ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

503111

Java Technology

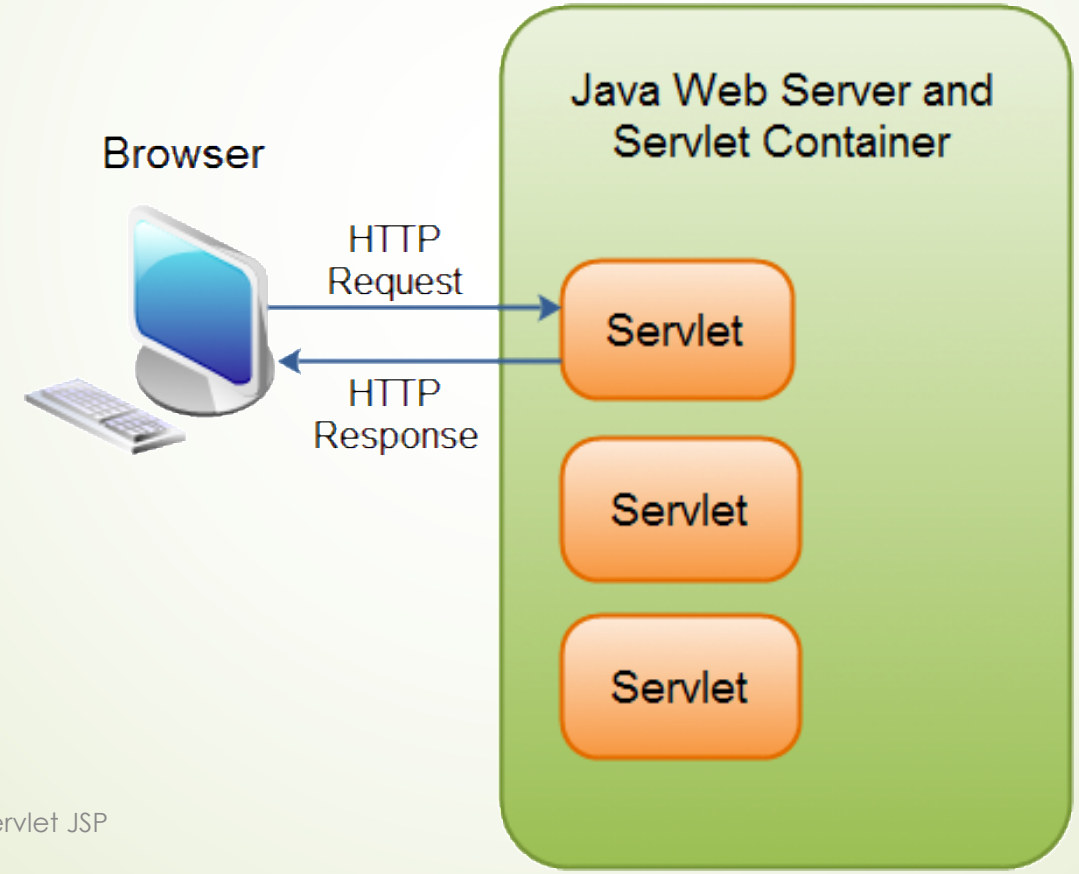
SERVLET - JSP

Introduction

- Java Servlets is a standard web technology for Java.
- Many modern Java web technologies are based on the foundation of Servlet.
- Java Servlets are part of the Java Enterprise Edition (Java EE).

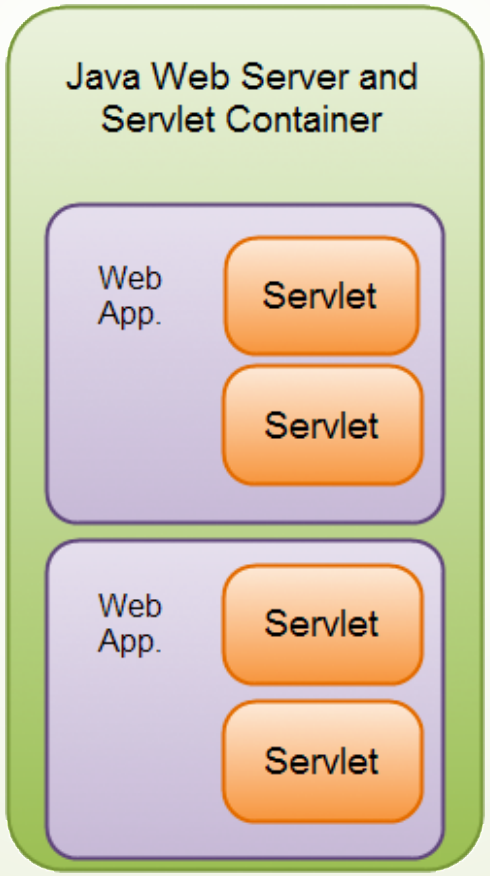
Servlet Overview

- A Java Servlet is a Java object that responds to HTTP requests.
- It runs inside a Servlet container. Here is an illustration of that:



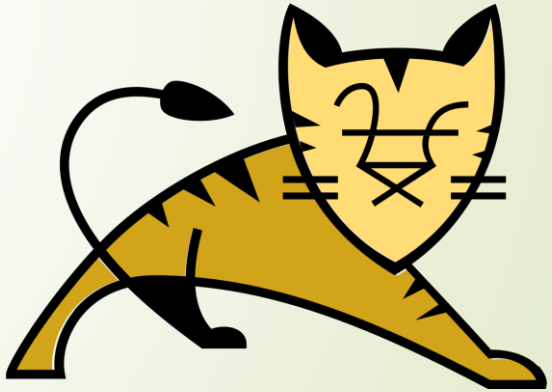
Servlet Overview

- A Servlet container may run multiple web applications at the same time, each having multiple servlets running inside.



Servlet Containers

- Java servlet containers are usually running inside a Java web server.
- A few common well known, free Java web servers are:
  - Jetty
  - Tomcat



Servlet Example

- Servlets are Java classes which service HTTP requests and extend javax.servlet.http.HttpServlet, an abstract class that implements the Servlet interface and is specially designed to handle HTTP requests.
- A servlet usually implements doGet() and doPost() methods, which handles the http get and the http post request respectively.

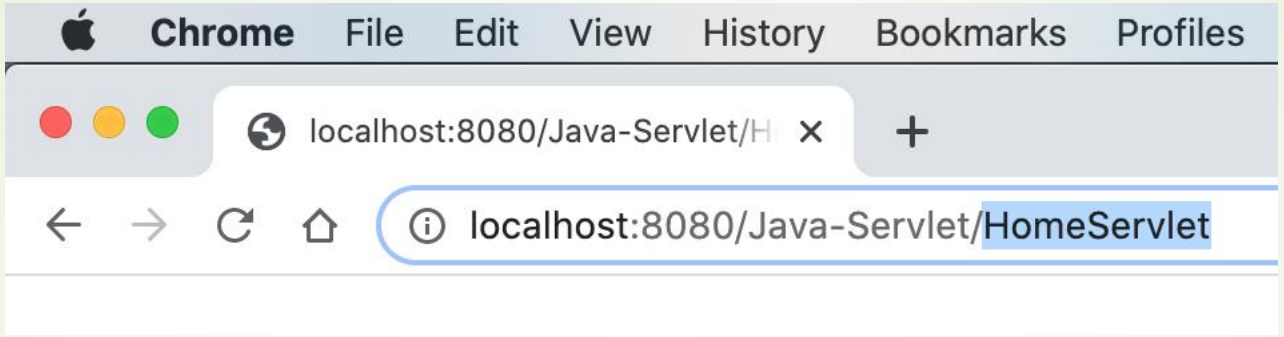
Servlet Example

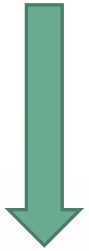
```
1  public class HomeServlet extends HttpServlet {
2      public HomeServlet() {
3      }
4      protected void doGet(HttpServletRequest request,
5       HttpServletResponse response){
6          response.getWriter().write("Hello GET method");
7      }
8
9      protected void doPost(HttpServletRequest request,
10      HttpServletResponse response){
11         response.getWriter().write("Hello POST method");
12     }
13 }
```

Register HomeServlet in web.xml

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee">

    <servlet>
        <servlet-name>HomeServlet</servlet-name>
        <servlet-class>com.example.web.HelloServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>HomeServlet</servlet-name>
        <url-pattern>/HomeServlet</url-pattern>
    </servlet-mapping>

