SUPINFO International University

Sign of Success

Java Server Pages

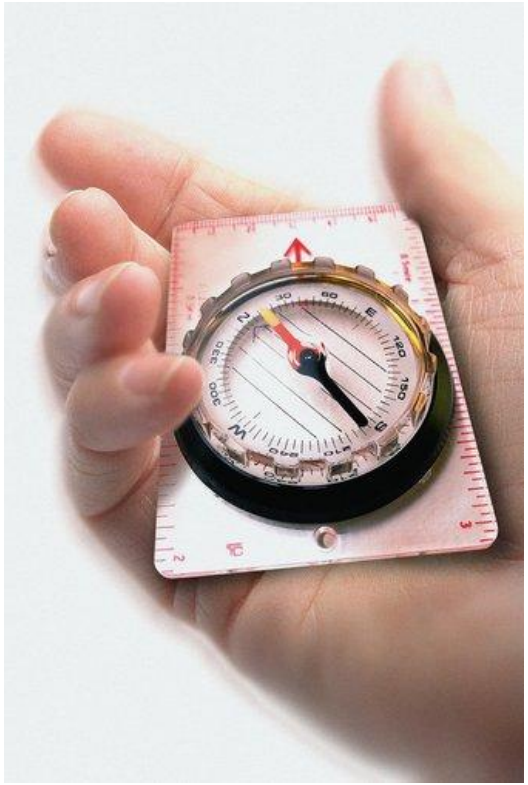
Java web pages

Course objectives

By completing this course you will be able to:

- Explain what JSP are
- Use Java Server Pages
- Use JSP Model 2 Architecture

Course plan



- Presentation
- Action elements
- Expression language
- Tag libraries
- JSP & Servlets

Java Server Pages

PRESENTATION

What are JSP?

- Web pages, but Java version
- JSP and Servlets work complementary
  - Used to separate request processing and HTML code generation
- Can contain Java code
  - Like PHP pages contain … PHP code
  - Thanks to scripting elements
- Can contain special markups to avoid Java code

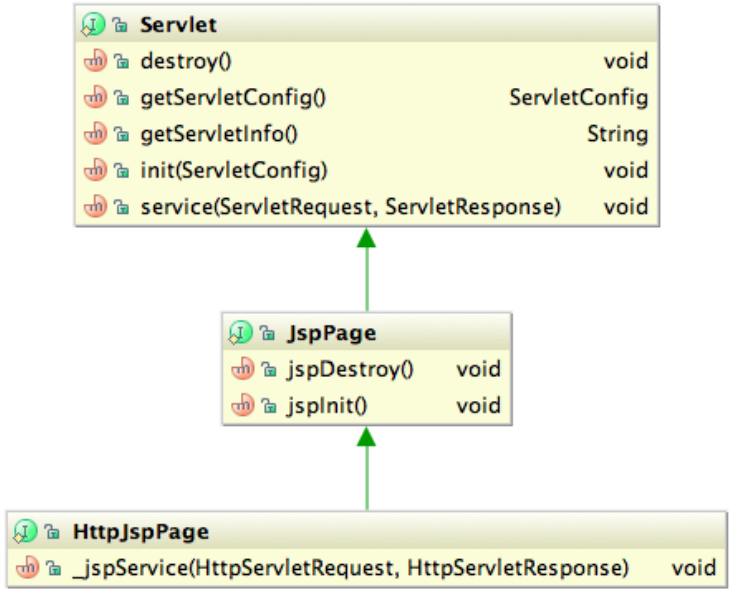
What are JSP?

- JSP example:

