JEE - JavaServer Pages (JSP) & Expression Language (EL)

Lecture 4 - JSP & EL

- **JSP**
- Scripting elements
- Directive elements
- Standard action elements

- **EL**
- Implicit Objects
- Getting Information

Servlet vs. JSP

Servlets

HTML in Java

```
public void doGet(request, response)
{
PrintWriter out = response.getWriter();
String name =
      request.getParameter(uName);
out.println("<html><body>");
out.println("Username:" + name);
out.println("</body></html>");
}
```

JSP

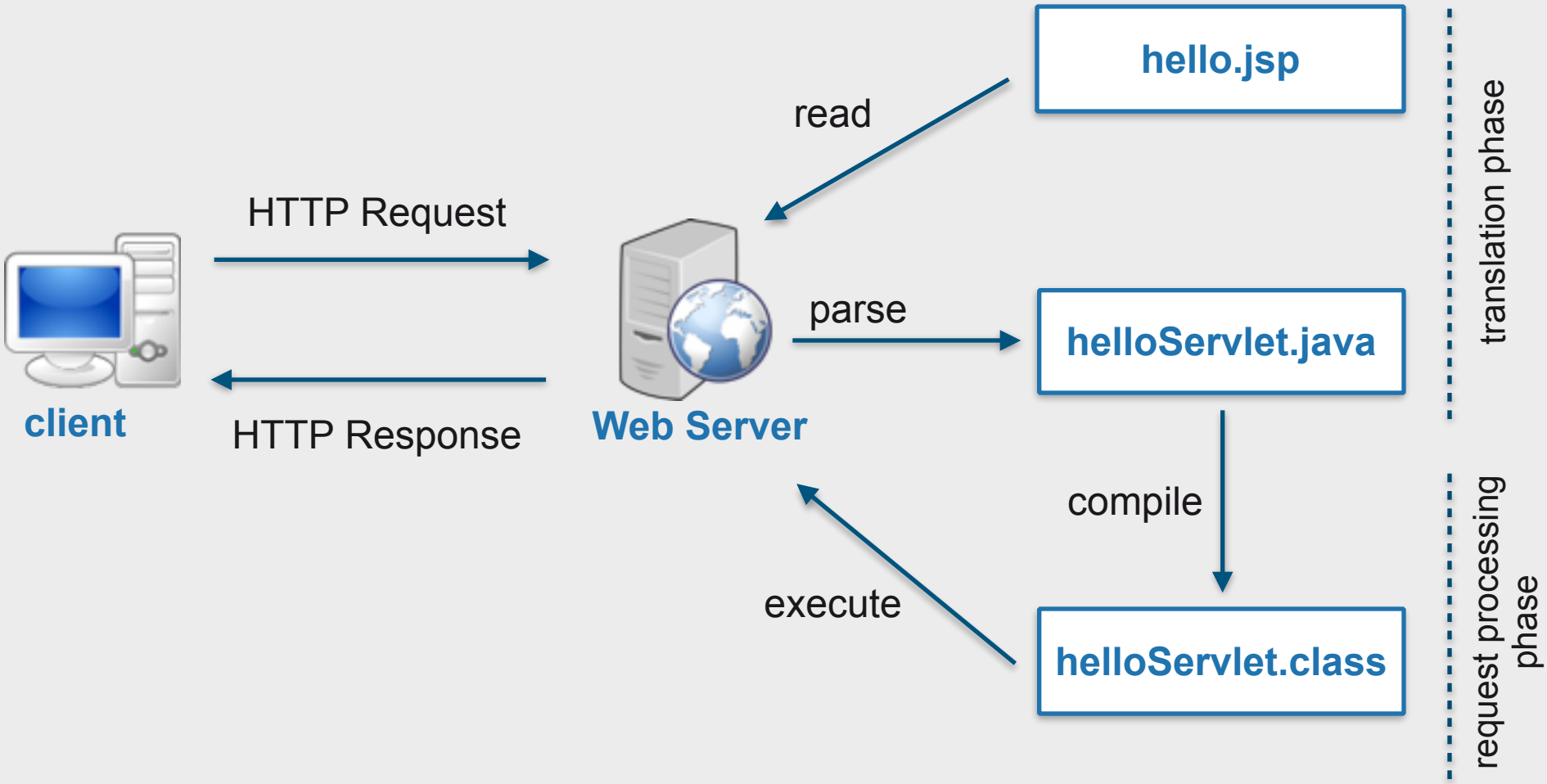
Java in HTML

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

Username: <%= name %>

</body>
</html>
```

JSP Architecture



Implicit Variables in JSP

- **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
- **session**
  - This is the HttpSession object associated with request

Implicit Variables in JSP

- **application**
  - This is the ServletContext object associated with the application context
- **config**
  - This is the ServletConfig object associated with the page
- **pageContext**
  - This encapsulates use of server-specific features like higher performance JspWriters

Implicit Variables in JSP

- page
  - This is a synonym for this. It is used to call the methods defined by the translated servlet class.
- Exception
  - It allows the exception data to be accessed by the designated JSP

- Examples:
  - out.print(dataType dt);
  - config.getServletName();
  - response.setStatus(int statusCode);
  - request.getParamter(String name);

JSP Elements

- There are mainly three types of tag tags (elements) in JSP, used to put java code in JSP files
  - JSP Scripting elements
  - JSP Directive elements
  - JSP Standard Action elements

JSP Scripting Elements

- There are four types of JSP Scripting elements
  - Declaration tag
  - Scripting tag
  - Expression tag
  - Comment tag

JSP Declaration tag

- It lets you declare variables and methods in jsp page
- Java code is inside tag
- Variable declaration must end with a semi-colon

- Syntax: <%!JavaCode;%>

```
<%! private int i=10;
private int square (int i){
        return i*i;
        }
%>
```

JSP Scripting tag

- It lets you insert java code in the jsp pages
- Syntax: <%JavaCode;%>

```
<%
String name = Fabien;
out.println("Name = " + name);
%>
```

JSP Expression tag

- It is used to insert Java values directly to the output
- Syntax: <%=JavaCode%>

