JAVA FULL STACK DEVELOPMENT PROGRAM

Session 11: JSP

OUTLINE

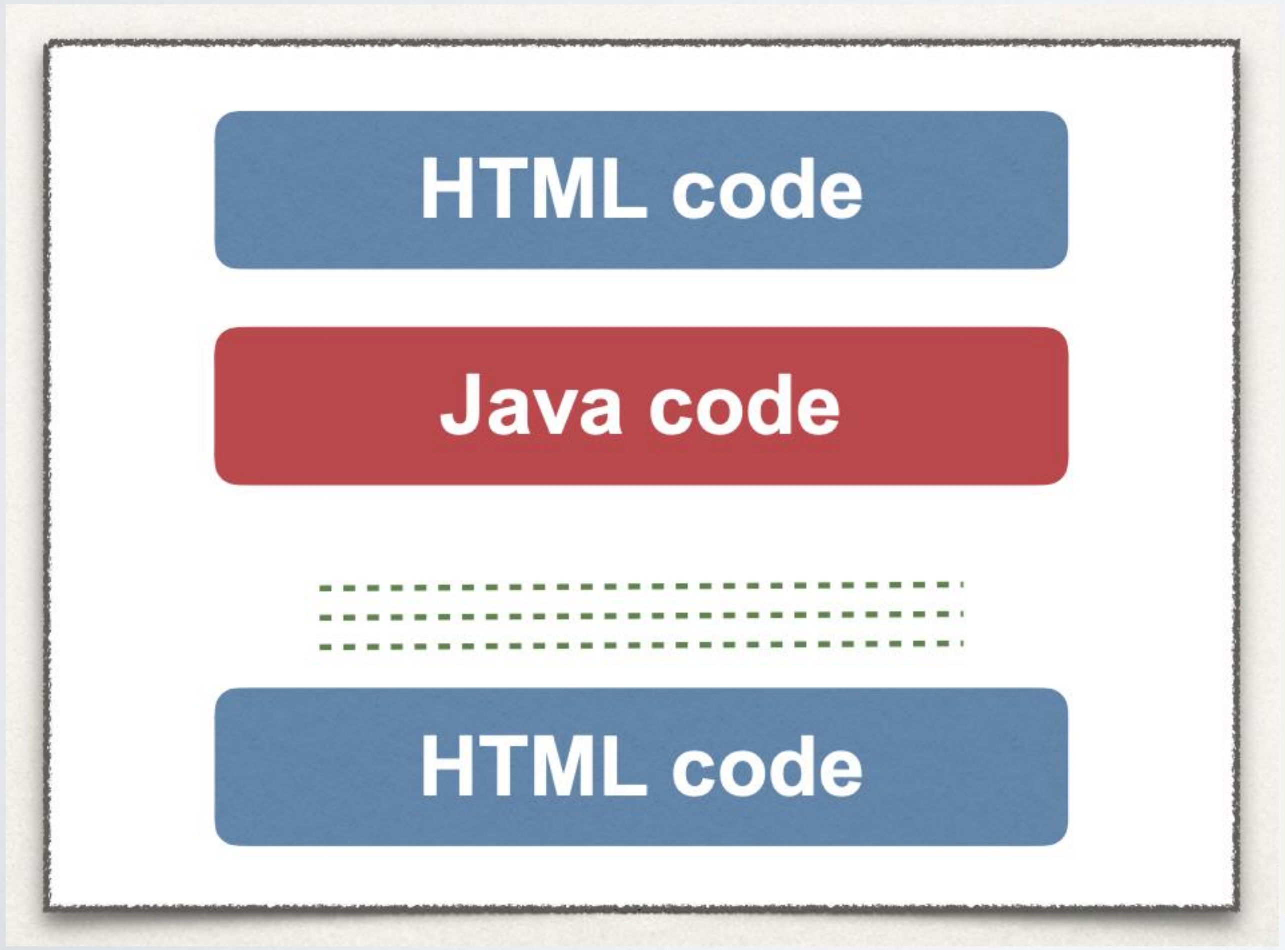
- Introduction to JSP
- JSP Architecture & Life Cycle
- JSP Elements
- JSP Build-in Object
- JSP Action
- MVC

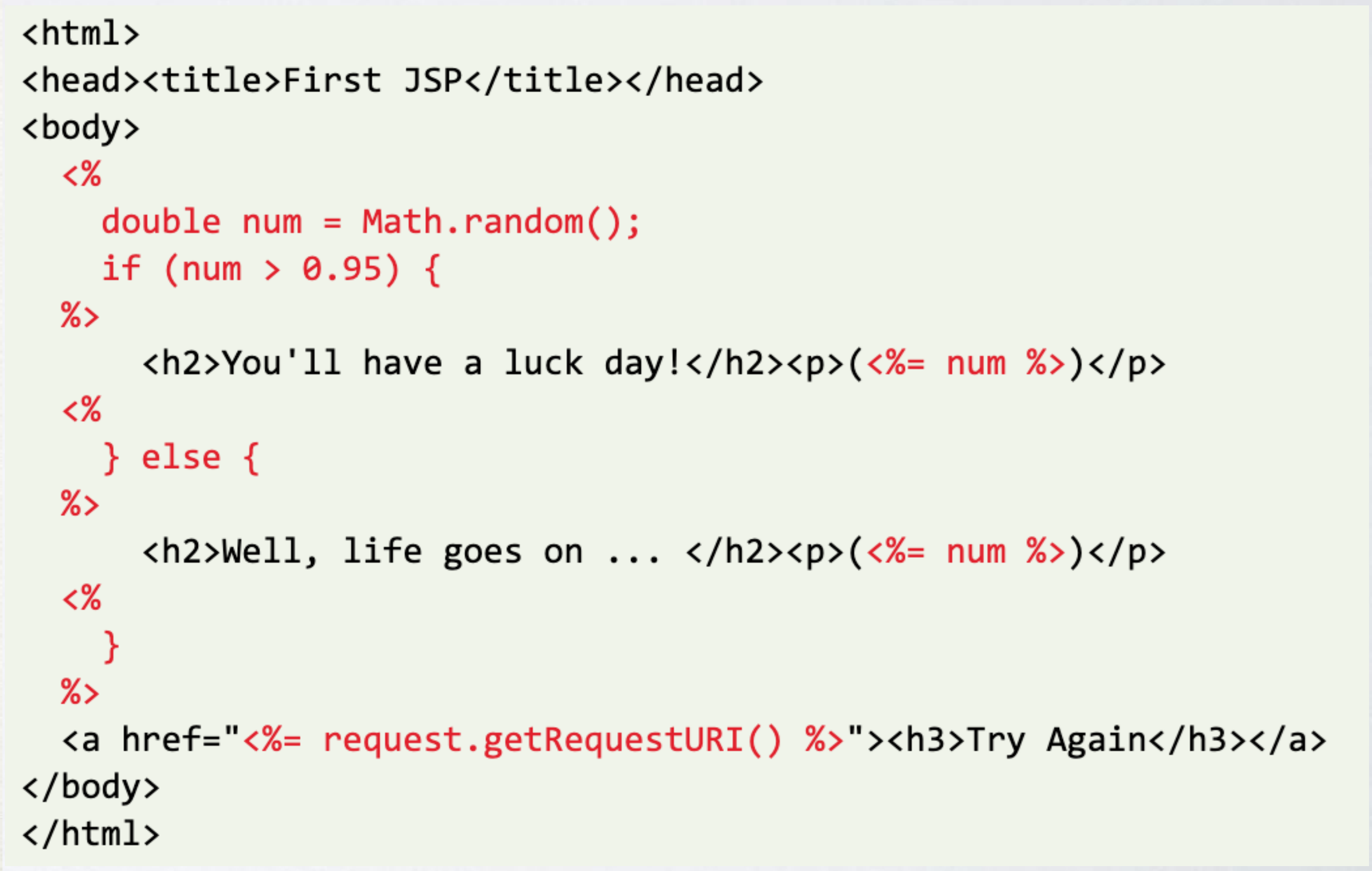
INTRODUCTION TO JSP

- JSP — Java Server Pages is a technology which is used to develop web pages by inserting Java code into the HTML pages by making special JSP tags
- The JSP tags which allow java code to be included into it are <% ----java code---- %>
- Dynamic content includes some fields like dropdown, checkboxes, etc. whose value will be fetched from the database.
- It can also be used to access JavaBeans objects.
- It can be used for separation of the view layer with the business logic in the web application.

