Java 4 WEB

Lesson 13 – Java Server Pages

Lesson goals

- Why not servlets
- What if not servlets
- Expression Language
- Tag Libraries

Servlet drawbacks

- Not simple to maintain - business logic mixed with presentation logic
- Slow development - servlet code needs to be updated and recompiled if we have to change the look of the application.
- Too much non-reusable copy-paste
- Servlet can be viewed as "***HTML inside Java***"

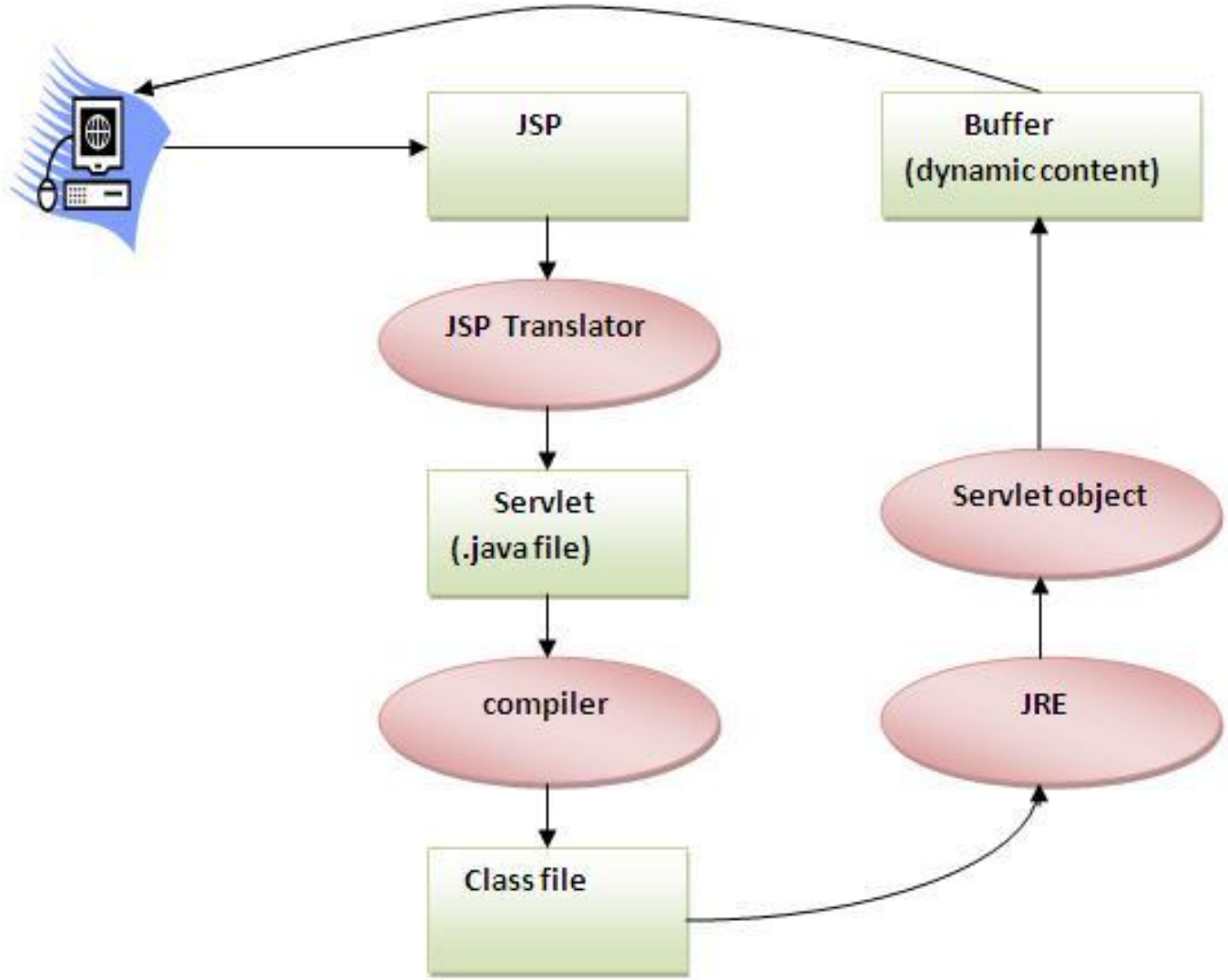
JSP (Java Server Page)