Current time = <% new java.util.Date()%>

JSP Comment tag

- HTML comments can be seen by the users through view source
- JSP comments are not visible to the end users

- Syntax: <%- - Comment here - -%>

```
<%- - This shows the current date and time- -%>
Current time = <% new java.util.Date()%>
```

Scripting Elements Example

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Scripting Element Example</title>
</head>
<body>
    <%--declaration tag --%>
    <%!int i = 10;%>

    <%--scriptlet tag --%>
    <%
        out.print("i = " + i);
    %>
    <BR />
    <BR />
    <%
        out.print("Loop execution:");
    %>
    <BR />
```

Scripting Elements Example

```
<%
        while (i < 20) {
            out.print("value of i = " + i);
            i++;
    %>
    <BR />
    <%
        }
    %>
    <BR />
    <%-- expression tag --%>
    <%!int a = 5, b = 15;%>
    The addition of a + b = 5 + 15 =
    <%=a + b%>
    <BR /> Current time:
    <%=new java.util.Date()%>
</body>
</html>
```

Scripting Elements Example

http://localhost:8080/FilteredSession/Example.jsp

i = 10

Loop execution:
value of i = 10
value of i = 11
value of i = 12
value of i = 13
value of i = 14
value of i = 15
value of i = 16
value of i = 17
value of i = 18
value of i = 19

The addition of a + b = 5 + 15 = 20
Current time: Tue Oct 21 11:31:24 CEST 2014

JSP Directive Elements

- They give special information about the page to JSP Engine
- They are handled only once at translation phase

- Syntax: **<%@ directive-name [attribute="value" attribute = "value" …]%>**

- Types of directives
  - page directive
  - include directive
  - taglib directive

Page directive

- It is used to specify attributes for the JSP page
  - e.g. making session data unavailable to a page

- Syntax:

```
<%@ page [attribute="value" attribute = "value" … ] %>
```

- Example:

```
<%@ page language="java" session="true" … %>
```

Page directive - attributes

Attribute	Description	Syntax	Example
language	The language to be used in JSP file	language="java"	<%@page language="java" %>
import	List of packages need by servlet	import="package.class, package.class"	<%@page import="java.util.*, java.io.*" %>
extends	superclass of servlet	extends = "package.class"	<%@page extends="com.ece.Login" %>
session	true binds to existing session or creates new. false means no session	session = "true | false"	<%@page session="true" %>

