JSP: java server pages

Dynamic web pages. Alternative to servlet. If u have more presentation logic, then we go for jsp. Otherwise, if we have more business logic: Servlet (java class)

Html(static)/ jsp(dynamic page)

Jsp -> translated into a servlet -> compiled to java class file -> executed

public abstract interface javax.servlet.jsp.JspPage extends javax.servlet.Servlet {

// Method descriptor #5 ()V

public abstract void jspInit();

// Method descriptor #5 ()V

public abstract void jspDestroy();

}

public abstract interface javax.servlet.jsp.HttpJspPage extends javax.servlet.jsp.JspPage {

// Method descriptor #5 (Ljavax/servlet/http/HttpServletRequest;Ljavax/servlet/http/HttpServletResponse;)V

public abstract void \_jspService(javax.servlet.http.HttpServletRequest arg0, javax.servlet.http.HttpServletResponse arg1) throws javax.servlet.ServletException, java.io.IOException;

}

Jsp page implements HttpJspPage interface=> jspInit();, jspDestroy();, \_jspService

Simplest way of creating a jsp is use .jsp extension for ur html page.

Jsp Tags:

1. **Scriptlet tag:** <% %> : Inside this u can write any java code.

System.out.println: print it on the console: IDE

Browser page:

HttpServletResponse response

PrintWriter writer=response.getWriter();

writer.println()

1. **Expression tag:** <%= %>. In an expression tag never use a semi-colon.
2. **Declaration tag** <%! %>: It should be used to place variables/ methods outside ur service method

**Implicit objects**: It can give u the already created object

1. out : PrintWriter
2. response
3. request
4. session
5. page
6. application

These implicit objects are only available inside the service method.

If a page is inside WebContent, I can access it directly by making a call to it in the browser. But if u place the jsp/html file inside WEB-INF folder, u cannot access it directly from the browser