- JSP is high-level abstraction of Java Servlets
- JSP is a text document that contains two types of text:
  - *static data* (HTML, SVG, WML, and XML)
  - JSP elements, which construct *dynamic content*
- JSPs servlet is cached and re-used until the original JSP is modified

JSP Example

```
<%@ page contentType="text/html; charset=UTF-8"%>
<html>
    <head>
        <title>First JSP</title>
    </head>
    <body>
        JSP Page
    </body>
</html>
```

JSP life cycle

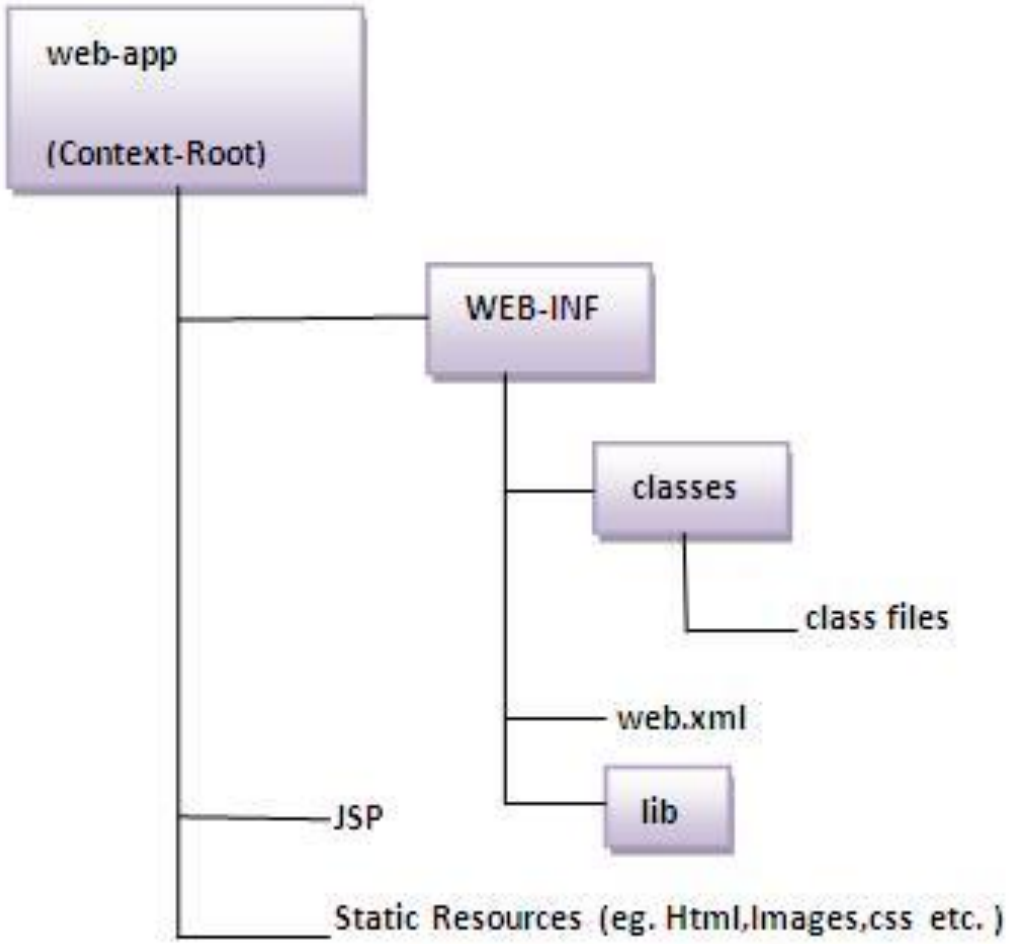


JSP vs Raw Servlet

- Extension to Servlet (supplement each other)
- Easier to maintain
- Faster Development: no necessity to recompile and redeploy
- Less code than Servlet