INTRODUCTION TO JSP

- JSP is processed on the server
- Results of Java code included in HTML returned to browser

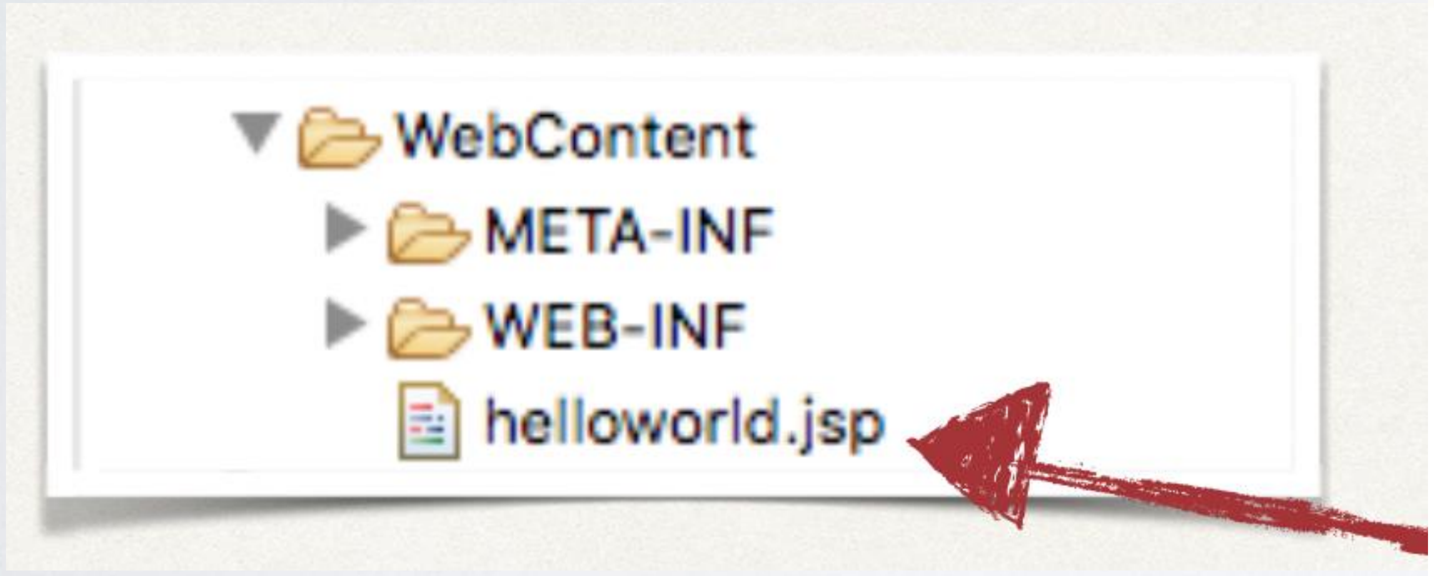


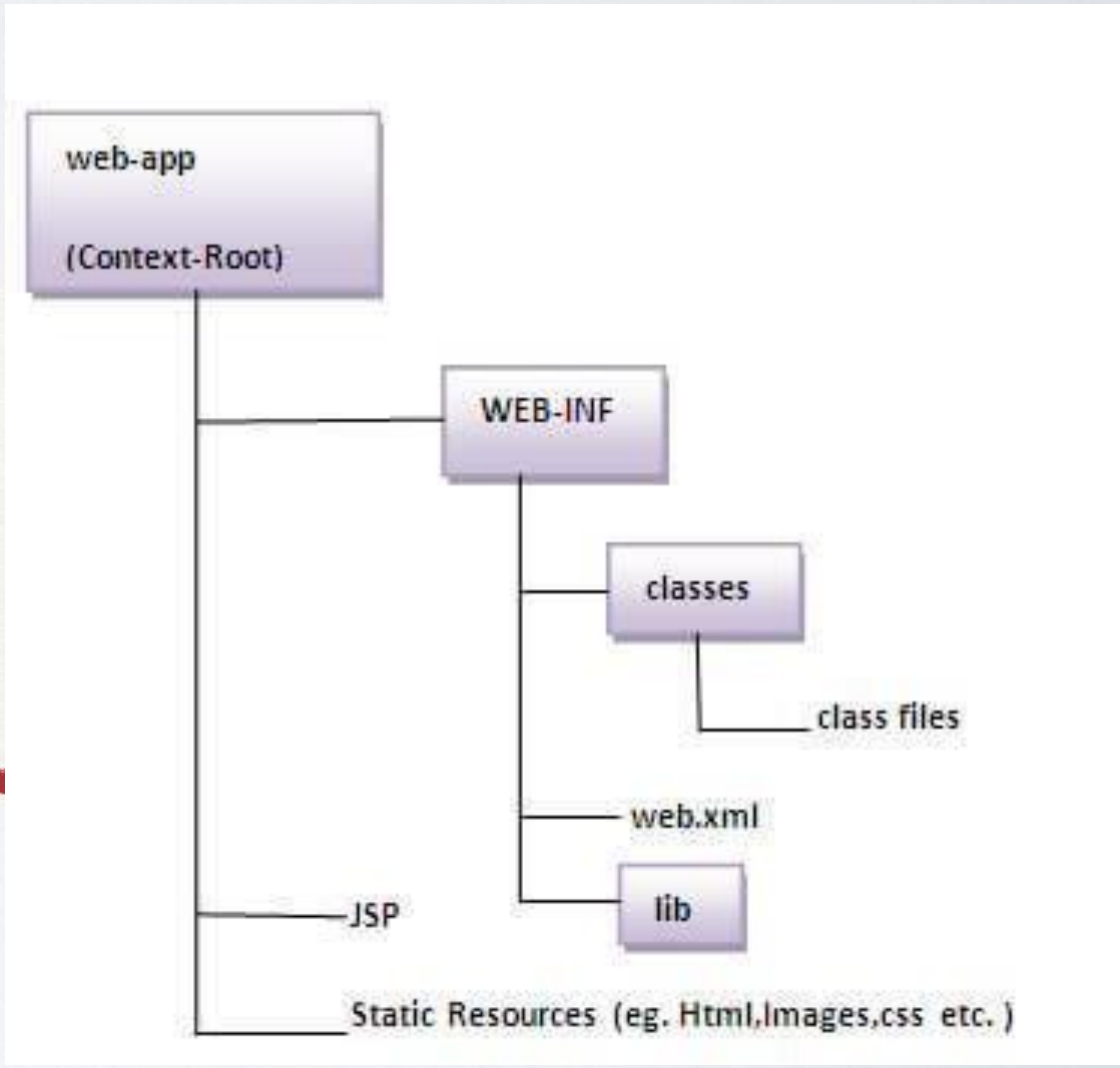


```
<html>
<head><title>First JSP</title></head>
<body>
  <%
    double num = Math.random();
    if (num > 0.95) {
  %>
      <h2>You'll have a luck day!</h2><p>(<%= num %>)</p>
  <%
    } else {
  %>
      <h2>Well, life goes on ... </h2><p>(<%= num %>)</p>
  <%
    }
  %>
  <a href="<%= request.getRequestURI() %>"><h3>Try Again</h3></a>
</body>
</html>
```

WHERE TO PLACE JSP

- The JSP file goes in your WebContent(webapp) folder
- Must have .jsp extension

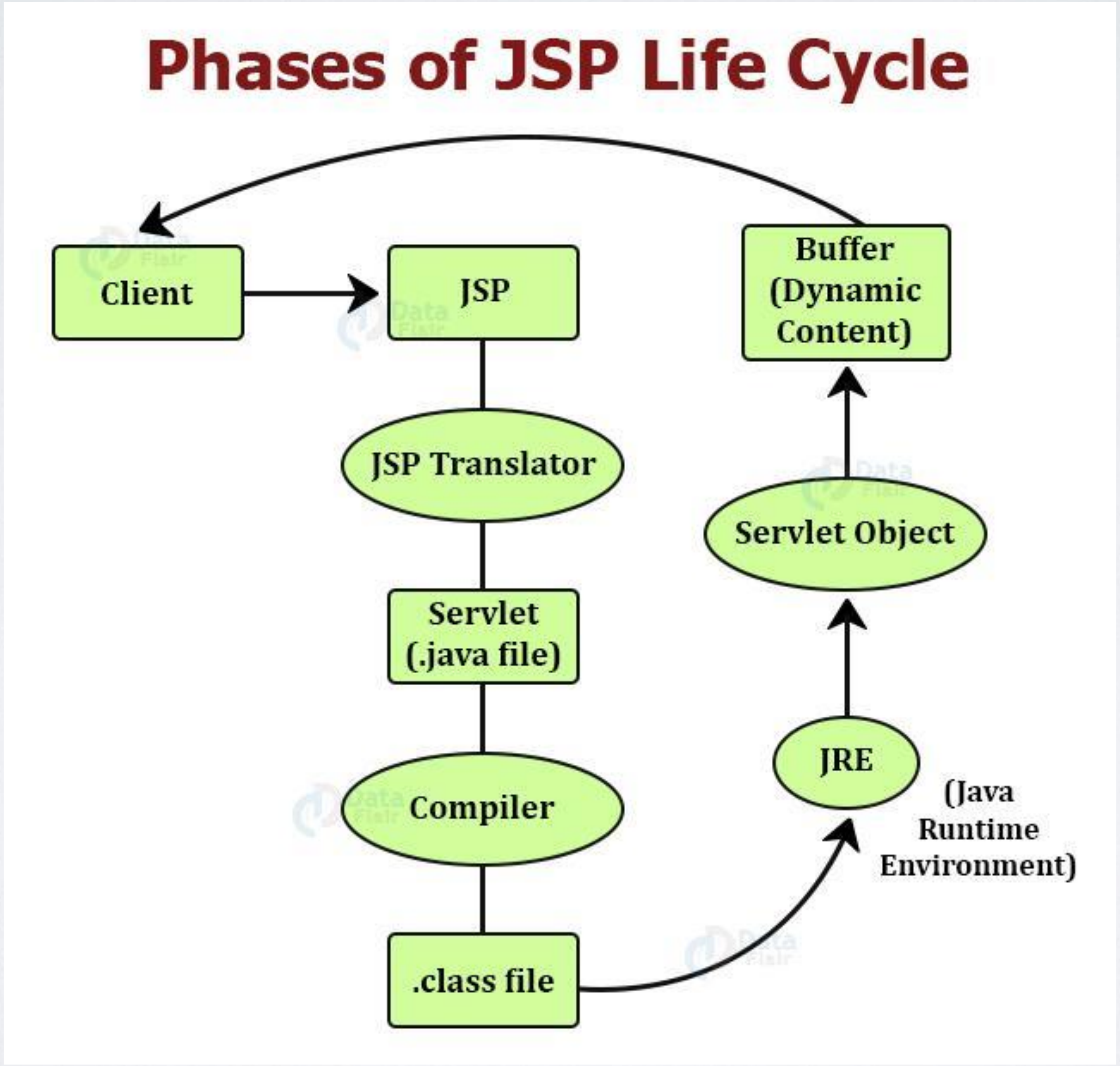




JSP LIFE CYCLE

- Translation of JSP page
- Compilation of JSP page (Compilation of JSP page into _jsp.java)
- Class Loading (_jsp.java is converted to class file _jsp.class)
- Instantiation (Object of generated servlet is created)
- Initialization (_jspinit() method is invoked by container)
- Request Processing (_jspservice() method is invoked by the container)
- Destroy (_jspDestroy() method invoked by the container)