</web-app>
```

Servlet Example

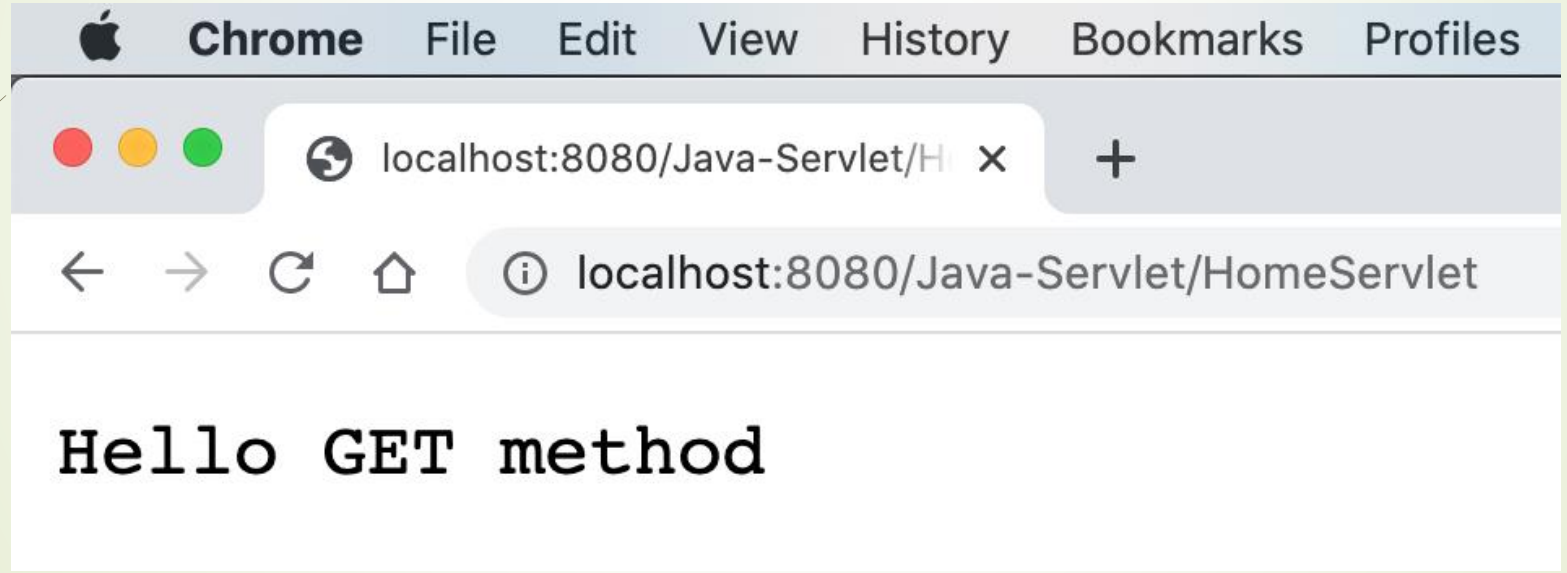




```
protected void doGet(HttpServletRequest request,
 HttpServletResponse response){
    response.getWriter().write("Hello GET method");
}
```

Servlet Example

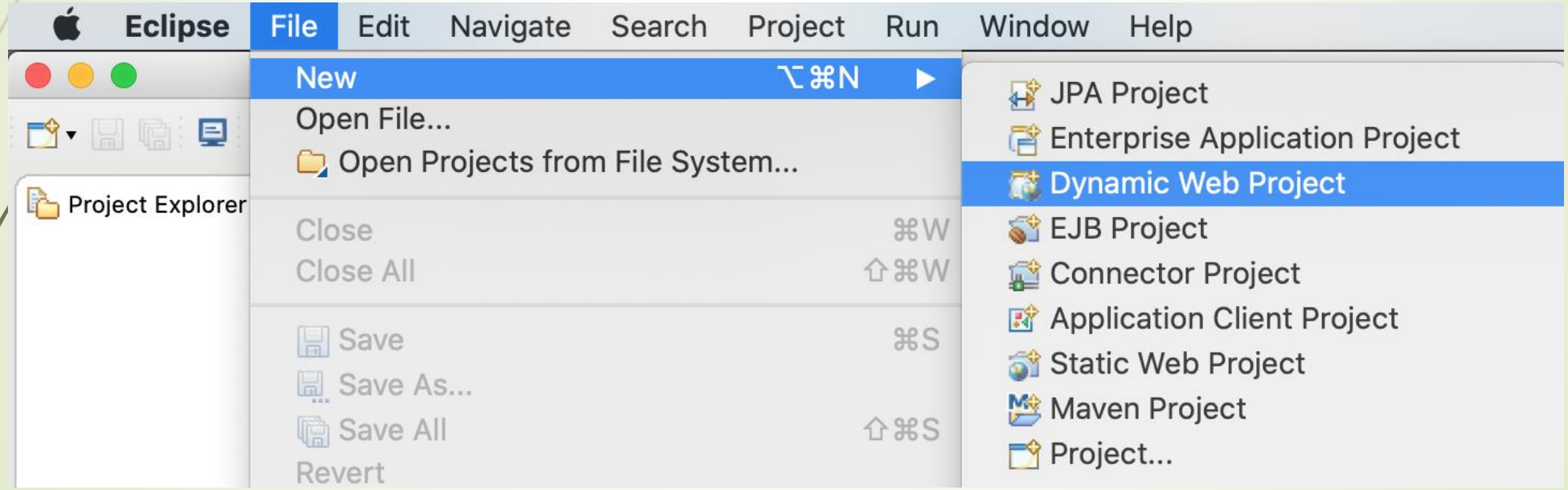
- Example result of accessing the servlet with GET request.



Create Servlet-JSP Project with Eclipse

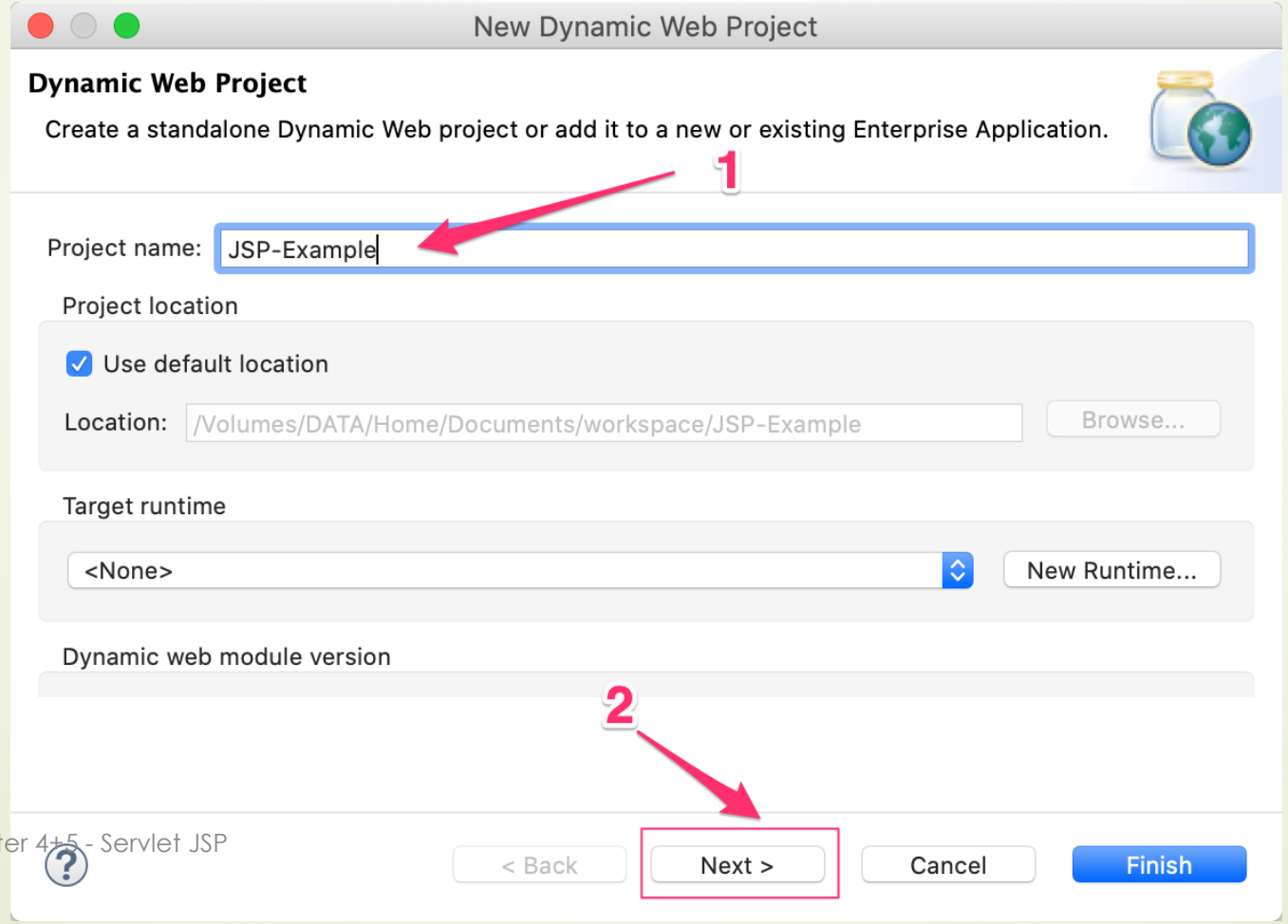
Getting Started with Servlet

1. Create a new Dynamic web project



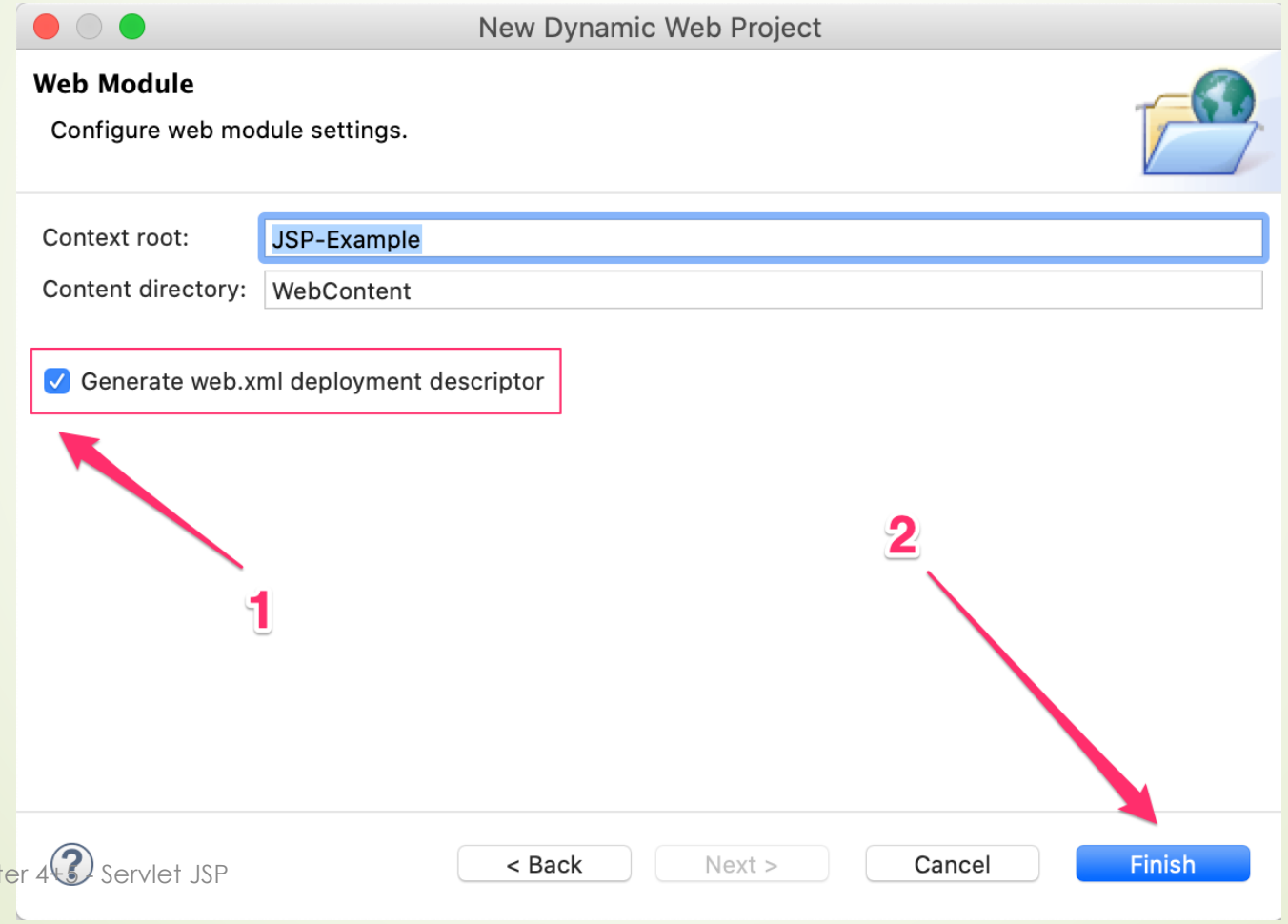
Getting Started with Servlet

Give the project a name and then click **Next**



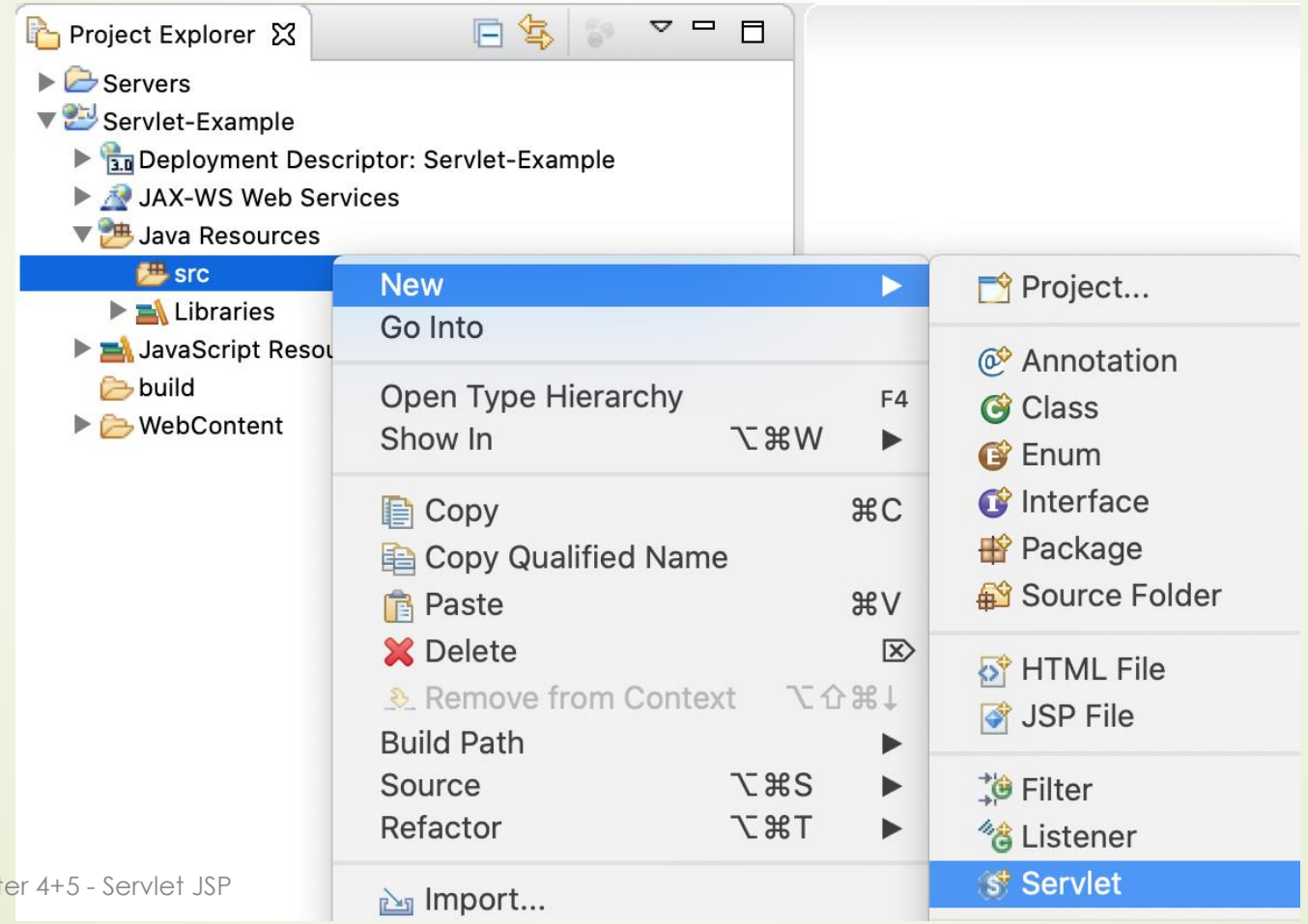
Getting Started with Servlet

Check the "**Generate web.xml....**" option.



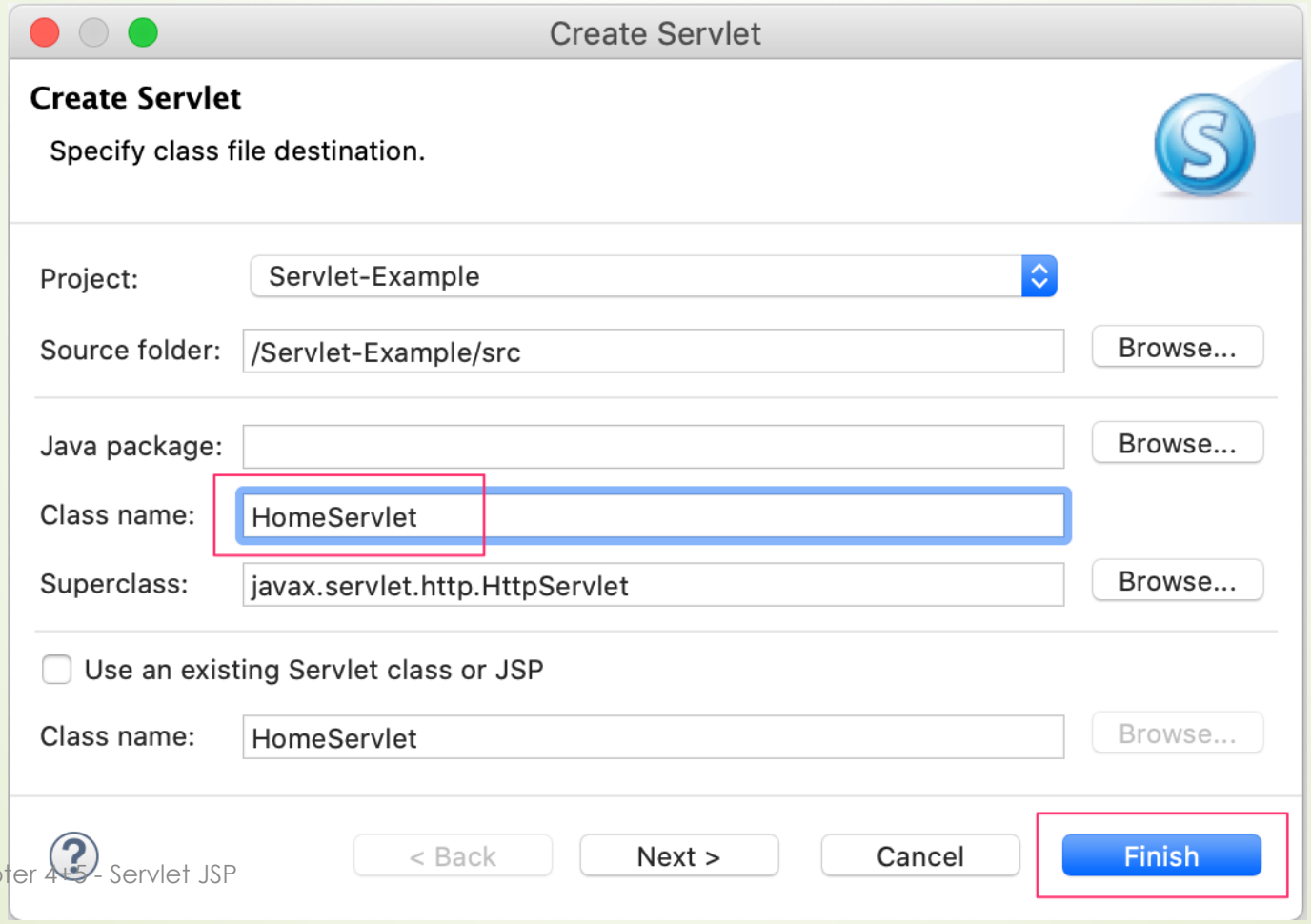
Getting Started with Servlet

2. Create a Servlet class



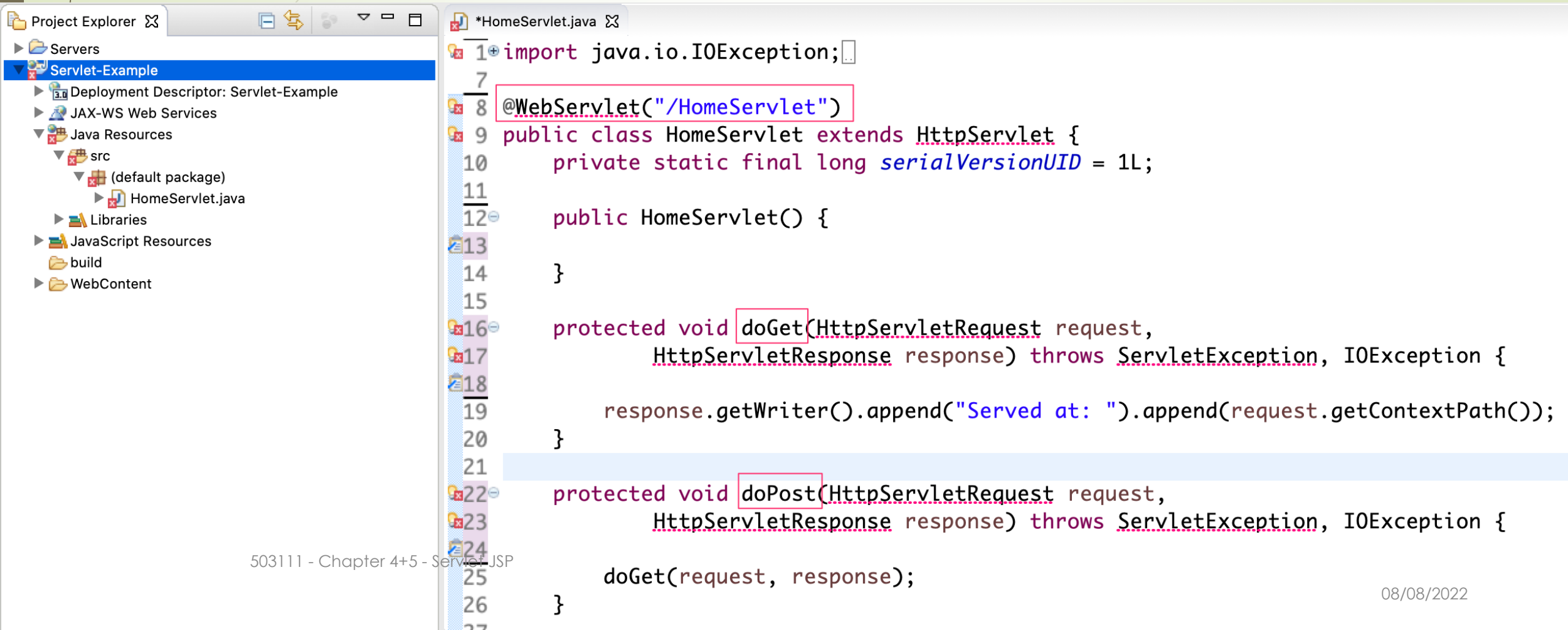