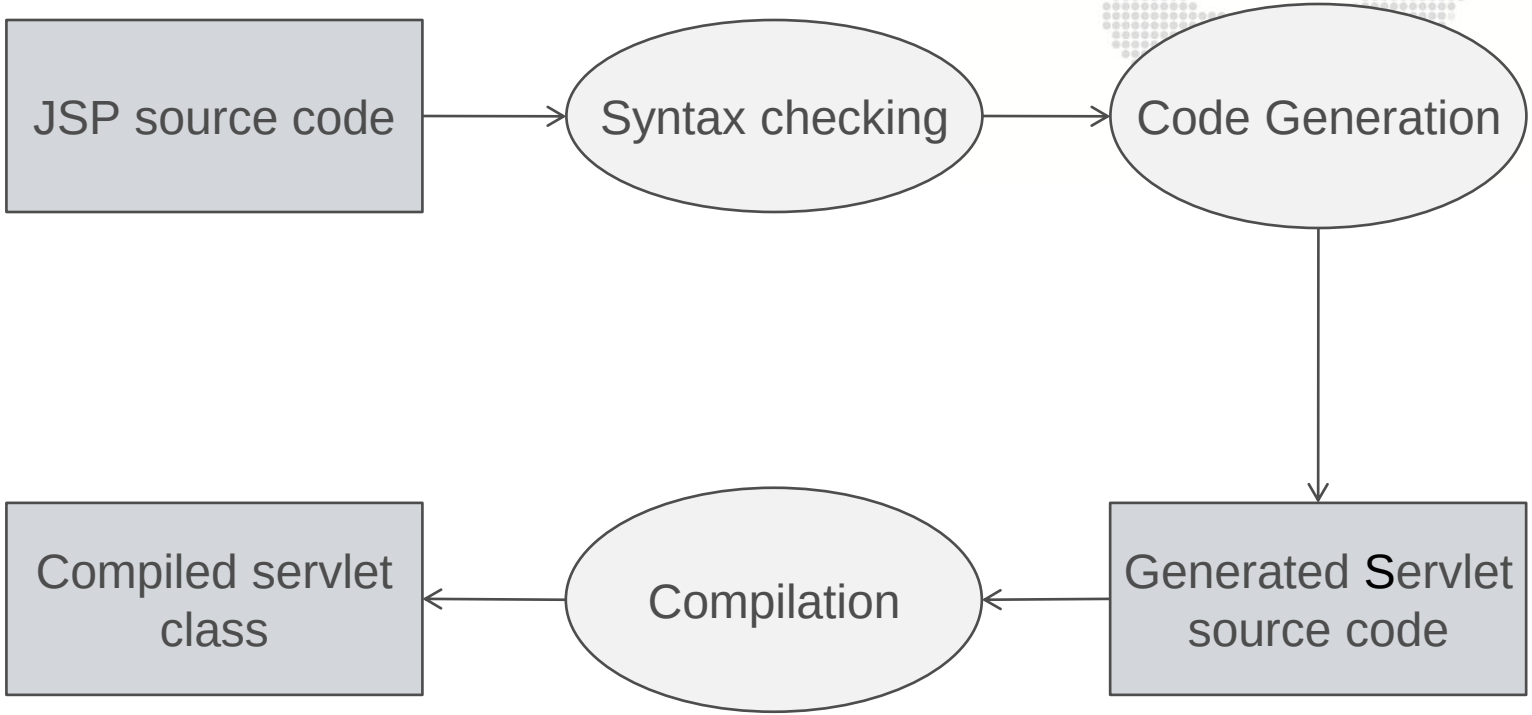
```
<%@ page language="java" %>
<%@ page import="java.util.Date" %>
<html>
   <head>
      <title>When am I ?</title>
   </head>
   <body>
      <%= new Date() %>
   </body>
</html>
```

JSP Life Cycle

- At the first request, JSP pages are translated into Java code (more exactly into a HttpJspPage class)
  - Which extends Servlet ☺



JSP Life Cycle



JSP Life Cycle

- Translated JSP example:

```
<%@ page language="java" %>
<%@ page import="java.util.Date" %>
<html>
   <head>
      <title>When am I ?</title>
   </head>
   <body>
      <%= new Date() %>
   </body>
</html>
```

Translated JSP Example

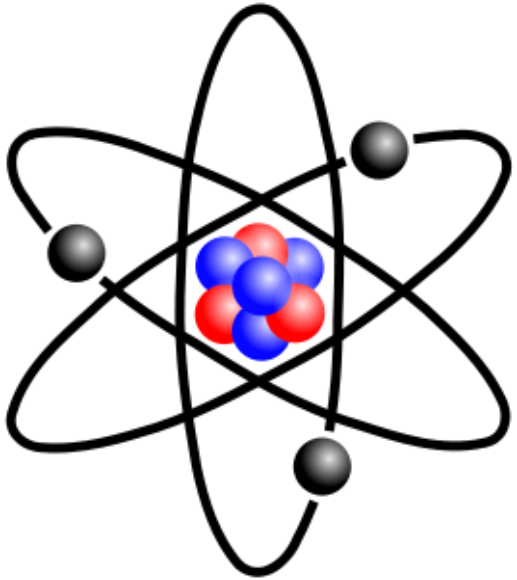
```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response) {
    //...
    PageContext pageContext = null;
    JspWriter out = null;
    //...
    pageContext = _jspxFactory.getPageContext(this,
        request, response, null, true, 8192, true);
    out = pageContext.getOut();
    //...
    out.write("\t<body>\n");
    out.print( new Date() );
    out.write("\n");
    out.write("\t</body>\n");
    //...
}
```