JSP LIFE CYCLE



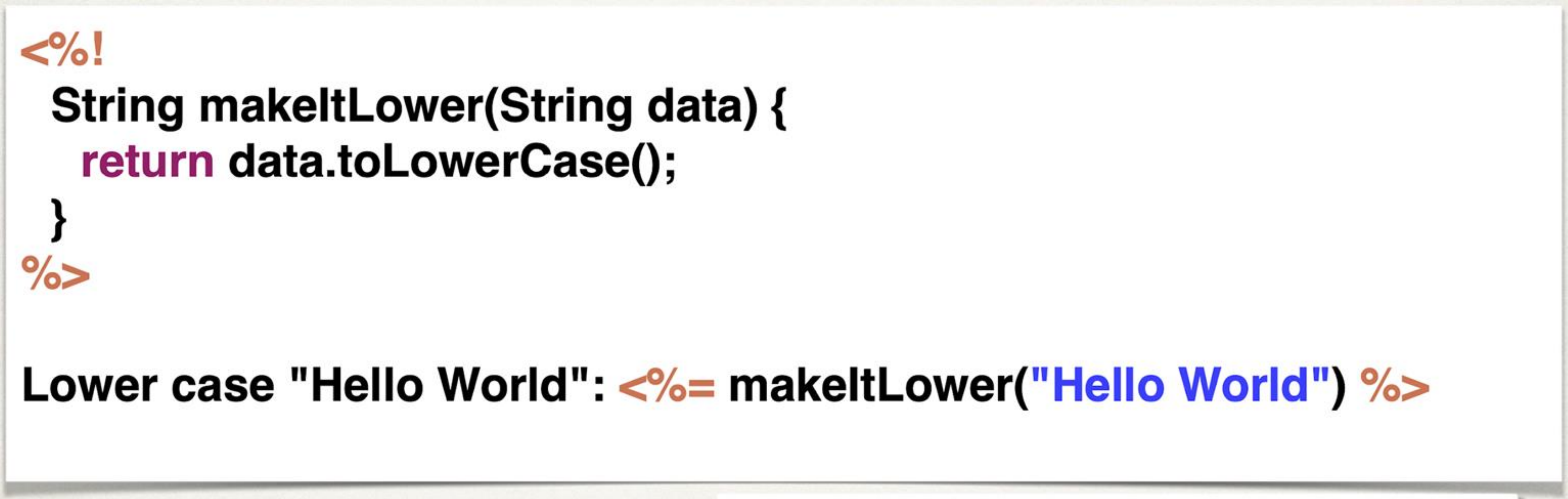
JSP ELEMENTS

- There are five common elements in JSP
  - Declaration
  - Scriptlet
  - Expression
  - Comment
  - Directive

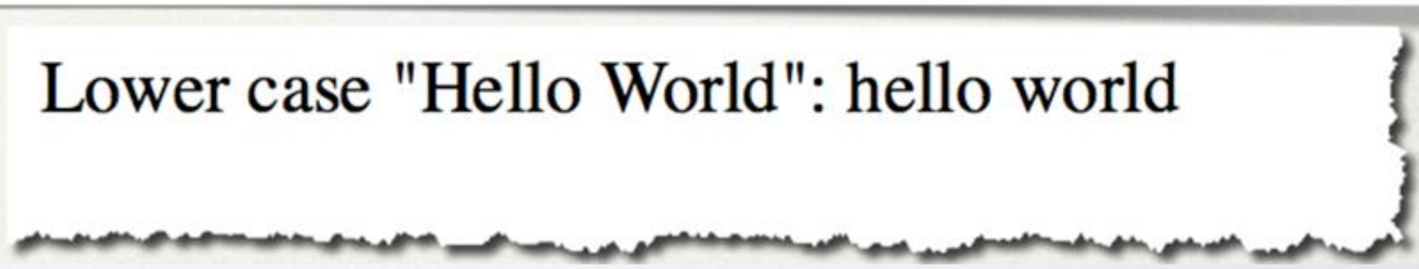
JSP DECLARATION

- A declaration tag is a piece of Java code for declaring variables, methods and classes.
- If we declare a variable or method inside declaration tag it means that the declaration is made inside the servlet class but outside the service method.
- We can declare a static member, an instance variable (can declare a number or string) and methods inside the declaration tag.
- Syntax
  - *<%! Declare var %>*

JSP DECLARATION



```
<%!
  String makeItLower(String data) {
    return data.toLowerCase();
  }
%>

Lower case "Hello World": <%= makeItLower("Hello World") %>
```

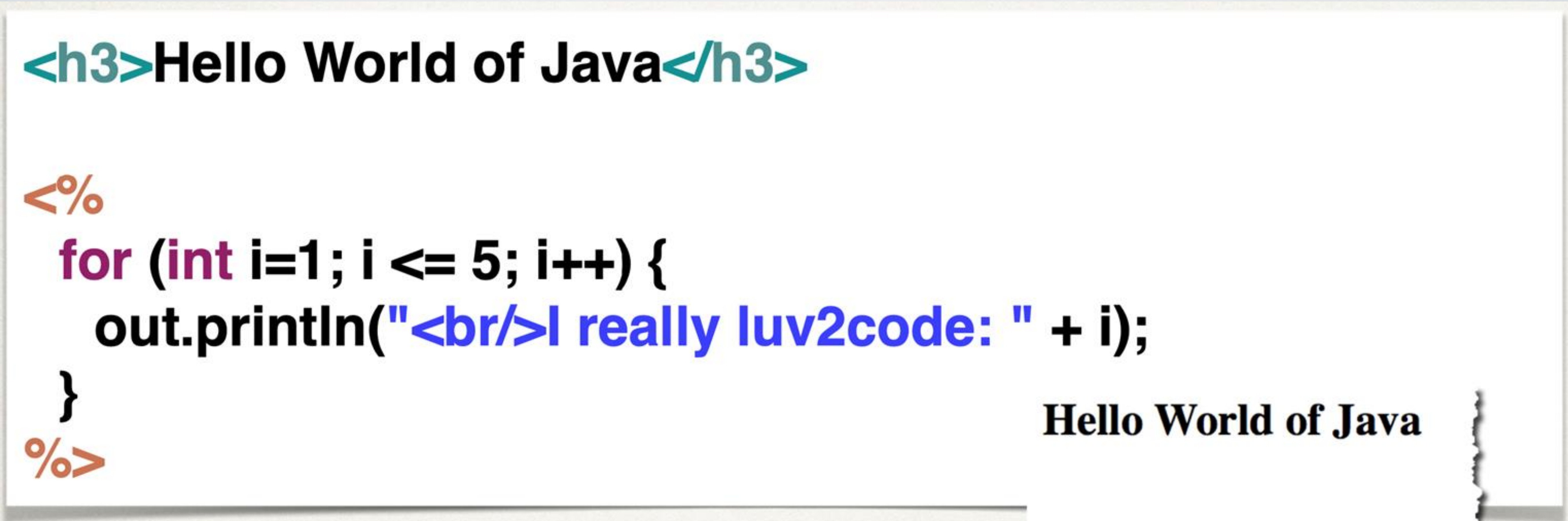


```
Lower case "Hello World": hello world
```

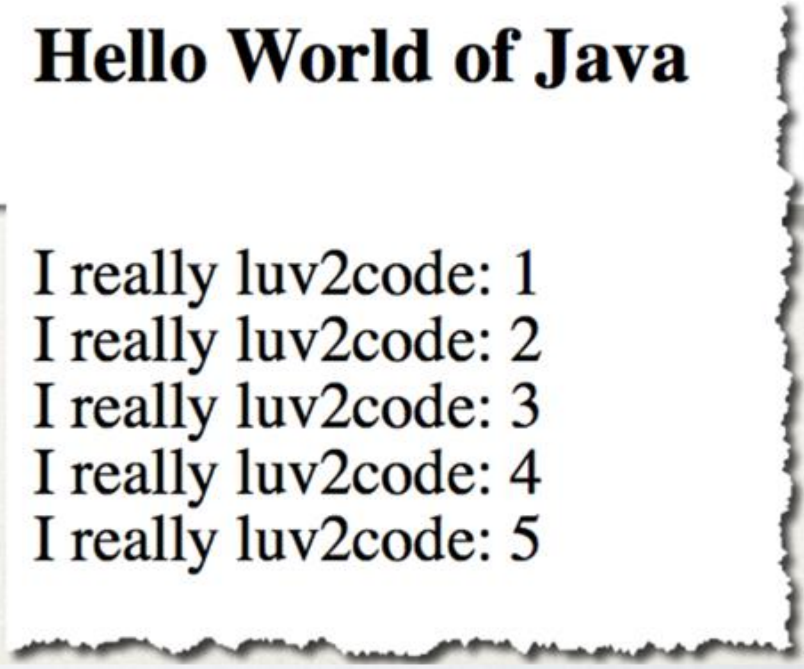
JSP SCRIPTLET

- Scriptlet tag allows to write Java code into JSP file.
- JSP container moves statements in _jspservice() method while generating servlet from jsp.
- For each request of the client, service method of the JSP gets invoked hence the code inside the Scriptlet executes for every request.
- A Scriptlet contains java code that is executed every time JSP is invoked.
- Syntax
  - *<% java code %>*

