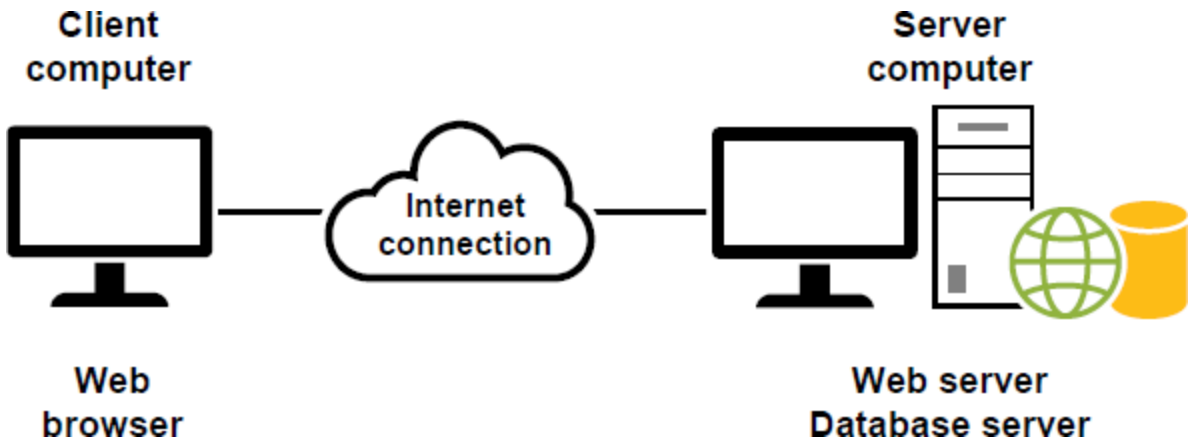
Web Java Application

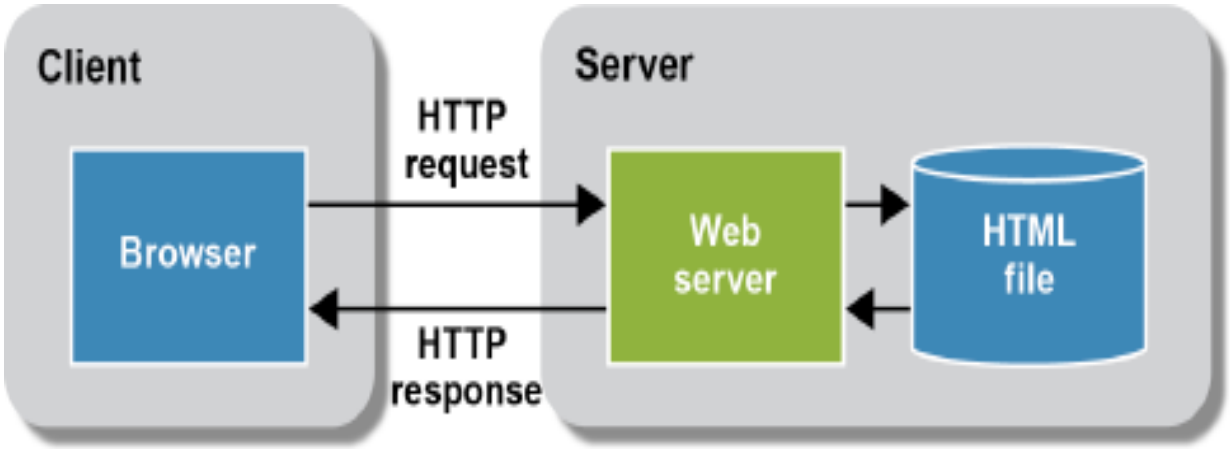
Tổng quan

Trường Đại học Công nghệ Sài gòn
Khoa Công nghệ Thông tin

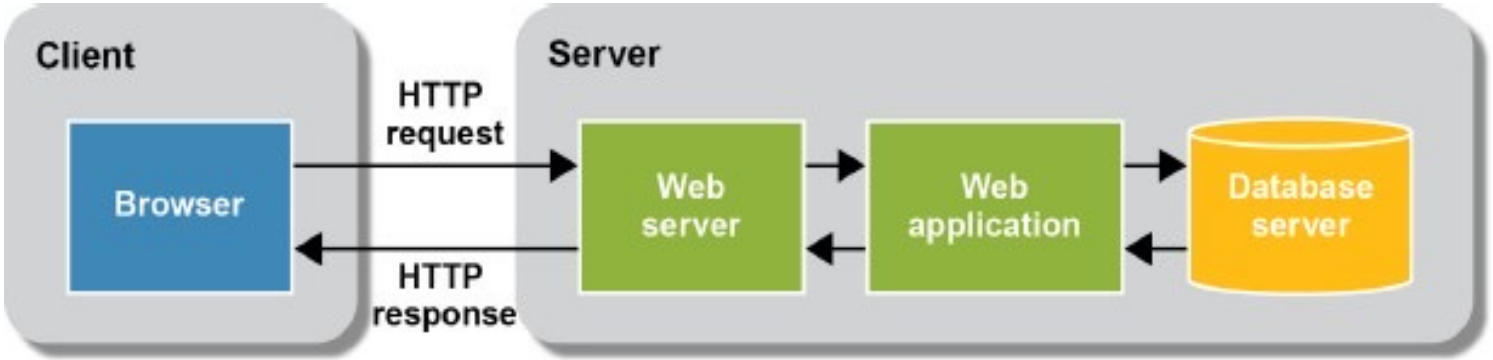
Các thành phần Web Aplication



- Xử lý static web page



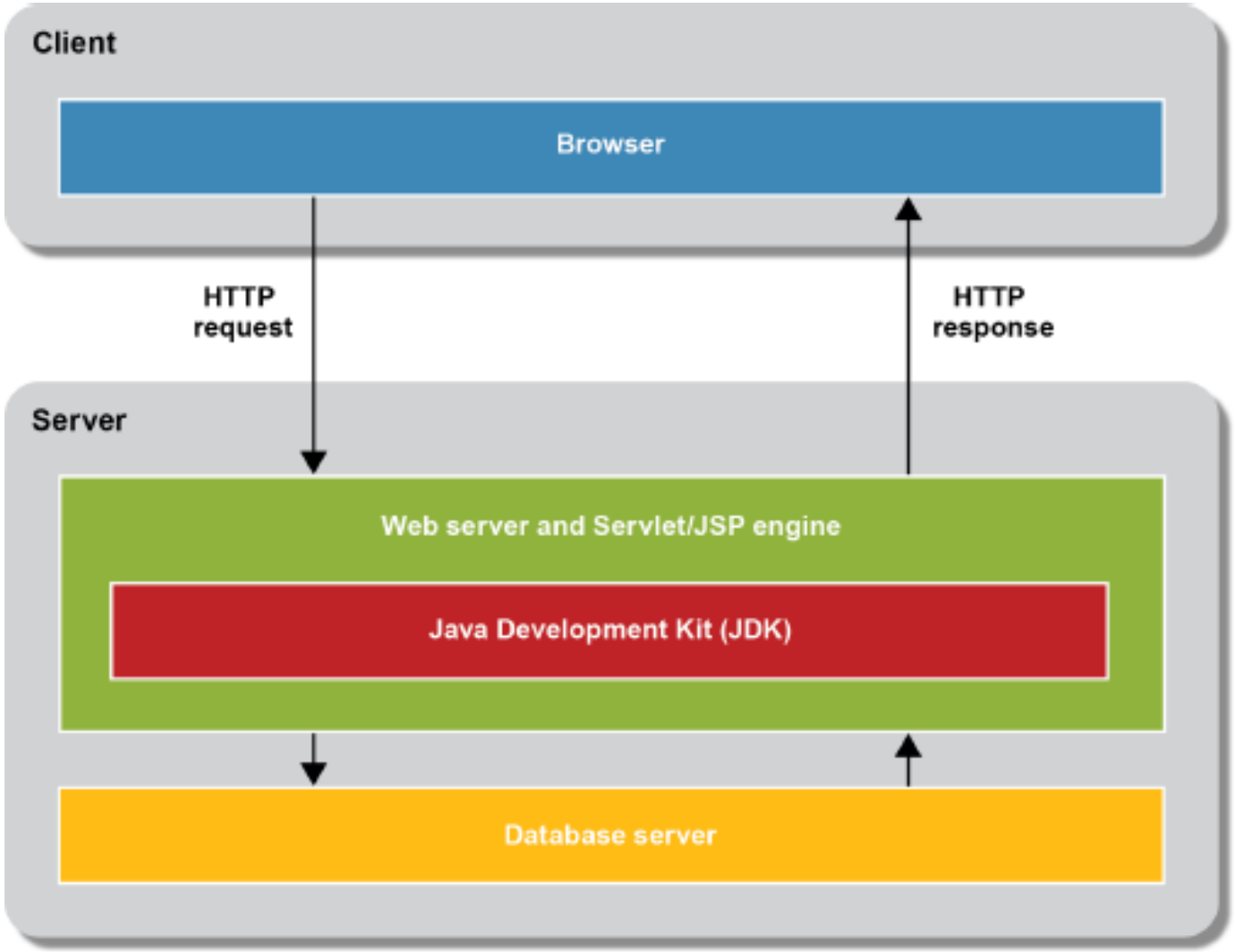
- Xử lý dynamic web page



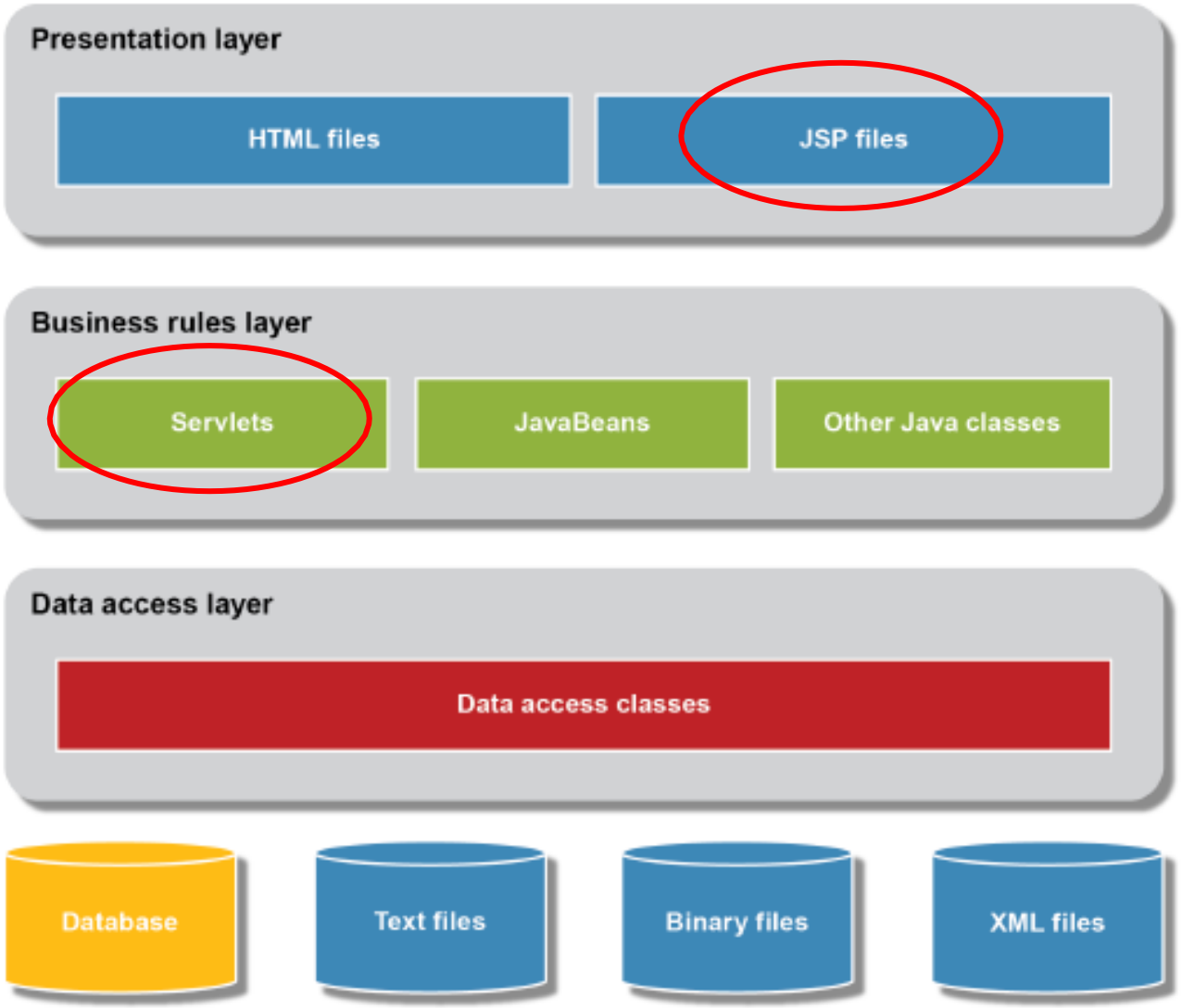
Ba phương pháp phát triển java web

- **Servlet/JSP**
- **JSF**
- **Spring frameWork**

Các thành phần của Servlet/JSP

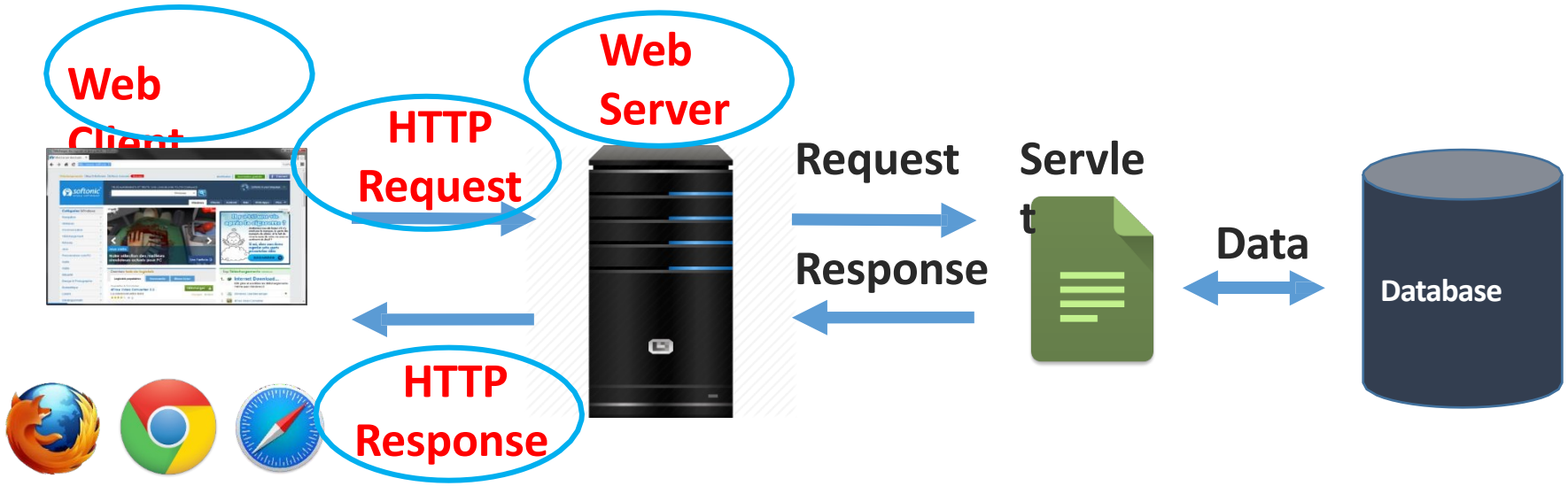


Kiến trúc cho ứng dụng Servlet/JSP

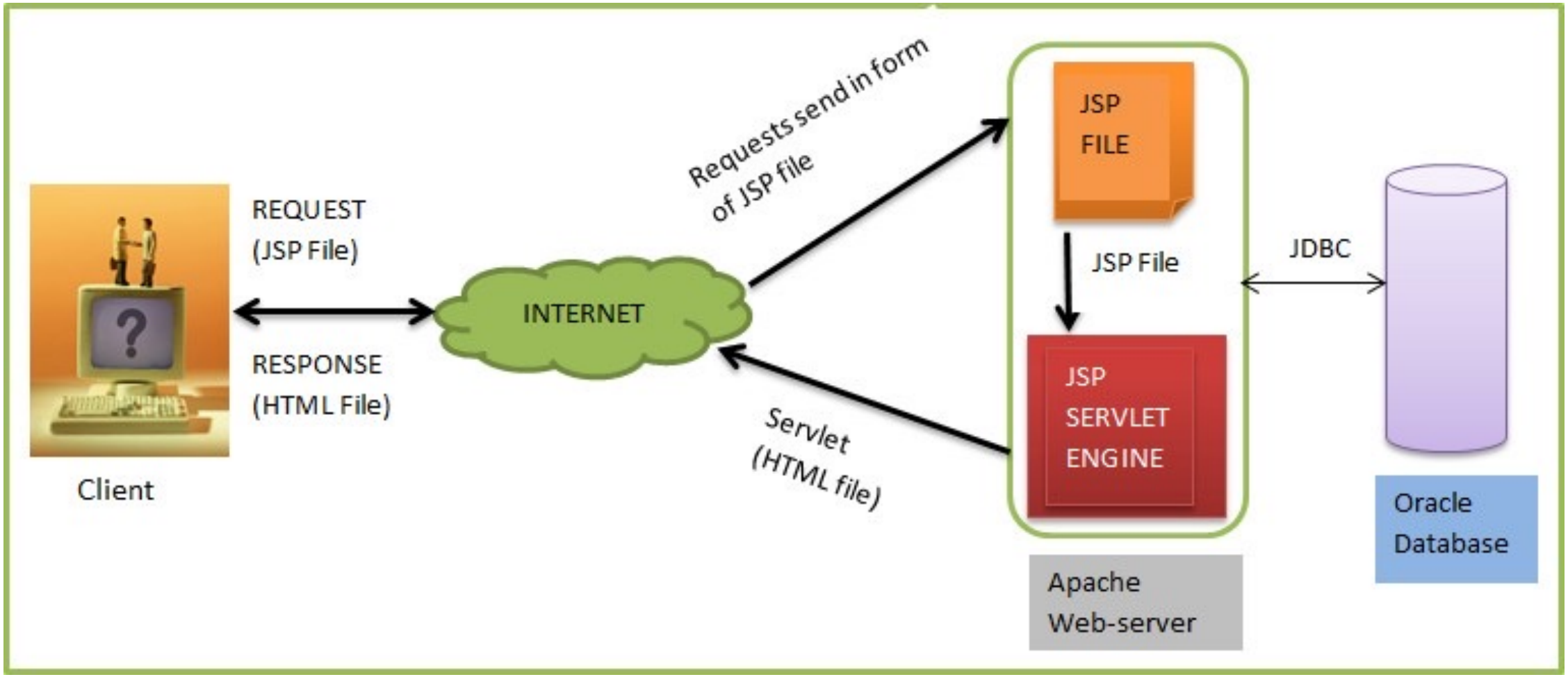


Java Servlets

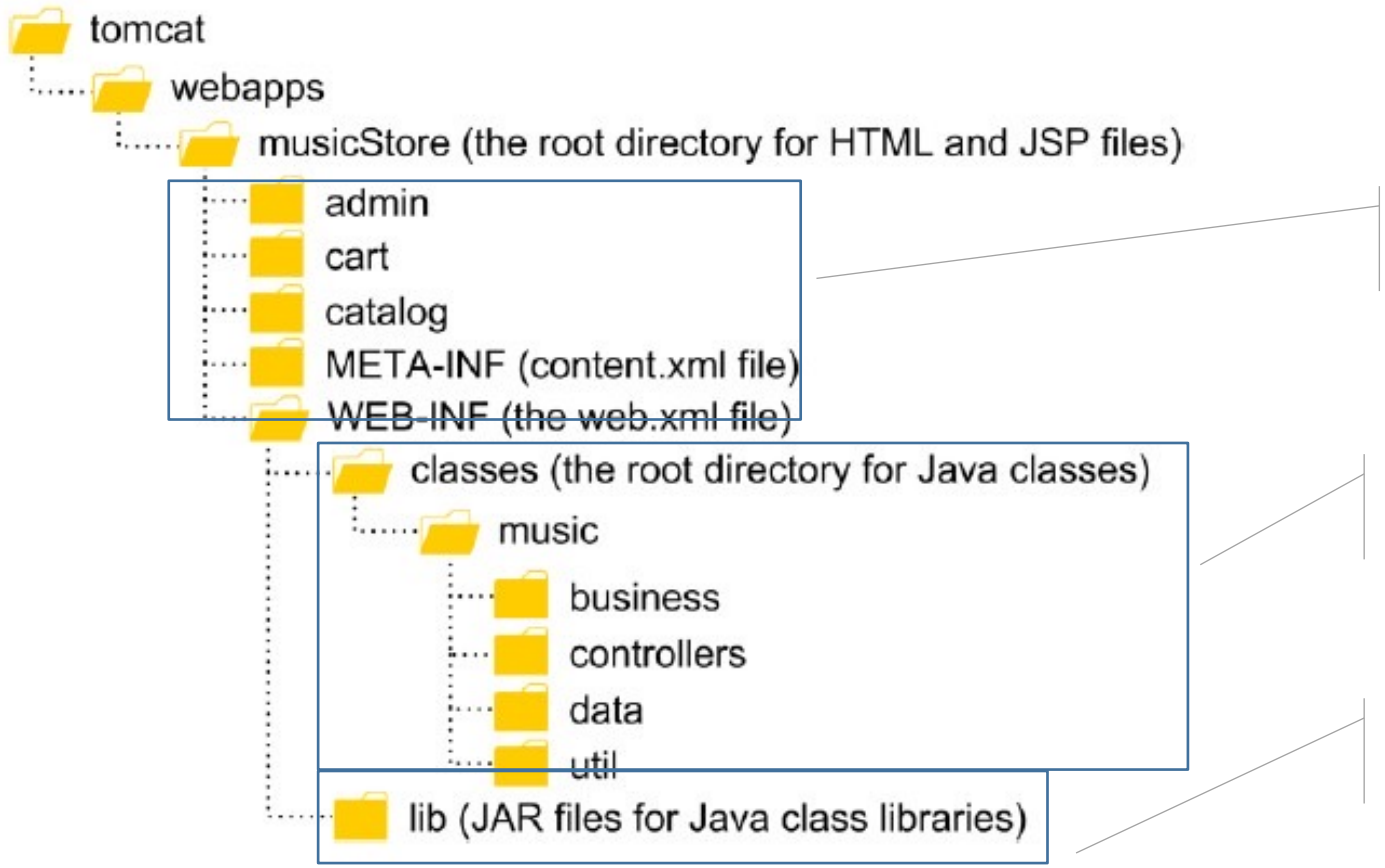
- Chương trình chạy trên server Web/Application và hoạt động như một tầng trung gian giữa một yêu cầu đến từ web browser



Kiến trúc JSP



Cấu trúc Web Application *.war



Chứa resource, jsp,css,html

Chứa java component

Chứa thư viện

Các thư mục và tập tin của Web Application

Thư mục	Mô tả
(root)	Chứa tập tin html và JSP
\WEB-INF	Chứa web.xml và không thể truy xuất trực tiếp từ web
\WEB-INF\classes	Chứa servlet và các class java của ứng dụng
\WEB-INF\lib	Chứa thư viện JAR file sử dụng trong ứng dụng
\META-INF	Chứa content.xml khai báo cấu hình ứng dụng

Cấu hình: tập tin web.xml

- Khai báo thông tin cấu hình web application
- Phiên bản cũ: Tất cả cấu hình được khai báo trong tập tin web.xml
- Web.xml=Deployment Descriptor
- Web.xml đặt trong thư mục WEB-INF