Questions ?

Java Server Pages

JSP ELEMENTS

Directive, action and scripting



Presentation

- Three types of JSP element :
  - Scripting elements
  - Directive elements
  - Action elements

Presentation

- Four elements type:

Element	Description
<% %>	Scriptlets: allow to write Java code inside the JSP
<%= %>	Expresion: Display the value of a variable or returned by a method
<%! %>	Declaration: declare a piece of code (variable, method, …) outside the _jspService(…) method. Be careful : use it to declare a writable variable is not thread-safe !
<%-- --%>	Comments: insert comments which will not be visible inside the generated HTML response

Scripting elements

```
<% String s1 = new String("Hello ! "); %>
<% for(int i = 0; i < 3 ; i++)  { %>
     <%= s1 %>
<% } %>

<br />

<%-- No compilation error because s2
                              is in a declaration ! --%>
<%= s2 %>

<%! String s2 = new String("How are you ?"); %>
```

Hello ! Hello ! Hello !
How are you ?

Directives

- In a JSP page, directives exist to fulfill your needs
- Three types of directive
  - **Page** directives
    - *<%@page ... %>*
  - **Inclusion** directives
    - *<%@include ... %>*
  - **Taglib** directives
    - *<%@taglib ... %>*

Page directive

- Define some page properties
- Page directives can have several attributes
  - Examples:

Property	Syntax
import	<%@ page import="java.util.List" %>
session	<%@ page session="true" %>
contentType	<%@ page contentType="text/html" %>
isELIgnored	<%@ page isELIgnored="true" %>

Include directive

- Include the contents of an HTML or JSP page
  - Happens during translation
  - Not a runtime operation !
- One mandatory attribute: file

- Example:

```
<%@ include file="template/header.html" %>
```

Taglib directive

- Reference a tag library
  - To make custom actions available in the JSP page

<%@taglib prefix="tags" "uri=http://supinfo.com/taglibs/tags" %>

- We'll see more about taglibs later ☺

Implicit objects – 1/2

- On a JSP you have an access to implicit objects
- Below the most useful

Variable name	Type	Description
out	JspWriter	Like the PrintWriter you get on a ServletResponse. Allow to write something on the page
request	HttpServletRequest	The servlet request

Implicit objects – 2/2

- On a JSP you have an access to implicit objects
- Below the most useful

Variable name	Type	Description
response	HttpServletResponse	The servlet response
session	HttpSession	The current user session
application	ServletContext	Context of the JSP page's servlet
config	ServletConfig	The servlet configuration

Implicit objects – Example

- A form on a page **page1.jsp**:

```
<form method="post" action="page2.jsp">
        <input type="text" name="username" />
        <input type="submit" value="OK" />
</form>
```

- On the **page2.jsp** we display the submitted value:

```
<% out.println(request.getParameter("username")); %>
<%-- Or more simplier ... --%>
<%= request.getParameter("username") %>
```

Questions ?

Exercices (1/5)

- Now it’s time to use JSP in our project!