Getting Started with Servlet

Give the servlet a name, for example: HomeServlet



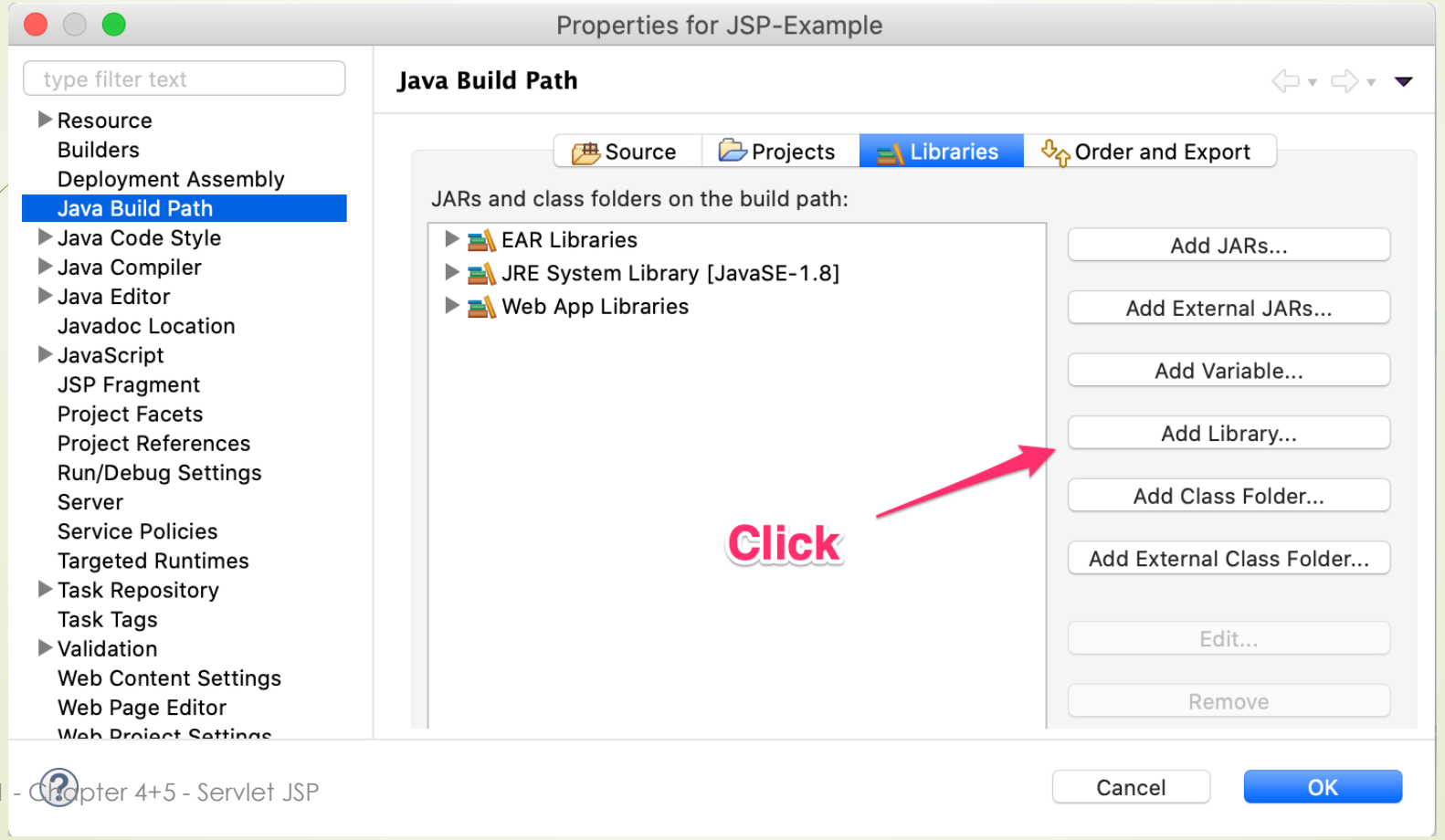
Getting Started with Servlet

The HomeServlet.java is created with some lines of code and errrors.



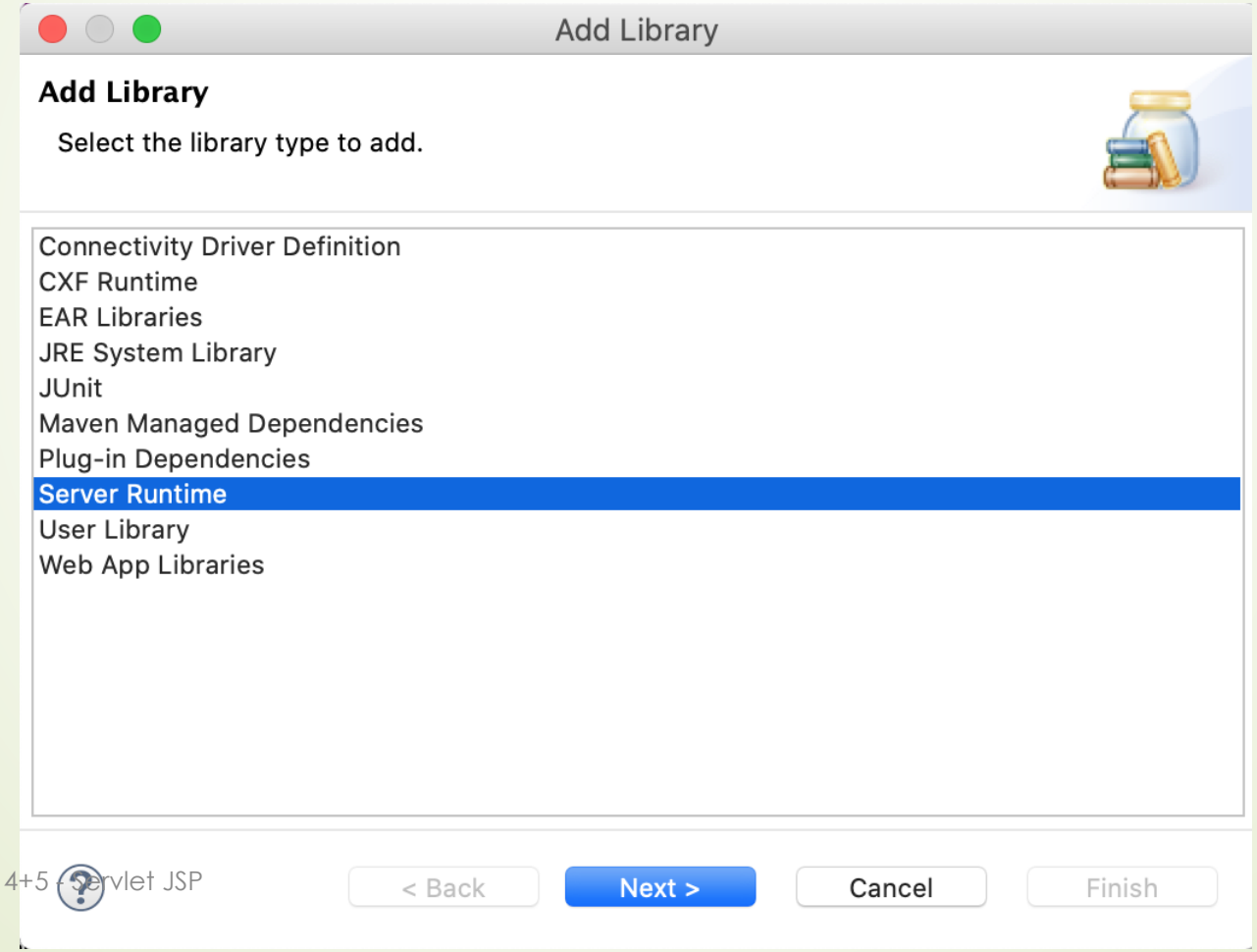
Getting Started with Servlet

3. Add servlet library to fix the erroor



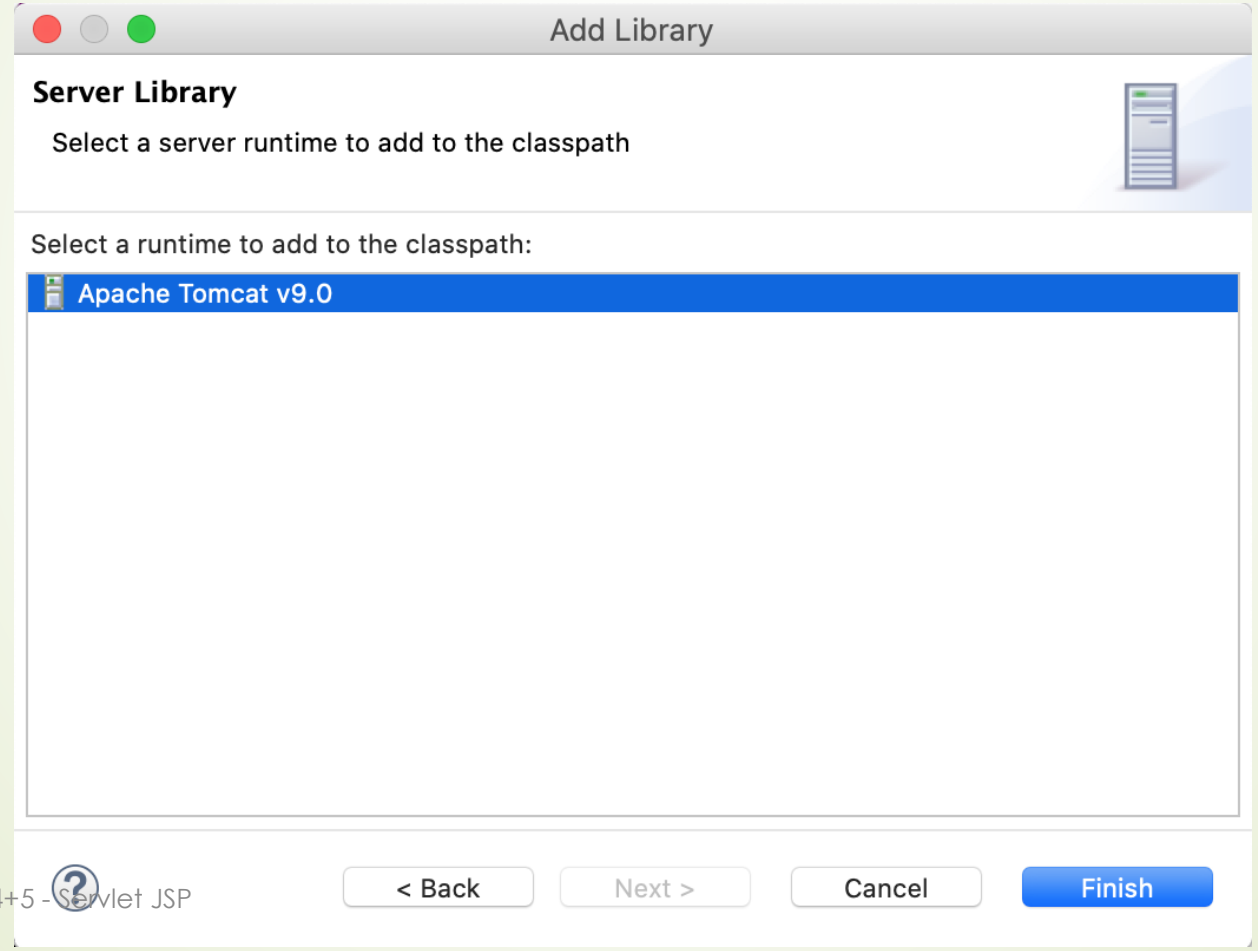
Getting Started with Servlet

Choose "Server Runtime" library and then hit "Next"



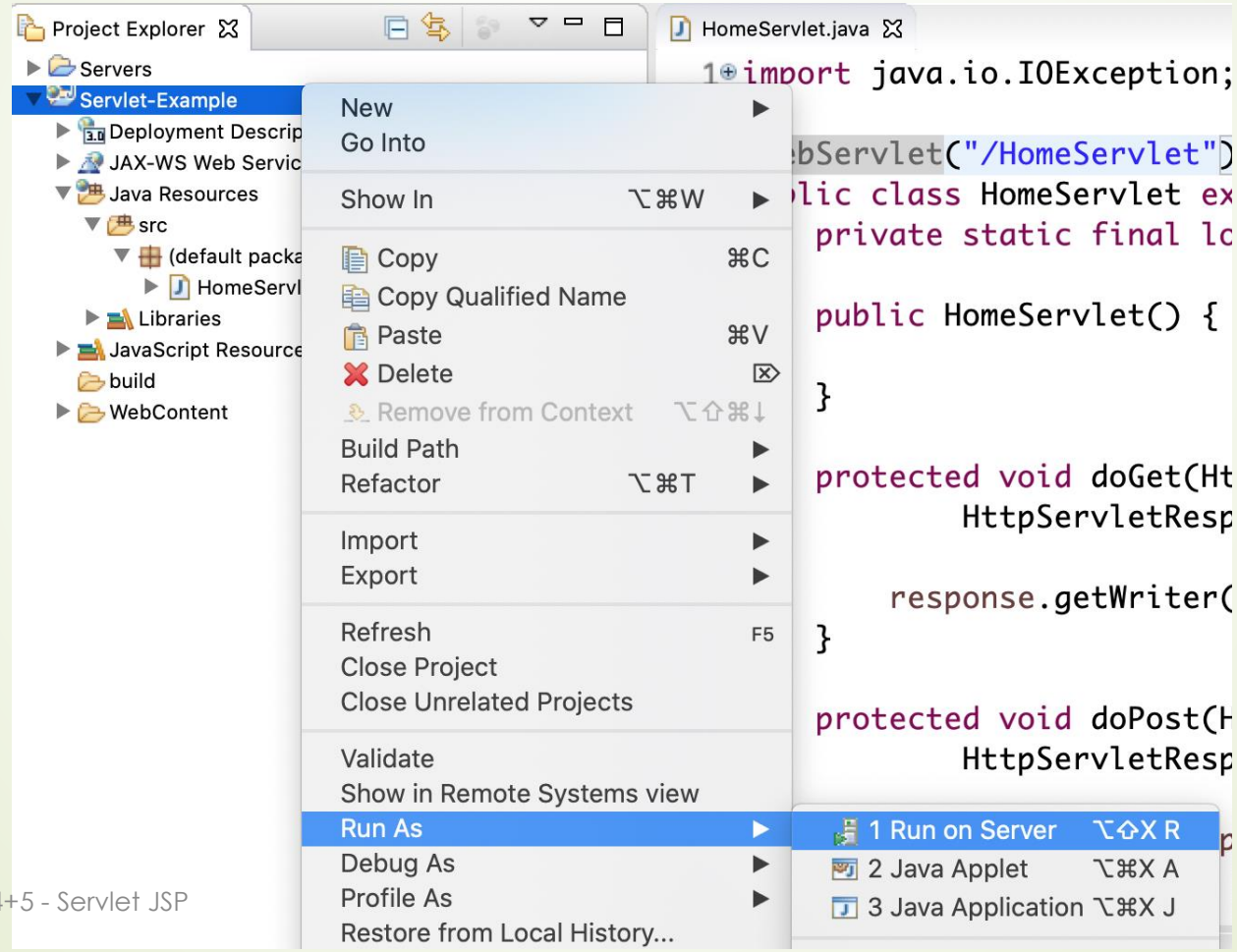
Getting Started with Servlet

Select an appropriate Apache Tomcat version and click "Finish".



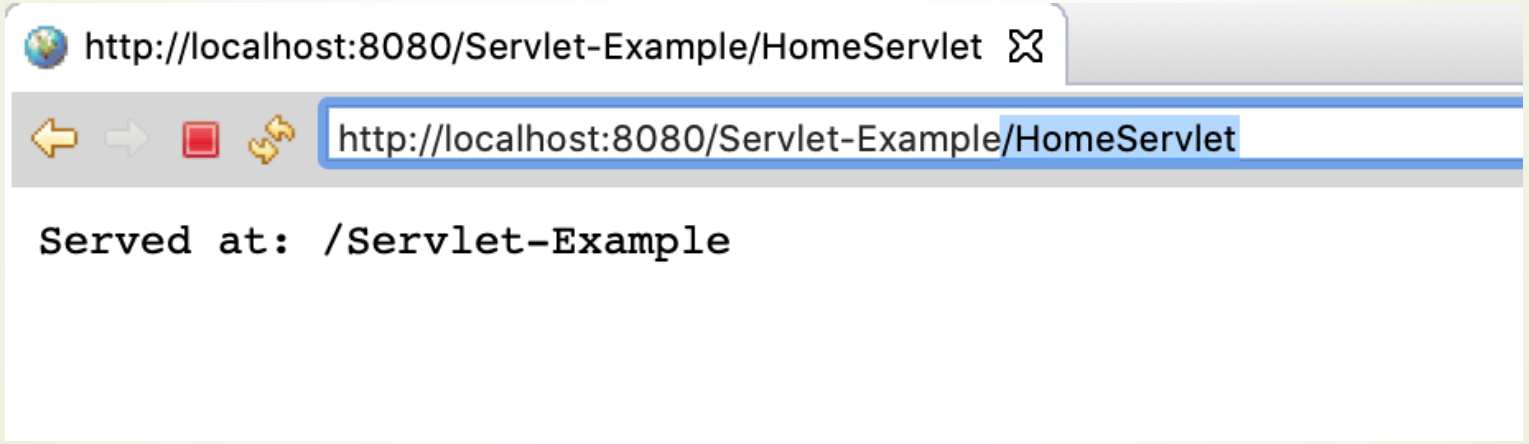
Getting Started with JSP

5. Run the project as "Run on server"



Getting Started with JSP

6. Check the result



HTTP Request

- When a browser requests for a web page, it sends lot of information to the web server.