JSP SCRIPTLET



```
<h3>Hello World of Java</h3>


<%
  for (int i=1; i <= 5; i++) {
    out.println("<br/>I really luv2code: " + i);
  }
%>
```

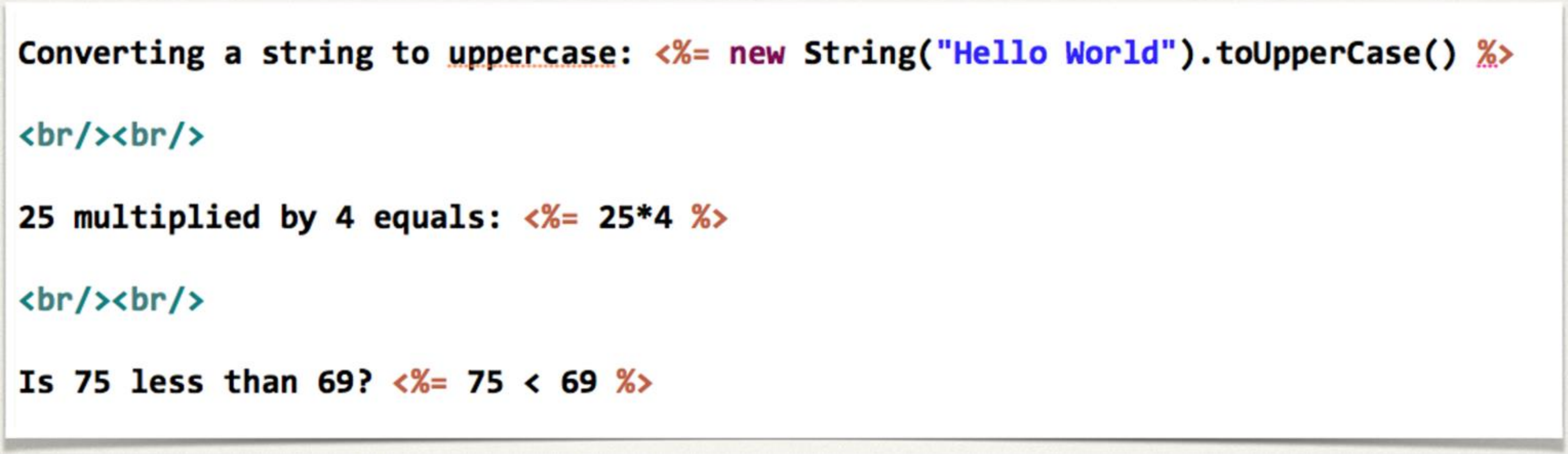


```
Hello World of Java


I really luv2code: 1
I really luv2code: 2
I really luv2code: 3
I really luv2code: 4
I really luv2code: 5
```

JSP EXPRESSION

- Expression tag evaluates the expression placed in it.
- It accesses the data stored in stored application.
- It allows create expressions like arithmetic and logical.
- It produces scriptless JSP page.
- Syntax
    - *<%= expression %>*

JSP EXPRESSION



```
Converting a string to uppercase: <%= new String("Hello World").toUpperCase() %>

<br/><br/>

25 multiplied by 4 equals: <%= 25*4 %>

<br/><br/>

Is 75 less than 69? <%= 75 < 69 %>
```

JSP COMMENT

- Comments are the one when JSP container wants to ignore certain texts and statements.
- When we want to hide certain content, then we can add that to the comments section.
- Syntax
  - *<% -- JSP Comments -- %>*

JSP DIRECTIVE

- JSP directives are the messages to JSP container. They provide global information about an entire JSP page.
- JSP directives are used to give special instruction to a container for translation of JSP to servlet code.
- In JSP life cycle phase, JSP has to be converted to a servlet which is the translation phase.
- They give instructions to the container on how to handle certain aspects of JSP processing
- Directives can have many attributes by comma separated as key-value pairs.
- Syntax
  - *<%@ directive attribute="" %>*

JSP DIRECTIVE

Directive	**Description**
<%@ page ... %>	defines page dependent properties such as language, session, errorPage etc.
<%@ include ... %>	defines file to be included.
<%@ taglib ... %>	declares tag library used in the page

JSP PAGE DIRECTIVE

- It provides attributes that get applied to entire JSP page.
- It defines page dependent attributes, such as scripting language, error page, and buffering requirements.
- It is used to provide instructions to a container that pertains to current JSP page.
- Syntax
  - *<%@ page…%>*

JSP PAGE DIRECTIVE

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    import="java.util.Date" pageEncoding="ISO-8859-1"%>
```

- Attribute ***language*** value is Java which is the underlying language in this case. Hence, the code in expression tags would be compiled using java compiler.(defines scripting language to be used in the page.)

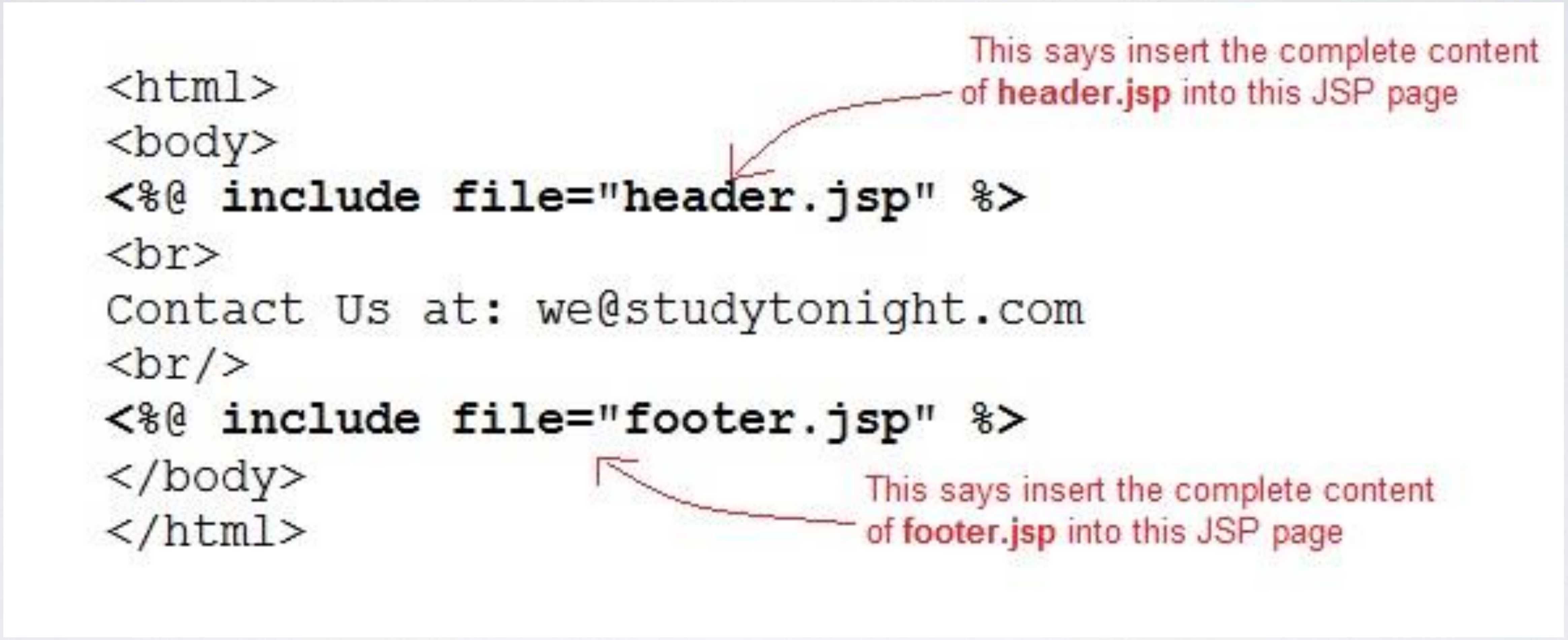
- The ***contentType*** is set as text/html — it sets character encoding for JSP and for generated response page(defines the MIME type for the JSP response.)

- The ***import*** is set as java.util.Date — we are importing Date class from java.util package (all utility classes), and it can use all methods of the following class.(defines the set of classes and packages that must be imported in servlet class definition)

JSP INCLUDE DIRECTIVE

- JSP "include directive” is used to include one file to the another file
- This included file can be HTML, JSP, text files, etc.
- It is also useful in creating templates with the user views and break the pages into header&footer and sidebar actions.
- It includes file during translation phase
- Syntax
  - *<%@ include file="filename.jsp" %>*

JSP INCLUDE DIRECTIVE