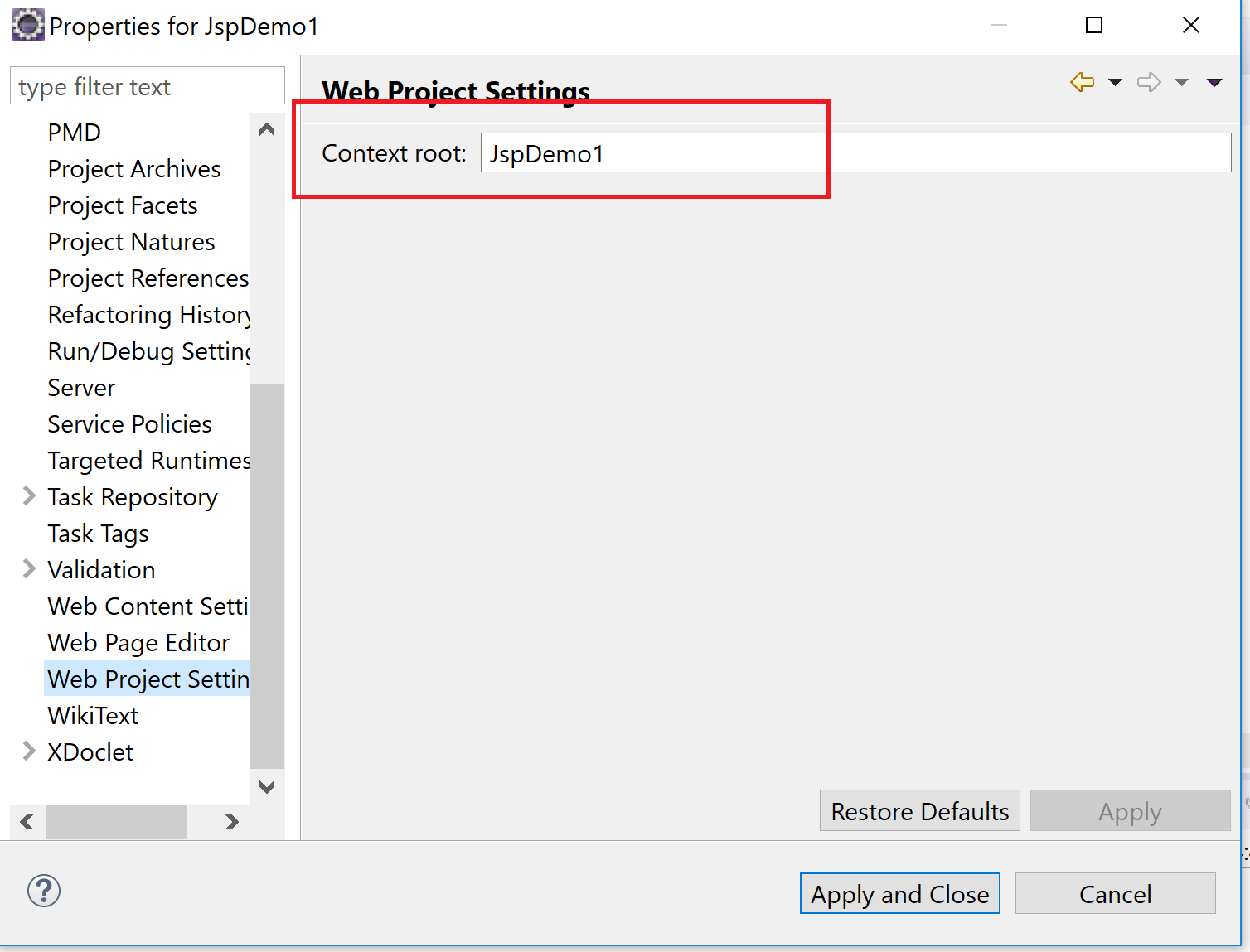
Directive tags

Action tags

Tag libraries

<http://localhost:8080/JspDemo1/Login.jsp>

Context Root: Is the name with which ur application is registered on the server.



Servlet

1. Implements Servlet
2. Extends HttpServlet
3. Extends GenericServlet

GenericServlet is an abstract class with 1 abstract method: service method. It is used for any kind of a protocol

HttpServlet: abstract class with no abstract method. It is used for ur http protocol only. It has overridden ur service method and this service method internally calls:

doGet()

doPost()

doPut()

doDelete()

based on the type of request.

@Override

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {

String username= req.getParameter("name");

String password= req.getParameter("password");

**if**(username!= **null** && username.trim()!="" && password != **null**

&& password.trim().length()!=0) {

}

}

Username would be null if the request is coming from a page which does not have this field and it would be empty if the field is present but the user did not enter a value for it.

Whatever code we write inside the body, scriptlet tag, goes inside the service method of the servlet.

**Tags:**

1. Scriptlet: any java code: service method
2. Expression: printing out after evaluating: service method
3. Declaration tag: declaring instance methods/ variables: outside service method

***Implicit objects***

PrintWriter writer= response.getWriter();

Request and response are automatically created by ur browser (http protocol)