- These information can be retrieved from the HttpRequest object:
  - Request Parameters (Both query string and form-urlencoded)
  - InputStream (raw data e.g Json document)
  - Headers
  - Cookies
  - Session

HTTP Request Parameter

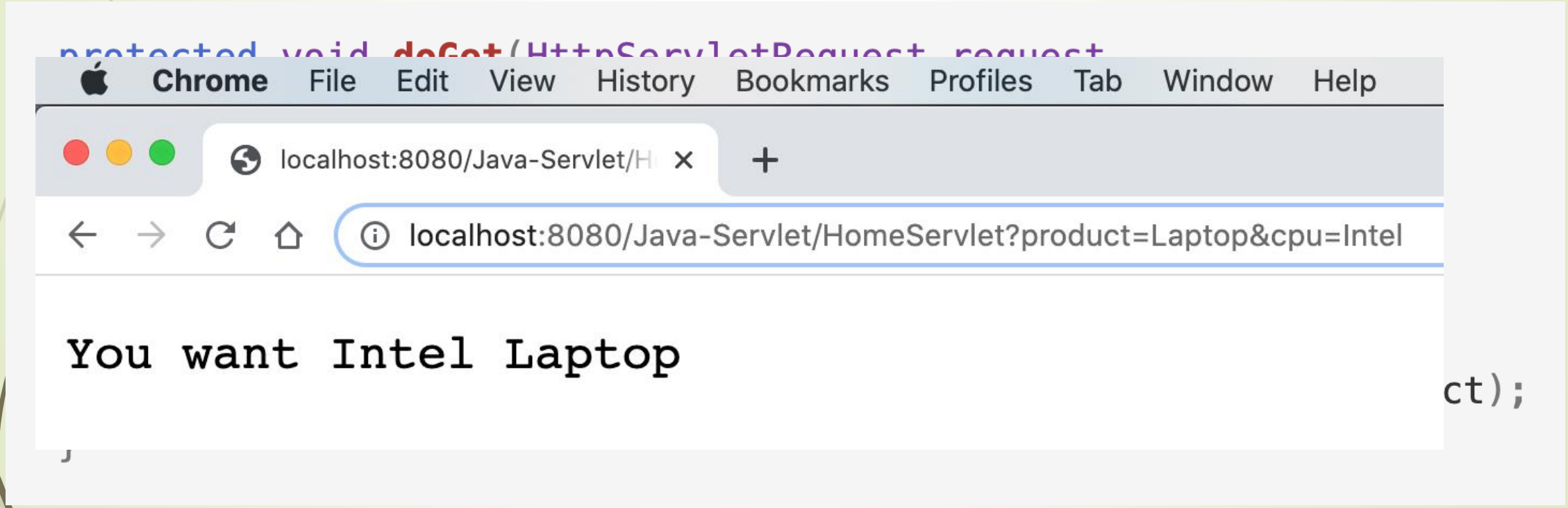
- The request parameters are parameters that are sent from the browser along with the request.
- Request parameters are typically sent as part of the URL (in the "query string"), or as part of the body of an HTTP request.

http://localhost/HomeServlet?product=laptop&cpu=intel

- There are two paramters in the above url
  1. product=laptop
  2. cpu=intel

HTTP Request Parameter

- These parameters can be access from the request object via the getParameter() method.



HTTP Session

- The HttpSession object represents a user session.
- Using session is a way to make data accessible across the various pages of an entire website.

