Web ontwikkeling 3

Les 1: No Scriptlets

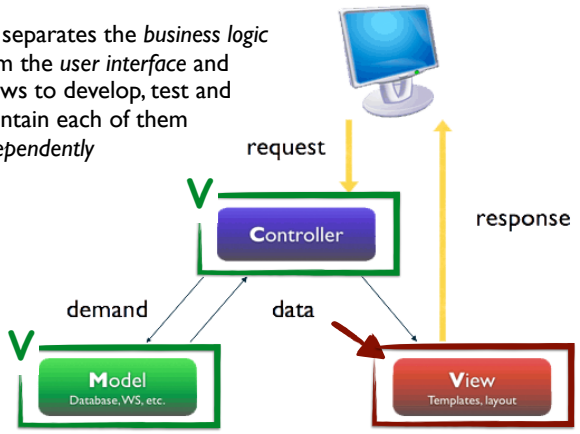
AGENDA

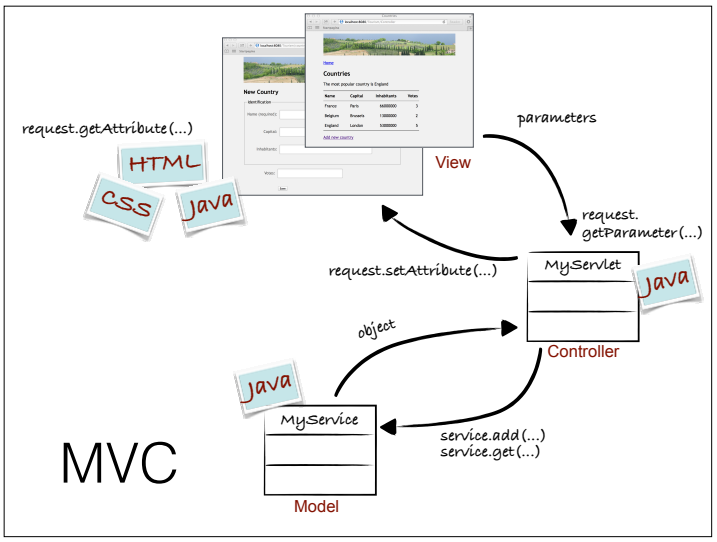
EXAM QUESTIONS…

- ☑ What is the advantage of using expression language instead of JSP scriptlets and JSP expressions
- ☑ What is the difference between the . operator and the [ ]
- ☑ What does JSTL stand for?
- ☑ Name the 6 language constructs of JSP
- ☑ Is JSTL used in the View, Model or Controller?
- ☑ When is a Java class a JavaBean?
- ☑ …

MVC

… separates the *business logic* from the *user interface* and allows to develop, test and maintain each of them *independently*



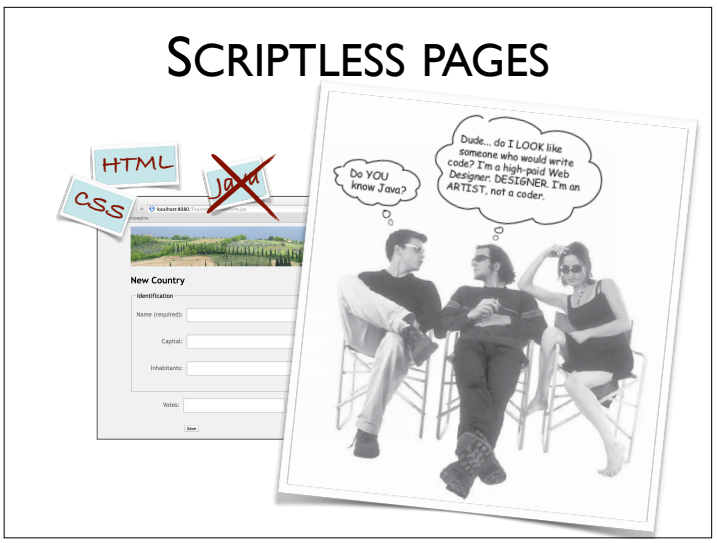


Recap workflow:

- parameters can be passed with HTTP request
- servlet reads them with request.getParameter()
- model is called to add or get an object
- model can return object(s)
- objects can be passed to view with request.setAttribute()
- objects can be read with request.getAttribute()

Technologies used:

- controller: Java
- model: Java
- view: Java, css, html



Problem:

- designers should be able to create view
- good designers or not necessarily good programmers (and vice versa)

Solution: avoid Java in view

JSP 2.0: EXPRESSION LANGUAGE

```
<td>
  <%= ((Country) request.getAttribute("country")).getName()%>
</td>
```

```
<td>
  ${country.name}
</td>
```

no cast!

property

JSP knows:
this must be an attribute

JSP scriplets and expressions are replaced by expression language
Easier:

- JSP assumes you are asking for an attribute —> no request.getAttribute()
- JSP can find out what type you're talking about —> no cast
- JSP expects a property after the dot —> no get-method

Does this mean we should make our properties public?

JAVABEAN

```
public class Country {
    private String name;

    public Country() {
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

public no-arg constructor

getter and setter
define a property

No, as long as we use the JavaBean conventions:

- getters an setters define a property (public getter to be able to read the property, public setter to modify it)
- a default constructor should be present (no arguments)

. OPERATOR

- Syntax:

```
${attribute.property}
${attribute.property.propertyOfProperty}
```

- Only for **Javabean** or **Map**

EXAMPLES . OPERATOR

```
private Country country = new Country("Belgium", "Brussels", 13000000, 5); ...
```

```
${country.votes}
```

```
private Address address = new Address("Herestraat", 49, "Leuven";
private Person customer = new Person("Bert", address; ...
```

```
${customer.address.street}
```

```
private Map<String, Country> countries = new HashMap<String, Country>();...
countries.put(country.getName(), country);
```

```
${countries.Belgium.capital}
```

```
private List<Country> countries = new ArrayList<Country>();
countries.add(country.getName(), country);
```

→ not possible

[ ] OPERATOR

- Syntax:

```
${attribute[“property”]}
${attribute[0]}
```

- For **Javabean, Map, Array, List, ...**

EXAMPLES [ ] OPERATOR

```
private Country country = new Country(“Belgium”, ”Brussels”, 13000000, 5); ..
```

```
${country[“votes”]}
```

```
private Address address = new Address(“Herestraat”, 49, “Leuven”;
private Person customer = new Person(“Bert”, address; ...
```

```
${customer[“address”][“street”]}
```

```
private Map<String, Country> countries = new HashMap<String, Country>();...
countries.put(country.getName(), country);
```

```
${countries[“Belgium”][“capital”]}
```

```
private List<Country> countries = new ArrayList<Country>();
countries.add(country.getName(), country);
```

```
${countries[0]}
```

WHAT IF …

I want to show a parameter and not an attribute ?

READ PARAMETER

http://localhost:8080/Tourism/Controller?votes=3

```
<td>
   ${param.votes}
</td>
```

parameter name

JSP knows:
this is probably not an attribute
I should look for a parameter

WHAT IF …

I want to be sure my project does not contain scriptlets ?

DISABLE SCRIPTLETS

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://
                       java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

   <jsp-config>
      <jsp-property-group>
         <url-pattern>*.jsp</url-pattern>
         <scripting-invalid>true</scripting-invalid>
      </jsp-property-group>
   </jsp-config>

   <welcome-file-list>
      <welcome-file>index.html</welcome-file>
   </welcome-file-list>
</web-app>
```

If you want to be sure that all script lets are replaced, you can disable them in your web.xml

HTTP Status 500 - /countryForm.jsp (line: 20, column: 5) Scripting elements (<%!, <jsp:declaration, <%=, <jsp:expression, <%, <jsp:scriptlet) are disallowed here.

type Exception report

message /countryForm.jsp (line: 20, column: 5) Scripting elements (<%!, <jsp:declaration, <%=, <jsp:expression, <%, <jsp:scriptlet) are disallowed here.

description The server encountered an internal error that prevented it from fulfilling this request.