Page directive - attributes

Attribute	Description	Syntax	Example
buffer	defines the buffer size for out. default is 8kb	buffer="size(kb) | none"	<%@page buffer="16kb" %>
isThreadSafe	choice between multithreading and single ThreadModel	isThreadSafe="true | false"	<%@page isThreadSafe="true" %>
autoFlush	true means flush buffer when full, false means to throw exception	autoFlush="true | false"	<%@page autoFlush="true" %>
pageEncoding	datatype of page encoding	pageEncoding="encocding"	<%@page pageEncoding = "ISO-8859-1" %>

Page directive - attributes

Attribute	Description	Syntax	Example
info	string for getServletInfo	info="information message"	<%@page buffer="16kb" %>
contentType	MIME type of output	contentType="MIME-Type"	<%@page contentType ="text/html; charset = UTF-8" %>
isELignored	expression language available?	isELIgnored="true | false"	<%@page isELignored ="true" %>
isErrorPage	Can current page act as error page ?	isErrorPage="true | false"	<%@page isErrorPage = "false" %>
errorPage	define error page URL for unchecked runtime exception	errorPage = "URL"	<%@page errorPage = "error.jsp" %>

Include directive

- It is used to insert code (static resource only) of a file inside jsp file at a specified place in the translation phase
- Headers, footers, tables and navigation menus that are common to multiple pages can be placed using it.

- Syntax:

```
<%@ include file="/folder_name/file_name"%>
```

- Example:

```
<%@ include file="footer.html"%>
```

Taglib directive

- It make custom actions available in the jsp file, using the tag libraries

- Syntax:

<%tablib uri="tag Library_path" prefix="tag_prefix"%>

- uri = Absolute path of a tag library descriptor
- prefix= Prefix to identify custom tags from a specific library

- Example:

<%@ taglib uri=/tlds/ColouredTable.tld" prefix="ct"@>

JSP to Servlet translation

```
<%@ page import="abc.*" %>

<html>
 <body>

   <% int i = 10; %>
   <%! int count = 0; %>

   Hello! Welcome

   <%! Public void hello()
         {
            out.println("Hello");
         } %>

</body>
</html>
```

```
import javax.servlet.HttpServlet.*
import abc.*;
public class Hello_jsp extends HttpServlet
{
 int count = 0;
 public void hello()
 {
   out.println("Hello");
 }

public void _jspService(req, res)
 {
  int i = 10;
  out.println("<html>\r<body>");
  out.println("Hello! Welcome");
 }
}
```

JSP Standard Action Elements

- They are used to create, modify or use other objects
- Only coded in strict XML syntax
- General usage
  - inserting a file
  - reuse JavaBeans component
  - forward to another page
  - generate HTML for a Java Plugin, etc.

Types of standard action types

1. <jsp:param>
2. <jsp:include>
3. <jsp:forward>
4. <jsp:fallback>
5. <jsp:plugin>
6. <jsp:useBean>
7. <jsp:setProperty>
8. <jsp:getProperty>

Standard action - <jsp:param>

- It provides other tags with additional information as name value pairs
- Used along with jsp:include, jsp:forward, jsp:plugin

- Syntax:

```
<jsp:param name="parameter_name"
value="parameter_value" />

<jsp:param name="parameter_name"
value="paramter_value> </jsp:param>
```

Standard action - <jsp:param>

- **Example:**

```
<jsp:param name="font_color" value="red" />

<jsp:param name="font_color" value="red">
<jsp:param>
```

Standard action - <jsp:include>

- This allows to include a static or dynamic resource in the JSP at request time.
- If page is buffered then the buffer is flushed prior to the inclusion
- Syntax:

```
<jsp:include page="file_name" flush="true|false"/>

<jsp:include page="file_name" flush="true|false">
<jsp:param name="parameter_name"
value="parameter_value"/>
</jsp:include>
```