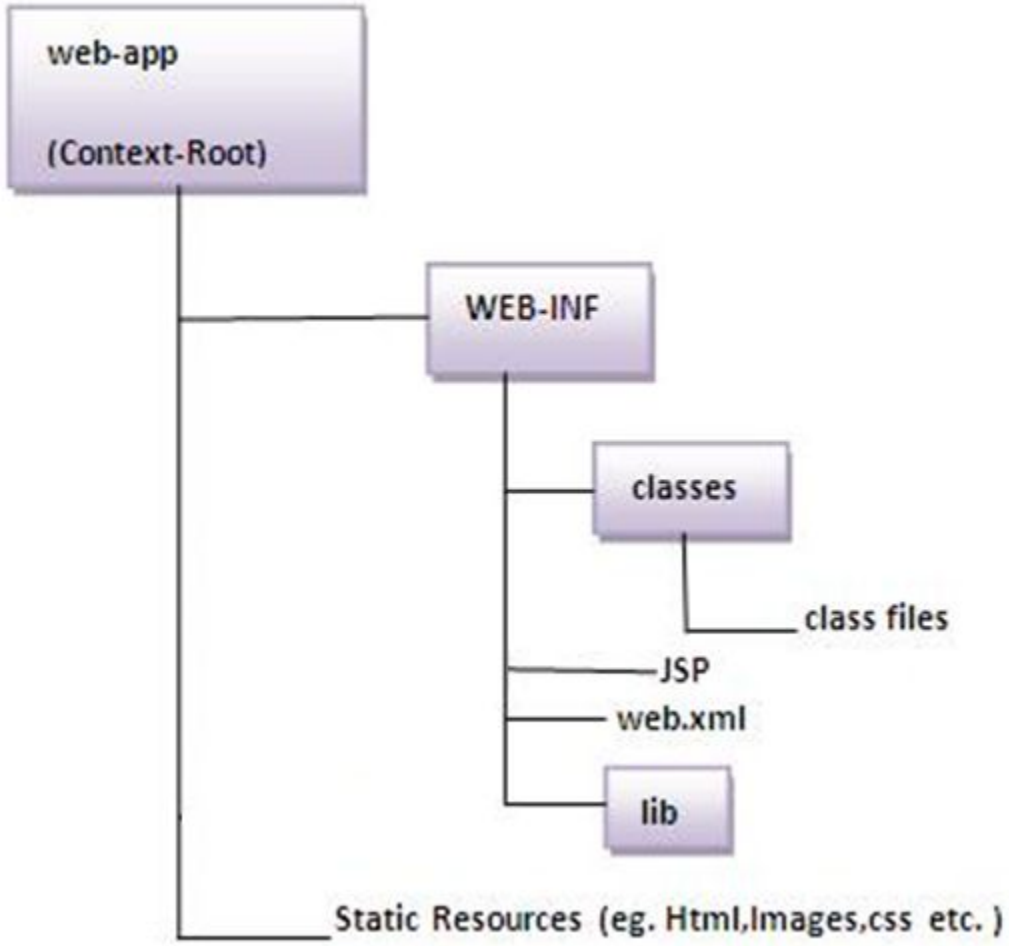
Folders structure with direct access to jsp

http://*localhost:8080/page.jsp* - available



Folders structure without direct access to jsp

http://*localhost:8080/page.jsp* - non available. Requires servlet mapping



JSP Example with Java inside HTML

1. JSP Scriptlet is used to used to execute java source code in JSP

```
<%
    int i=0;
%>
```

2. JSP Expression - evaluates a single Java expression and display its result.

Current Time: `<%= java.util.Calendar.getInstance().getTime()  %>`

JSP Example with Java inside HTML

3. Declaration tag <%! field or method declaration %>

```
<%!
    int square(int a){ return a * a;}
%>
Square : <%= square(10) %>
```

4. Directives tag <%@ JSP directives %>

```
<%@ page contentType="text/html; charset=UTF-8"%>
<%@ page import="java.util.*" %>
<%@ include file="some-another-part.jsp" %>
```

JSP Example with Java inside HTML (scriptlet)

```
<%@ page contentType="text/html; charset=UTF-8"%>
<html>
    <head>
        <title>First JSP</title>
    </head>
    <body>
        <%
            double num = Math.random();
            if (num > 0.5) {
        %>
            <h2>You'll be geek!</h2><p>(<%= num %>)</p>
        <%
            } else {
        %>
            <h2>Well, you won't be geek ... </h2><p>(<%= num %>)</p>
        <%
            }
        %>
    </body>
</html>
```

MVC

Architecture of building applications is called MVC

- Model - classes of business logic and long-term storage
- View - JSP pages
- Controller - servlet.

