiểm tra lớn hơn
<= or le	Kiểm tra nhỏ hơn hoặc bằng
> = or ge	Kiểm tra lớn hơn hoặc bằng
&& or And	Kiểm tra AND logic
|| or Or	Kiểm tra logic HOẶC
! Or not	Phần bổ sung Boolean một bậc
Empty	Kiểm tra các giá trị biến trống

```
<% int num1=10; int num2 = 50;
int num3 = num1+num2;
if(num3 != 0 || num3 > 0){
   int num4= num1*num2;
   out.println("Number 4 is " +num4);
   out.println("Number 3 is " +num3);
}%>
```

JSP Implicit Objects

- out : <% out.println("num1 is " +num1); %>
- Request : <% String uname= request.getParameter(« uname »);%>
- Response: <%response.setContentType("text/html"); %>
- Config: <% String servletName = config.getServletName();%>
- application: <% application.getContextPath(); %>
- Session: <% session.setAttribute("user","JSP"); %>
- pageContext:
  - <% pageContext.setAttribute("student","student",pageContext.PAGE_SCOPE);
  - String name = (String)pageContext.getAttribute("student");
  - out.println("student name is " +name);
  - %>

SP Operations: Insert, Update, Delete, Select

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql"
prefix="sql"%>
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1“ pageEncoding="ISO-8859-1"%>
```

- Select

```
<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost/dbTest"
user="gururoot" password="guru"/>
<sql:query dataSource="${snapshot}" var="result">
        SELECT * from user;
</sql:query>
<table>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
<c:forEach var="row" items="${result.rows}">
    <tr>
        <td><c:out value="${row.emp_id}"/></td>
        <td><c:out value="${row.emp_name}"/></td>
    </tr>
</c:forEach>
</table>
```

- Insert

```
<sql:setDataSource var=" guru " driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost/dbTest"
user="root" password="123"/>
<sql:update dataSource="${guru}" var="guruvar">
INSERT INTO guru_test VALUES (3, 'emp emp3');
</sql:update>
```

- Insert

```
<sql:setDataSource var=" guru " driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost/dbTest"
user="root" password="123"/>
<c:set var="guruid" value="3"/>
<sql:update dataSource="${guru}" var="guruvar">
DELETE FROM guru_test WHERE emp_id = ?
<sql:param value="${guruid}" />
</sql:update>
```

- Delete

```
<sql:setDataSource var=" guru " driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost/dbTest"
user="root" password="123"/>
<c:set var="guruid" value="2"/>
<sql:update dataSource="${guru}" var="guruvar">
UPDATE guru_test SET emp_name='emp guru99'
<sql:param value="${guruid}" />
</sql:update>
```

Bài tập

- Viết một chương trình tính giai thừa của một số nguyên dương n. Với n được nhập từ bàn phím. Ví dụ, n = 8 thì kết quả đầu ra phải là 1*2*3*4*5*6*7*8 = 40320.
- Viết chương trình giải phương trình bậc 2: ax2 + bx + c = 0.
- Dãy số Fibonacci được định nghĩa như sau: F0 = 0, F1 = 1, F2 = 1, Fn = F(n-1) + F(n-2) với n >= 2. Ví dụ: 0, 1, 1, 2, 3, 5, 8, ... Hãy viết chương trình tìm n số Fibonacci đầu tiên.
- **JSP File Upload & File Download Program**

Thông tin liên lạc

- Nguyễn Hữu Trung
- 0908617108
- trungnh@hcmute.edu.vn
- utex.hcmute.edu.vn