- Create a JSP page named: **listProduct.jsp**
  - Display all the products stored in memory

- Create a JSP page named: **showProduct.jsp**
  - Display details about the product with the id in the URL

Exercices (2/5)

- Create a new JSP page
  - With name: **addProduct.jsp**
  - With an HTML form containing fields to define a new product
  - Put it inside a folder named **auth**
    - To execute the filter…

Exercices (3/5)

- Create a HttpServlet
  - Name it **AddProductServlet**
  - Bind it to **/auth/addProduct** url-pattern
  - Override the **doPost(…)** method
    - Retrieve the form parameters
    - Create a new **SupProduct** object
    - Add it in memory with the DAO
    - Redirect the user to the **ShowProductServlet** to display the new product

Exercices (4/5)

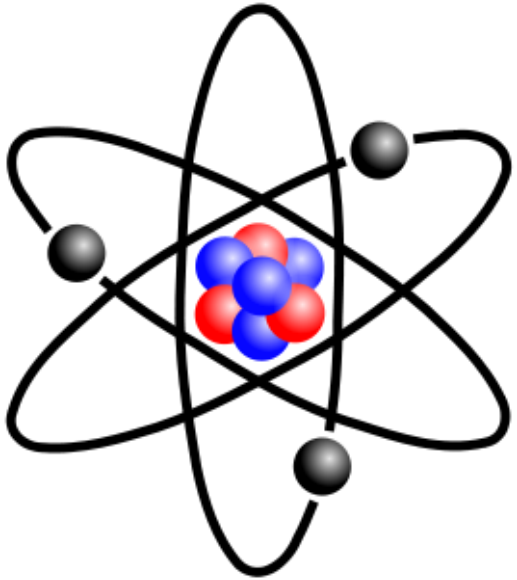
- Create a HttpServlet
  - Name it **LogoutServlet**
  - Bind it to **/logout** url-pattern
  - Override the **doGet(…)** method
    - Remove the session attribute "**username**"
    - Redirect the user to the login page
  - Override the **doPost(…)** method
    - Call the doGet(…) method to do the same thing for GET or POST methods

Exercices (5/5)

- Create a jsp page named: header.jsp
  - With HTML links to:
    - Product listing page
    - Add product page (if the user is logged)
    - Logout or Login Servlet (in function of if the user is logged or not)
  - Include it inside all your JSP pages
- Create a jsp page named: footer.jsp
  - With some information of your choice
  - Include it inside all your JSP pages

Java Server Pages

ACTION ELEMENTS



Introduction

- It provides
  - File inclusions
  - Redirection
  - Object instantiation and handling
  - …

- Syntax: **<jsp:action … >**, where action will be replaced by the action you decide to use

useBean

- Allow to work with JavaBeans inside a JSP page

- JavaBeans are classes following these characteristics
  - Have a public constructor without arguments
  - Have no public attributes
  - Have a getter and a setter for each attribute
  - Implement Serializable

useBean

- How to use that action?

<jsp:useBean … />

- You can pass attributes to that markup
  - **id**: instance name for the bean
  - **class**: Java class name of the JavaBean to use
  - **scope**: page | request | session | application
  - **beanName**: instantiate a Bean from either a class or Serializable template
  - **type**: change the Bean type if its already exists

useBean – Example

- These snippets are equivalent:

```
<jsp:useBean id="student"
                 class="com.supinfo.sun.beans.Student"
                 scope="session" />
```

```
<%
    Student student = null;
    student = (Student) session.getAttribute("student");

    if(student == null) {
        student = new Student();
    }
%>
```

useBean

- Allow to set a property of a JavaBean
- Must **always** be called **after** a <jsp:useBean … />

- Three ways to set properties by using attributes
  1. property="propName" value="Terry"
  2. property="propName" param="paramName"
  3. property="*"

setProperty example

```
<jsp:useBean id="student"
    class="com.supinfo.sun.beans.Student"
    scope="session" />

<jsp:setProperty
    name="student"
    property="idBooster"
    value="300" />

<%-- Set the firstName value with the value of a
request parameter named firstName --%>
<jsp:setProperty
    name="student"
    property="firstName"
    param="firstName" />

<%-- Set all the property values of the bean with
request parameters with the same name --%>
<jsp:setProperty name="student" property="*" />
```

getProperty

- Allow to get a property of a JavaBean
- Must **always** be called **after** a <jsp:useBean … />

```
<jsp:useBean id="student"
                  class="com.supinfo.sun.beans.Student"
                  scope="session" />

<%-- Displays the value of the firstName property --%>
<jsp:getProperty name="student" property="firstName" />
```

include

- Allow to include the result of another resources inside the current one
  - Runtime operation
  - Similar to RequestDispatcher.include(…)
- Use the page attribute to indicate the resource to include

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

Action Elements

forward

- Allow to forward the user to another page
- Use the page attribute to specify the forward's destination

```
<jsp:forward page="aPage.jsp" />
```

param

- Allow to send parameters to
  - <jsp:include … /> or <jsp:forward … />
- Use the attribute
  - name to indicate the parameter's name
  - value to specify the parameter's value

```
<jsp:forward page="aPage.jsp">
        <jsp:param name="idBooster" value="300" />
        <jsp:param name="firstName" value="Alick" />
        <jsp:param name="lastName" value="MOURIESSE" />
</jsp:forward>
```

Questions ?