**Sessions in a web application**

**Directives**

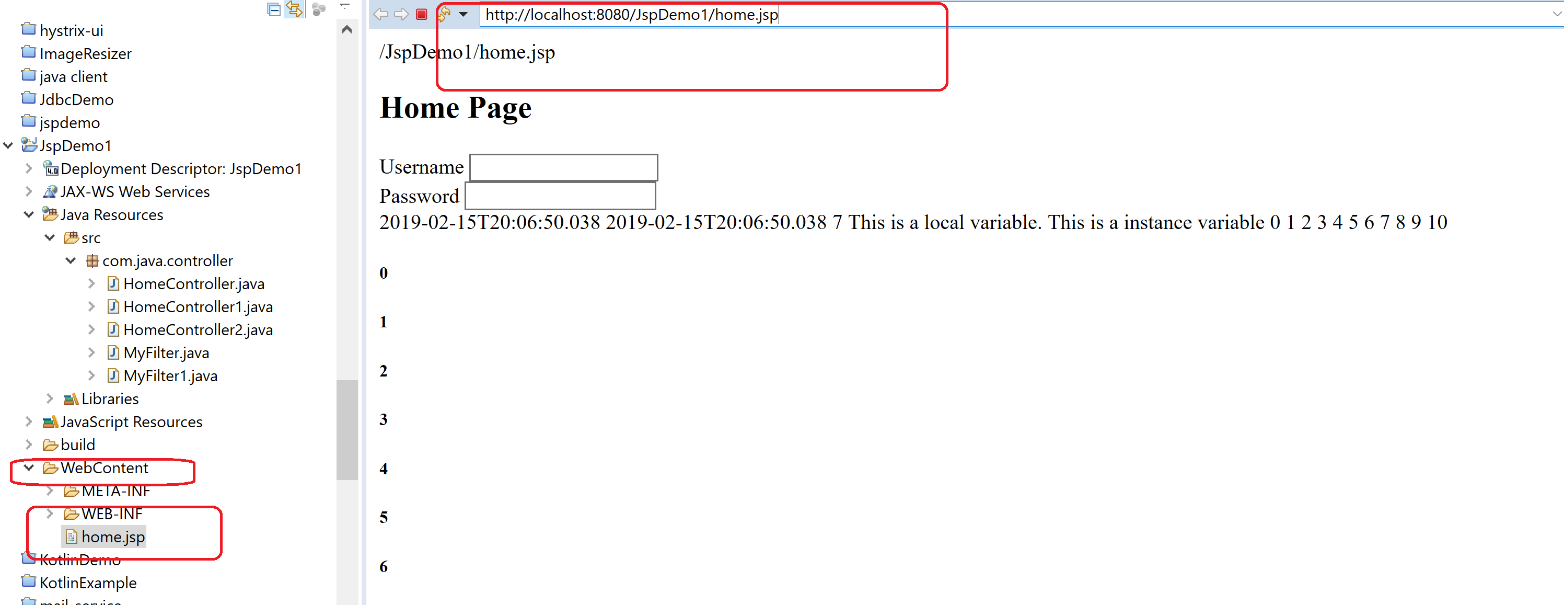
**Action tags**

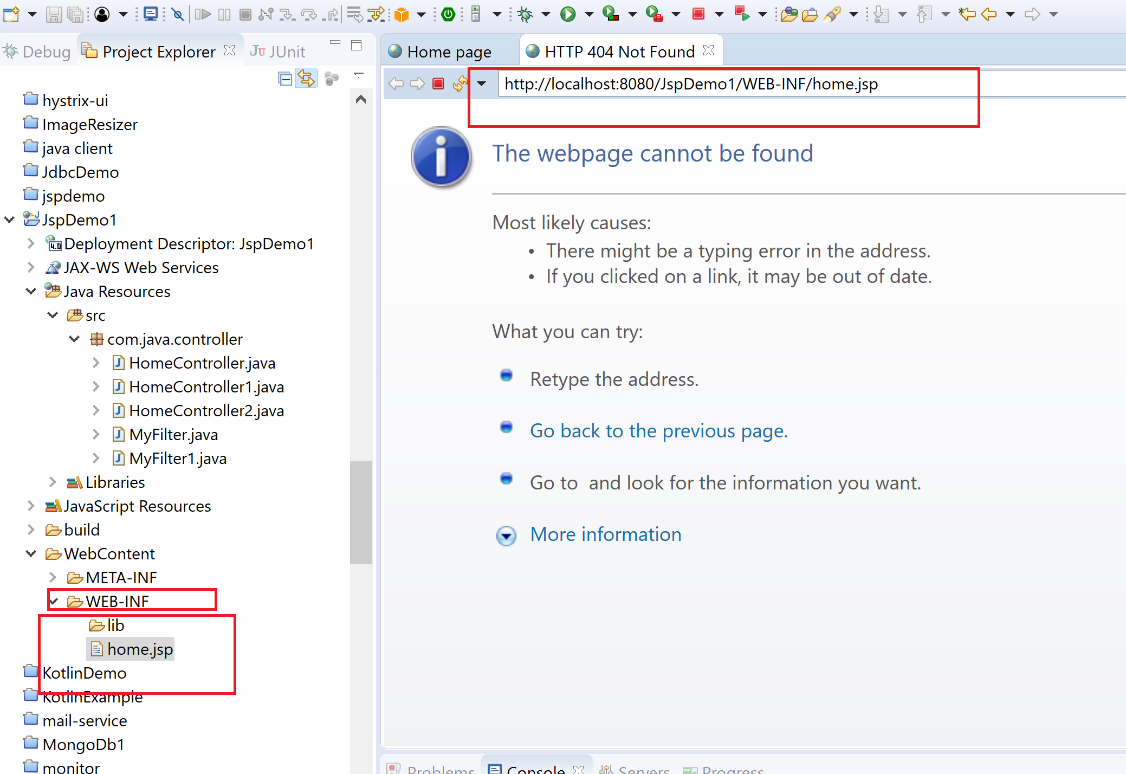
**Directive in a jsp:** These would be resolved at compile time.