```
<html>
<body>
<%@ include file="header.jsp" %>
<br>
Contact Us at: we@studytonight.com
<br/>
<%@ include file="footer.jsp" %>
</body>
</html>
```

This says insert the complete content of **header.jsp** into this JSP page

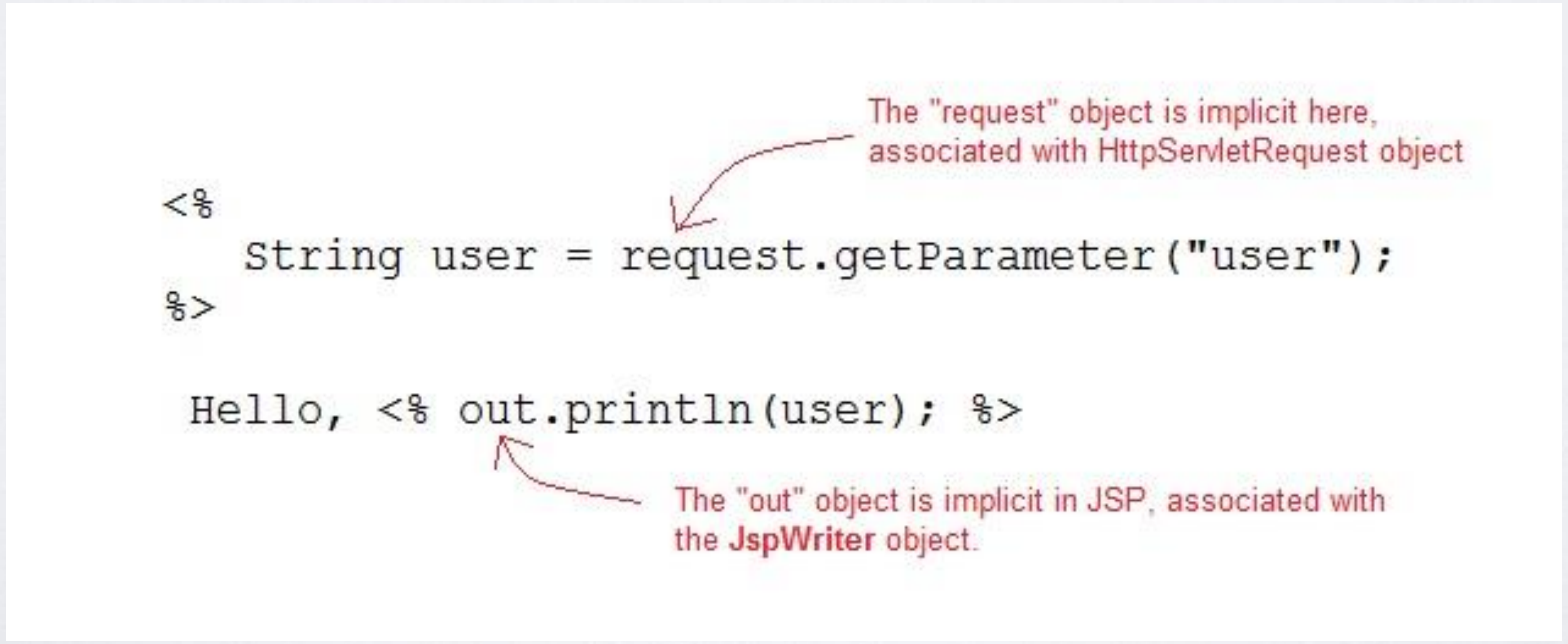
This says insert the complete content of **footer.jsp** into this JSP page

JSP TAGLIB DIRECTIVE

- JSP taglib directive is used to define the tag library with "taglib" as the prefix, which we can use in JSP.
- JSP taglib directive is used in the JSP pages using the JSP standard tag libraries
- It uses a set of custom tags, identifies the location of the library and provides means of identifying custom tags in JSP page.
- Syntax
    - *<%@ taglib uri="uri" prefix="value"%>*

Jsp Implicit Objects

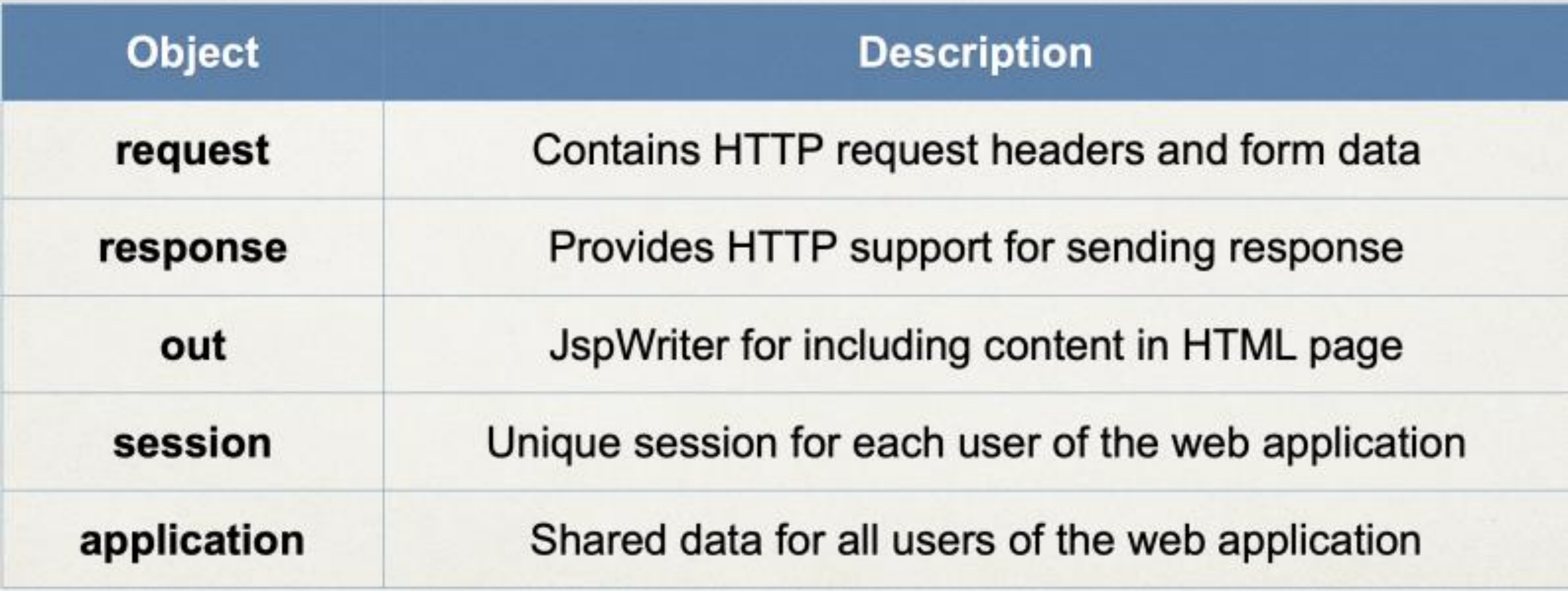
- There several build-in object in JSP. We can use them directly.
- represent some commonly used objects for servlets that JSP page developers might need to use.