Cấu hình: tập tin web.xml

Configuration: web.xml file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3    <display-name>Temperature_Table_MVC_webXML</display-name>
4    <welcome-file-list>
5      <welcome-file>index.html</welcome-file>
6      <welcome-file>index.htm</welcome-file>
7      <welcome-file>index.jsp</welcome-file>
8    </welcome-file-list>
9      <servlet>
10         <servlet-name>converter</servlet-name>
11         <servlet-class>TemperatureConversionServlet</servlet-class>
12     </servlet>
13     <servlet-mapping>
14         <servlet-name>converter</servlet-name>
15         <url-pattern>/Convert</url-pattern>
16     </servlet-mapping>
17 </web-app>
```

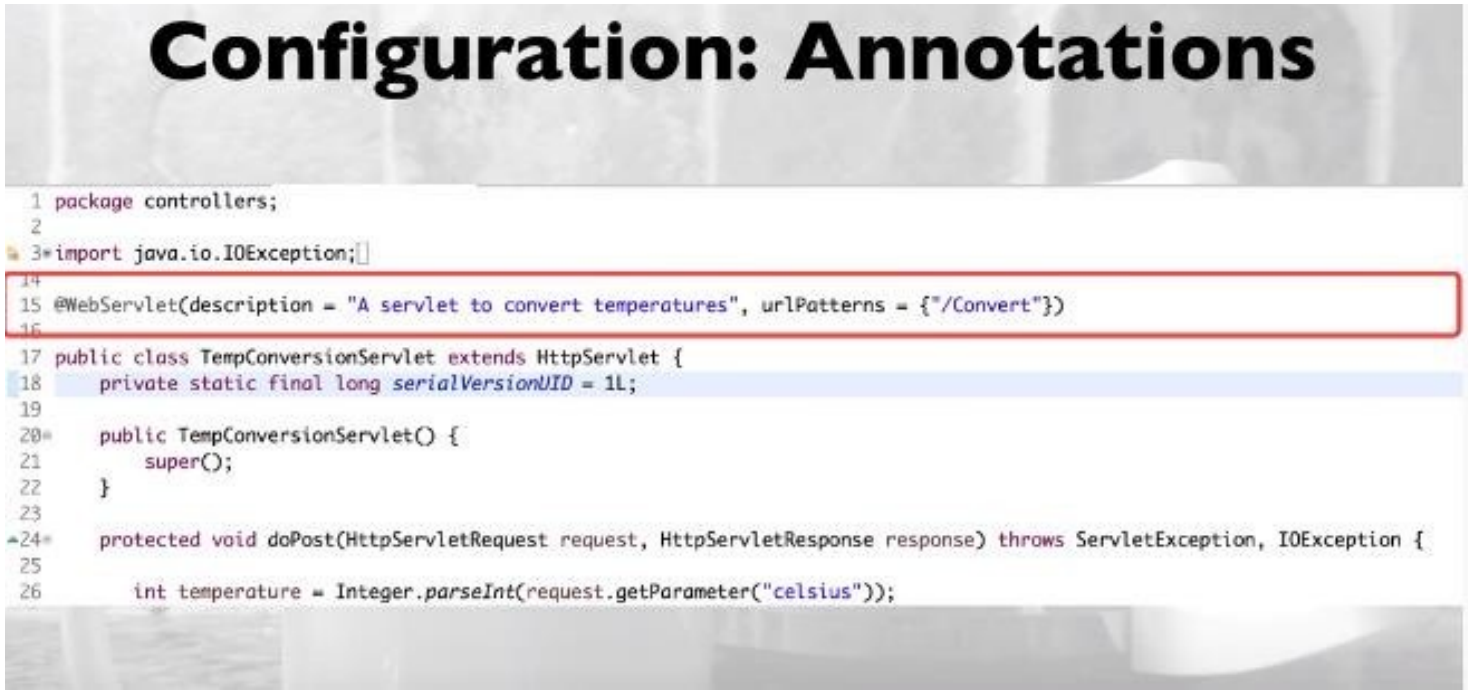
Cấu hình: Annotations

- Bắt đầu từ phiên bản Servlet API 3.0 , sử dụng một API mới

  javax.servlet.annotation

- Cung cấp các dạng annotation thay thế việc khai báo trong web.xml

- Phiên bản tomcat 7 trở lên



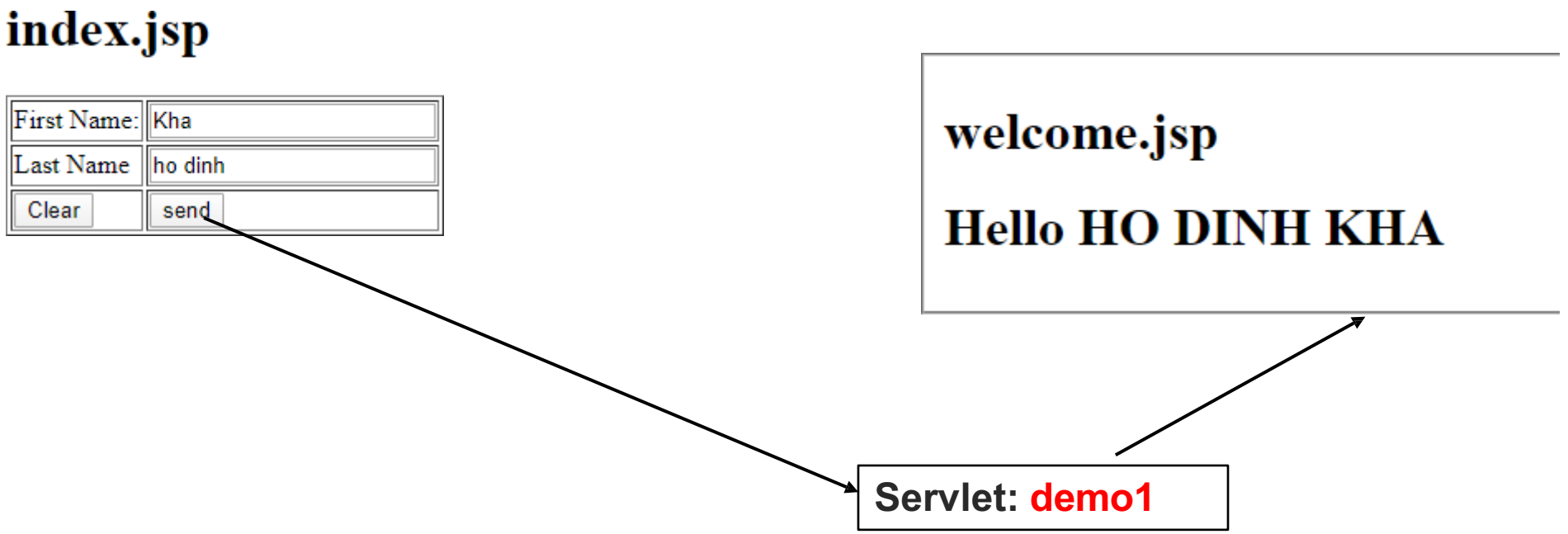
Công nghệ và công cụ phát triển Wen app

- Web server phổ biến
  - **Tomcat 9**
  - **Glassfish**
  - **Jboss**
  - **IBM Webphere**
  - **Oracle WebLogic**

- Database Server
  - **MySql 5**
  - **Oracle**
  - **SQL server**

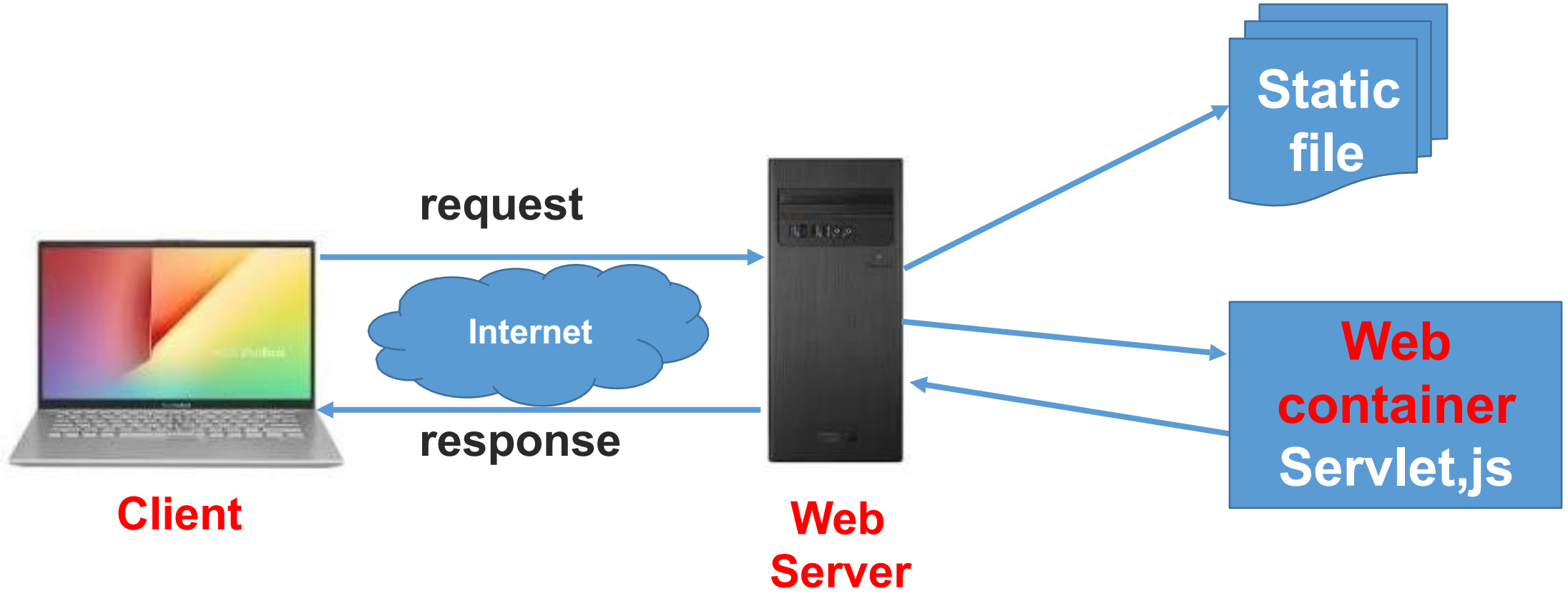
- **IDE**
  - **Apache NetBean**
  - **Eclipse**
  - **IntelliJ IDEA**

CaseStudy Tạo Web App



Java Servlets

Dynamic Web Model



Servlet

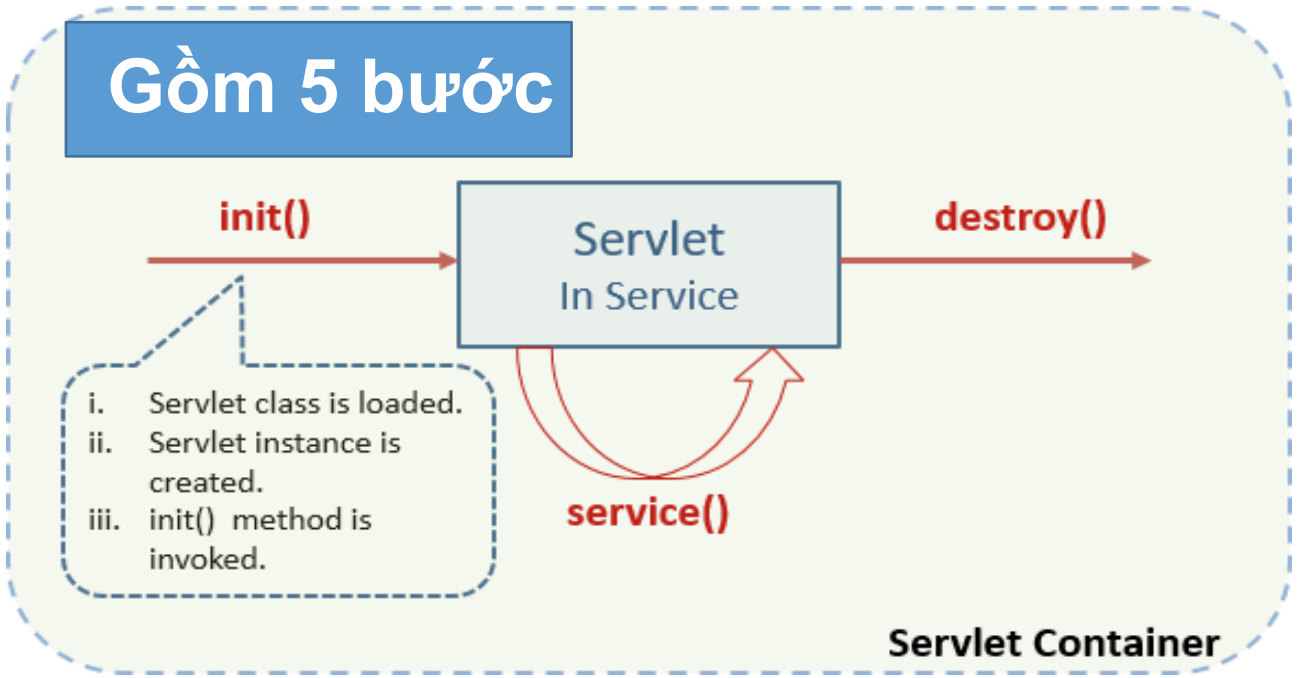
- Công nghệ Servlet dùng tạo ra các ứng dụng web động
- Công nghệ Servlet mạnh mẽ và có thể mở rộng

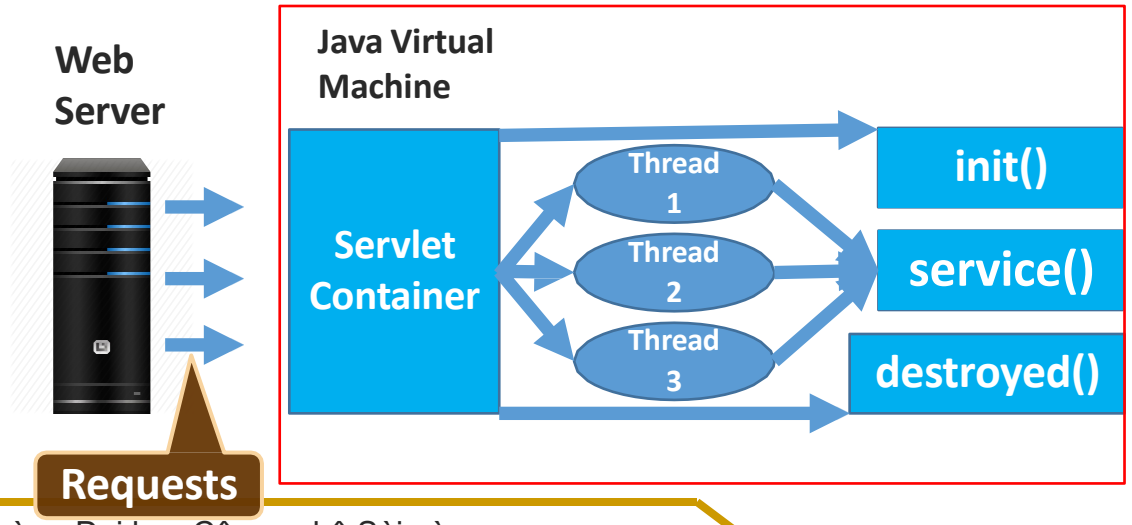
Server side: back end	Client side: Front end
Php	Java script
ASP.NET	VBScript
C++	HTML
Java và JSP	CSS
Python	AJAX
Ruby và Rails	JQuery
……	….

Servlet?

- Servlet là các đối tượng Java, mở rộng chức năng của một HTTP server, do đó được viết bằng ngôn ngữ java
- Cơ chế hoạt động theo mô hình CGI mở rộng
- Chương trình servlet:
  - Thường thừa kế class HttpServlet. Không có phương thức main
  - Phải được dịch ra dạng byte-code và khai báo với web server hoặc sử dụng tiền sử lý

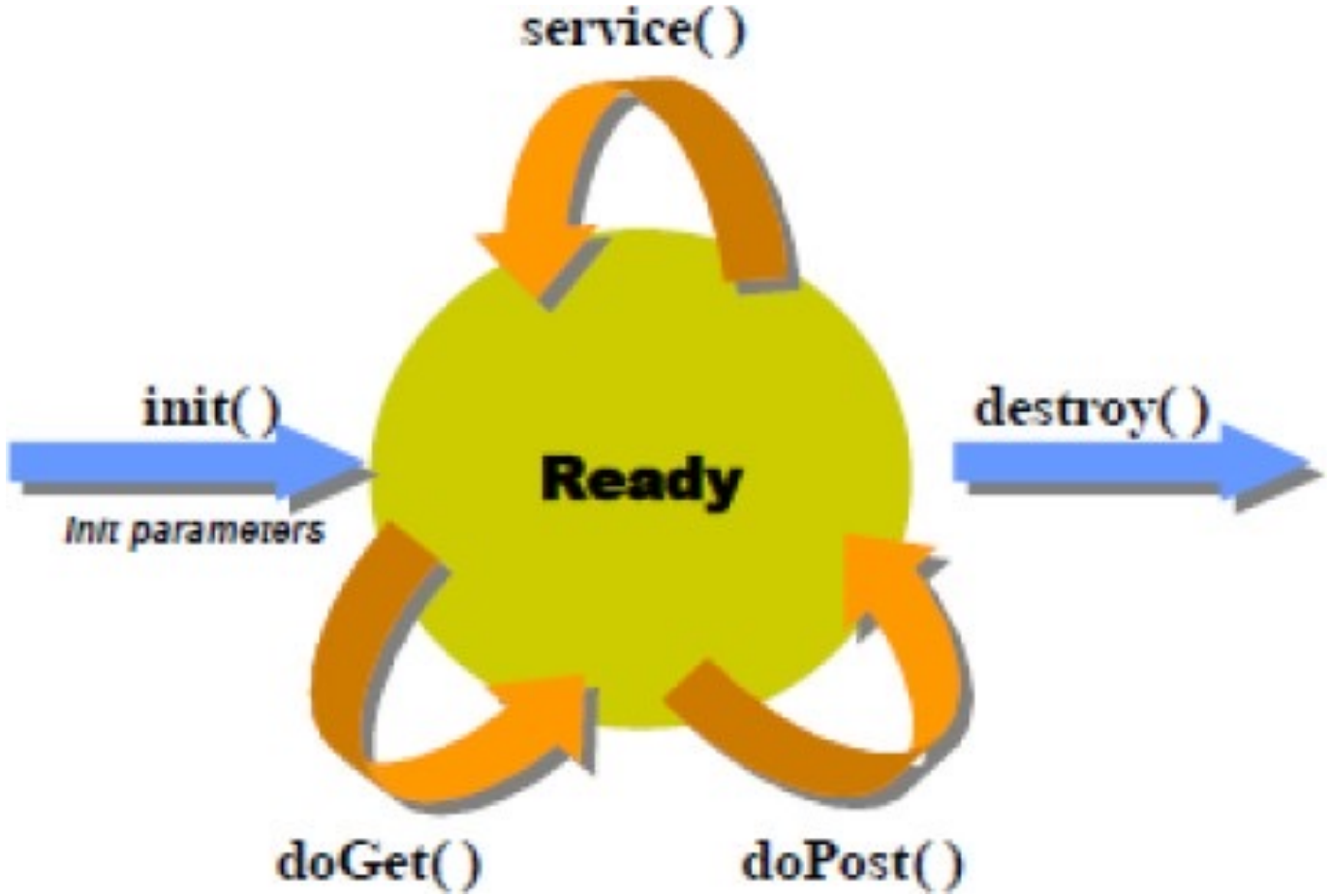
Vòng đời của Servlet





- **init()**: được gọi sau khi đối tượng servlet được tạo.
- **service()**: được viện dẫn mỗi lần một yêu cầu được nhận
- **destroy()**: được viện dẫn trước khi servlet bị hủy
- **getServletConfig()**: thông số cấu hình
- **getServletInfo()**: thông tin servlet

Các phương thức trong vòng đời



Phương thức init()

- Được nạp chỉ một lần sau khi servlet được tạo
  Ví dụ: Tạo kết nối database
- Dùng khởi tạo Servlet