Using JSP with Servlet

```java
@WebServlet(name = "userPageServlet", urlPatterns = "/userpage")
public class UserServlet extends HttpServlet {
    @Override
    protected void doGet(
            HttpServletRequest req, HttpServletResponse resp
    ) throws ServletException, IOException {
        req.setAttribute("username", "Johny");
        req.getRequestDispatcher("/WEB-INF/pages/userIntro.jsp").forward(req, resp);
    }
}
```

UserServlet.java

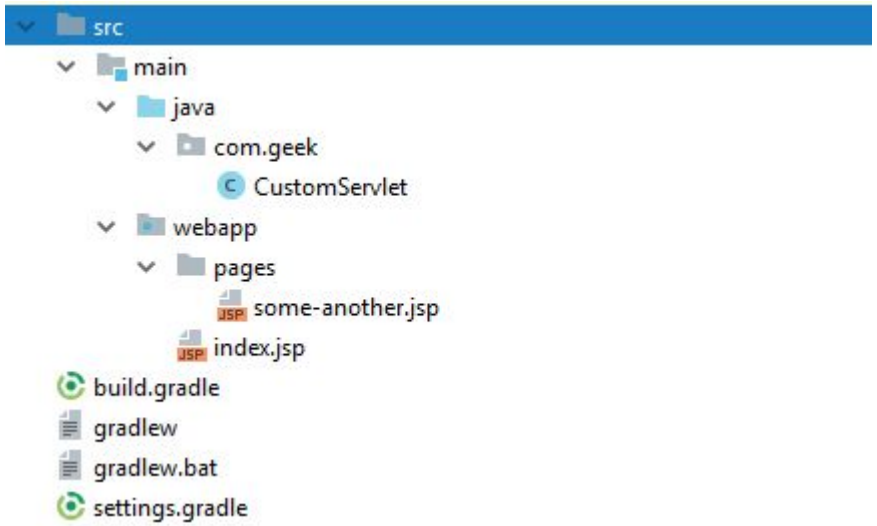
```jsp
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
    <head>
        <title>Second JSP</title>
    </head>

    <body>
        Hello ${username}
    </body>
</html>
```

userIntro.jsp

index.jsp vs custom.jsp

```
@WebServlet(name = "custom", urlPatterns = "/custom-page")
public class CustomServlet extends HttpServlet {
    @Override
    protected void doGet(
        HttpServletRequest req, HttpServletResponse resp
    ) throws ServletException, IOException {
        req.getRequestDispatcher("/pages/some-another.jsp").forward(req, resp);
    }
}
```



JSP expression language (EL)

```
${username} , ${user.name}
<p> Hello, ${author.name}</p>, <some:tag value="${expr}"/>
```

Bean is searched by container in the next order:

- page
- request
- session
- application scopes
- otherwise returns null

JSP Implicit Objects

Object	Type
out	JspWriter
request	HttpServletRequest
response	HttpServletResponse
config	ServletConfig
application	ServletContext
session	HttpSession
pageContext	PageContext
page	Object
exception	Throwable

Tag libraries

Advantages of using Tag Libs:

- get rid of "scriptlets"
- a simple HTML-like syntax
- JSP code can be modified by HTML developers
- code reuse

JSP Tags syntax

```
<someTagLib:tag attr="…"…> body </someTagLib:tag>
```

or if no body

```
<someTagLib:tag attr="…" …/>
```

JSP Tags types

1. Predefined (start with "jsp:")

```
<jsp:include page="hello.jsp"/>
```

2. External (custom tag libraries).

```
<c:set var="username" value="Johny" />
<c:out value="${username}" />
```

JSP Tag Example

```
<%@ page contentType="text/html; charset=UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<h2>Hello,
    <c:choose>
        <c:when test="${not empty param.name}">
            <c:out value="${param.name}"/>
        </c:when>
        <c:otherwise>
            <c:out value="anonymous"/>
        </c:otherwise>
    </c:choose>
</h2>
<jsp:include page="footer.jsp"/>
```

JSTL

The standard JSP tag library (JSL) is an extension of the JSP specification that adds a JSP tag library for general purposes, such as parsing XML data, conditional processing, creating loops, and supporting internationalization.