exception

```
org.apache.jasper.JasperException: /countryForm.jsp (line: 20, column: 5) Scripting elements (
        org.apache.jasper.compiler.DefaultErrorHandler.jspError(DefaultErrorHandler.java:42)
        org.apache.jasper.compiler.ErrorDispatcher.dispatch(ErrorDispatcher.java:443)
        org.apache.jasper.compiler.ErrorDispatcher.jspError(ErrorDispatcher.java:89)
        org.apache.jasper.compiler.Validator$ValidateVisitor.visit(Validator.java:721)
        org.apache.jasper.compiler.Node$Scriptlet.accept(Node.java:932)
        org.apache.jasper.compiler.Node$Nodes.visit(Node.java:2375)
        org.apache.jasper.compiler.Node$Visitor.visitBody(Node.java:2427)
        org.apache.jasper.compiler.Node$Visitor.visit(Node.java:2433)
        org.apache.jasper.compiler.Node$Root.accept(Node.java:474)
        org.apache.jasper.compiler.Node$Nodes.visit(Node.java:2375)
        org.apache.jasper.compiler.Validator.validateExDirectives(Validator.java:1817)
        org.apache.jasper.compiler.Compiler.generateJava(Compiler.java:217)
        org.apache.jasper.compiler.Compiler.compile(Compiler.java:373)
        org.apache.jasper.compiler.Compiler.compile(Compiler.java:353)
        org.apache.jasper.compiler.Compiler.compile(Compiler.java:340)
        org.apache.jasper.JspCompilationContext.compile(JspCompilationContext.java:657)
        org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:357)
        org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:390)
        org.apache.jasper.servlet.JspServlet.service(JspServlet.java:334)
```

There is much more you can do with expression language.
This does not mean you have to use all these possibilities: they might lead to bad design.
Remember: your view should contain any other than presentation logic!

REFACTOR

```
<fieldset>
  <legend>Identification</legend>
  <p class="form-group <%=request.getAttribute("nameClass")%>">
    <label class="control-label" for="name">Name (required): </label>
    <input id="name" name="name" type="text"
      value="<%=request.getParameter("name") != null ? request.getParameter("name") : ""%>">
  </p>
  <p class="form-group <%=request.getAttribute("capitalClass")%>">
    <label class="control-label" for="capital">Capital: </label>
    <input id="capital" type="text" name="capital"
      value="<%=request.getParameter("capital") != null ?
                request.getParameter("capital") : ""%>">
  </p>
  <p class="form-group <%=request.getAttribute("inhabitantsClass")%>">
    <label class="control-label" for="inhabitants">Inhabitants:</label>
    <input id="inhabitants" name="inhabitants" type="text"
      value="<%=request.getParameter("inhabitants") != null ?
                request.getParameter("inhabitants") : ""%>">
  </p>
</fieldset>
```

Assignment: refactor this, using expression language.

RESULT

```
<fieldset>
  <legend>Identification</legend>
  <p class="form-group ${nameClass}">
    <label class="control-label" for="name">Name (required): </label>
    <input id="name" name="name" type="text" value="${param.name}">
  </p>
  <p class="form-group ${capitalClass}">
    <label class="control-label" for="capital">Capital: </label>
    <input id="capital" type="text" name="capital" value="${param.capital}">
  </p>
  <p class="form-group ${inhabitantsClass}">
    <label class="control-label" for="inhabitants">Inhabitants:</label>
    <input id="inhabitants" name="inhabitants" type="text" value="${param.inhabitants}">
  </p>
</fieldset>
```