```
public void init(ServletConfig config)
                        throws ServletException
{
   //initialization…
}
```

Đối tượng được dùng bởi trình chứa, truyền thông tin tới Servlet trong khi khởi tạo

Phương thức Service()

- Phương thức chính thực hiện hoạt động tác vụ
- Trình chứa gọi Service() tiếp nhận các yêu cầu từ client và xuất các đáp ứng tới client.
- Mỗi lần server gọi servlet nó sẽ gọi phương thức service().

Syntax:

```
public void service(ServletRequest request,
                    ServletResponse response)
          throws ServletException, IOException
{
      …
      …
}
```

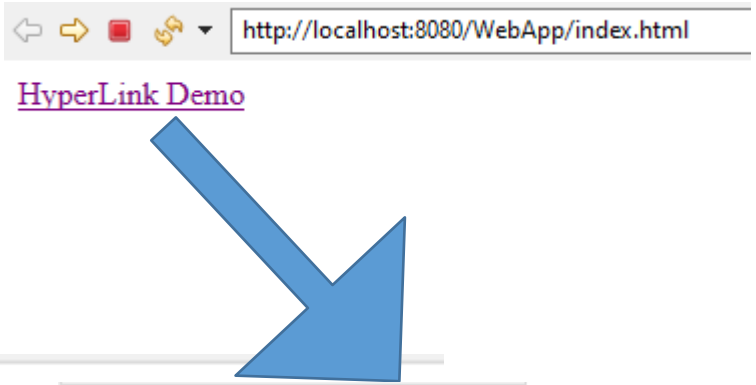
- Kiểm tra kiểu HTTP request (GET,POST,PUT….) và gọi phương thức tương ứng doGet, doPost, doPut…
- doGet, doPost dùng phổ biến nhất.

Demo Servlet

- Tạo Dynamic Web Application
- doGet,doPost
- Servlet Config
- Servlet Context
- HttpServletRequest
- Inteface RequestDispatcher
- sendRedirect()
- Session Manager

Lập trình gọi Servlet từ HTML

- Index.htm




```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="ISO-8859-1">
5  <title>Insert title here</title>
6  </head>
7  <body>
8   <a href="/WebApp/MyServlet">HyperLink Demo</a>
9  </body>
10 </html>
```

URL servlet

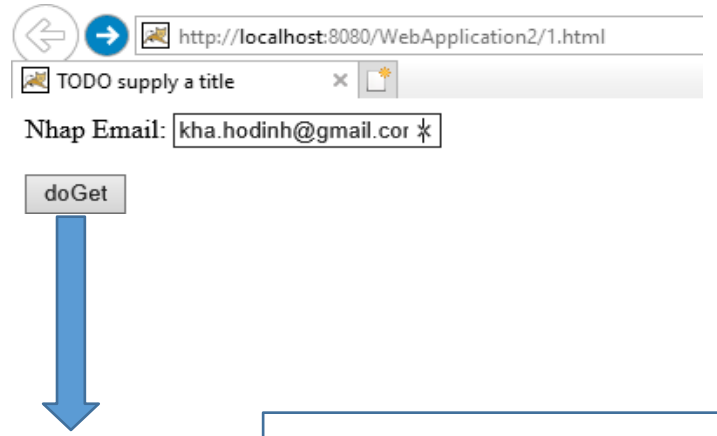
Context Path

Phương thức doGet và doPost

doGet	doPost
- Dữ liệu được gởi như một phần header có kích thước giới hạn - Hiển thị trong address bar - Truyền qua mạng khi submit bởi người dùng	- Dữ liệu được gởi như một phần body có kích thước không giới hạn - Không hiển thị trong address bar - Truyền qua mạng được mã hóa

Sử dụng doGet()

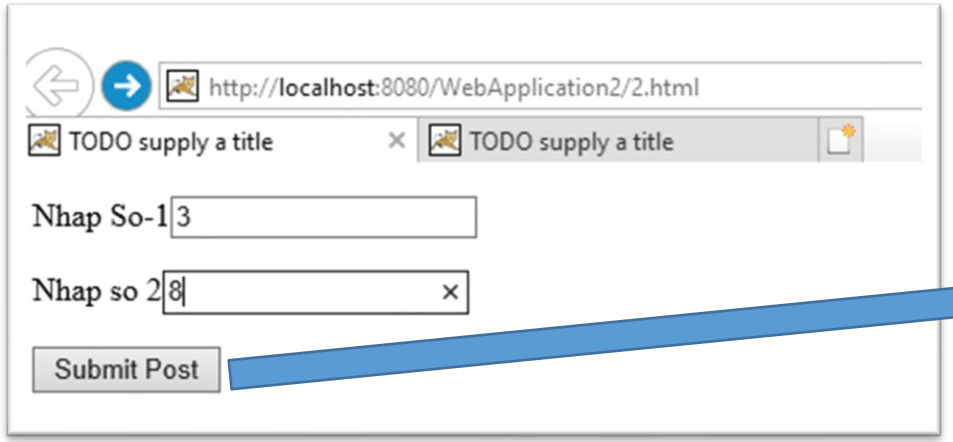
```html
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <form action="/WebApplication2/doGetDemo">
            Nhap Email: <input type="text" name="email" value="" />
            <p> <input type="submit" value="doGet" /></p>
        </form>
    </body>
</html>
```

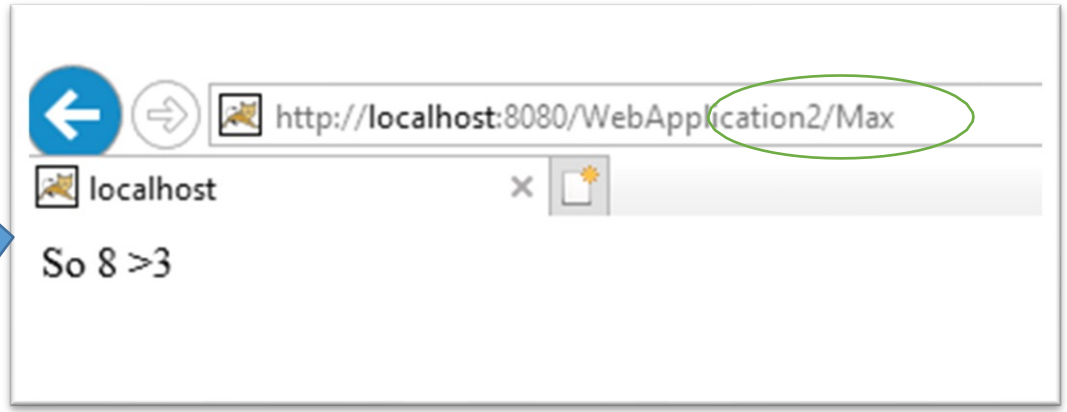



```java
@WebServlet("/doGetDemo")
public class doGetDemo extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public doGetDemo() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        PrintWriter out=response.getWriter();
        String email=request.getParameter("email");
        out.println("Email:"+email);

    }
```

doPost()