1. @Page: Importing java library/ error page/ content/ scripting language/ parent class
2. @taglib: What all tags u can use
3. @include: If u want u to include some other page at compile time: headers/ footers/ jsp

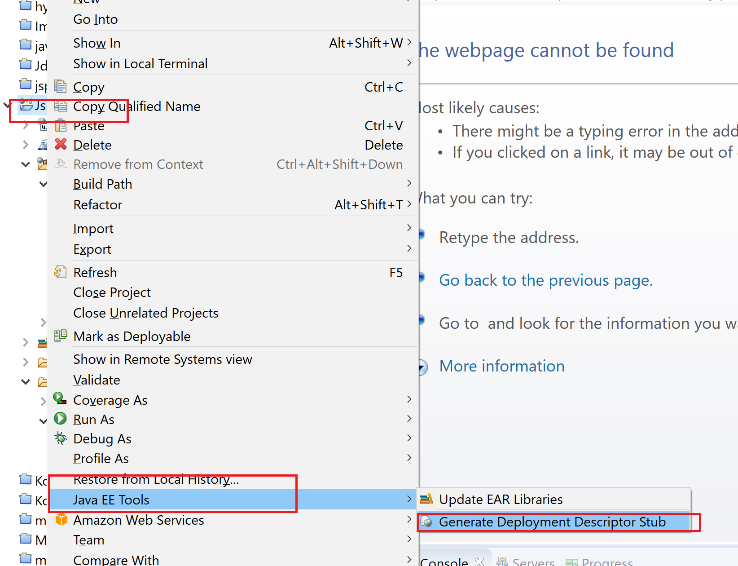
Note:

If I include my view pages inside my WebContent I can access them directly. But if I add them in a WEB-INF folder I cannot access them directly from outside.