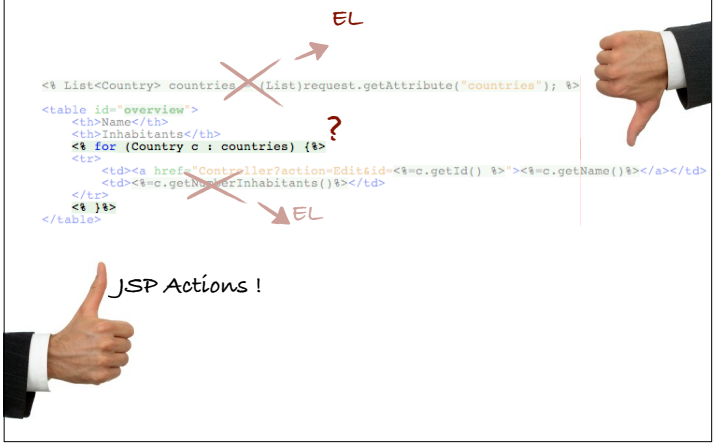
REMARK

```
<fieldset>
  <legend>Identification</legend>
  <p class="form-group ${nameClass}">
    <label class="control-label" for="name">Name (required): </label>
    <input id="name" name="name" type="text" value="${country.name}">
  </p>
  <p class="form-group ${capitalClass}">
    <label class="control-label" for="capital">Capital: </label>
    <input id="capital" type="text" name="capital" value="${country.capital}">
  </p>
  <p class="form-group ${inhabitantsClass}">
    <label class="control-label" for="inhabitants">Inhabitants:</label>
    <input id="inhabitants" name="inhabitants" type="text" value="${country.inhabitants}">
  </p>
</fieldset>
```

you could also pass country as an attribute

AGENDA

SCRIPTLESS PAGES

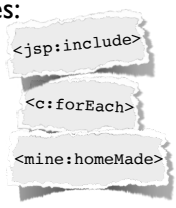


Example country overview: some parts we can refactor with expression language, but other parts we can't. Expression language is not enough. That's why there is something else called JSP Actions.

JSP ACTIONS

components of a 'tag library'

- = predefined XML *tags*
- Categories:
  - standard
  - custom



JSP actions are tags you can use next to the normal html tags. They offer you extra functionality, without using to much Java in your pages.

STANDARD ACTIONS

STANDARD ACTIONS

- defined in JSP itself
- begin with default `jsp:` prefix

<jsp:include>	Includes the response from a servlet or JSP page during the request processing phase
<jsp:param>	Adds a parameter value to a request handed off to another servlet
<jsp:plugin>	Generates HTML that contains the appropriate browser-dependent elements (OBJECT or EMBED) needed to execute an applet with the Java Plug-in software

STANDARD ACTIONS

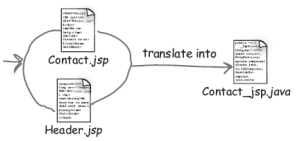
```
<jsp:include page="header.jsp"/>
```

```
<jsp:include page="header.jsp">
   <jsp:param name="title"value="Countries"/>
</jsp:include>
```

```
<jsp:plugin type="applet" code="Molecule.class" codebase="/html">
  <jsp:params>
    <jsp:param name="molecule" value="molecules/benzene.mol" />
  </jsp:params>
  <jsp:fallback>
    <p>Unable to load applet</p>
  </jsp:fallback>
</jsp:plugin>
```

INCLUDE

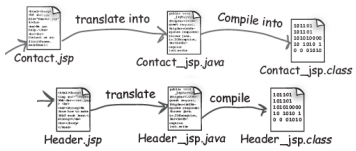
```
<%@include file="header.jsp" %>
```



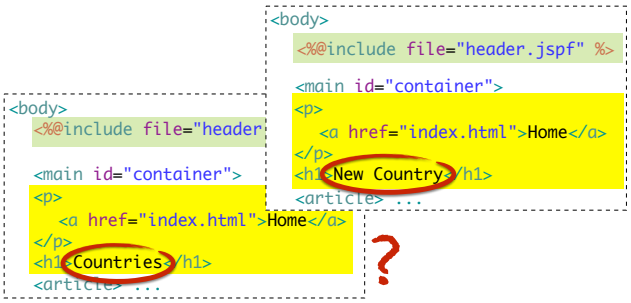
inserts SOURCE of "Header.jsp" at translation time

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

inserts RESPONSE of "Header.jsp" at runtime



DRY



DRY


```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<p>
  <a href="index.html">Home</a>
</p>
<h1>${param.title}</h1>
```

title.jsp

STANDARD

`<jsp:useBean>`	Makes a JavaBeans component available in a page
`<jsp:getProperty>`	Gets a property value from a JavaBeans component and adds it to the response
`<jsp:setProperty>`	Sets a JavaBeans component property value
`<jsp:forward>`	Forwards the processing of a request to servlet or JSP page

Replaced by Expression Language, ...