```
protected void doGet(HttpServletRequest req,
        HttpServletResponse res){

    HttpSession session = req.getSession();

    // do something with the session (read/write)
    // then send a response back to the client
}
```

HTTP Session

- The getAttribute() method is used to retrieve data from a user session.

```
HttpSession session = request.getSession();
String email = (String) session.getAttribute("email");

if (email != null) {
    // user đã đăng nhập, chuyển hướng vào trang profile
}else {
    // user chưa đăng nhập, chuyển hướng đến trang login
}
```

HTTP Session

- The setA data to a user sess

```
protected void doPost(HttpServletRequest req,
        HttpServletResponse res){

    // đọc thông tin từ client
    String email = request.getParameter("email");
    String password = request.getParameter("password");

    // kiểm tra đăng nhập
    if (login(email, password) == true) {

        HttpSession session = request.getSession();
        session.setAttribute("emai", email);

        // sau đó redirect tới trang profile

    }else {
        // thông báo đăng nhập thất bại
    }
```

HTTP Response

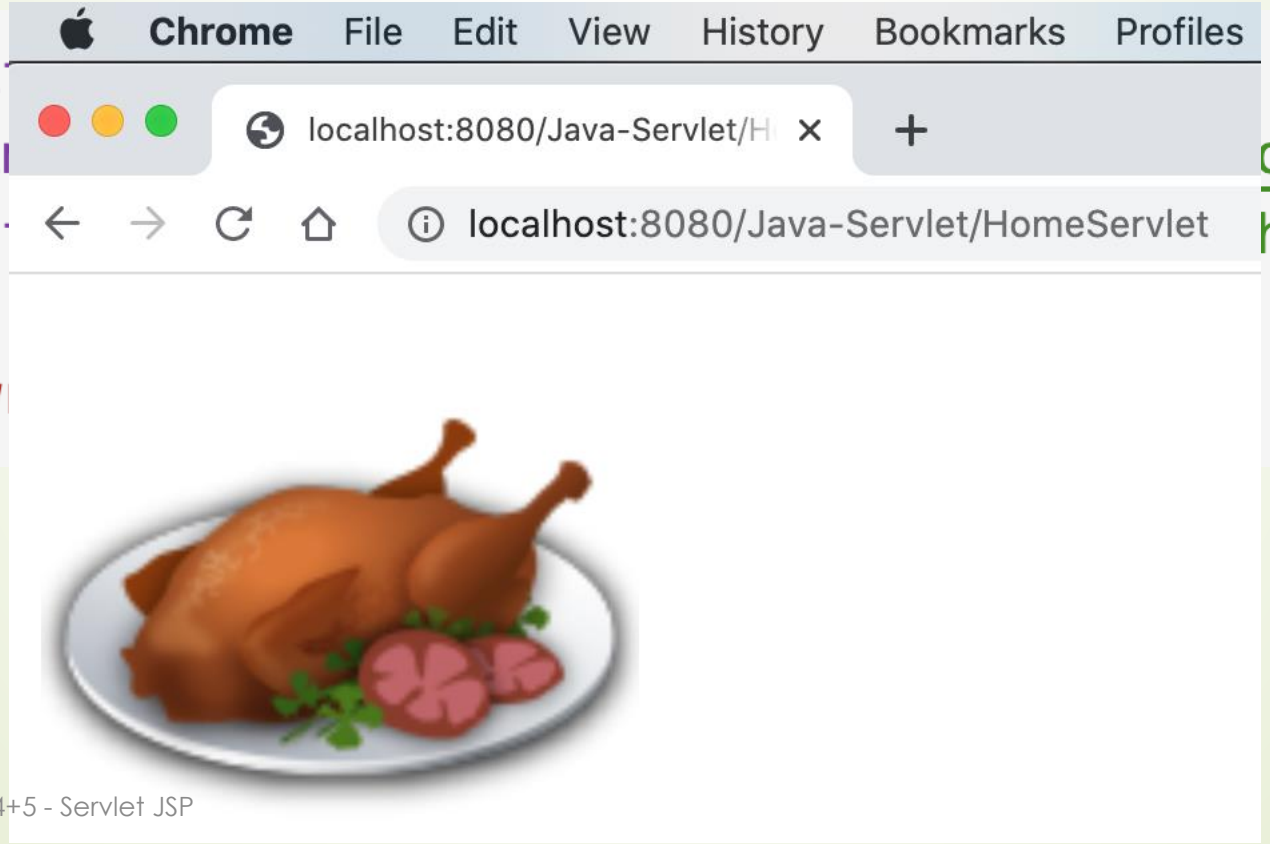
- The purpose of the HttpResponse object is to represent the HTTP response your web application sends back to the browser.
- The HttpResponse object has a lot of methods, so I will just cover the most commonly used here.

HTTP Response - HTML

- To send HTML back to the browser, you have to obtain the a PrintWriter from the HttpResponse object.

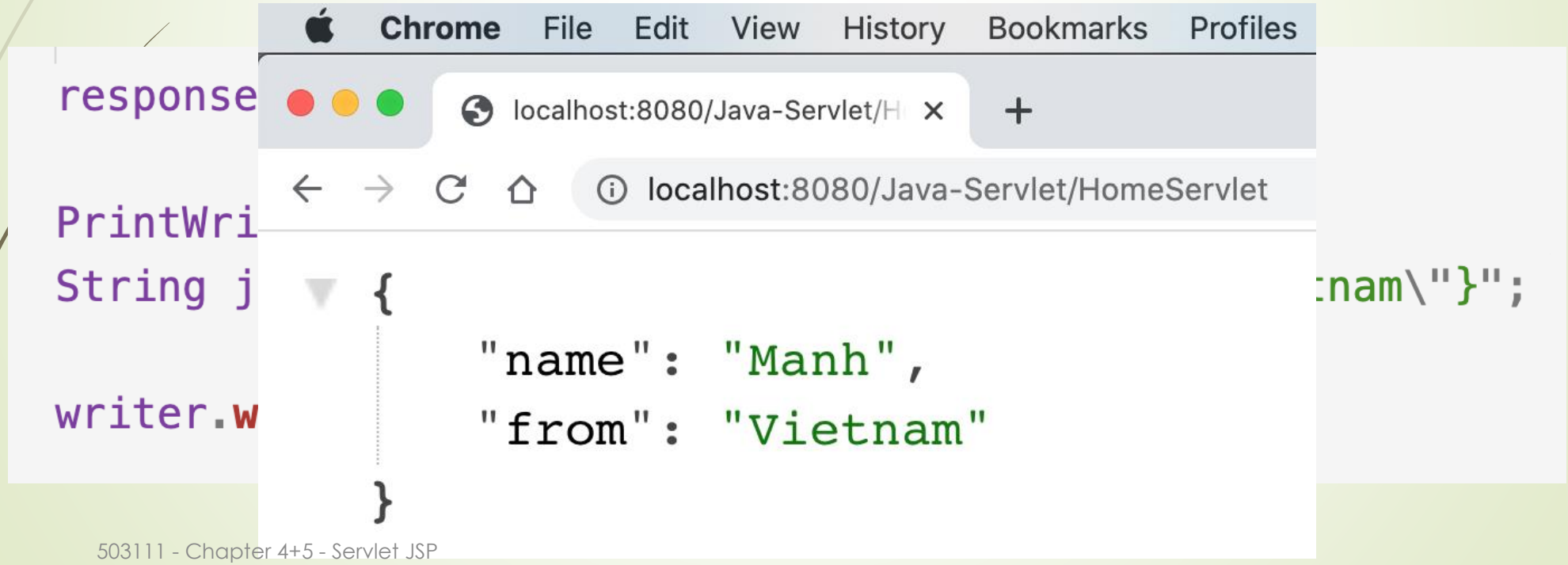
```
PrintWri
String i
String h

writer.w
```

```
o.png'>";
html>";
```



HTTP Response - JSON

- The Content-Type header tells the browser to interpret the content according to its type. The content type 'application-json' is used for JSON data format.`



HTTP Response – Redirect

- You can redirect the browser to a different URL from your servlet.
- When redirecting, we can only send the data via query string.

```
protected void doGet(HttpServletRequest request,
        HttpServletResponse response){

    String url = "http://http://localhost:8080/Java-Servlet/";
    String query = "product=phone&color=black";
    String link = url + "?" + query;