```
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <form action="/WebApplication2/Max" method="POST">
            <p>Nhap So-1<input type="text" name="no1" value="" /></p>
            <p>Nhap so 2<input type="text" name="no2" value="" /></p>
            <p><input type="submit" value="Submit Post" /></p>
        </form>
    </body>
</html>
```





Inteface RequestDispatcher

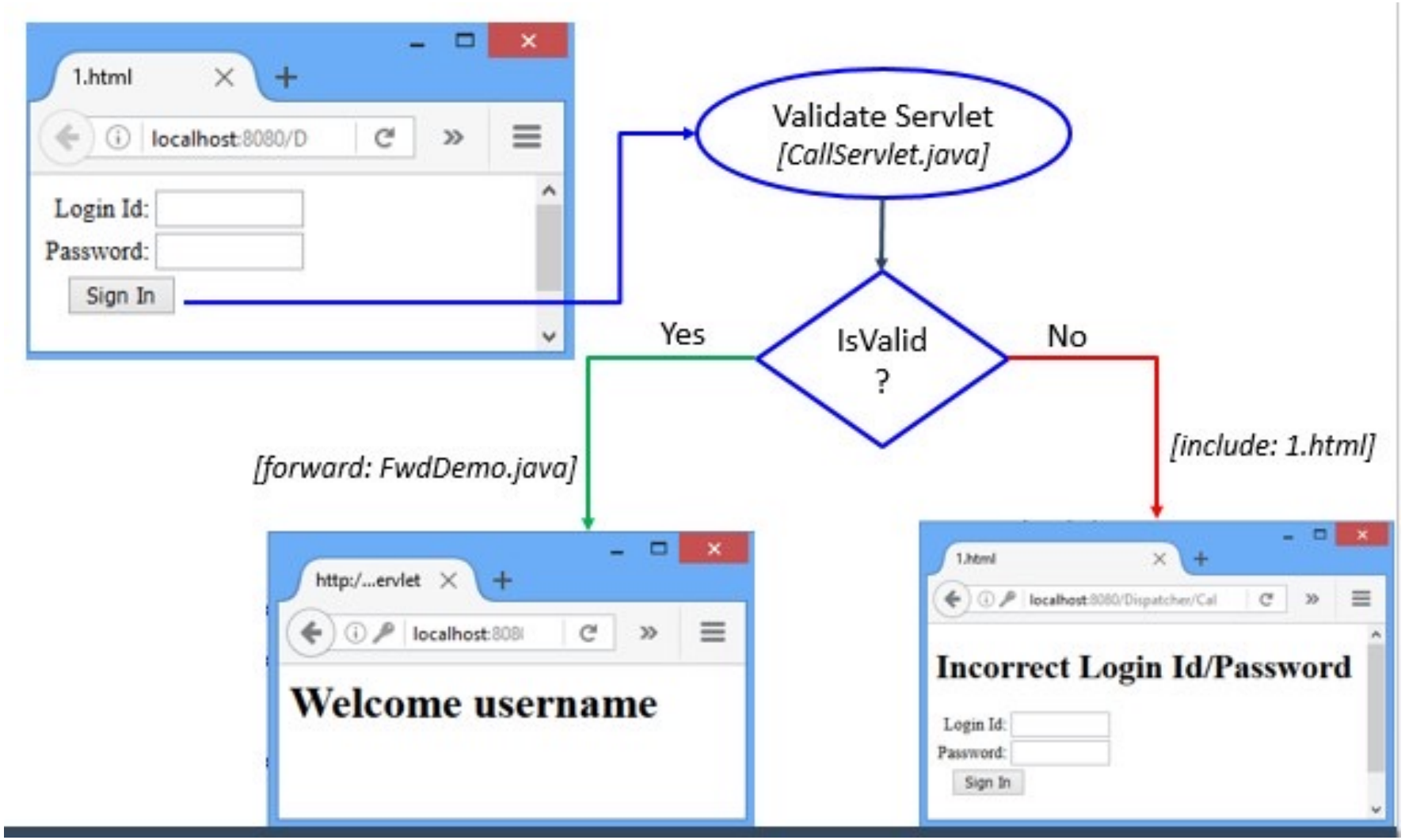
- Cung cấp cơ chế điều vận các yêu cầu tới resource khác (servlet, html,jsp,…..)
- Còn được dùng để bao hàm nội dung của resource khác
- Là một cách để cộng tác giữa các resource

Tên method	Ý nghĩa	Cú pháp
void forward(ServletRequest req, ServletRespone res)	Chuyển hướng request từ servlet này tới resource khác	ReQuestDispatcher rd=request.getRequestDispatcher(“resource”) rd.forward(request,respone)
void include(ServletRequest req, ServletRespone res)	Bao hàm nội dung của resource(servlet,jsp, html…) trong respone	ReQuestDispatcher rd=request.getRequestDispatcher(“resource”) rd.include (request,respone)

Inteface RequestDispatcher

Tên method	Ý nghĩa	Cú pháp
void forward(ServletRequest req, ServletRespone res)	Chuyển hướng request từ servlet này tới resource khác	ReQuestDispatcher rd=request.getRequestDispatcher(“resource”) rd.forward(request,respone)
void include(ServletRequest req, ServletRespone res)	Bao hàm nội dung của resource(servlet,jsp, html…) trong response	ReQuestDispatcher rd=request.getRequestDispatcher(“resource”) rd.include (request,respone)

Ví dụ RequestDispatcher



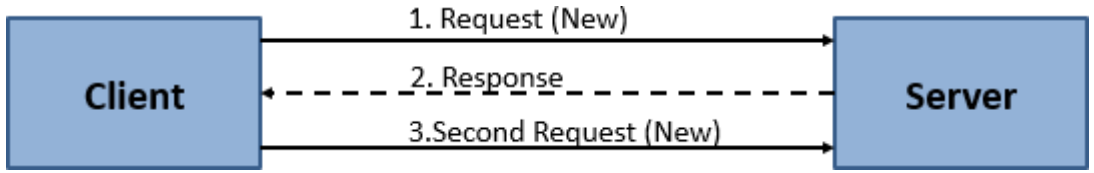
sendRedirect()

Phương thức này của interface HTTPServletRespone được dùng chuyển hướng respone tới resource khác