Example - <jsp:include>

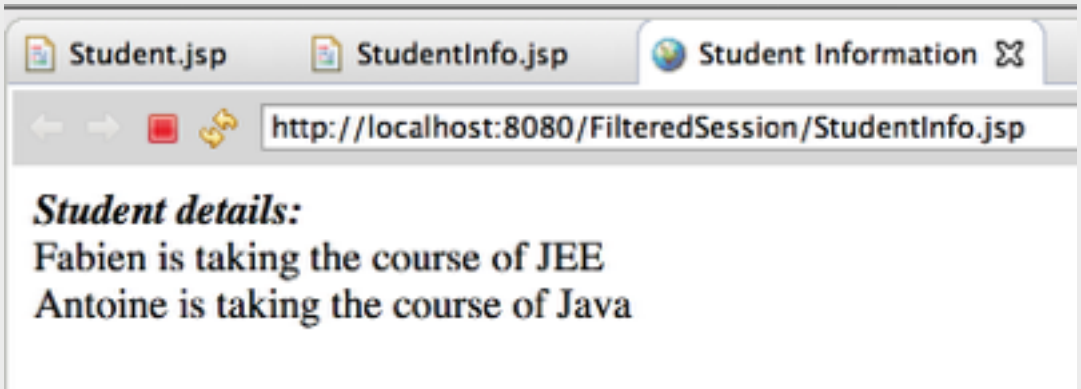
Student.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Student</title>
</head>
<body>
    <b><i>Student details:</i></b>
    <br>
    <%
        out.print(request.getParameter("name1") + " is taking the course of "
                + request.getParameter("course1"));
    %>
    <br>
    <%
        out.print(request.getParameter("name2") + " is taking the course of "
                + request.getParameter("course2"));
    %>
</body>
</html>
```

Example - <jsp:include>

StudentInfo.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Student Information</title>
</head>
<body>
    <jsp:include page="Student.jsp">
        <jsp:param value="Fabien" name="name1" />
        <jsp:param value="JEE" name="course1" />
        <jsp:param value="Antoine" name="name2" />
        <jsp:param value="Java" name="course2" />
    </jsp:include>
</body>
</html>
```



Directive include vs Action include

Include	Syntax	Inclusion time	Content type	Parsing
Directive	<%@include file="file_name"%>	Translation phase	Static	Container
Action	<jsp:include page=file_name"%>	Request processing phase	Static or dynamic	Not parsed, included in the container

Standard action - <jsp:forward>

- This is used to forward the request to another page

- Syntax:

```
<jsp:forward page="file_name"/>

<jsp:forward page="file_name">
<jsp:param name="parameter_name
value="paramater_value"/>
</jsp:forward>
```

Standard action - <jsp:fallback>

- Used in conjunction with <jsp:plugin>
- This element can be used to specify an error string to be sent to the user in case the plugin fails

- Syntax:

<jsp:fallback> text message for user </jsp:fallback>

Standard action - <jsp:plugin>

- Generates browser-specific code that makes an Object or Embed tag for the Java plugin
- Can be used for applets and JavaBeans
- Syntax:

```
<jsp:plugin type="bean|apple"
        code ="className.class"
        codebase="path of className.class in WebRoot"
        [name= "name of bean or applet]
        [align="bottom|top|middle|left|right]
        [height:"diplayPixels"] ….. >
```

Standard action - <jsp:plugin>

[<jsp:params>

<jsp:param name="parameter_name" value="parameter_value"/>

<jsp:param name="parameter_name" value="parameter_value"/>

…….

</jsp:params>]

[<jsp:fallback> text message </jsp:fallback>]

</jsp:plugin>

Standard action - <jsp:useBean>

- It is used to find and instantiate a JavaBean
- A common way of interaction between web pages
- Scopes:
  - page: (default) within a JSP page
  - request: within the same request
  - session: all JSPs in the same session
  - application: within the same context

Standard action - <jsp:useBean>

- **Syntax:**

```
<jsp:useBean id= "bean_id"
scope= "page | request | session | application"
class= “packageName.className"
beanName="packageName.className" >
</jsp:useBean>
```

Standard action - <jsp:setProperty>

- It is used to set the value of a bean's property

- Syntax:

```
<jsp:setProperty name="bean_id" property= "*"
| property="propertyName" param="parameterName"
| property="propertyName" value="propertyValue" />
```

Standard action - <jsp:getProperty>

- It is used to retrieve the value of a bean's property, convert it into a string and insert it into the output.