Java Server Pages

EXPRESSION LANGUAGE

Presentation

- Language allowing to use Java objects on JSP
  - Don’t replace scriptlets
  - But JSTL does…
- Succinct and robust syntax : **${expression}**
- Example: **${sessionScope.user.name}**

EL Literals

- Five Kinds of EL Literal
  - Boolean : **${true}**
  - Integer : **${12345}**
  - Floating point : **${1.16876}**
  - Strings : **${"Hello"}** or **${'World'}**
  - Null : **${null}**

EL Literals

- Five Kinds of EL Literal
  - Boolean : **${true}**
  - Integer : **${12345}**
  - Floating point : **${1.16876}**
  - Strings : **${"Hello"}** or **${'World'}**
  - Null : **${null}**

EL Operators

- Subset of Java Operators
  - Arithmetic:

Operation	Operator
Addition	**+**
Substraction	**-**
Multiplication	*****
Division	**/** or **div**
Module	**%** or **mod**

EL Operators

- Subset of Java Operators
  - Relational:

Operation	Operator
Greater than	**>** or **gt**
Greater than or equals	**>=** or **ge**
Equals	**==** or **eq**
Lower than	**<** or **lt**
Lower than or equals	**<=** or **le**
Not equals	**!=** or **ne**

EL Operators

- Subset of Java Operators
  - Logical:

Operation	Operator
Logical « and »	**&&** or **and**
Logical « or »	**||** or **or**
Logical « not »	**!** or **not**

EL Operators

- empty: Returns true if null or empty
  - Syntax:

${empty obj}

- Returns true if:
  - obj is an empty string
  - obj is an empty array
  - obj is an empty map or collection

EL Operators

- Any attribute in any scope can be displayed in EL
- If an attribute called “title” exists
  - You can retrieve it with this expression : **${title}**
- If an attribute called person exists and if it is a complex object
  - To call the getName() method on it you can use
    - **${person.name}**
    - **${person["name"]}**
    - **${person['name']}**

Implicit objects

- Some implicit objects exists
  - **pageScope**: to access page scope attributes
  - **requestScope**: to access request scope attributes
  - **sessionScope**: to access session scope attributes
  - **applicationScope**: to access application scope attributes
  - **param**: to access request parameters
  - **cookie**: to access cookies
  - …

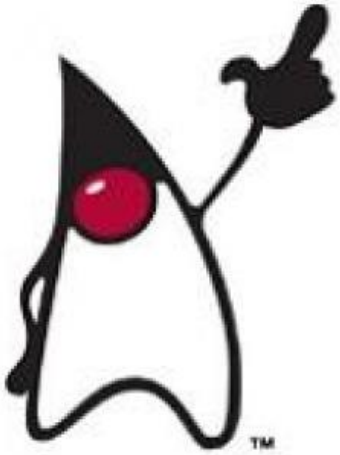
Questions ?

Exercices

- Create a HttpServlet
  - Name it **RemoveProductServlet**
  - Bind it to **/auth/removeProduct** url-pattern
  - Override the **doPost(…)** method
    - Retrieve the product Id in request parameters
    - Remove the corresponding object in memory
    - Redirect the user to the product listing
- Update your **listProduct.jsp** page
  - Add a link to the **RemoveProductServlet** for each product

Java Server Pages

TAGLIBS



Presentation

- Personalized tags
- Like JavaBeans, provide a Java and JSP code separation
- Make it easy to manage a Web application

How does it work

- A Tag handler for each set of tags
  - Java class implementing javax.servlet.jsp.tagtext.Tag
- A Tag Library Descriptor (TLD) file
  - Link between JSP pages and Tag Handler
  - Describes a set of tags

- Path to the TLD could be specified in the deployment descriptor (optional)