- respone.sendRedirect("http://stu.edu.vn");
- respone.sendRedirect("/1.html");
- respone.sendRedirect("http://localhost:8080/1.html");

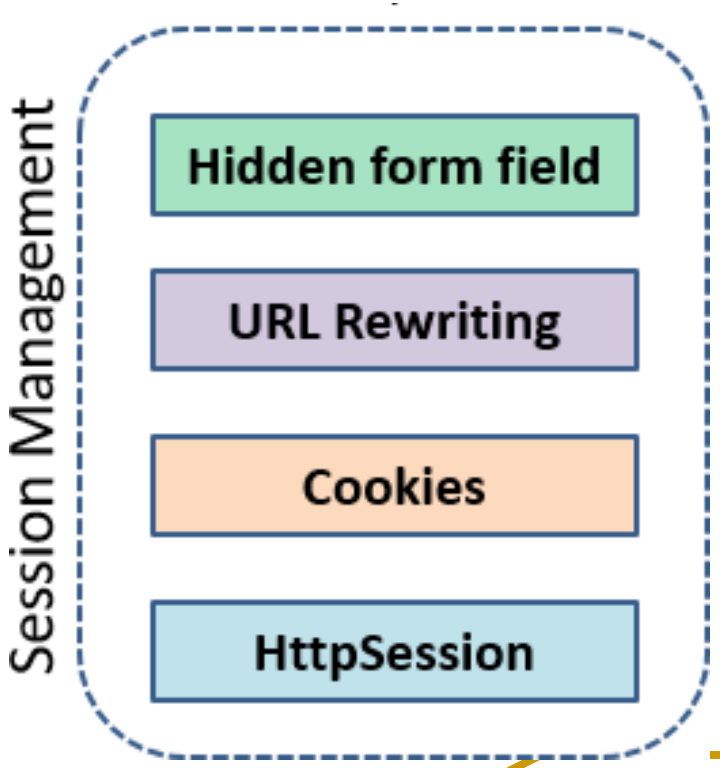
Session Manager

- **Session?**



- **Phương pháp quản lý Seesion**

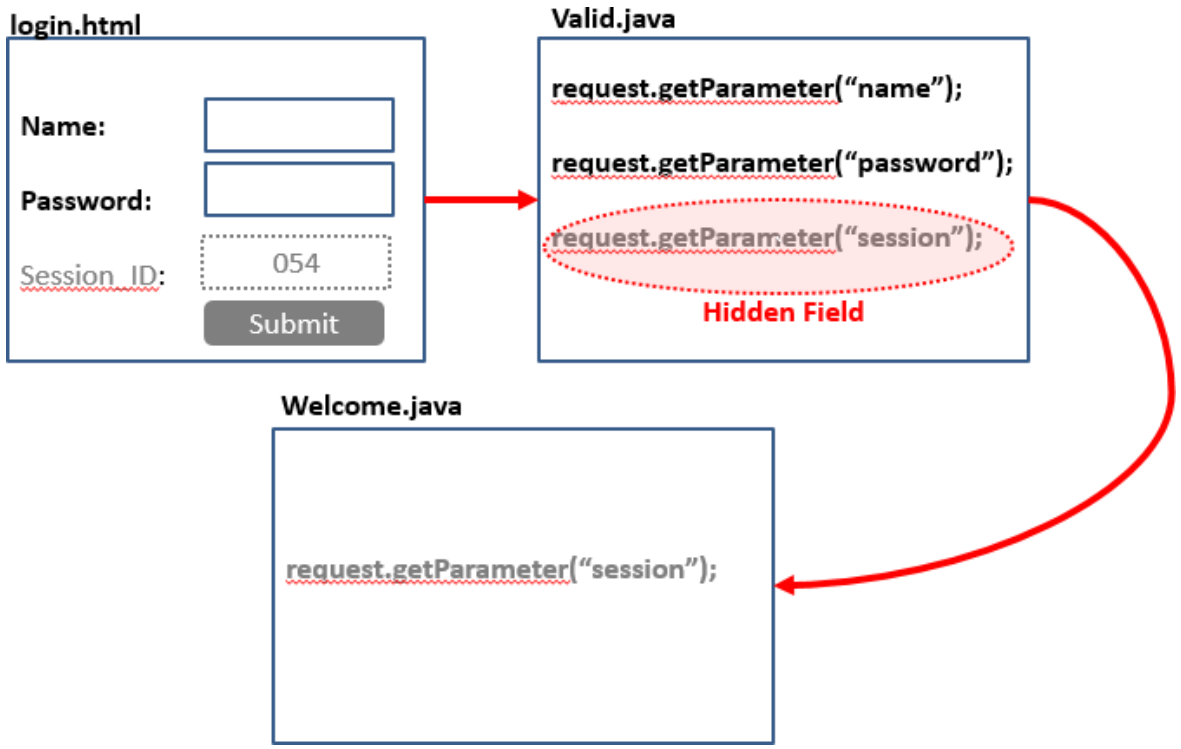
CASE STUDY



Session Manager

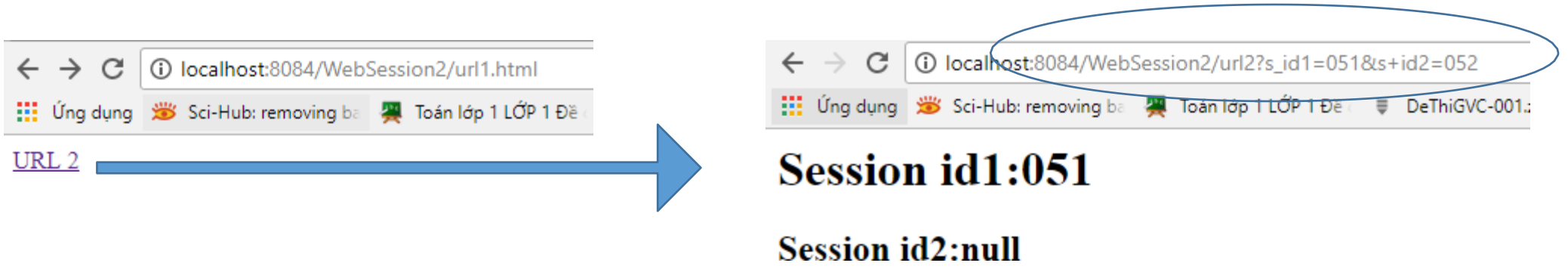
Session Manager: Hidden Form field Project: WebSession1

- <input type="hidden" name="session_id" value="054" />



Session Manager: URL ReWriting Project: WebSession2

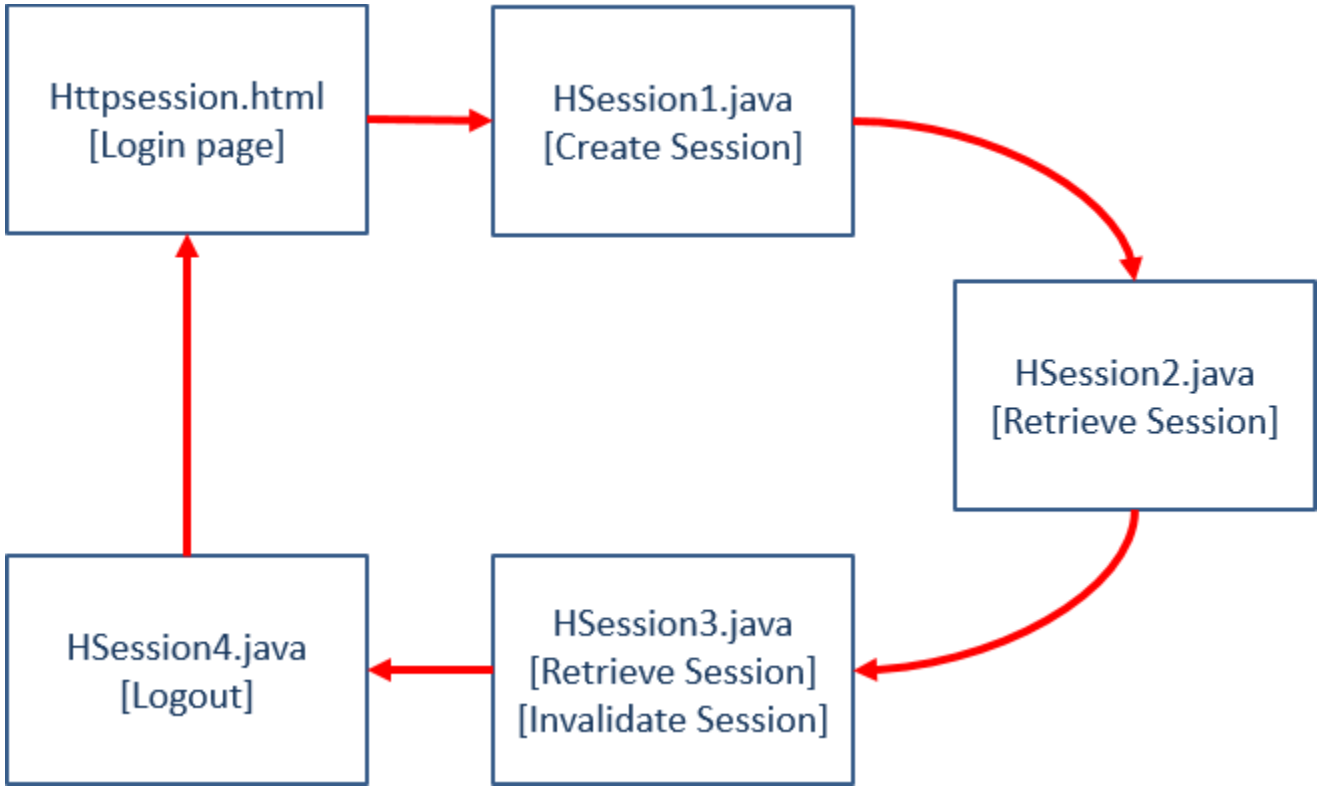
- URL?name1=value1&name2=value2



Session Manager: Cookie Project: WebSession3

- **Tạo**
  - **Cookie c=new Cookie(“session_id”,”054”);**
  - **response.addCookie(c);**
- **Truy xuất:**
  - **Cookie c[]=request.getCookies();**
  - **c[i].getName(); c[i].getValue();**

Session Manager: HttpSession Project: WebSession4

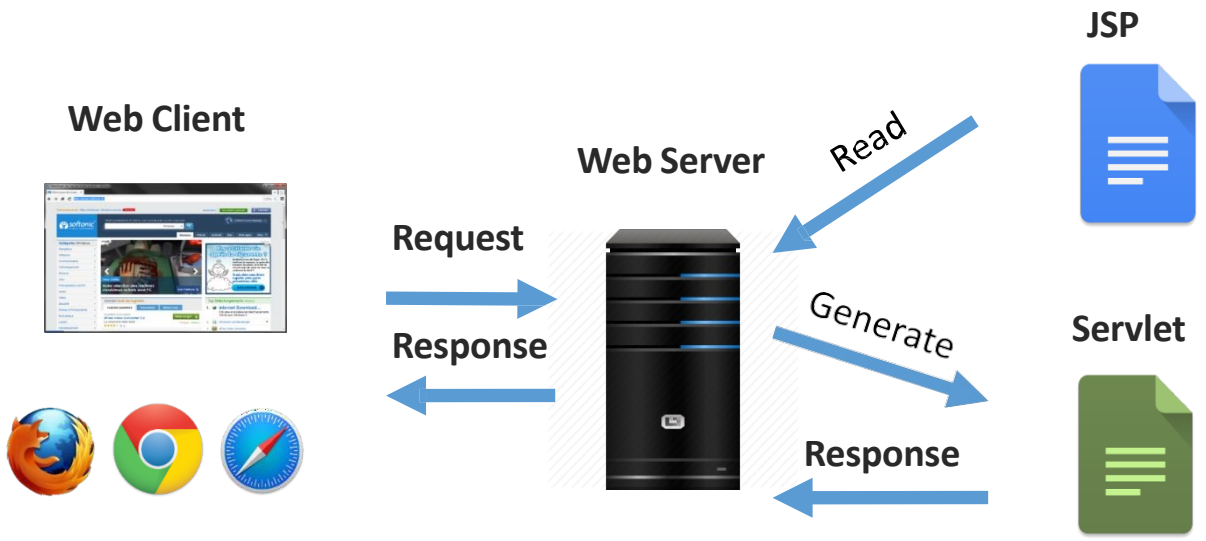


JSP
Java Server Pages

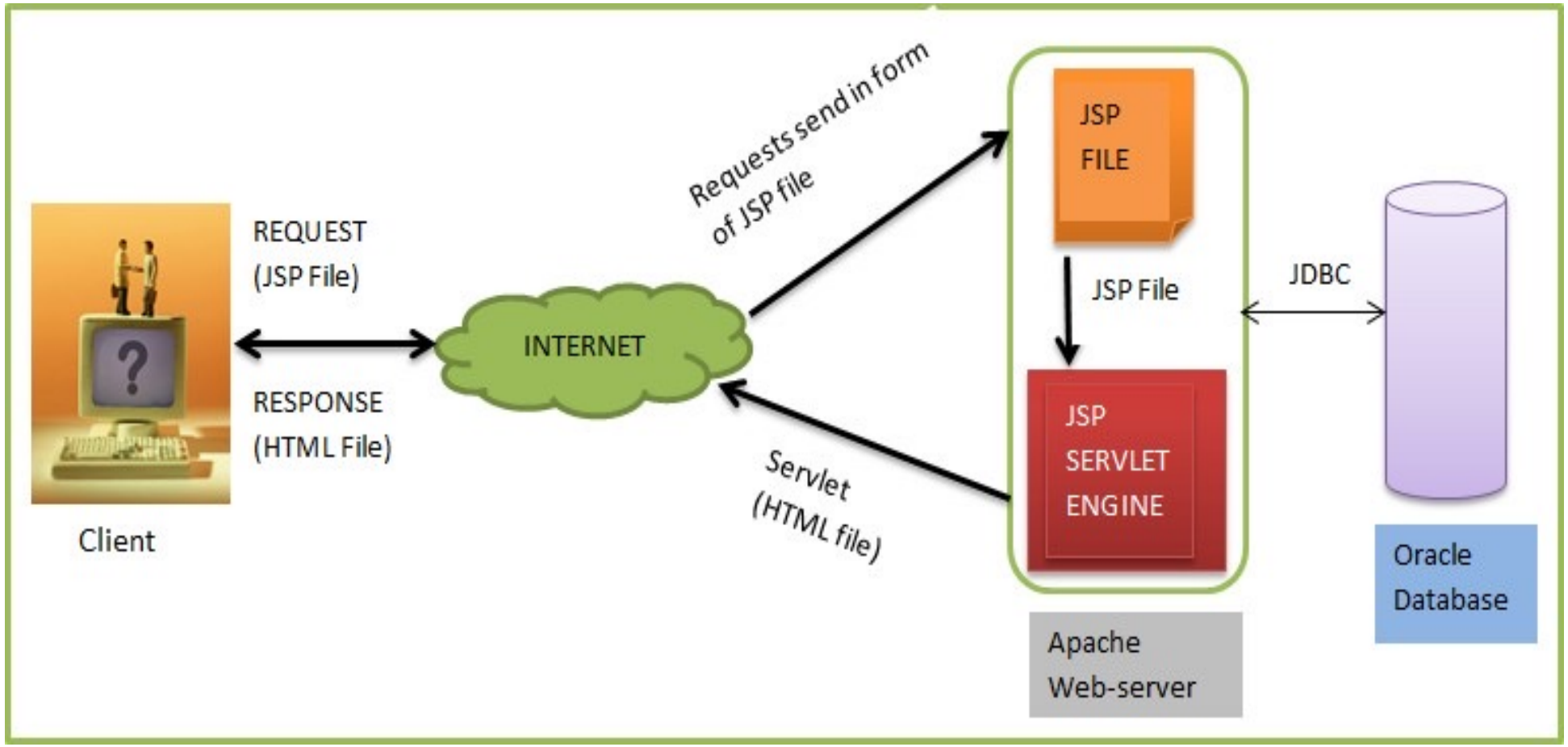
Trường Đại học Công nghệ Sài gòn
Khoa Công nghệ Thông tin

Tổng quan JSP

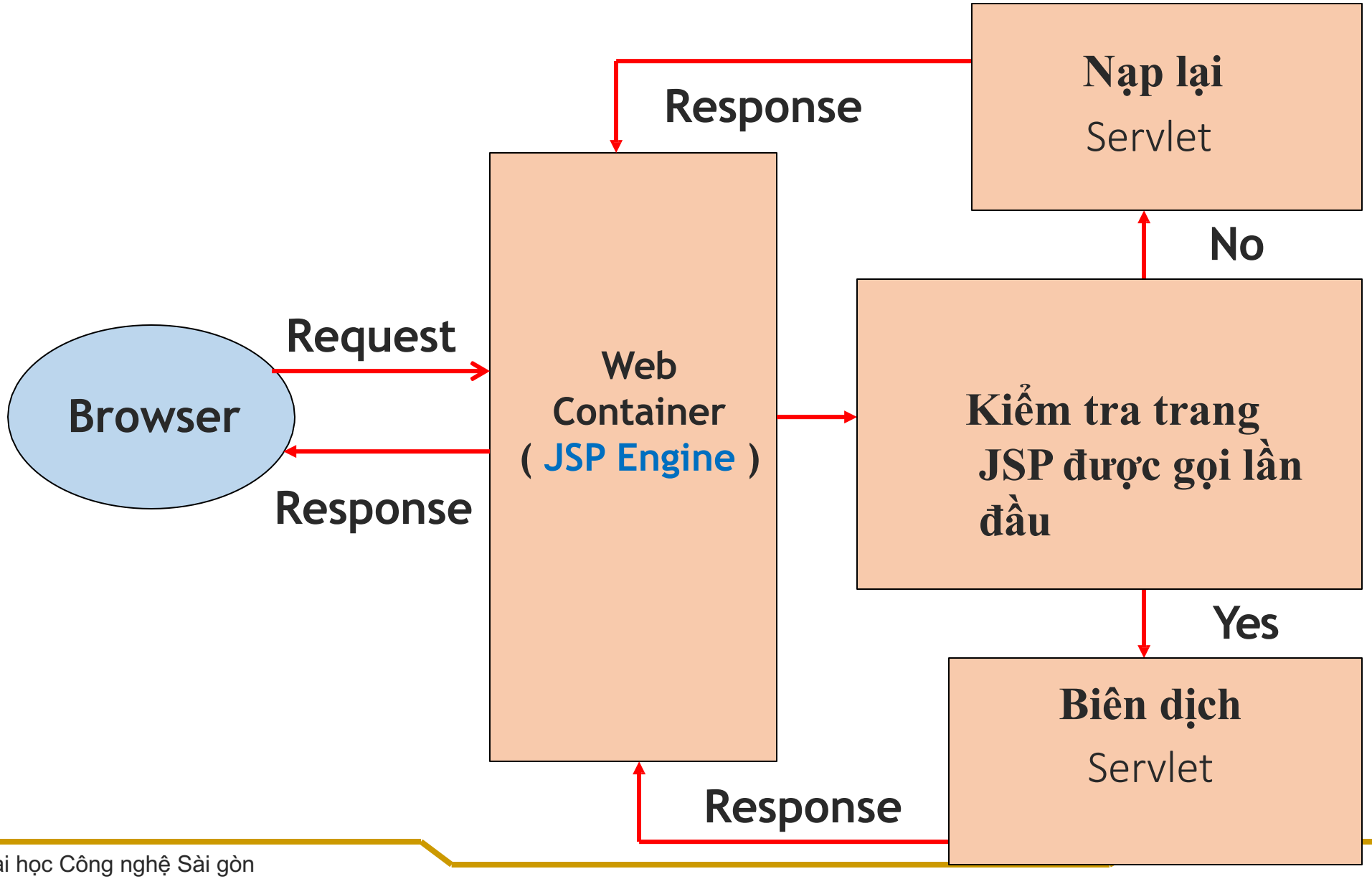
- Công nghệ cho phép phát triển các trang web hổ trợ nội dung động
- Cho phép chèn java code vào HTML page thông qua thẻ JSP
- Có phần mở rộng *.jsp



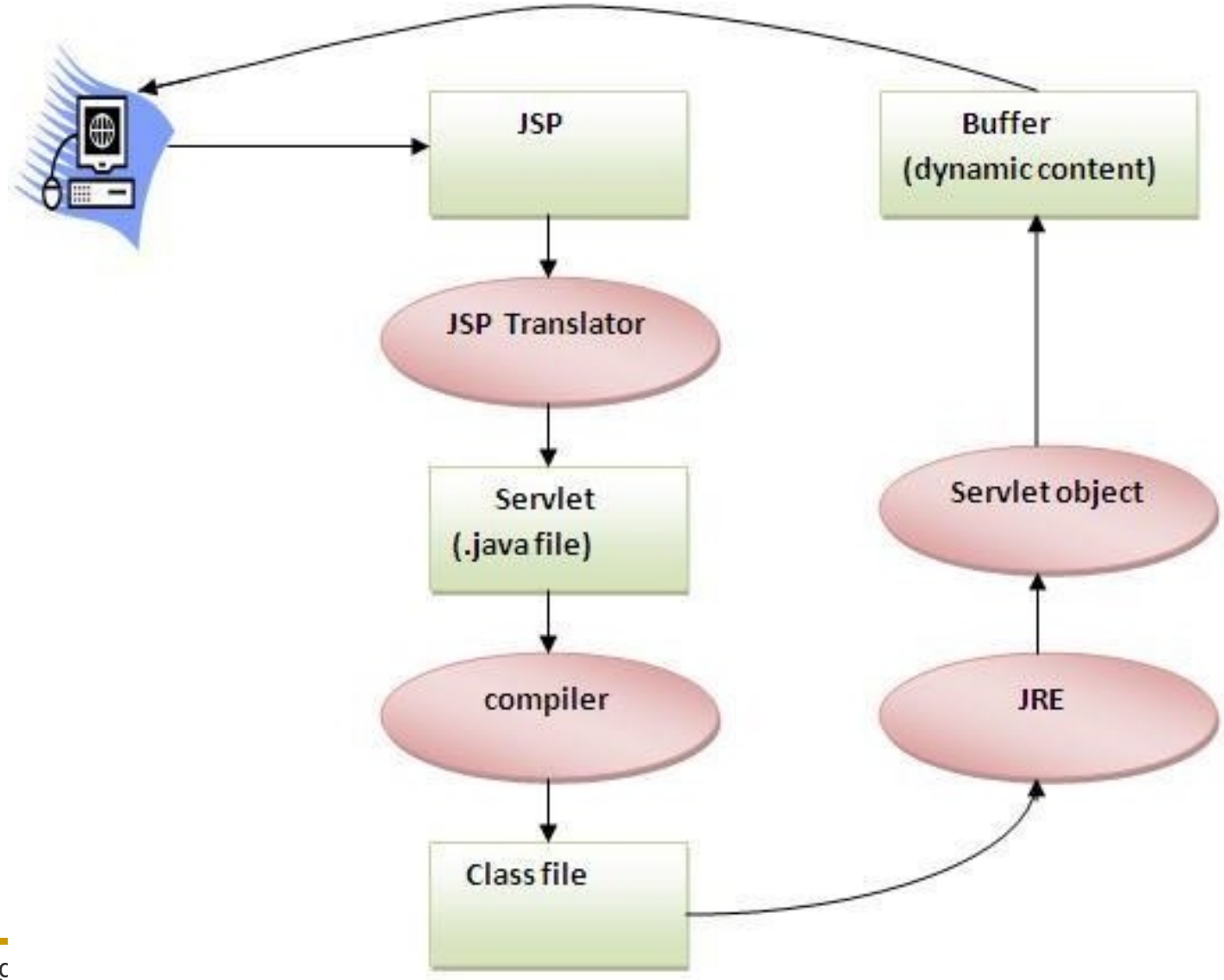
JSP - Architecture



Chu kỳ Request-Response JSP Page



Chu kỳ Request-Response JSP Page



JSP Scripting elements

Trường Đại học Công nghệ Sài gòn
Khoa Công nghệ Thông tin

JSP comments <%-- JSP comment--%>

- Tương tự HTMLcomments
  - <!-- HTML comment -- >
- JSP comments không gởi tới trình duyệt

Index.jsp