- Syntax:

```
<jsp:getProperty name="bean_id"
property="propertyName"/>
```

JSP-JavaBeans Example

```
package com.ece.jee;

public class BackgroundColor {
    int red, blue, green;

public BackgroundColor() {
        super();
        this.red = 0;
        this.blue = 0;
        this.green = 255;
    }

public int getRed() {
        return red;
    }

public void setRed(int red) {
        this.red = red;
    }
```

JSP-JavaBeans Example

```
public int getBlue() {
        return blue;
    }

public void setBlue(int blue) {
        this.blue = blue;
    }

public int getGreen() {
        return green;
    }

public void setGreen(int green) {
        this.green = green;
    }
}
```

JSP-JavaBeans Example

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<jsp:useBean id="bgc" class="com.ece.jee.BackgroundColor"
    scope="session" />
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>

<jsp:setProperty name="bgc" property="red" param="red" />
<jsp:setProperty name="bgc" property="green" param="green" />
<jsp:setProperty name="bgc" property="blue" param="blue" />
```

JSP-JavaBeans Example

```
<body
    style="background: rgb(<jsp:getProperty name="bgc" property="red"/>,
    <jsp:getProperty name="bgc" property="green"/>,
    <jsp:getProperty name="bgc" property="blue"/>)">

    <h3>Colorful Hello !!!</h3>

    <form name="colorForm" action="ColorExample.jsp" method="post">
        Red: <input type="text" name="red"> Green: <input type="text"
            name="green"> Blue: <input type="text" name="blue"> <input
            type="submit">
    </form>

</body>
</html>
```

Initialization parameters

In Deployment Descriptor:

```
<servlet>
 <servlet-name>InitJSP</servlet-name>
 <jsp-file>/Example.jsp</jsp-file>
 <init-param>
 <param-name>userName</param-name>
 <param-value>Abcd</param-value>
 </init-param>
 </servlet>

 <servlet-mapping>
 <servlet-name>InitJSP</servlet-name>
 <url-pattern>/Example.jsp</url-pattern>
 </servlet-mapping>
```

In JSP:

```
Our initialization parameter has username value =
     <%= getServletConfig().getInitParameter("userName") %>
```

Overriding Init()

```
<%!
public void jspInit(){
    String configUser =
getServletConfig().getInitParameter("userName");
    getServletContext().setAttribute("contextUser", configUser);
}
%>

Our initialization parameter has username value =
    <%= config.getInitParameter("userName") %>
<br/>
Servlet context value for user =
    <%= application.getAttribute("contextUser") %>
```

Our initialization parameter has username value = Abcd
Servlet context value for user = Abcd

JSP Documents

Syntax Elements	Standard Syntax	XML Syntax
Comments	<%--.. --%>	<!-- .. -->
Declarations	<%! ..%>	<jsp:declaration> .. </jsp:declaration>
Directives	<%@ include .. %>	<jsp:directive.include .. />
	<%@ page .. %>	<jsp:directive.page .. />
	<%@ taglib .. %>	xmlns:prefix="tag library URL"
Expressions	<%= ..%>	<jsp:expression> .. </jsp:expression>
Scriptlets	<% ..%>	<jsp:scriptlet> .. </jsp:scriptlet>

Expression Language (EL)

- Incorporated in JSP2.0
- Accessing bean using simple syntax
  - ${name} for a simple variable
  - ${name.foo.bar} for a nested property
- Kinds of expressions
  - Arithmetic

```
<jsp:setProperty name="box" property="perimeter"
        value="${2*box.width+2*box.height}"/>
```

  - Logical