JSTL Examples

- **Core Tags** - basic tags, provide iteration, exception handling, url, forward and redirect response, etc.
- **Formatting and Localization Tags** - formatting tags, provide opportunities for formatting Numbers, Dates and support for i18n localization and resource bundles.
- **SQL Tags** - tags for working with SQL, support for working with databases like MySQL, Oracle, etc.
- **XML Tags** - tags for working with XML documents. For example, for parsing XML, converting XML data, and executing XPath expressions.
- **JSTL Functions Tags** - function-tags for processing strings, provides a set of functions that allow you to perform various operations with strings, etc. For example, by concatenating or splitting strings.

JSTL core tags

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

```
<c:out>
<c:remove>
<c:if>
<c:when>
<c:import>
<c:set>
<c:choose>
<c:otherwise>
<c:forEach>
<c:param>
```

JSTL formatting tags

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
```

```
<fmt:formatNumber/>
<fmt:formatDate>
<fmt:setTimezone>
<fmt:requestEncoding>
<fmt:parseNumber>
<fmt:parseDate>
<fmt:setLocale>
<fmt:timeZone>
<fmt:message>
```

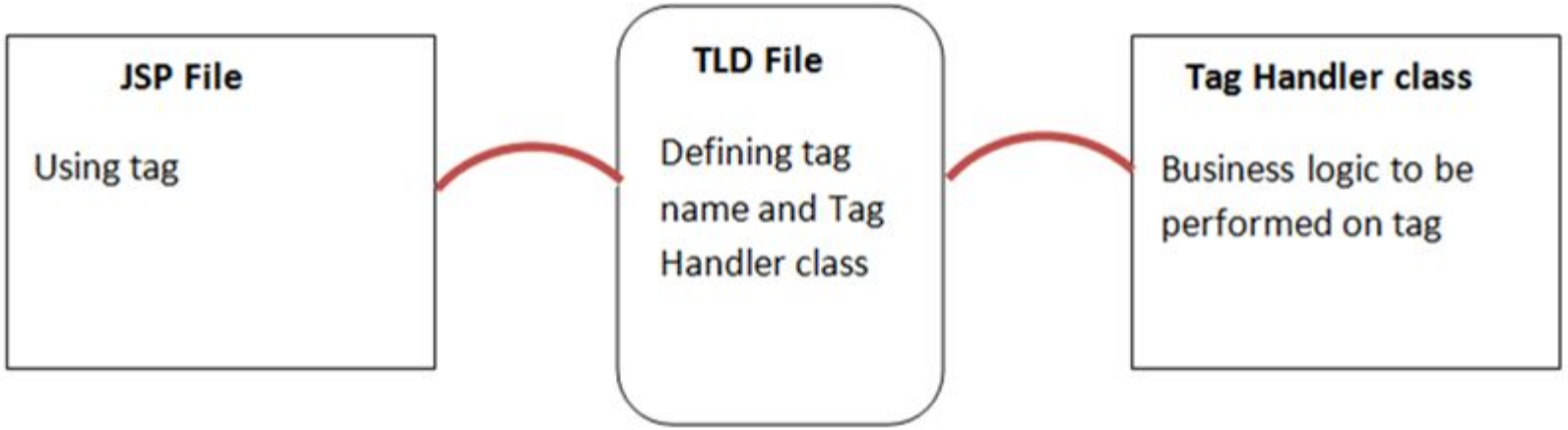
JSTL other tags

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

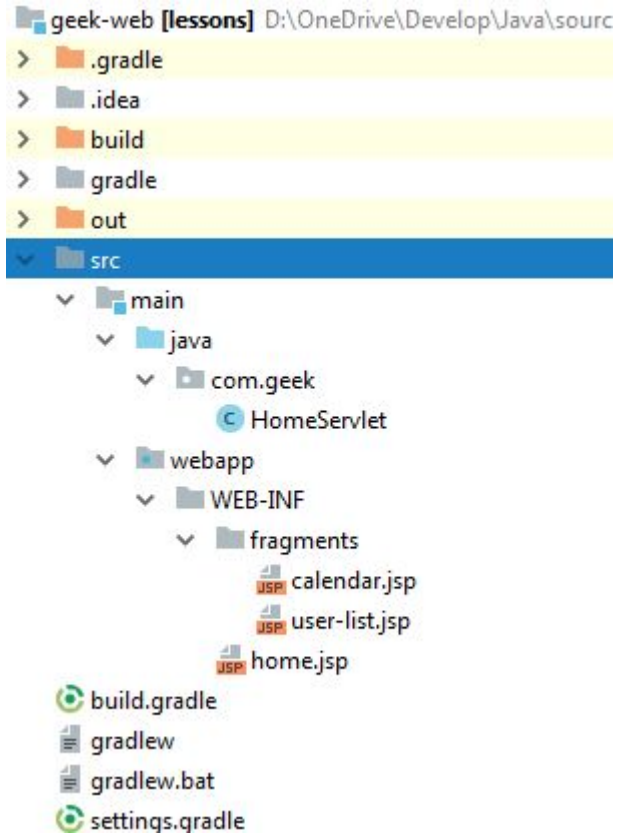
```
${fn:contains()}
${fn:endsWith()}
${fn:indexOf()}
${fn:split()}
${fn:substring()}
${fn:toUpperCase()}
${fn:escapeXml()}
${fn:join()}
${fn:replace()}
${fn:startsWith()}
${fn:toLowerCase()}
${fn:trim()}
```

Creating custom tag library

- Extend classes TagSupport or BodyTagSupport (JSP Custom Tag Handler) `group: 'javax.servlet.jsp', name: 'jsp-api', version: '2.0'`
- Write Tag Lib Definition file (my-tag-lib.tld)
- Connect your tag library into JSP file and use it



Example: structure + servlet