```
<html>
        <head>
                <title> HTML and JSP Comments </title>
        </head>
        <body>
                <h2> Comments </h2>
                <!-- This HTML Comment-visible in the page source -->
                <%-- This JSP comment-Not visible in the page source --%>
        </body>
</html>
```

Ví dụ Trang JSP đơn giản

```
<html>
        <head><title> ☹ Random JSP Example ☺ </title></head>

<body> <center>
    <%
        double num = Math.random();
        if (num > 0.95) {
    %>
        <h2>You are having a lucky day!</h2>  <p>( <%= num %>
    <%    )</p>
        }
        else {
    %>
        <h2>Well, life goes on ... </h2>          <%= num %> )</p>
    <%    <p>(
        }
    %>

    <a href=" <%=  request.getRequestURI() %> "> <h3>  Try Again ☺
    </h3></a>

</center></body>

</html>
```

Thẻ JSP Scripting

- Sử dụng chèn code Java vào HTML
- Các dạng thẻ scripting:

1. Declaration tag

   <%! Khai báo biến hoặc phương thức %>

   Ví dụ <%! int data=50; %>

2. Expression tag

   <%= biểu thức %>

   Ví dụ: Current Time: <%=

   java.util.Calendar.getInstance().getTime() %>

Thẻ JSP Scripting

- Sử dụng chèn code Java vào HTML
- Các dạng thẻ scripting:

3. Directive tag

   <%@ directive attribute="value" %>

4. Scriptlet tag

   <% java source code; %>

5. Action tag

   <jsp:action_name attribute = "value" />

   Ví dụ <jsp:include page = "date.jsp" flush = "true" />

Ví dụ: Thẻ JSP Declaration

test1.jsp

```
<html>
<body>
        <%! int data=50; %>
        <%= "Value of the variable is:"+ data %>
</body>
</html>
```

test2.jsp

```
<html>
<body>
        <%!
                int cube(int n){
                        return n*n*n*;
                }
        %>
        <%= "Cube of 3 is:“ +cube(3)  %>
</body>
</html>
```

Ví dụ : Thẻ JSP expression

test1.jsp

```
<html>
<body>
            <%= "welcome to jsp" %>
</body>
</html>
```

test2.jsp

```
<html>
<body>
            Current Time: <%=  java.util.Calendar.getInstance().getTime()
 %>         Current Date and Time is : <%=  new java.util.Date()  %>
</body>
</html>
```

Ví dụ: Thẻ JSP expression

index.jsp

```
<html>
<body>
<form        action=“welcome.jsp“ >
          <input type  =“text"       name="uname“
>
          <input type  =“submit"   value="go">
          <br>
</form>
</body>
</html>
```

welcome.jsp

```
<html>
<body>
          <%= “Welcome “ + request.getParameter("uname") %>
</body>
</html>
```

Directive tag (<%@ directive…. %>)

- Chuyển tới trình chứa yêu cầu cách thức chuyển jsp vào servlet
- Có 3 dạng:
  1. page : Xử lý thông tin trang
  2. Include : Các tập tin đính kèm
  3. Tag library: thư viện sử dụng

Ví dụ: JSP Directive

- **Import java packages**

  <%@ page import="java.util.*, java.sql.*" %>

- **Multiple import statements**

  <%@ page import="java.util.*" %>

  <%@ page import="java.sql.*" %>

- **Include file tại thời điểm thông dịch**

  <%@ include file="header.html" %>

```
<%@page import="java.util.Date” %>
Today is: <%= new Date() %>

<%@  include=“test.html" %>
Today is: <%= new Date() %>
```

Ví dụ: Thẻ JSP Scriptlet

index.html

```
<html>  <body>
<form     action="welcome.jsp“ >
          <input type ="text" name="uname“ >
          <input type ="submit“ value="go"> <br>

</form>
</body>  </html>
```

welcome.jsp

```
<html>  <body>
          <%   String name = request.getParameter("uname");

          out.print("welcome :“ + name );
          %>
</body>  </html>
```

Action Tag

Có 2 vai trò chính của thẻ action:

1. **Cho phép sử dụng đối tượng Javabean tại server**
2. **Truyển điều khiển giữa các trang**

Ví dụ: <jsp:useBean> ... </jsp:useBean>

Các thẻ JSP Action

JSP Action Tags	Description
jsp:forward	Chuyển hướng request và response tới tài nguyên khác.
jsp:include	includes tài nguyên.
jsp:useBean	Tạo hay định vị đối tượng bean
jsp:setProperty	Đặt giá trị thuộc tính cho bean
jsp:getProperty	Xuất giá trị thuộc tính bean.
jsp:plugin	Nhúng thành phần khác (ví dụ applet)
jsp:param	Đặt giá trị tham số chủ yếu được dùng trong forward và include.
jsp:fallback	Sử dụng xuất thông báo nếu plugin đang hoạt động, dùng trong thẻ jsp:plugin.

Ví dụ (jsp:forward)

first.jsp

```
<html>
<body>
        <h2>this is the first jsp page</h2>
        <jsp:forward page="second.jsp" />
</body>
</html>
```

second.jsp

```
<html>
<body>
                <h2>this is the second jsp page</h2>
                <% out.print("Today is:" + java.util.Calendar.getInstance().getTime());
%>
</body>
</html>
```

Ví dụ: (jsp:include)

first.jsp

```
<html>
<body>
        <h2>this is the first jsp page</h2>
        <jsp:include page="second.jsp" />
        <h2>end of first page</h2>
</body>
</html>
```

second.jsp

```
<html>
<body>
                <h2>this is the second jsp page</h2>
                <% out.print("Today is:" + java.util.Calendar.getInstance().getTime());
%>
```

Ví dụ: (jsp:param)

first.jsp

```
<html>
<body>
        <h2>this is the first jsp page</h2>
        <jsp:forward page="second.jsp" />
        <jsp:param name="name" value="ABC" />
</body>
</html>
```

second.jsp

```
<html>
<body>
                <h2>this is the second jsp page</h2>
                <% out.print("Today is:" + java.util.Calendar.getInstance().getTime());  %>
                <%= request.getParameter("name") %>
```

JSP Implicit Objects

Trường Đại học Công nghệ Sài gòn
Khoa Công nghệ Thông tin

JSP Implicit Objects

- **Được tạo bởi web container và sử dụng trên tất cả trang JSP**

Object/Variable	Type
out	javax.servlet.jsp.JspWriter
request	javax.servlet.http.HttpServletRequest
response	javax.servlet.http.HttpServletResponse
config	javax.servlet.http.ServletConfig
application	javax.servlet.http.ServletContext
session	javax.servlet.http.HttpSession
pageContext	javax.servlet.jsp.PageContext
page	javax.lang.Object
exception	javax.lang.Throwable

Out Object

- **Xuất dữ liệu tới buffer**

index.jsp

```
<html>
<body>
        <%   out.print(2*5);   %>
</body>
</html>
```

index.jsp

```
<html>
<body>
        <% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
</body>
</html>
```

Request Object

- Được tạo bởi trình chứa web cho mỗi lần yêu cầu
- Đối tượng JSP request là một đới tượng dạng HttpServletRequest
- Cung cấp thông tin kết hợp với tử yêu cầu
- Thường được sử dụng tìm giá thông thông số
- Được tạo từ trình chứa web
- Các thông tin đi kèm request:
  - parameter,
  - header information,
  - remote address,
  - server name,
  - server port,
  - content type,
  - character encoding etc.

index.html

```
<html> <body>
<form action="welcome.jsp“ >
      <input type="text" name="uname“ >
      <input type ="submit"   value="go"> <br>
</form>
</body> </html>
```

welcome.jsp