```
<c:if test="${bean1.a < 3}" >   ... </c:if>
```

Operators in EL

Operator	Description
.	Access a bean property or Map entry
[]	Access an array or List element
( )	Group a subexpression to change the evaluation order
+	Addition
-	Subtraction or negation of a value
*	Multiplication
/ or div	Division
% or mod	Modulo (remainder)

Operators in EL

Operator	Description
== or eq	Test for equality
!= or ne	Test for inequality
< or lt	Test for less than
> or gt	Test for greater than
<= or le	Test for less than or equal
>= or gt	Test for greater than or equal
&& or and	Test for logical AND
|| or or	Test for logical OR
! or not	Unary Boolean complement
empty	Test for null, empty String, array or Collection.
func(args)	A function call

Implicit Objects in EL

Implicit object	Description
pageScope	Scoped variables from page scope
requestScope	Scoped variables from request scope
sessionScope	Scoped variables from session scope
applicationScope	Scoped variables from application scope
param	Request parameters as strings
paramValues	Request parameters as collections of strings
header	HTTP request headers as strings
headerValues	HTTP request headers as collections of strings
initParam	Context-initialization parameters
cookie	Cookie values
pageContext	The JSP PageContext object for the current page

Implicit Objects in EL

- Not the same as implicit objects of JSP
  - except pageContext
- pageContext is used to access other JSP implicit objects
- Syntax:

  ${pageContext.request.queryString}

Scope Objects

- These scope objects allow access to variables stored at different access levels
  - requestScope
  - sessionScope
  - applicationScope
  - pageScope
- Example

```
Servlet context value for user =
    <%= application.getAttribute("contextUser") %>
```

- Using EL

```
Servlet context value for user = ${applicationScope.contextUser}
```

Handling Attributes

- EL is used to read values, not to set values, JSP serves as “view” in MVC

- In Servlet

```
request.setAttribute(“contextUser”,cu);
```

- Using EL

```
${requestScope[“contextUser”].name}
${sessionScope[“contextUser”].name}
${applicationScope[“contextUser”].name}
```

Param & ParamValues

- Gives you access to parameter values using request.getParameter and request.getParameterValues methods

- Example: for a parameter named password

  ${param.password}

  ${param[“password”]}

Getting Header Information

- In JSP

```
<%= request.getHeader("host")%>
```

- With EL

```
${header.host}
${header["host"]}
```

Init Parameter

- Deployment Descriptor

```
<context-param>
<param-name>name</param-name>
<param-value>Antoine</param-value>
</context-param>
```

- With expression

```
<%= application.getInitParameter("name") %>
```

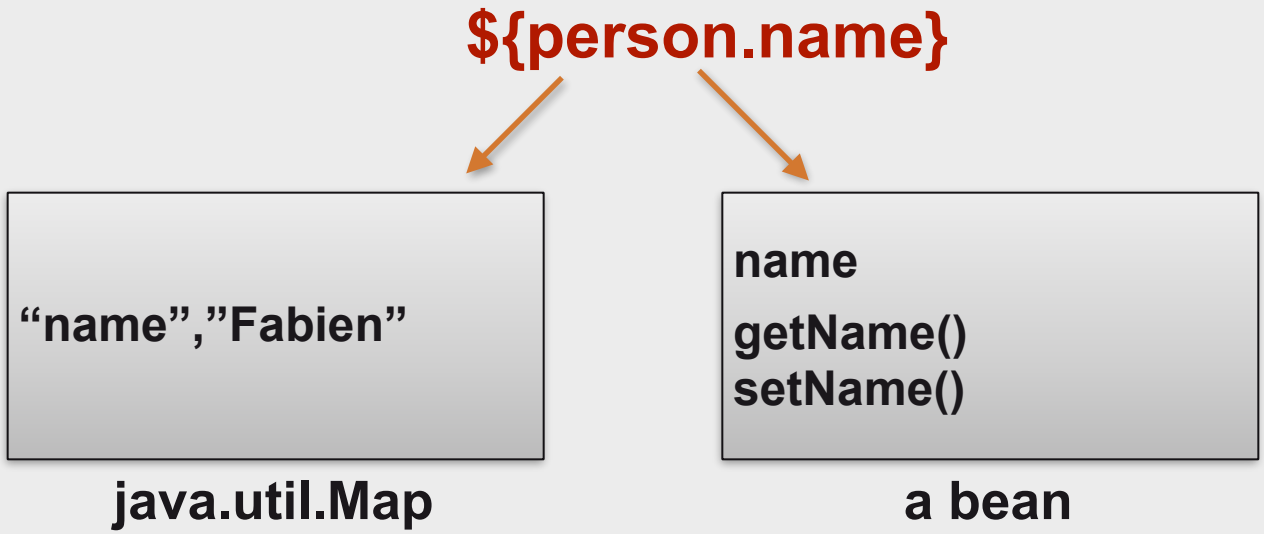
- With EL