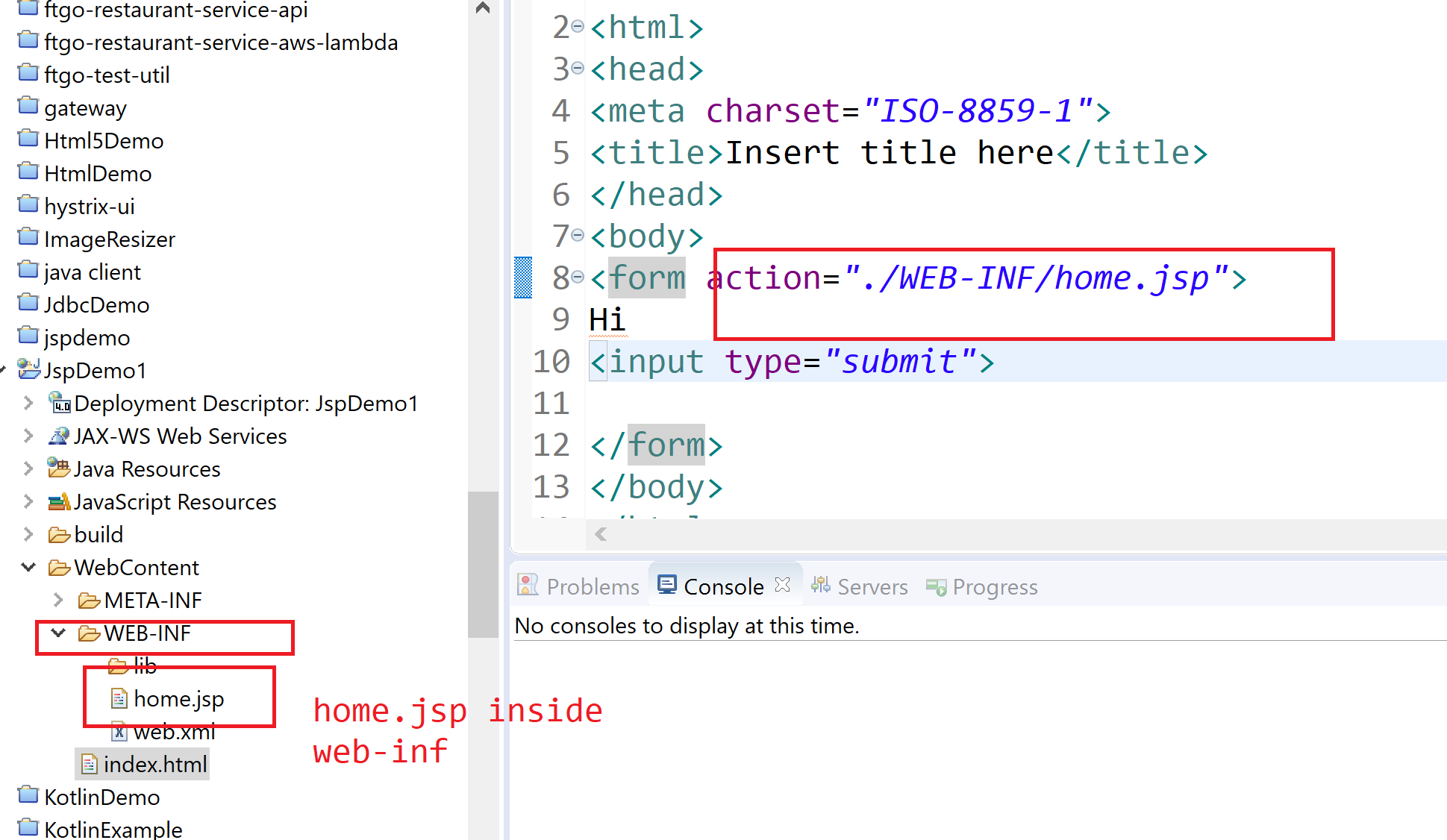


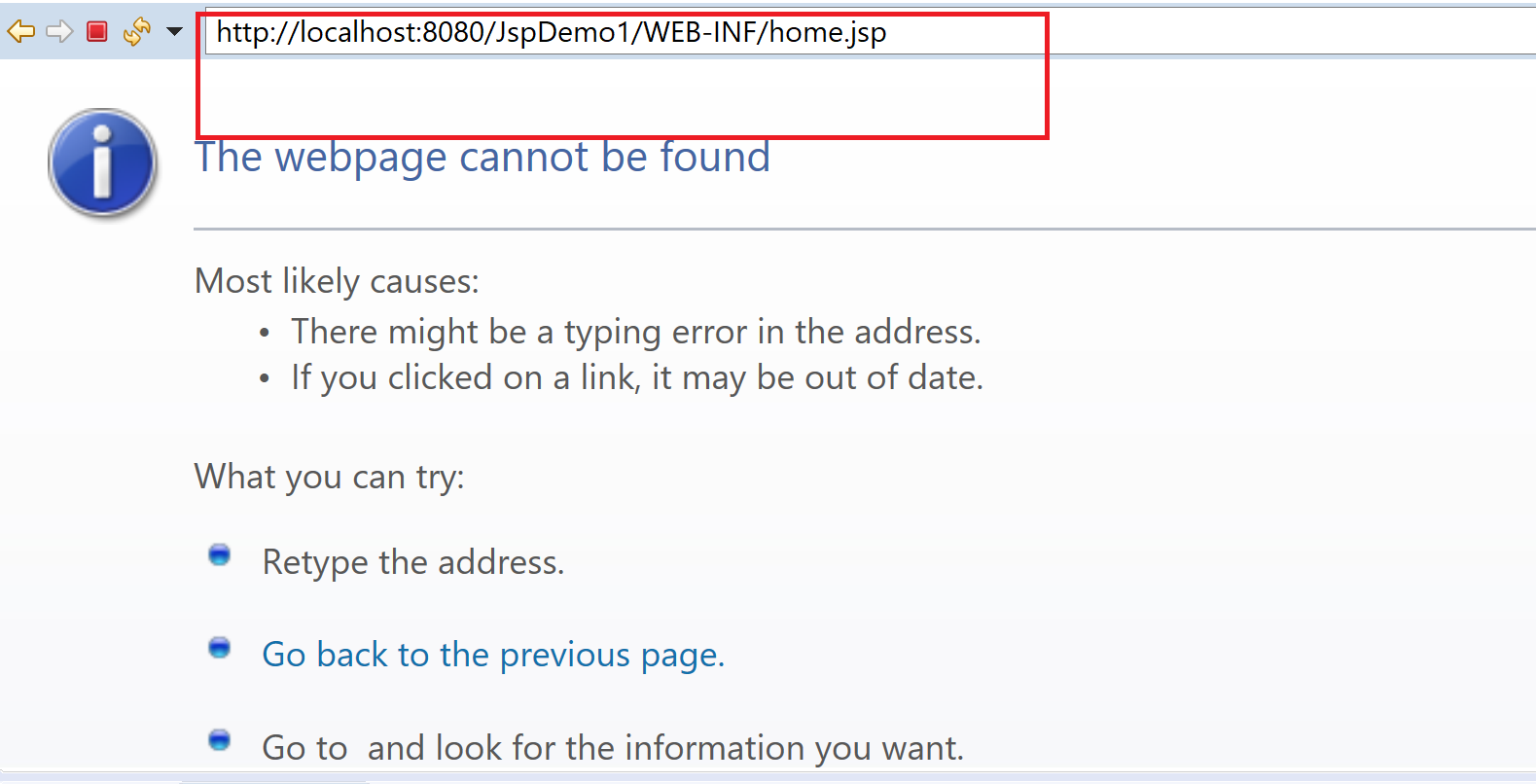


If u do not see web.xml, u can generate it by:



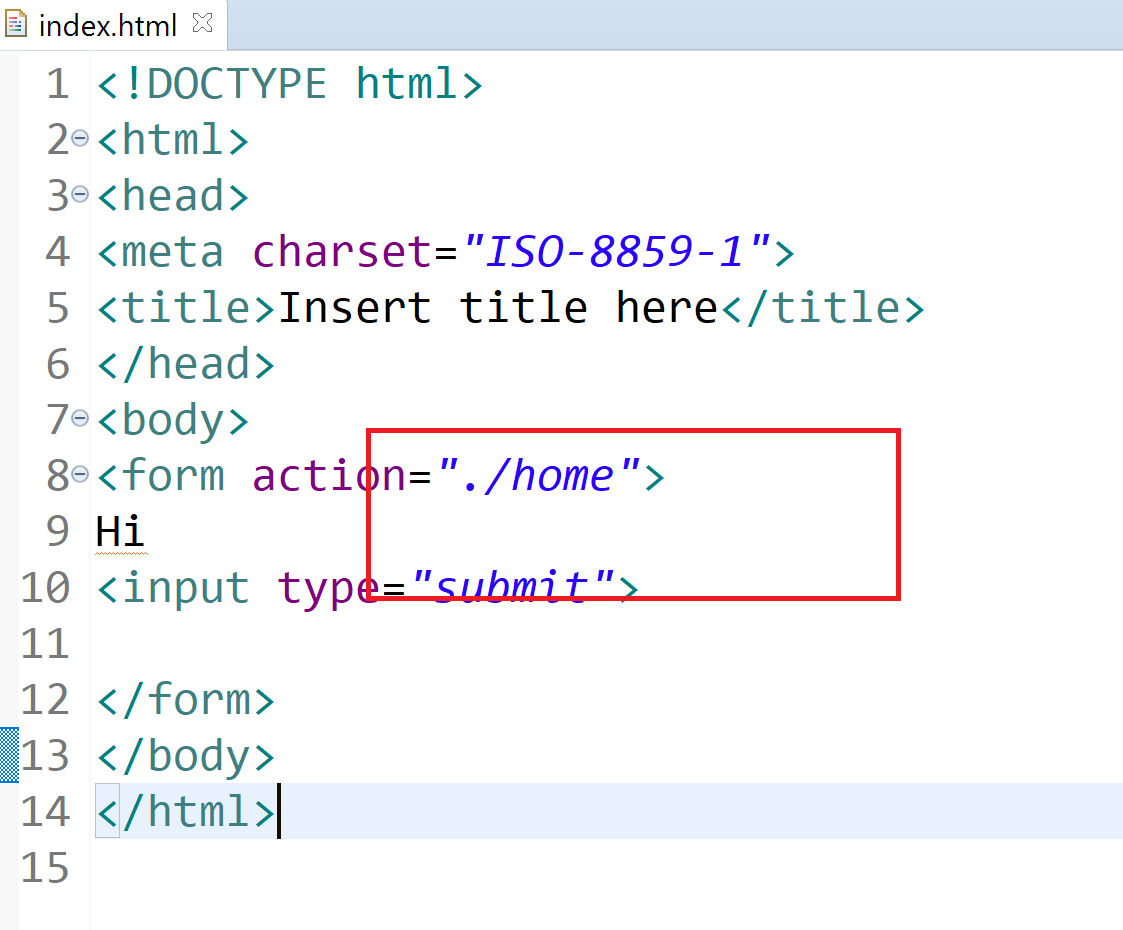
We cannot access files inside Web-INF directly





So we need to do mapping for it in web.xml:





Directive:

1. @Page:

Jsp page extends org.apache.jasper.runtime.HttpJspBase by default