Deployment

- More optimized code
  - Modify the path only in the **web.xml** file

```
<jsp-config>
	<taglib>
		<taglib-uri>uriName</taglib-uri>
		<taglib-location>TLDPath</taglib-location>
	</taglib>
</jsp-config>
```

Declaration and use

- First declare your taglib
  - Thanks to
    <%@taglib uri="uriLocation" prefix="aPrefix" %>
  - Then use it
    <aPrefix:tagName attribute="blue" … />

Declaration and use

- Taglib inclusion

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

<html>
        <head><title>My Page</title></head>
        <body>
                <c:out value="Hello !" />
        </body>
</html>
```

- Displays “Hello !”

Java Standard Tag Library

- Also known as JSTL
- Sun specifications
- 4 libraries
  - **Core**: base functions
  - **XML**: XML treatments
  - **Format**: Internationalization
  - **Database**: SQL queries
- Avoid scriptlet usage !

JSTL: Core

- Some Core library available tags
  - **out**: print in the output stream
  - **set**: instantiate or modify a variable
  - **if**: define a conditional block
  - **catch**: catch exceptions
  - …
- Taglib information:
  - URI: http://java.sun.com/jsp/jstl/core
  - Usually prefix: c

JSTL Core example

```
<%@taglib
    uri="http://java.sun.com/jsp/jstl/core"
    prefix="c" %>
...
<table>
    <c:forEach items="${tickets}" var="t">
        <tr>
            <td><c:out value="${t.name}" /></td>
            <td><c:out value="${t.content}" /></td>
            <td>
                <c:if test="${t.editable}">
                    <a href="edit?id=${t.id}">Edit</a>
                </c:if>
            </td>
        </tr>
    </c:forEach>
</table>
...
```

JSTL: Format

- Format library available tags
  - **setBundle**: specify the ResourceBundle to use
  - **message**: print a message associated to a key in the ResourceBundle
  - **param**: used for dynamic message
- Remember
  - A ResourceBundle is composed of **.properties** files. Each file is a *translation* for a specific language

JSTL: Format example

- Considering the file Msg_en.properties at the path:
  - com/supinfo/sun/supcommerce/lang/Msg_en.properties
  - This file contains:

```
message.hello = Hello World!
```

- How to use locale:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt"
    prefix="fmt" %>
<fmt:setBundle
    basename="com.supinfo.sun.supcommerce.lang.Msg" />
<fmt:message key="message.hello" />
```

Questions ?

Exercices

- Download JSTL libraries and put them in the lib directory of your web project

- Refactor your **listProduct.jsp** page to use JSTL instead of Java iteration
  - Use Expression Language instead of scripting elements when you can

Java Server Pages

SERVLETS & JSP

Best practices

- Don’t overuse Java code in HTML pages
  - Avoid spaghetti code
  - Easier to read
  - Easier to maintain
- Use custom tags
  - Easier to read for HTML contents developer