    response.sendRedirect(link);
}
```

JSP

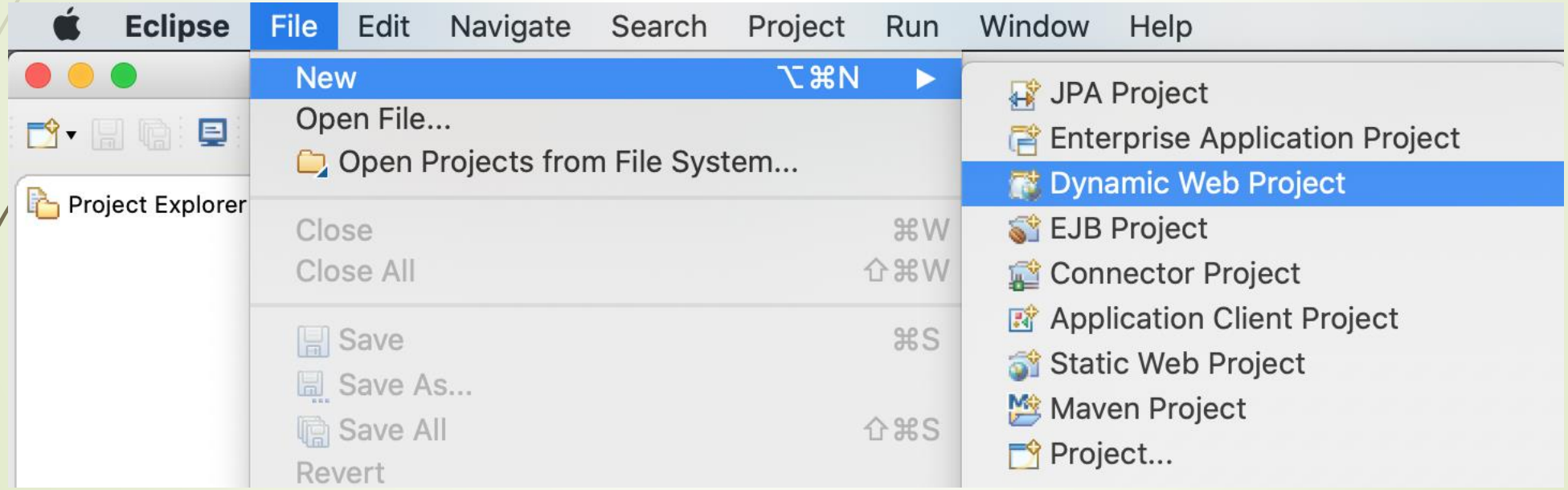
Java Server Page

JSP Overview

- JavaServer Pages (JSP) is a technology for developing Webpages that supports dynamic content.
- This helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.
- JavaServer Pages are built on top of the Java Servlets API.

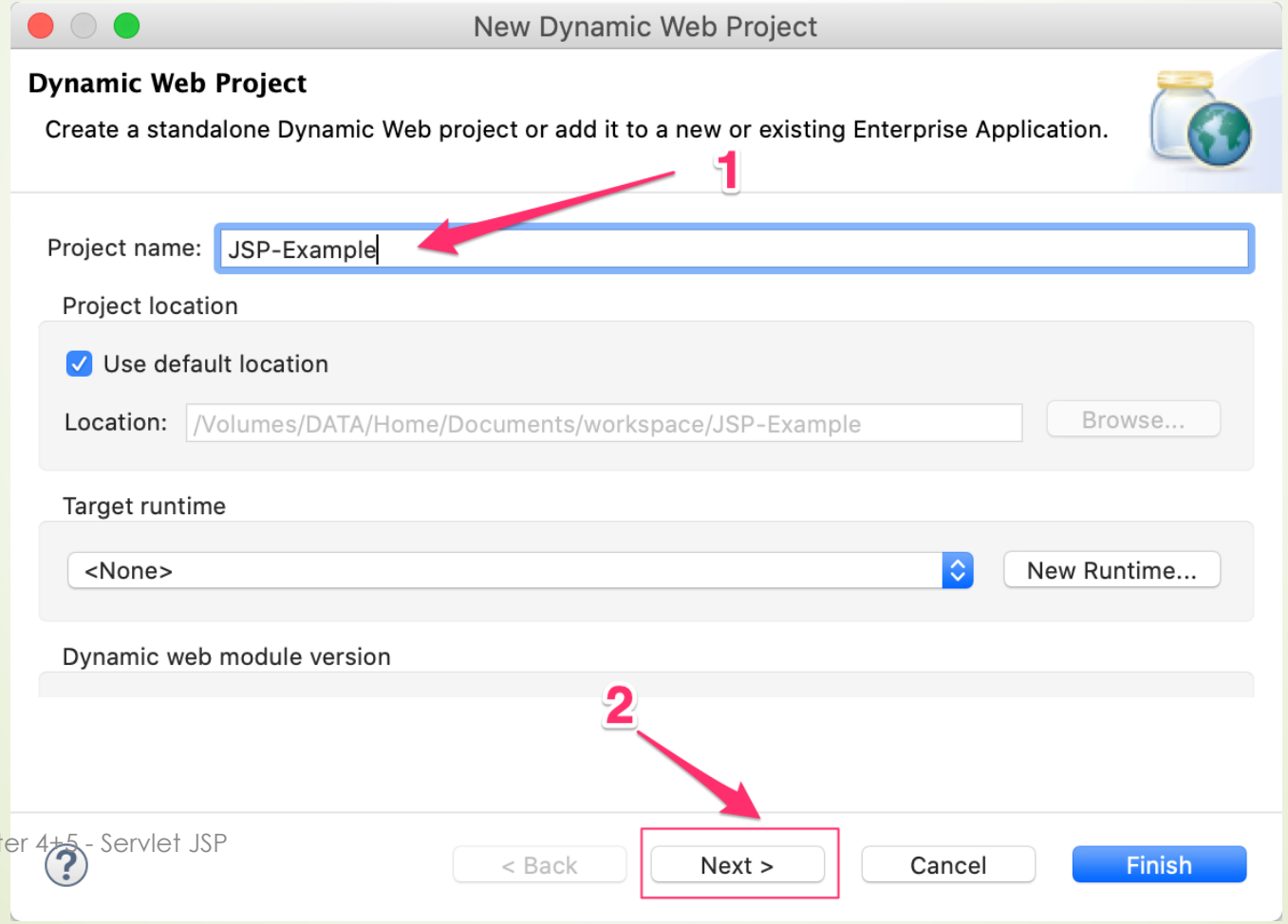
Getting Started with JSP

1. Create a new Dynamic web project



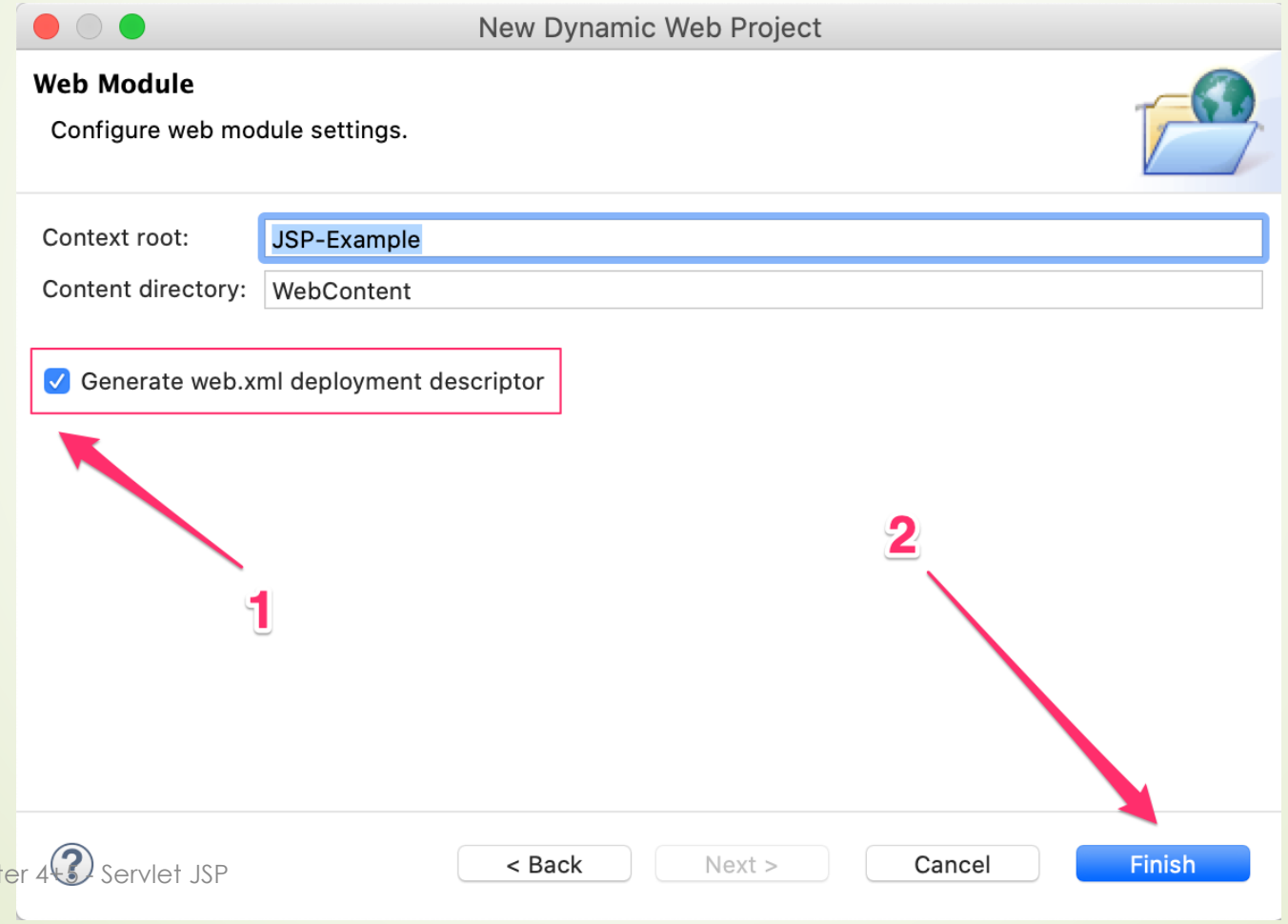
Getting Started with JSP

Give the project a name and then click **Next**



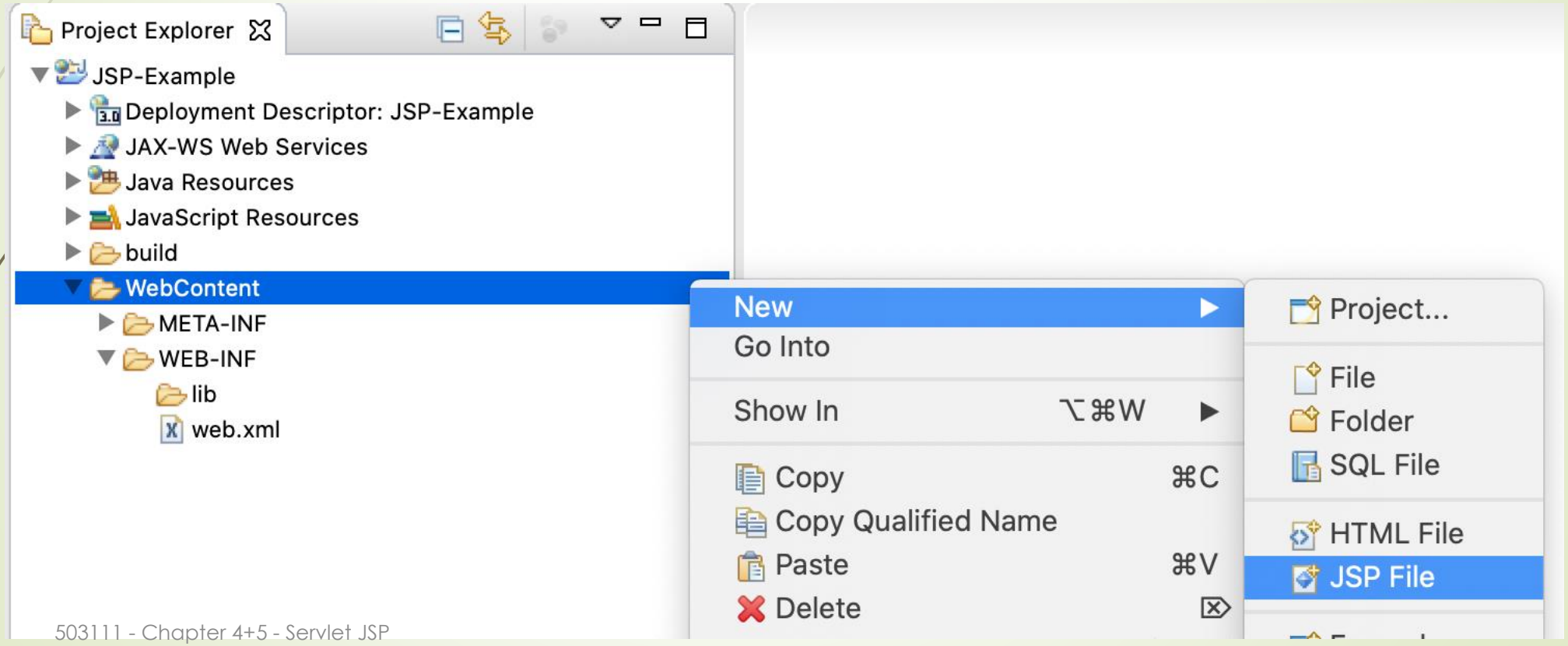
Getting Started with JSP

Check the "**Generate web.xml....**" option.



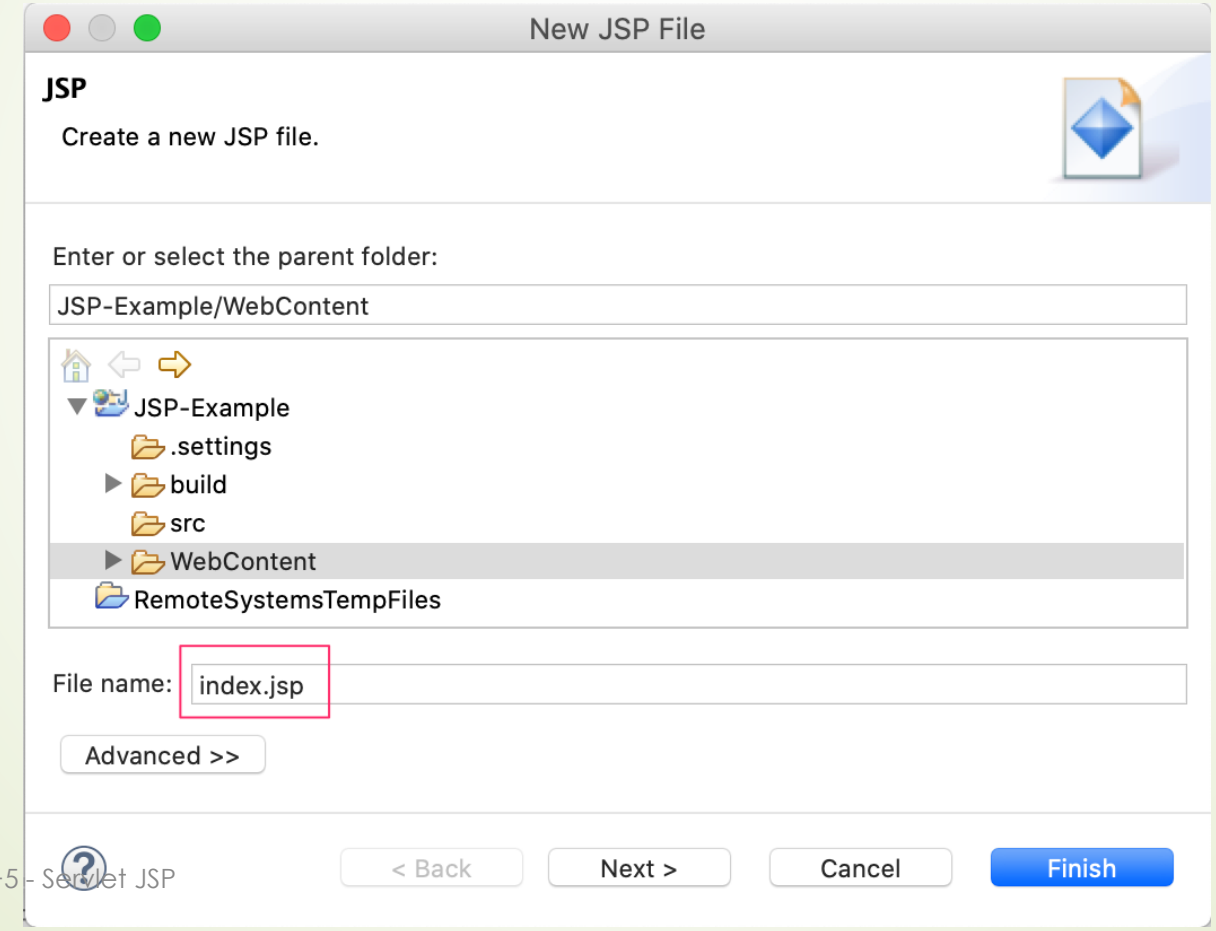
Getting Started with JSP

2. Create an index.jsp file inside WebContent folder



Getting Started with JSP

Named the file as index.jsp



Getting Started with JSP

3. Put some HTML 5 and JSP code in the file.

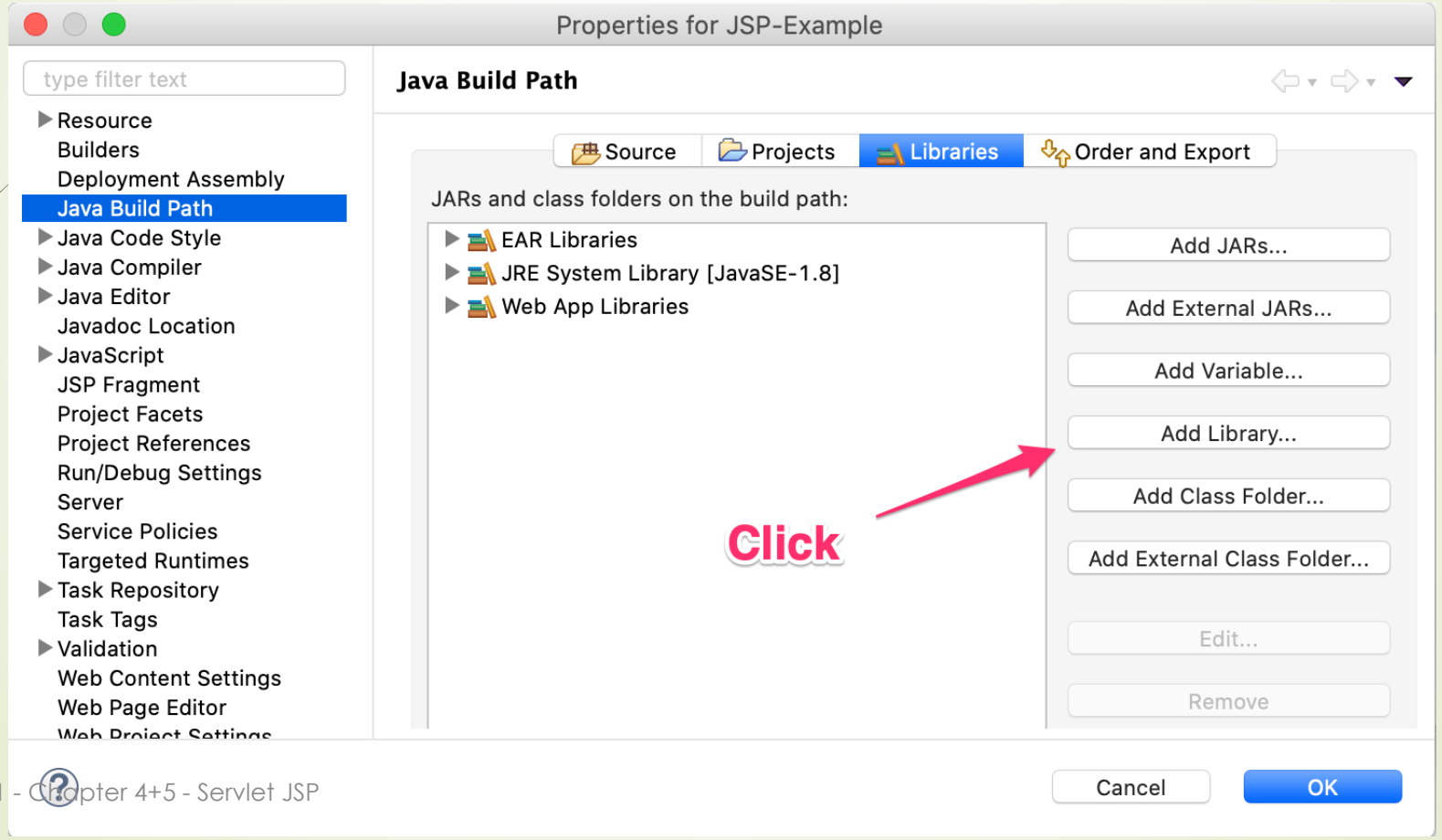
index.jsp