```
<html> <body>
     <%  String name = request.getParameter("uname");
           out.print("welcome :“ + name );
     %>
</body> </html>
```

JSTL (JavaServer Pages Standard Tag Library)
EL (Expression Language)

Trường Đại học Công nghệ Sài gòn
Khoa Công nghệ Thông tin

Tổng quan

- JSTL cung cấp 2 tình huống
  - Một tập các thẻ rất mạnh
- Hổ trợ các chức năng phổ biến trên nhiều ứng dụng web
- Sử dụng đơn giản: hướng cú pháp giống html

- EL Expression language
- Được sử dụng trong JSTL để truy xuất thuộc tính javaBean
- Tương thích phiện bản JSP 2.0

Tổng quan

- Thẻ JSTL bao gồm các nhóm sau:
  - Core Tags: Nhóm thẻ cơ bản
  - Formatting tags: Nhóm thẻ định dạng
  - SQL tags: Nhóm thẻ SQL
  - XML tags: Nhóm thẻ XML
  - JSTL Functions: Nhóm hàm JSTL

EL Expression language

EL là ngôn ngữ script đơn giản

- **Giúp việc truy xuất các thành phần java đơn giản hơn**
- **Cú pháp đơn giản hướng biểu thức**
- **Tự động ép kiểu**

Cú pháp EL

- **Biểu thức**
  - **Thông dịch khi «run time»**
  - **Cú pháp: ${biểu thức}**
- **Toán tử**
  - **. : truy xuất thuộc tính đối tượng**
  - **[ ]: truy xuất phần tử trong tập**
  - **Số học: +,-,/,*,%**
  - **So sánh: ==,!=,<,>,<=,>=,eq,ne,lt,gt,le,ge**
  - **Luận lý: !,&&,||, and, or, not, empty**

Toán tử cơ bản trong EL

Toán tử	Miêu tả
.	Truy cập một đặc tính của Bean hoặc Map entry
[]	Truy cập một phân tử mảng hoặc List
( )	Nhóm một subexpression để thay đổi thứ tự ước lượng
+	Phép cộng
-	Phép trừ hoặc phủ định một giá trị
*	Phép nhân
/ hoặc div	Phép chia
% hoặc mod	Phép chia lấy phần dư
== hoặc eq	Kiểm tra có bằng hay không
!= hoặc ne	Kiểm tra tính không bằng
< hoặc lt	Kiểm tra tính nhỏ hơn
> hoặc gt	Kiểm tra tính lớn hơn
<= hoặc le	Kiểm tra tính nhỏ hơn hoặc bằng
>= hoặc ge	Kiểm tra tính lớn hơn hoặc bằng
&& hoặc and	Phép AND logic
|| hoặc or	Phép OR logic
! hoặc not	Phần bù Boolean một ngôi
empty	Kiểm tra các giá trị biến rỗng

Cài đặt thư viện JSTL

- Để sử dụng Standard Taglib từ Jakarta Taglib, đơn giản bạn chỉ cần sao chép các JAR file(jstl...jar) vào thư mục 'lib' của project
- Để sử dụng JSTL trong trang JSP cần khai báo thẻ <taglib> trên cùng của mỗi JSP.

Cài đặt thư viện JSTL

- **Nhóm core tag**:<%@ taglib prefix="c"
  uri="http://java.sun.com/jsp/html/core" %>
- **Nhóm Formatting Tags**: <%@ taglib prefix="fmt"
  uri="http://java.sun.com/jsp/html/fmt" %>
- **Nhóm SQL tag**: <%@ taglib prefix="sql"
  uri="http://java.sun.com/jsp/html/sql" %>
- **Nhóm Function tag**: <%@ taglib prefix="fn"
  uri="http://java.sun.com/jsp/html/functions" %>

Nhóm Core Tag

Thẻ	Miêu tả
Thẻ <c:out > trong JSTL	Giống <%= ... >, nhưng cho các Expression
Thẻ <c:set > trong JSTL	Thiết lập kết quả của một ước lượng Expression trong một 'scope'
Thẻ <c:remove > trong JSTL	Gỡ bỏ một biến mục tiêu (từ một biến scope cụ thể, nếu đã xác định)
Thẻ <c:catch> trong JSTL	Bắt bất kỳ Throwable mà xuất hiện trong thân của nó và trưng bày nó một cách tùy ý
Thẻ <c:if> trong JSTL	Thẻ điều kiện đơn giản, mà ước lượng phần thân của nó nếu điều kiện đã cho là true
Thẻ <c:choose> trong JSTL	Thẻ điều kiện đơn giản mà thiết lập một context cho các hoạt động điều kiện loại trừ, được đánh dấu bởi <when> và <otherwise>
Thẻ <c:when> trong JSTL	Thẻ phụ của <choose> mà include phần thân của nó nếu điều kiện được ước lượng là true

Nhóm core tag

Thẻ	Miêu tả
Thẻ <c:otherwise > trong JSTL	Thẻ phụ của <choose> mà theo sau thẻ <when> và chỉ chạy nếu tất cả điều kiện trước được ước lượng là 'false'
Thẻ <c:import> trong JSTL	Thu nhận một URL tuyệt đối hoặc quan hệ và trưng bày nội dung của nó tới hoặc trang đó, một String trong 'var', hoặc một Reader trong 'varReader'
Thẻ <c:forEach > trong JSTL	Thẻ lặp cơ bản, chấp nhận nhiều kiểu tập hợp khác nhau và hỗ trợ subsetting (chia tập con) và tính năng khác
Thẻ <c:forTokens> trong JSTL	Lặp qua các token, được phân biệt bởi các dấu phân tách (delimiter) đã cung cấp
Thẻ <c:param> trong JSTL	Thêm một parameter tới một URL của thẻ đang chứa 'import'
Thẻ <c:redirect > trong JSTL	Redirect tới một URL mới
Thẻ <c:url> trong JSTL	Tạo một URL với các tham số truy vấn tùy ý

Nhóm Formatting Tags

Thẻ	Miêu tả
<fmt:formatNumber>	Trả lại giá trị số với định dạng cụ thể
<fmt:parseNumber>	Parse biểu diễn chuỗi của một số, tiền tệ, phần trăm
<fmt:formatDate>	Định dạng một date/time bởi sử dụng Style và Pattern đã cho
<fmt:parseDate>	Parse biểu diễn chuỗi của một date/time
<fmt:bundle>	Tải một Resource Bundle để được sử dụng bởi phần thân thẻ
<fmt:setLocale>	Lưu giữ Locale đã cho trong biến cấu hình locale
<fmt:setBundle>	Tải một Resource Bundle và lưu giữ nó trong biến scope đã đặt tên hoặc biến cấu hình bundle
<fmt:timeZone>	Xác định timezone cho bất kỳ định dạng time nào hoặc parse các action được lặp trong phần thân của nó
<fmt:setTimeZone>	Lưu giữ timezone đã cung cấp biến cấu hình time zone đó
<fmt:message>	Hiển thị một thông báo đa ngôn ngữ
<fmt:requestEncoding>	Thiết lập mã hóa ký tự cho request

Nhóm SQL Tags trong JSTL

Thẻ	Miêu tả
<sql:setDataSource>	Tạo một DataSource đơn giản chỉ thích hợp cho prototype
<sql:query>	Thực thi SQL query được định nghĩa trong phần thân của nó hoặc thông qua thuộc tính sql
<sql:update>	Thực thi SQL update được định nghĩa trong phần thân của nó hoặc thông quan thuộc tính sql
<sql:param>	Thiết lập một parameter trong một lệnh SQL tới giá trị đã xác định
<sql:dateParam>	Thiết lập một parameter trong một lệnh SQL tới giá trị java.util.Date đã xác định
<sql:transaction >	Cung cấp các phần tử database action được lặp với một Connection đã chia sẻ, thiết lập để thực thi tất cả các lệnh

Nhóm JSTL Functions

<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/html/functions" %>

Hàm	Miêu tả
fn:contains()	Kiểm tra nếu một chuỗi input chứa chuỗi phụ đã cho
fn:containsIgnoreCase()	Kiểm tra nếu một chuỗi input chứa chuỗi phụ đã cho trong trường hợp không phân biệt kiểu chữ
fn:endsWith()	Kiểm tra nếu một chuỗi input kết thúc với suffix đã cho
fn:escapeXml()	Các ký tự thoát mà có thể được phiên dịch như XML markup
fn:indexOf()	Trả về index bên trong một chuỗi về sự xuất hiện đầu tiên của chuỗi phụ
fn:join()	Kết hợp tất cả phần tử trong một mảng thành một chuỗi
fn:length()	Trả về số item trong một tập hợp, hoặc số ký tự trong một chuỗi
fn:replace()	Trả về một chuỗi là kết quả của việc thay thế một chuỗi input với một chuỗi đã cho

Nhóm JSTL Functions

<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/html/functions" %>

Hàm	Miêu tả
fn:split()	Chia một chuỗi thành một mảng các chuỗi phụ
fn:startsWith()	Kiểm tra nếu một chuỗi input bắt đầu với prefix đã cho
fn:substring()	Trả về một tập con của một chuỗi
fn:substringAfter()	Trả về một tập con của một chuỗi ở sau một chuỗi phụ đã cho
fn:substringBefore()	Trả về một tập con của một chuỗi ở trước một chuỗi phụ đã cho
fn:toLowerCase()	Biến đổi tất cả ký tự của một chuỗi thành chữ thường
fn:toUpperCase()	Biến đổi tất cả ký tự của một chuỗi thành chữ hoa
fn:trim()	Gỡ bỏ các khoảng trống trắng từ hai đầu của một chuỗi

Mô hình MVC

Nội Dung

- Mô hình MVC (Model-View-Control)
- JSP / Servlet
- Cấu trúc ứng dụng web java
- Mô hình MVC trong ứng dụng web java
- Ví dụ mẫu

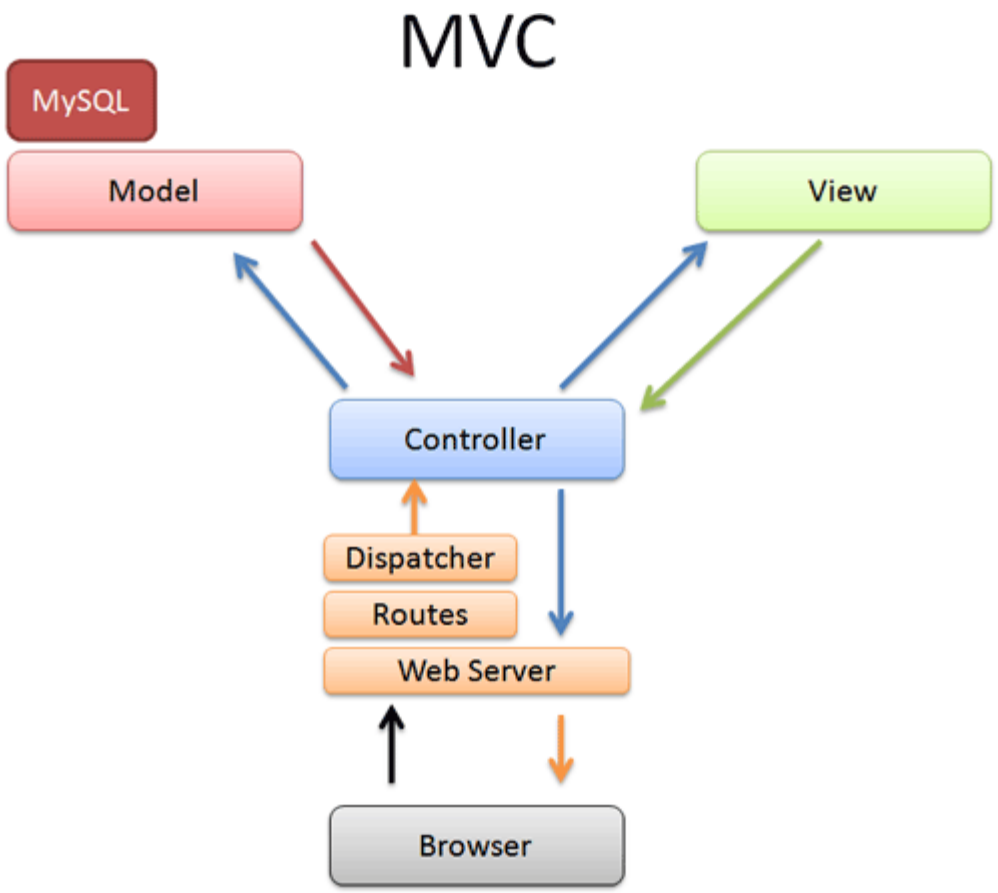
So sánh JSP và Servlet

JSP	Servlet
ISP là “Webpage Scripting language” Được chuyển vào servlet khi chạy	Là một chương trình java được biên dịch
Nhúng java code vào HTML <html><% java code %></html>	Sử dụng thẻ html thông qua lệnh print .. out.print(“html code”);
Chạy chậm	Chạy nhanh
Sử dụng java thiết kế giao diện. Trong mô hình MVC JSP đóng vai trò View	Xử lý tác vụ. Trong mô hình MVC Servlet đóng vai trò Controller
Có thể xây dựng các tag mới sử dụng Java API	Không xây dựng thêm
Không cần biên dịch lại khi thay đổi	Khó khăn khi kết xuất kết quả
Mạnh hiển thị	Mạnh xử lý

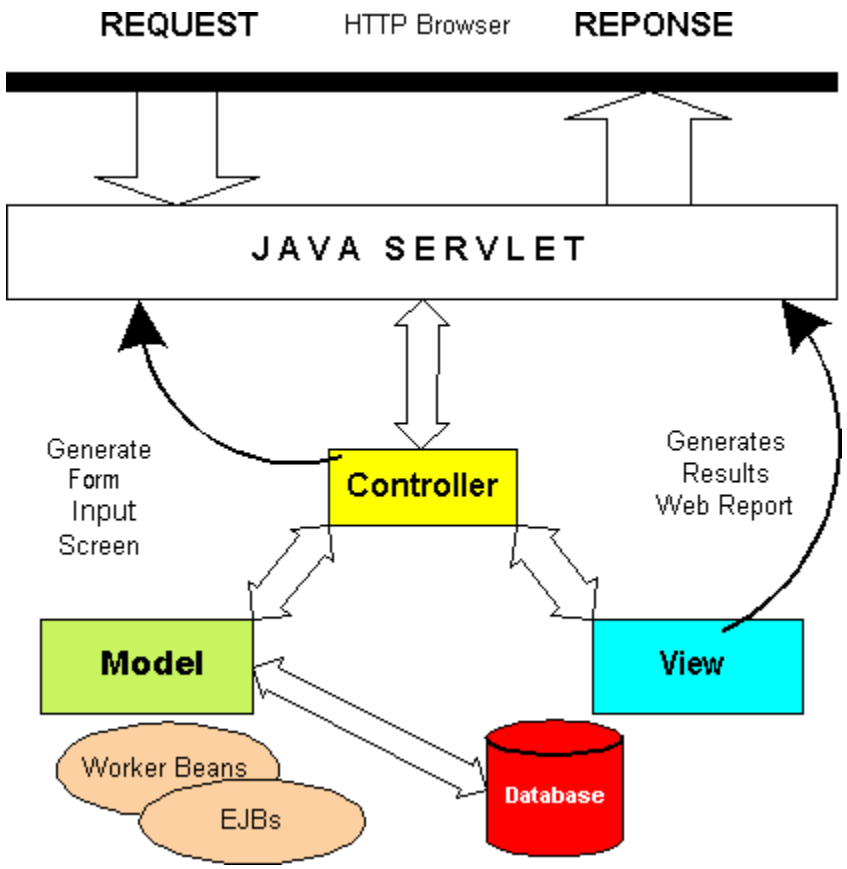
Mô hình MVC

- Ý tưởng chính của MVC là tách rời mức tác nghiệp (business logic) và mức thể hiện (UI) trong chương trình, cho phép chúng thay đổi độc lập mà không làm ảnh hưởng lẫn nhau.
- MVC tách rời business logic và UI thành 3 vai trò trong ứng dụng:
  - *Model*: có nhiệm vụ đóng gói dữ liệu của ứng dụng cho *view* thể hiện.
  - *View*: chỉ hiển thị dữ liệu, không có bất cứ sự tác nghiệp nào.(JSP)
  - *Controller*: có nhiệm vụ nhận yêu cầu từ người dùng, gọi các dịch vụ tác nghiệp, sau đó trả về dữ liệu cho *view* hiển thị. (Servlet)
- Một *model* có thể có nhiều *view* khác nhau.

Mô hình MVC (tt)



Mô hình MVC trong ứng dụng web java



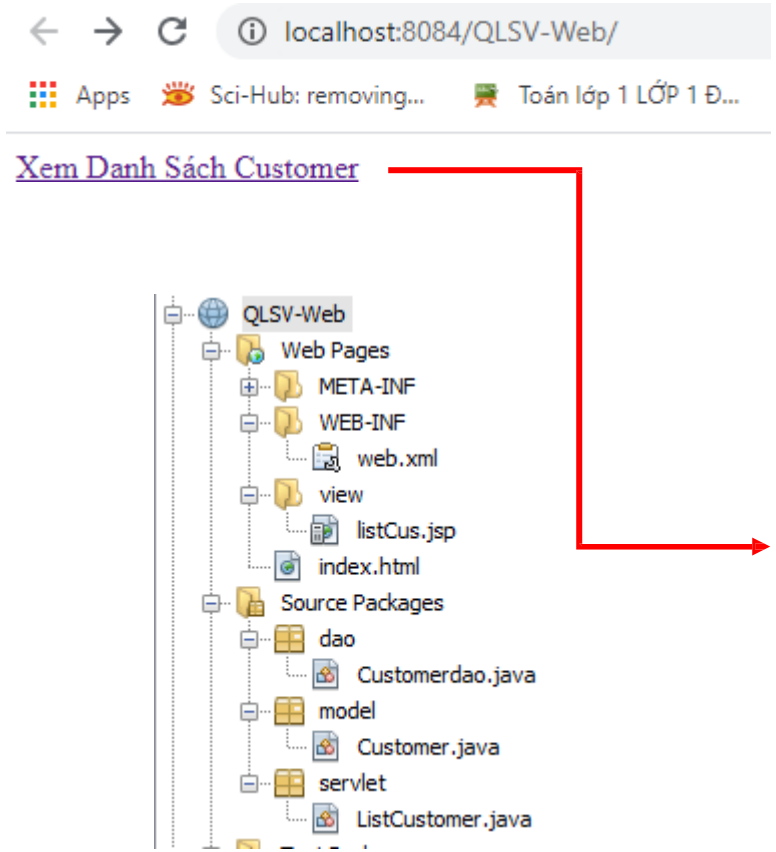
Mô hình MVC trong ứng dụng web java (tt)

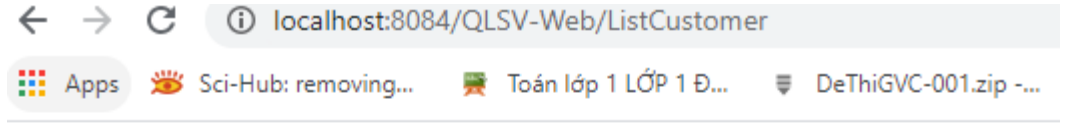
- Định nghĩa các Bean mô tả dữ liệu
- Dùng Servlet xử lí yêu cầu
  - Đọc tham số yêu cầu
  - Kiểm tra tính hợp lệ
  - Gọi tầng Business Logic
  - Kết quả được đặt vào các Bean ở bước 1
- Nạp các Bean vào request, session .. .
- Chuyển yêu cầu tới trang JSP tương ứng (VIEW)
- Các trang JSP trình bày kết quả từ các Bean

CaseStudy

- Viết ứng dụng Web hiển thị danh sách Customer sử dụng mô hình MVC

ID	NAME	GEN
1	Hồ Đình Khả	Female
2	Trần Thị Như Ý	Male
3	Lương An Vinh	Female
4	Đoàn Trình Dục	Female
5	Lê Triệu Ngọc Đức	Female





Danh Sách Customer

ID	Name	Gen
1	Hồ Đình Khả	Female
2	Trần Thị Như Ý	Male
3	Hồ Đình Khả	Female
4	Hồ Đình Khả	Female
5	Hồ Đình Khả	Female