```
<%
   String user = request.getParameter("user");
%>

 Hello, <% out.println(user); %>
```

The "request" object is implicit here, associated with HttpServletRequest object
The "out" object is implicit in JSP, associated with the **JspWriter** object.

Jsp Implicit Objects



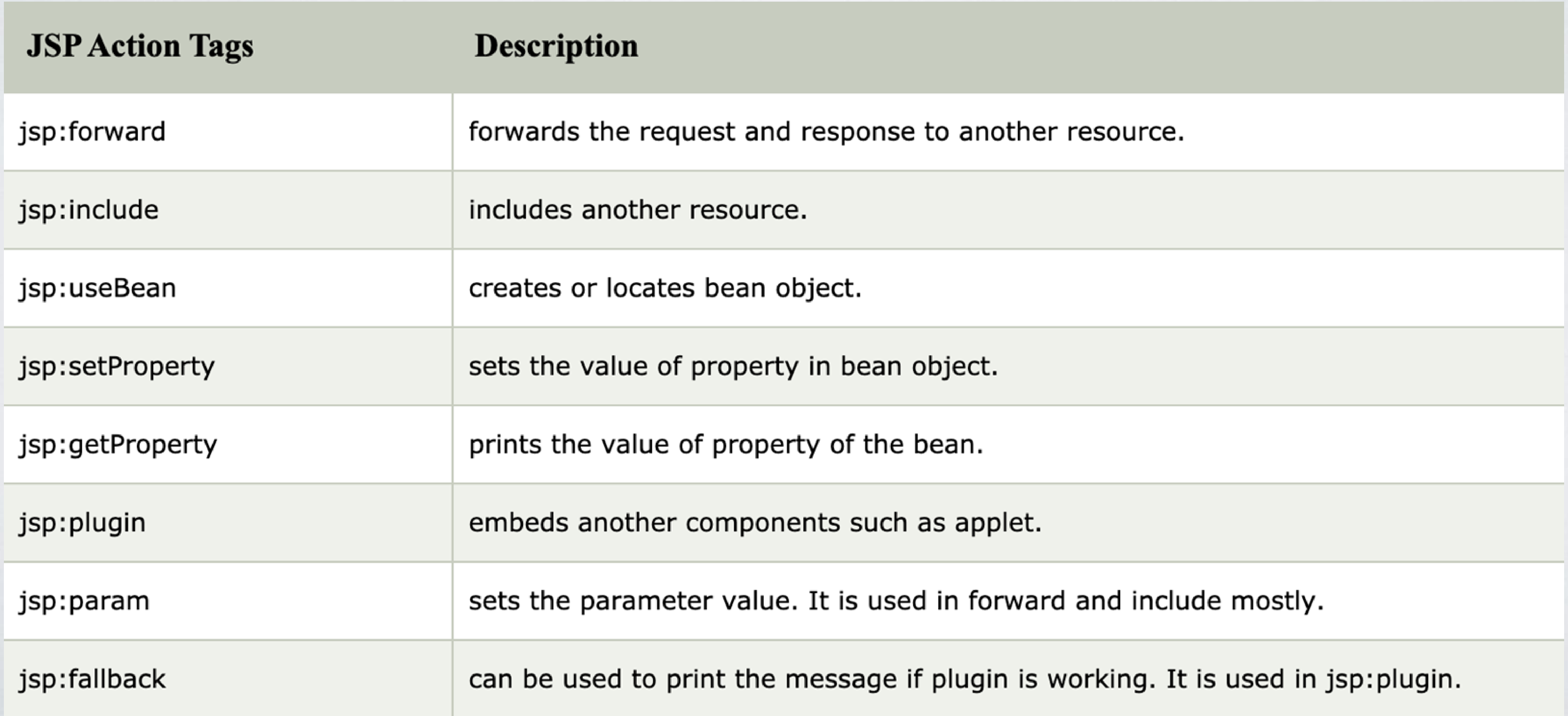
Object	Description
request	Contains HTTP request headers and form data
response	Provides HTTP support for sending response
out	JspWriter for including content in HTML page
session	Unique session for each user of the web application
application	Shared data for all users of the web application

JSP ACTION

- JSP actions use the construct in XML syntax to control the behavior of the servlet engine.
- We can dynamically insert a file, reuse the beans components, forward user to another page, etc. through JSP Actions like include and forward.
- Unlike directives, actions are re-evaluated each time the page is accessed.
- Syntax
    - *<jsp:action_name attribute="value" />*

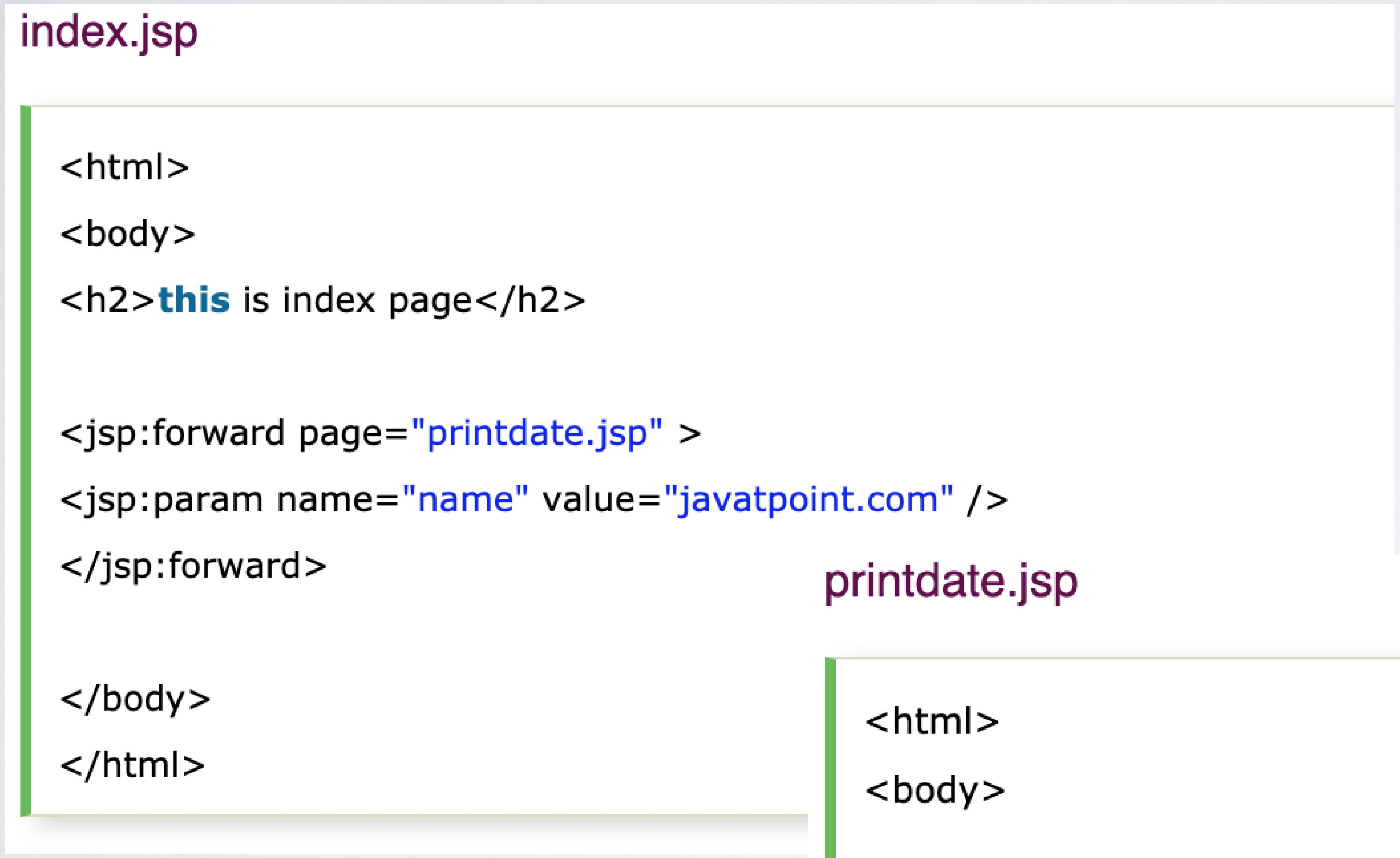
JSP ACTION

- There are 11 types of Standard Action Tags
  - https://www.guru99.com/jsp-action-tags.html
- List of commonly used Action Tags



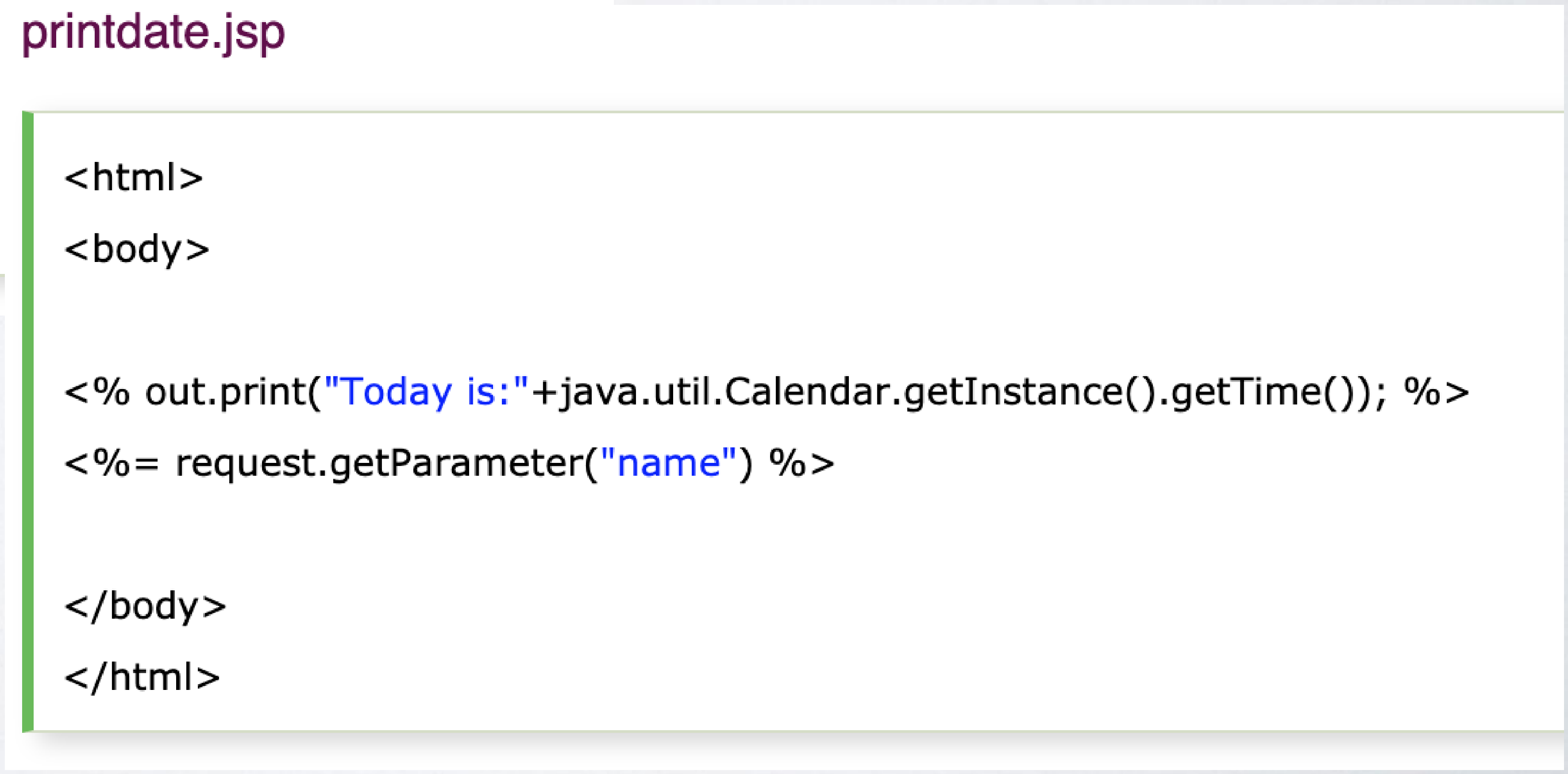
JSP Action Tags	Description
jsp:forward	forwards the request and response to another resource.
jsp:include	includes another resource.
jsp:useBean	creates or locates bean object.
jsp:setProperty	sets the value of property in bean object.
jsp:getProperty	prints the value of property of the bean.
jsp:plugin	embeds another components such as applet.
jsp:param	sets the parameter value. It is used in forward and include mostly.
jsp:fallback	can be used to print the message if plugin is working. It is used in jsp:plugin.

JSP ACTION



index.jsp