CUSTOM ACTIONS

CUSTOM ACTIONS

- user-defined JSP language element
- NOT defined in JSP itself → include !
- Example: **JSP Standard Tag Library**

Custom actions are located in extra libraries. You will have to include them in your build path, and you will have to reference them from your JSP page.

JSTL

- **J**SP **S**tandard **T**ag **L**ibrary
- Bundels recurring functionality:
  - iteration
  - choice
  - ...

ITERATION

```
<c:forEach var="country" items="${countries}">
   <tr>
       <td>${country.name}</td>
       ...
   </tr>
</c:forEach>
```

CHOICE: IF

```
<c:if test="${country.numberInhabitants > 1000000}">
      ...
</c:if>
```

CHOICE: IF ELSE

```
<c:choose>
      <c:when test="${country.numberInhabitants > 1000000}">
            ...
      </c:when>
      <c:when test="${country.numberInhabitants > 50000000}">
            ...
      </c:when>
      <c:otherwise>
            ...
      </c:otherwise>
</c:choose>
```

USE LIBRARIES

Add in WEB-INF/lib:

jstl-1.2.jar

JSP page:

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

LIBRARIES

- Core: `http://java.sun.com/jsp/jstl/core`
- XML: `http://java.sun.com/jsp/jstl/xml`
- Internationalization: `http://java.sun.com/jsp/jstl/fmt`
- SQL: `http://java.sun.com/jsp/jstl/sql`
- Functions: `http://java.sun.com/jsp/jstl/functions`

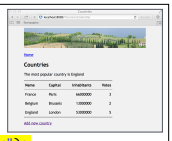
In this course, you will be using the Core library.

YOUR OWN ACTIONS

SUMMARY JSP

Scriplet:	`<% List<Country> countries = (List)request.getAttribute("countries"); %>`
Expression:	`<%= country.getName() %>`
Directive:	`<%@ include file="header.jsp" %>`
Declaration:	`<%! int counter = 0; %>`
EL:	`${ country.name }`
Action:	`<jsp:include page="header.jsp"/>`

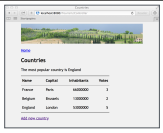
REFACTOR



```
<h1>Countries</h1>
<% Country mostPopularCountry = (Country) request.getAttribute("popular");
if (mostPopularCountry != null) { %>
  <p> The most popular country is <%=mostPopularCountry.getName()%> </p>
<% } %>
<% Collection<Country> countries = (Collection<Country>) request.getAttribute("countries");
if (countries != null) { %>
 <table id="overview">
   <tr>
     <th>Name</th>
     <th>Capital</th>
     <th class="getal">Inhabitants</th>
     <th class="getal">Votes</th>
   </tr>
  <% for (Country country : countries) { %>
   <tr>
     <td><%=country.getName()%></td>
     <td><%=country.getCapital()%></td>
     <td class="getal"><%=country.getNumberInhabitants()%></td>
     <td class="getal"><%=country.getVotes()%></td>
   </tr>
  <% } %>
  </table>
<% } %>
```

Assignment: refactor this, using expression language and JSP actions.

RESULT



```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

...
<h1>Countries</h1>
    <p> The most popular country is ${popular.name} </p>
    <table id="overview">
        <tr>
            <th>Name</th>
            <th>Capital</th>
            <th class="number">Inhabitants</th>
            <th class="number">Votes</th>
        </tr>
        <c:forEach var="country" items="${countries}">
        <tr>
            <td>${country.name}</td>
            <td>${country.capital}</td>
            <td class="number">${country.numberInhabitants}</td>
            <td class="number">${country.votes}</td>
        </tr>
        </c:forEach>
    </table>
```

AGENDA