```
1  <%@page import="java.util.Date"%>
2  <%@ page language="java" contentType="text/html; charset=UTF-8"
3      pageEncoding="UTF-8"%>
4  <!DOCTYPE html>
5  <html>
6      <head>
7      <title>Page Title</title>
8      </head>
9  <body>
10
11     <h1>Welcome to JSP</h1>
12     <% out.println(new Date().toString()); %>
13
14 </body>
15 </html>
```

The superclass "javax.servlet.http.HttpServlet" was not found on the Java Build Path

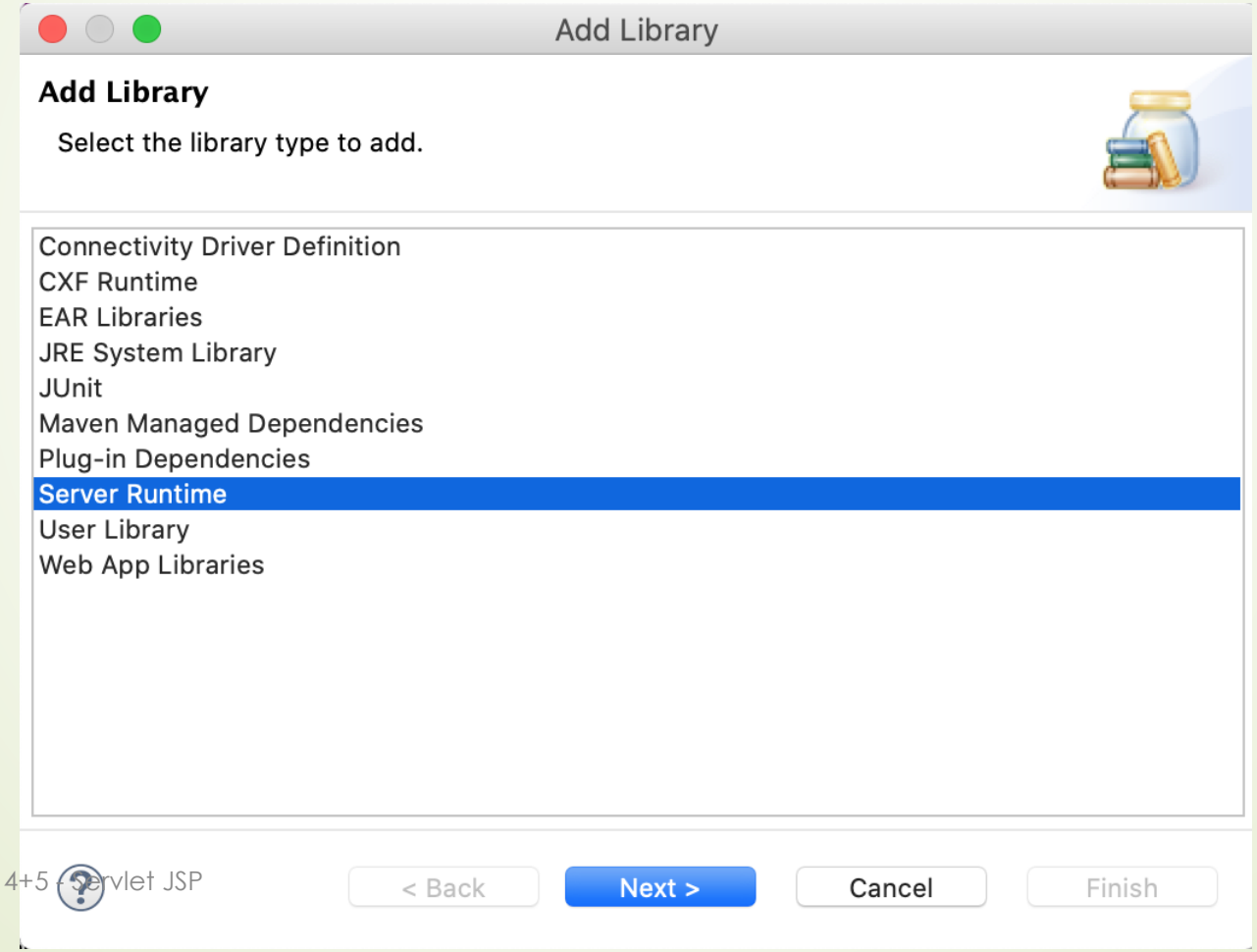
Getting Started with JSP

4. Add servlet library to fix the erroor



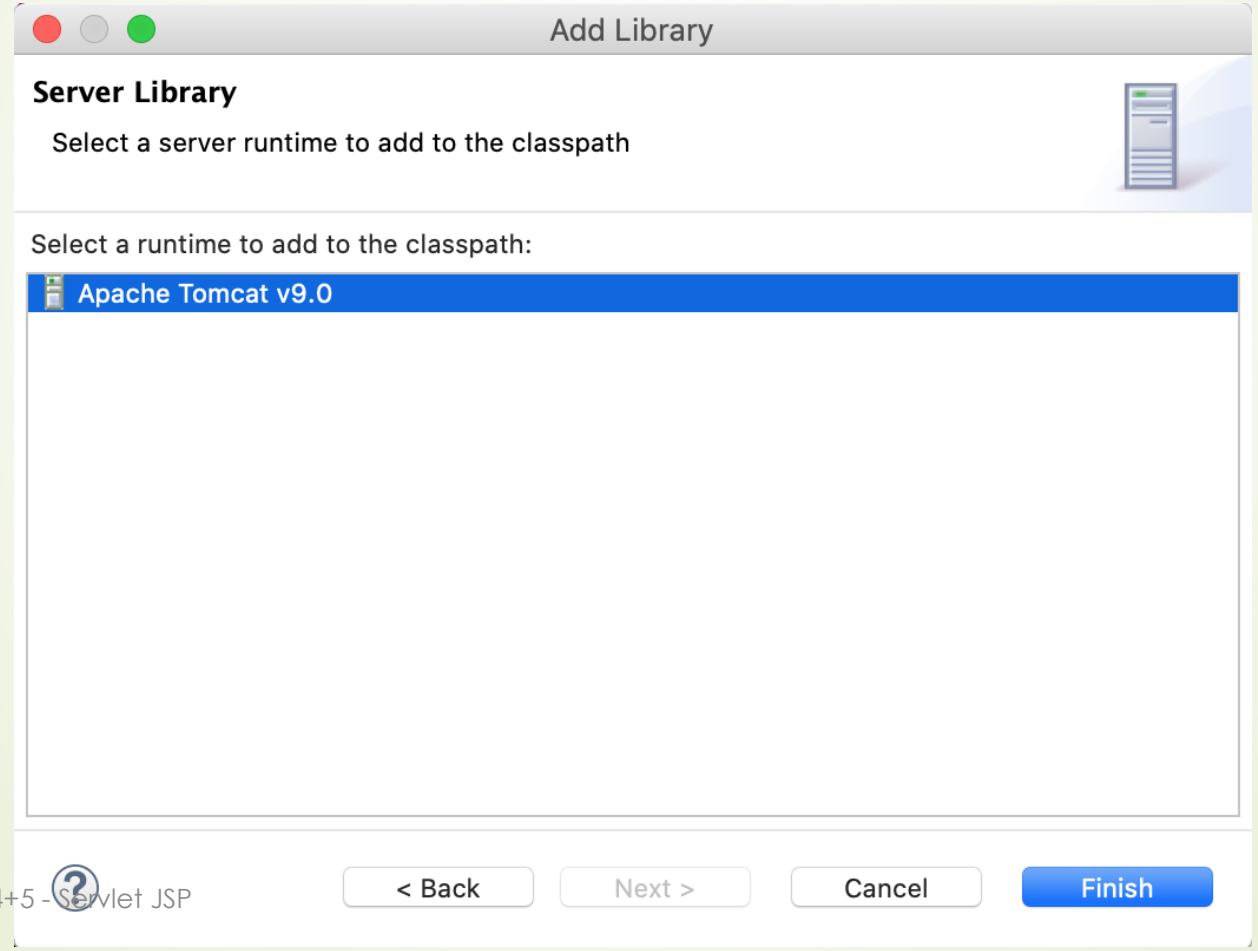
Getting Started with JSP

Choose “Server Runtime” library and then hit “Next”



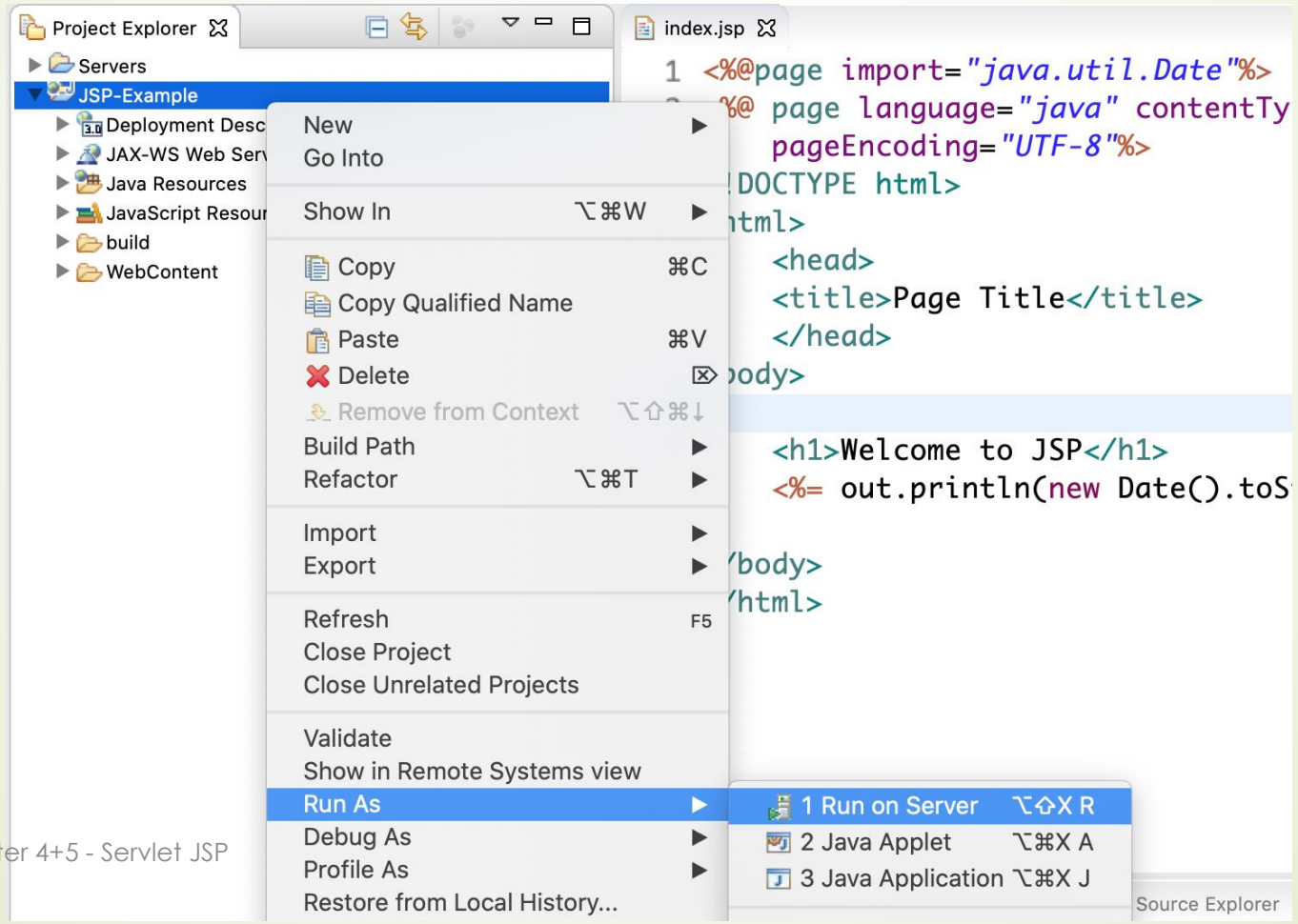
Getting Started with JSP

Select an appropriate Apache Tomcat version and click “Finish”.



Getting Started with JSP

5. Run the project as "Run on server"



Getting Started with JSP

6. Check the result

JSP Syntax

- The Scriptlet
- JSP Expression
- JSP Declarations
- JSP Comments
- JSP Directives
- JSP Actions
- JSP Implicit Objects
- Decision-Making Statements
- Loop Statements

The Scriptlets

- Syntax

  <% code fragment %>

- A scriptlet can contains:
  - Java language statements.
  - Variable declaration
  - Java Expression.
- A scriptlet can not contains:
  - Method declaration
  - HTML code.

The Scriptlets

*index.jsp

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      pageEncodi
3  <!DOCTYPE html
4  <html>
5  <head>
6      <meta http
7  </head>
8  <body>
9      <%
10         int si
11         for (i
12             ou
13         }
14     %>
15 </body>
16 </html>
```