```
<html>
<body>
<h2>this is index page</h2>

<jsp:forward page="printdate.jsp" >
<jsp:param name="name" value="javatpoint.com" />
</jsp:forward>

</body>
</html>
```

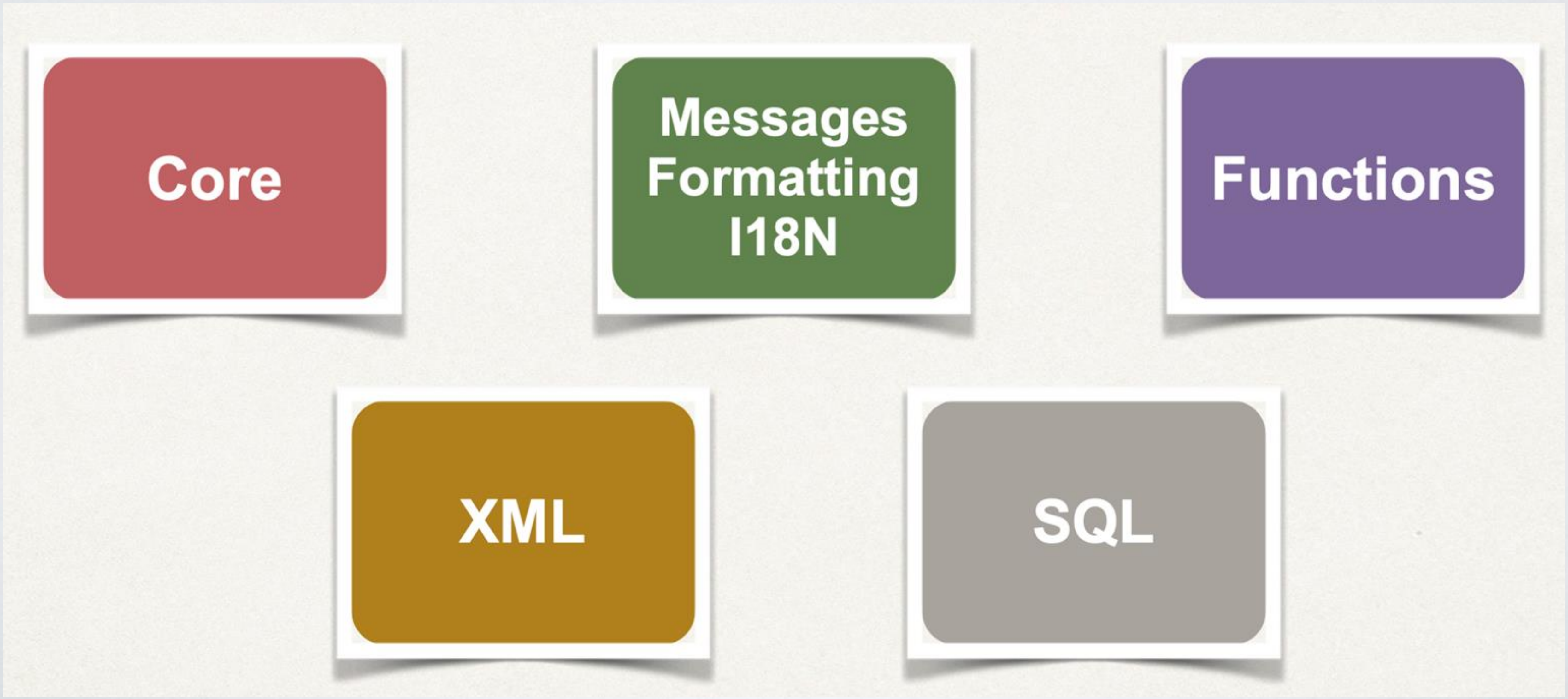


printdate.jsp

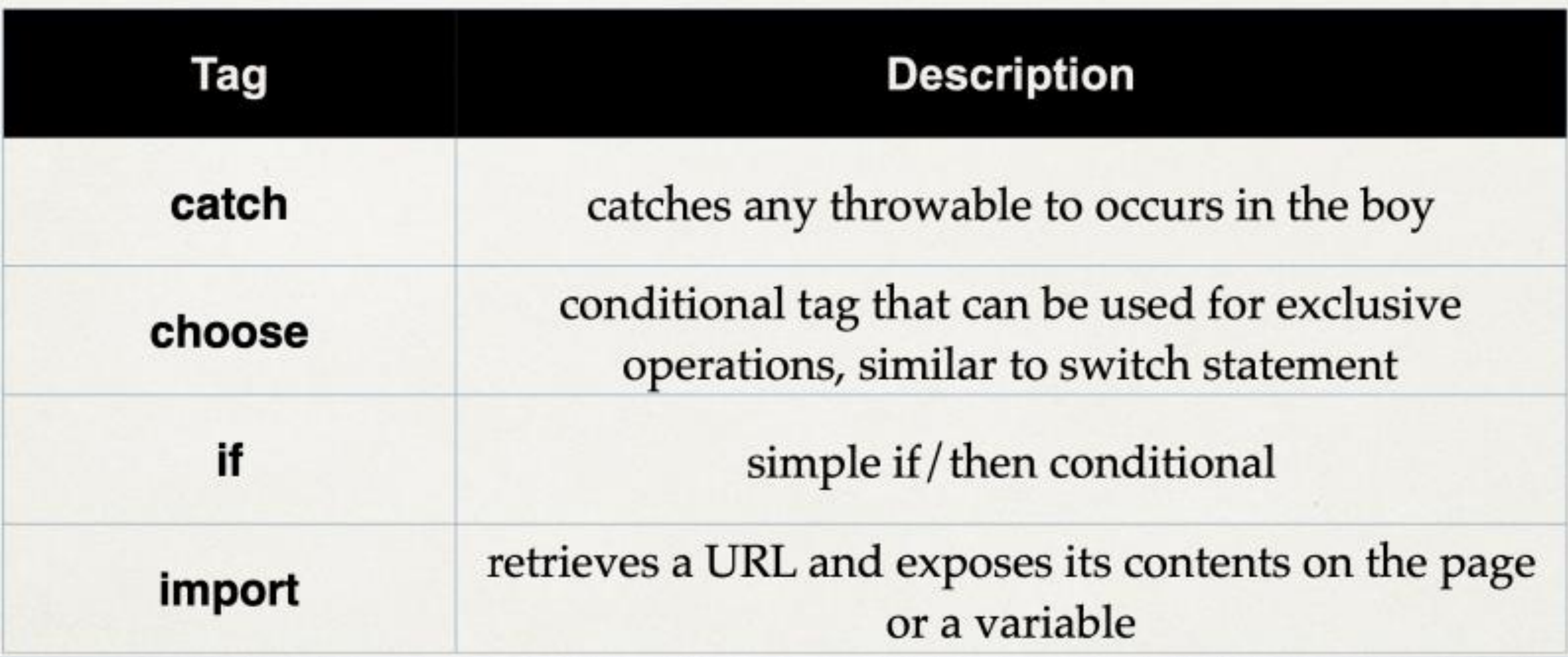
```
<html>
<body>

<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
<%= request.getParameter("name") %>