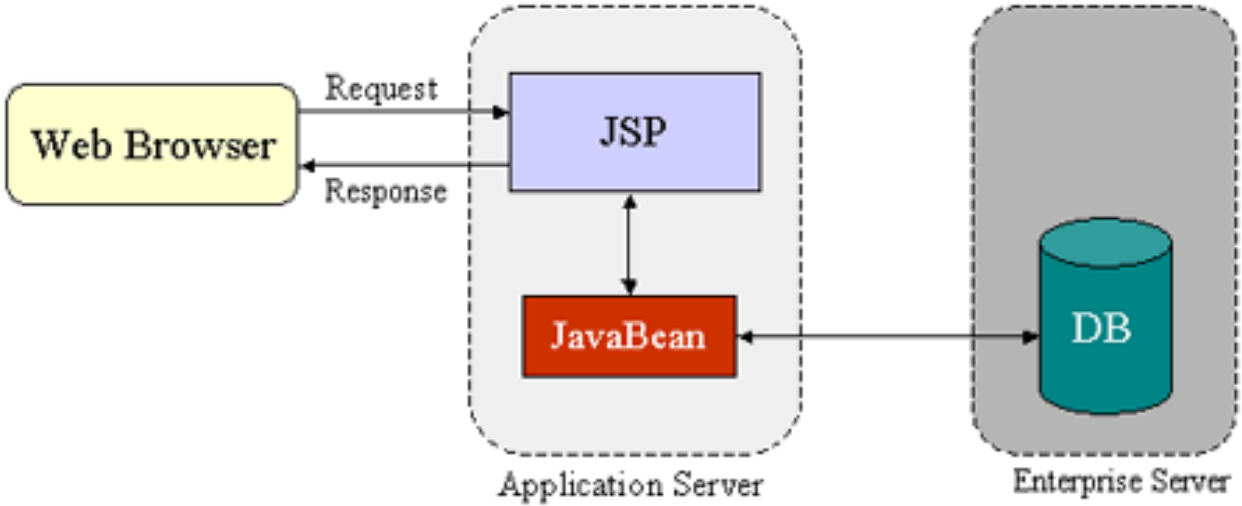
Best practices

- Use Expression Language
  - Easier syntax

- Redirect the user instead of forward his request when you can
  - Bookmarkable URL
  - Better for **Previous** and **Next** buttons of the browser

JSP Model 1 Architecture

- One of the both approaches to use JSP
- The JSP page is responsible for processing requests

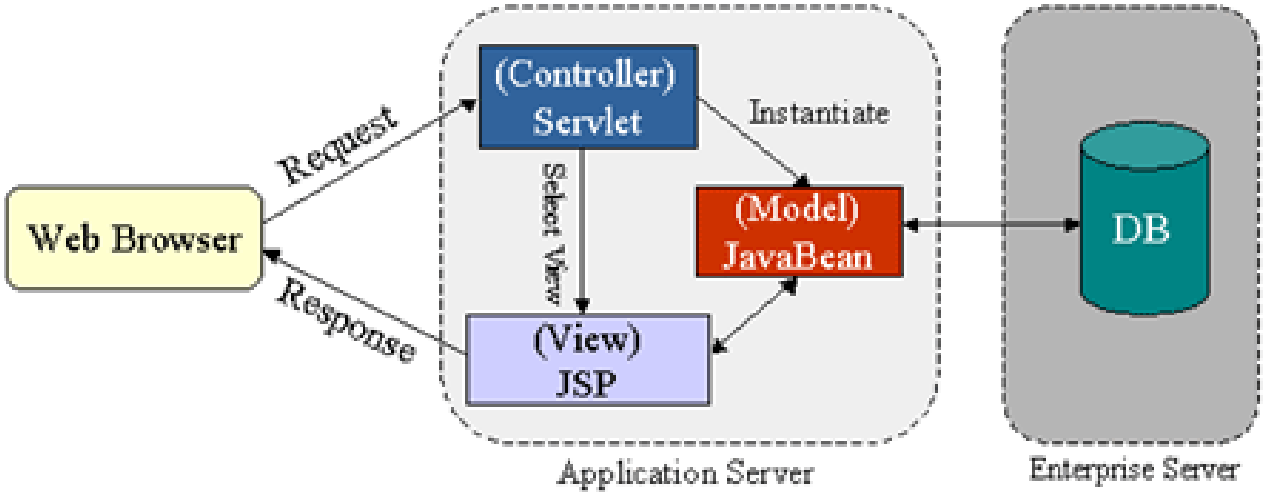


Best practices

- JSP and Servlets work complementary
  - JSP pages used to generate HTML response
  - Servlet for processing tasks
- Servlet acts as a Controller
  - Process request, create beans, …
  - Decide to which JSP page to forward the request
- JSP acts as a View
  - Retrieves objects created by the Controller
  - Use them to construct the HTML response

JSP Model 2 Architecture

- Model View Controller (MVC) design pattern!



Questions ?

Exercices (1/2)

- Now you understand how Servlets and JSP are complementary:
  - Refactor **ListProductServlet** and **listProduct.jsp** to implement **JSP Model 2 Architecture**
    - Put the product list in request attributes
    - Forward the request to the JSP
    - Retrieve the list in the JSP page
  - Refactor **AddProductServlet** and **addProduct.jsp**
    - Override the **doGet(…)** method
    - Forward the request to the JSP page

Exercices (2/2)

- Now you understand how Servlets and JSP are complementary:
  - Refactor **ShowProductServlet** and **showProduct.jsp**
  - Refactor **LoginServlet** and **login.jsp**
  - Replace all scripting elements in your JSP page by JSTL + EL !

The end

Thanks for your attention