http://localhost:8080/Servlet-Example/

http://localhost:8080/Servlet-Example/

```
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 9
Item 10
```

JSP Declarations

- A declaration declares one or more variables or methods that you can use in Java code later in the JSP file.

```
<body>
    <%!
        double PI = 3.14;
        double circleArea(double radius) {
            return PI * radius * radius;
        }
    %>

    <h1>Scriptlet & Declarations</h1>

    <%
        out.println(circleArea(5));
    %>
</body>
```

JSP Expression

- A JSP expression element contains a scripting language expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file.

```
<html>
    <body>
        <%!
            double PI = 3.14;
            double circleArea(double radius) {
                return PI * radius * radius;
            }
        %>
        <p>Diện tích HCN là <%= circleArea(5.0) %>
    </body>
</html>
```

JSP Comments

- JSP comment marks text or statements that the JSP container should ignore.

```
<body>
    <%!
        double PI = 3.14;
        double circleArea(double radius) {
            return PI * radius * radius;
        }
    %>
    <%-- Tính và in ra diện tích HCN với bán kính 5.0 --%>
    <p>Diện tích HCN là <%= circleArea(5.0) %>
</body>
```

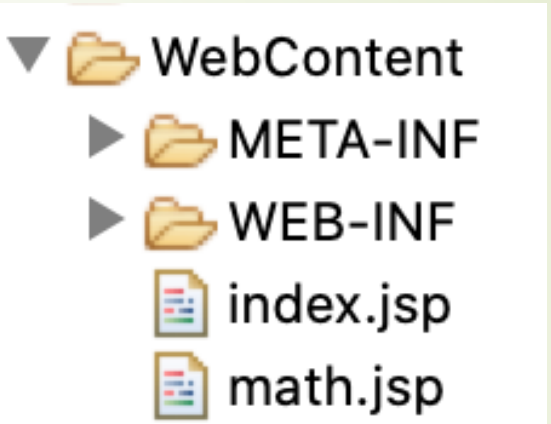
JSP Directives

- A JSP directive affects the overall structure of the servlet class. It usually has the following form: `<%@ directive attribute="value" %>`
- There are three types of directive tag:
  - <%@ page ... %>
    - Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.
  - <%@ include ... %>
    - Includes a file during the translation phase.
  - <%@ taglib ... %>
    - Declares a tag library, containing custom actions, used in the page

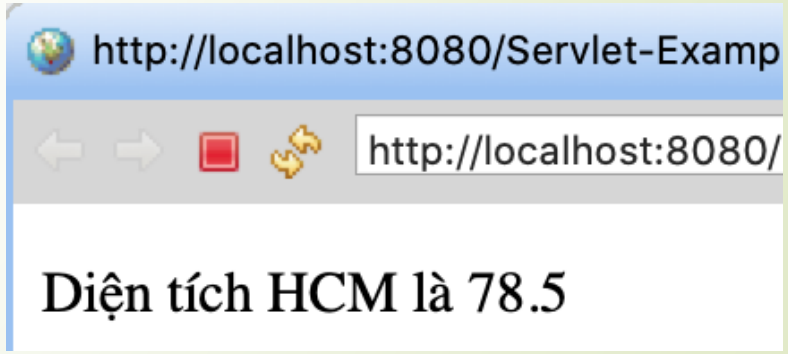
JSP Directives

math.jsp

```
1 <%!
2     double PI = 3.14;
3     double circleArea(double radius) {
4         return PI * radius * radius;
5     }
6 %>
```




```
<body>

    <%@ include file="math.jsp" %>

    <%-- Tính diện tích HCM có bán kính là 5 --%>
    <p>Diện tích HCM là <%= circleArea(5) %></p>

</body>
```



JSP Actions

- JSP actions use **constructs** in XML syntax to control the behavior of the servlet engine.
- You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.

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

JSP Actions

- Action elements are basically predefined functions.
- Following lists out the common JSP Actions:
  - jsp:include
    - Includes a file at the time the page is requested.
  - jsp:forward
    - Forwards the requester to a new page.
  - jsp:text
    - Used to write template text in JSP pages and documents.

JSP Actions

index.jsp

```jsp
<html>
    <body>
        <h1>Index Page</h1>

        <jsp:forward page="home.jsp">
            <jsp:param name="user" value="mvmanh" />
            <jsp:param name="theme" value="dark" />
        </jsp:forward>
    </body>
</html>
```

http://localhost:8080/Servlet-Example/

http://localhost:8080/Servlet-Example/

Home Page

mvmanh dark

*home.jsp

```jsp
<html>
<body>
    <h1>Home Page</h1>

    <%= request.getParameter("user") %>
    <%= request.getParameter("theme") %>
</body>
</html>
```

JSP Implicit Objects

- JSP supports nine automatically defined variables, which are also called implicit objects.
- Some of the most common objects are:
  - Request
    - This is the HttpServletRequest object associated with the request.
  - Response
    - This is the HttpServletResponse object associated with the response to the client
  - Out
    - This is the PrintWriter object used to send output to the client
  - Session
    - This is the HttpSession object associated with the request

JSP Implicit Objects

index.jsp

```
10      <%
11          String email = request.getParameter("email");
12          String password = request.getParameter("password");
13
14          if (login(email, password)) {
15              response.sendRedirect("error.jsp");
16          }
17          else {
18              session.setAttribute("email", email);
19              response.sendRedirect("home.jsp");
20          }
21      %>
22      </body>
23  </html>
```

JSP Implicit Objects

index.jsp

```
1  <html>
2      <body>
3          <%
4              String email = (String) session.getAttribute("email");
5
6              if (email == null) {
7                  out.println("You have not logged in");
8              }else {
9                  out.println("You have logged in with email: " + email);
10             }
11         %>
12
13     </body>
14 </html>
```

Decision-Making Statements

- The **if...else** block starts out like an ordinary Scriptlet, but the Scriptlet is closed at each line with HTML text included between the Scriptlet tags.

*index.jsp

```
1  <%! int day = 3; %>
2  <html>
3      <body>
4          <% if (day == 1 || day == 7) { %>
5              <p> Today is weekend</p>
6          <% } else { %>
7              <p> Today is not weekend</p>
8          <% } %>
9      </body>
10 </html>
```

index.jsp

```
1  <%! int day = 3; %>
2  <html>
3      <body>
4       <%
5            if (day == 1 || day == 7) {
6                out.println("<p> Today is weekend</p>");
7            }else {
8                out.println("<p> Today is not weekend</p>");
9            }
10      %>
11     </body>
12 </html>
```

Loop Statements

- You can also use three basic types of looping blocks in Java: **for, while, and do…while** blocks in your JSP programming.

```
<table border=1>
    <tr>
        <td>STT</td>
        <td>Name</td>
    </tr>
    <%
        for (int i = 1; i <= 5; i
            out.println("<tr>");
                out.println("<td>
                out.println("<td>
            out.println("</tr>");
        }
    %>
</table>
```


```
border=1>
r>
  <td>STT</td>
  <td>Name</td>
tr>

  <% for (int i = 1; i <= 5; i++) {%>
      <tr>
          <td><%= i %></td>
          <td>Item <%= i %></td>
      </tr>
  <% } %>

</table>
```

Servlet JSP
CRUD Examples