</body>
</html>
```

JSTL

- JSTL — JSP Standard Tag Library
- It is a collection of custom JSP tag libraries that provide common web development functionality.

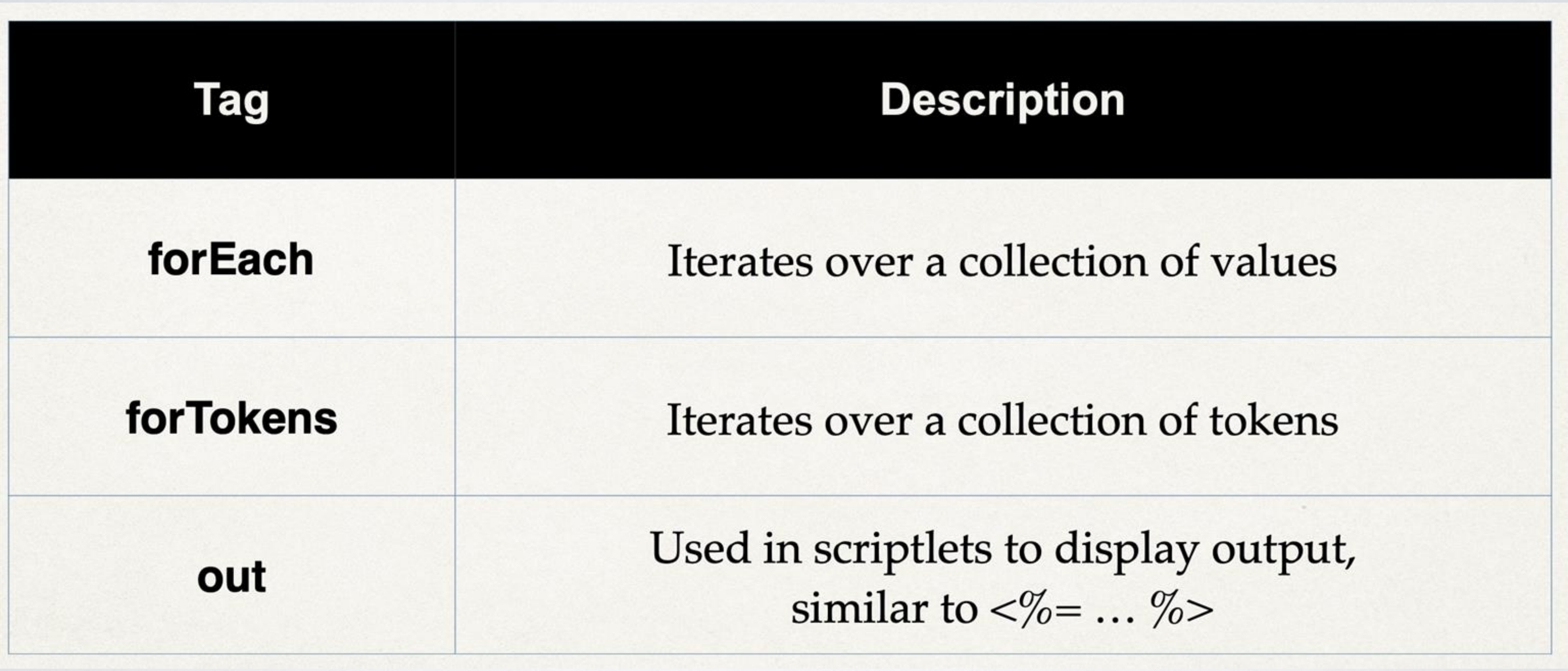


JSTL CORE



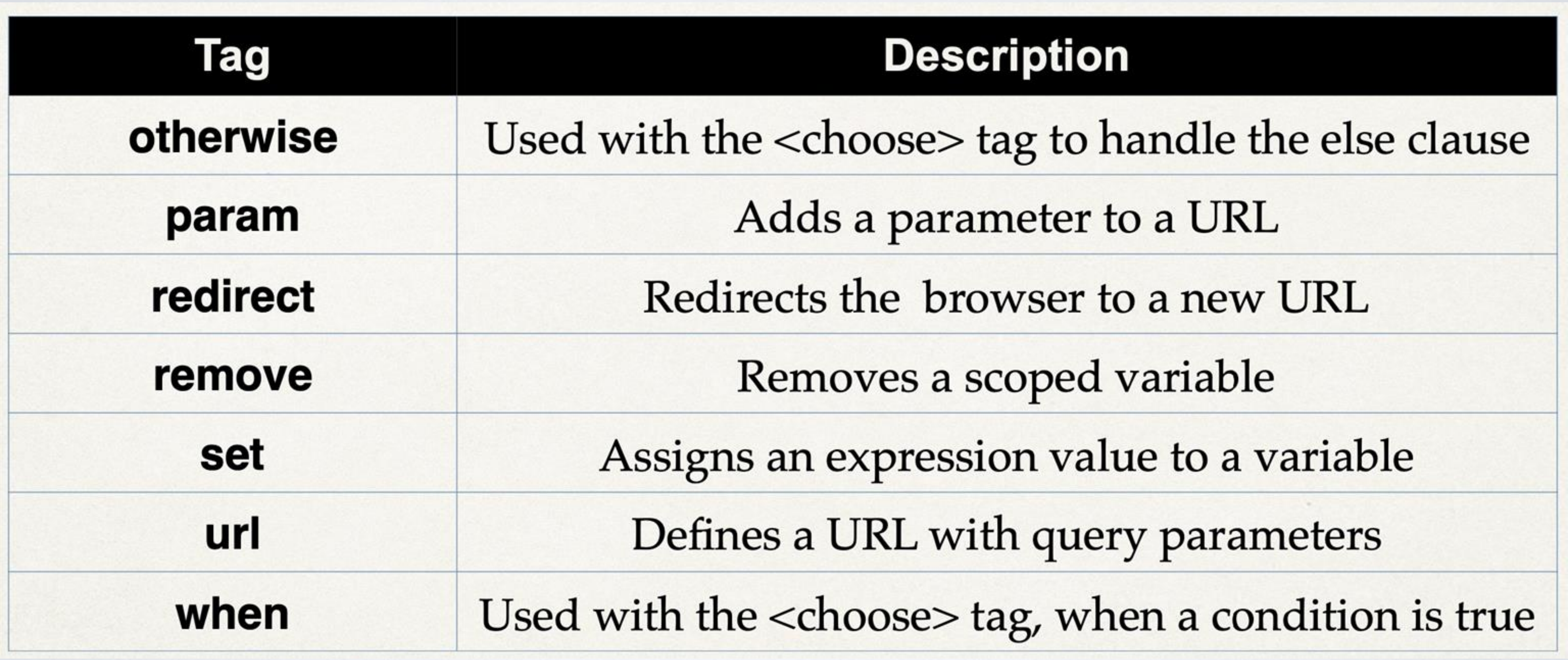
Tag	Description
catch	catches any throwable to occurs in the boy
choose	conditional tag that can be used for exclusive operations, similar to switch statement
if	simple if/then conditional
import	retrieves a URL and exposes its contents on the page or a variable

JSTL CORE



Tag	Description
forEach	Iterates over a collection of values
forTokens	Iterates over a collection of tokens
out	Used in scriptlets to display output, similar to <%= … %>

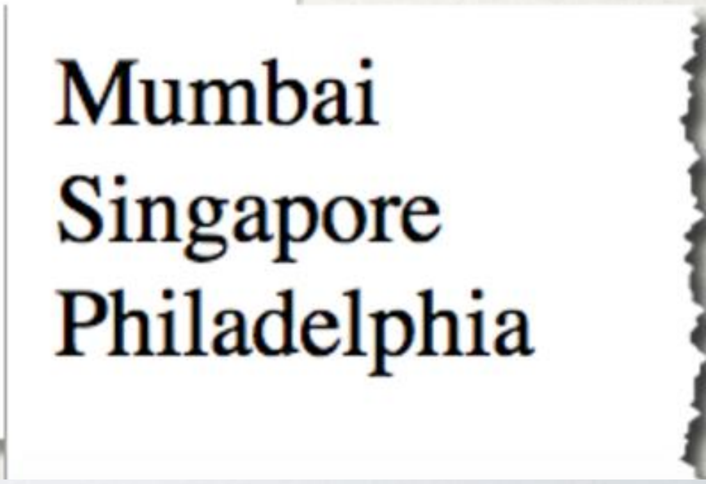
JSTL CORE



Tag	Description
otherwise	Used with the <choose> tag to handle the else clause
param	Adds a parameter to a URL
redirect	Redirects the browser to a new URL
remove	Removes a scoped variable
set	Assigns an expression value to a variable
url	Defines a URL with query parameters
when	Used with the <choose> tag, when a condition is true

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

```
<%
  // just create some sample data ... normally provided by MVC
  String[] cities = {"Mumbai", "Singapore", "Philadelphia"};

  pageContext.setAttribute("myCities", cities);
%>


<body>


  <c:forEach var="tempCity" items="${myCities}">
    ${tempCity} <br/>
  </c:forEach>

</body>
```



```
Mumbai
Singapore
Philadelphia
```

HTTP SESSION

- The problem with HTTP
  - HTTP is ***stateless*** — we don’t know if two HTTP requests comes from the same user or not.
- In order to associate a request to any other request, we need a way to store user data between HTTP requests
  - URL parameters — Passing same credential with all request
  - Cookies — Storing the credential on client side (Unsafe)
  - Session — Storing the credential on server side, give it an "id", and let the client only know (and pass back at every http request) that id. That is SESSION!

COOKIE

- Cookie is small piece of information that is sent by web server in response ***header*** and gets stored in the browser cookies

- It is a key value pair sent by the server to the client.

- This pair is automatically attached with every request to the server from where it was downloaded and then sent to the server.

- By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.

SESSION MANAGEMENT

- Several way to maintain a session between client and server
  - User Authentication — Send the credential with every request once authenticated. Won't work if user changed the browser
  - HTML Hidden Field — Set a hidden filed to the HTML through Servlet. Only works with form submitting.
  - URL Rewriting — Sending the session Id with request and response.
  - Cookie — When client make further request, it adds the cookie to the request header and we can utilize it to keep track of the session
  - Session Management API

SESSION MANAGEMENT API

- Servlet provides Session Management through HttpSession Interface.
- It provides us following methods
  - *HttpSession getSession()* – Ths method always returns a HttpSession object. It returns the session object attached with the request, if the request has no session attached, then it creates a new session and return it.
  - *HttpSession getSession(boolean flag)* – This method returns HttpSession object if true return a new Session; if false return current session or null.
  - *String getId()* – Returns a string containing the unique identifier assigned to this session.
  - *Object getAttribute(String name)* – Returns the object bound with the specified name in this session, or null if no object is bound under the name. Some other methods to work with Session attributes are *getAttributeNames(), removeAttribute(String name)* and *setAttribute(String name, Object value*).

SESSION MANAGEMENT API (CONT.)

- It provides following methods:
  - *long getCreationTime()* – Returns the time when ths session was created, measured in milliseconds since midnight January 1, 1970 GMT. We can get last accessed time with *getLastAccessedTime()* method.
  - *setMaxInactiveInterval(int interval)* – Specifies the time, in seconds, between client requests before the servlet container will invalidate this session. We can get session timeout value from *getMaxInactiveInterval()* method.
  - *boolean isNew()* – Returns true if the client does not yet know about the session or if the client chooses not to join the session.
  - *void invalidate()* – Invalidates this session then unbinds any objects bound to it.

HOW IT WORKS?

- When we use *HttpServletRequest getSession()* method and it creates a new request, it creates the new *HttpSession* object and also add a *Cookie* to the response object with name *JSESSIONID* and value as session id.
- This cookie is used to identify the HttpSession object in further requests from client by sending the *JSESSIONID*
- *JSESSIONID* cookie is used for session tracking, so we should not use it for our application purposes to avoid any session related issues

ANY QUESTIONS?