```java
@WebServlet(name = "homeServlet", urlPatterns = "/home")
public class HomeServlet extends HttpServlet {
    @Override
    protected void doGet(
        HttpServletRequest req, HttpServletResponse resp
    ) throws ServletException, IOException {
        req.setAttribute("users", Arrays.asList("Rikki", "Tommy", "Johny"));
        req.getRequestDispatcher("/WEB-INF/home.jsp").forward(req, resp);
    }
}
```

Example: home.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>


<html>
    <head><title>Home</title></head>
    <body>
        <jsp:include page="fragments/calendar.jsp"/>
        <br><br>

        <jsp:include page="fragments/user-list.jsp"/>
    </body>
</html>
```

Example: calendar.jsp

```
<%@ page import="java.util.Date" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<c:set var="date" value="<%=new Date()%>"/>
Today is <fmt:formatDate value="${date}" pattern="YYYY-MM-dd HH:mm:ss"/>
```

Example: user-list.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>


<c:forEach var="user" items="${users}">
    <br>Username: <c:out value="${user}"/>
    <c:choose>
        <c:when test="${user == 'Tommy'}">
            need Jerry
        </c:when>
        <c:when test="${user == 'Rikki'}">
            need Tikki
        </c:when>
        <c:otherwise>
            need a gun
        </c:otherwise>
    </c:choose>
</c:forEach>
```

Literature

1. Java EE tutorial (3-9)
2. JSP Tutorial
3. Introduction to Java Server Pages

Homework Task 1

1. Implement a simple editing form on Servlet-JSP (JSTL) data stored in the session.
2. Implement output of data using Custom Taglib (keep "add" inside the JSP)
3. If any error happens – log and redirect user to custom error page
4. Add request blocking filter. Only user with Google Chrome 65 or later can access site – otherwise block requests and show error page.
5. Add request logging filter. Log endpoints path and total execution time. Should measure all actions (even when request blocked)
6. Add request blocking filter. Deny all operations if time between 1AM-7AM. Microsoft Edge users should never come here and be stopped by user-agent filter.
7. Attributes lists should not be shared between two browsers

Homework 1

Name	Value	Action
Cat	Tom	delete
	Dave	delete
	Kate	delete
Dog	Bim	delete
	Rex	delete
Mouse	Jerry	delete
	Mikky	delete
		add