```
${initParam.name}
```

Accessing properties

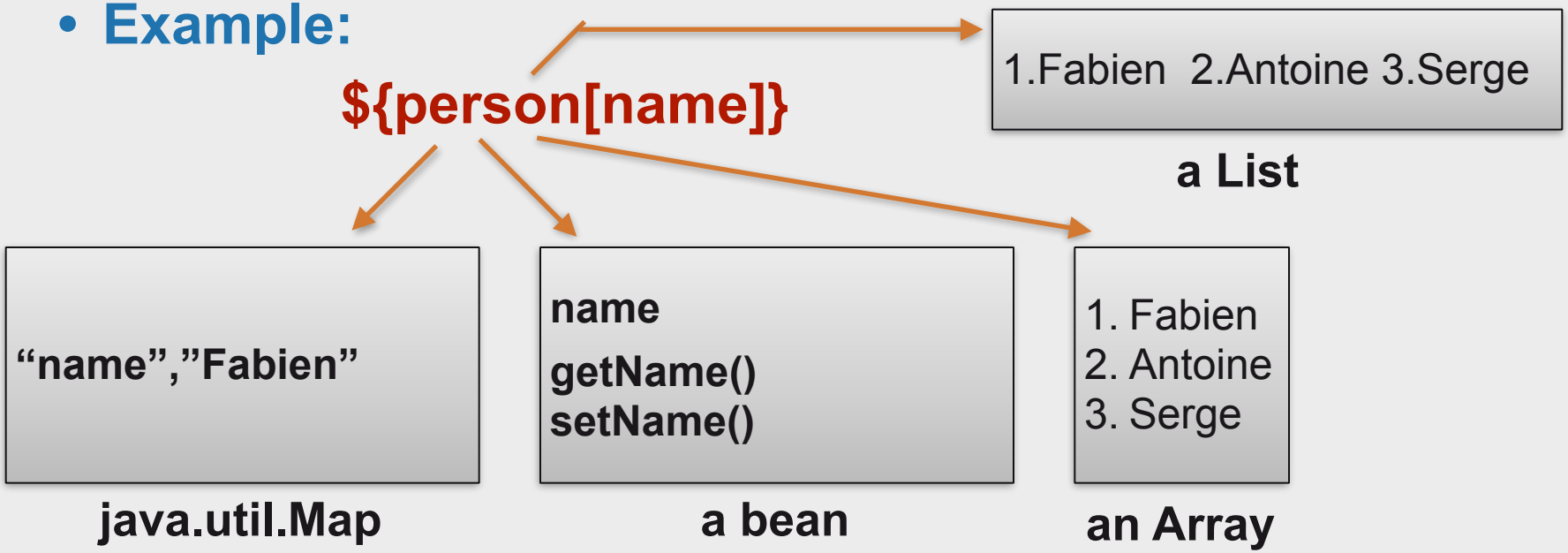
- If the expression has a variable followed by a dot, then this variable must be a bean or a map

- Example:



Accessing properties

- If the expression has a variable followed by a bracket, then this variable must be a bean, a map, a List or an Array
- Example:



Array

```
<%!
String[] nameList = {"Alpha","Bravo","Charlie"};
%>

<% request.setAttribute("name", nameList);%>

Name : ${name}
<BR />
Name : ${name[0]}
<BR />
Name : ${name["0"]}
```

```
Name : [Ljava.lang.String;@198c5c7a
Name : Alpha
Name : Alpha
```

HashMap

```
<%!
Map<String,String> studentMap = new HashMap<String,String>();
%>

<%
studentMap.put("name","Fabien");
request.setAttribute("studentMap", studentMap);
%>

Name : ${studentMap.name}
<BR />
Name : ${studentMap[name]}
<BR />
Name : ${studentMap["name"]}
```

Name : Fabien
Name :
Name : Fabien

Requesting Parameters

- In HTML

```
<form action="UserBean.jsp">
First Name : <input type="text" name="firstName">
Last Name: <input type="text" name="lastName">
<input type="submit">
</form>
```

- In JSP

```
${param.firstName}
${param.lastName}
```