

Aula de hoje

- Ponto da situação /questões?
 - Teste: Essencialmente semanas 1-4 e 7-10
 - Resultados da avaliação por pares #1/3
- Realização de um tutorial sobre JavaServer Pages
 - A entregar até ao final da semana
- Simulacro de teste
 - Regresso à *Main room* pelas 16h30.

Avaliação por pares

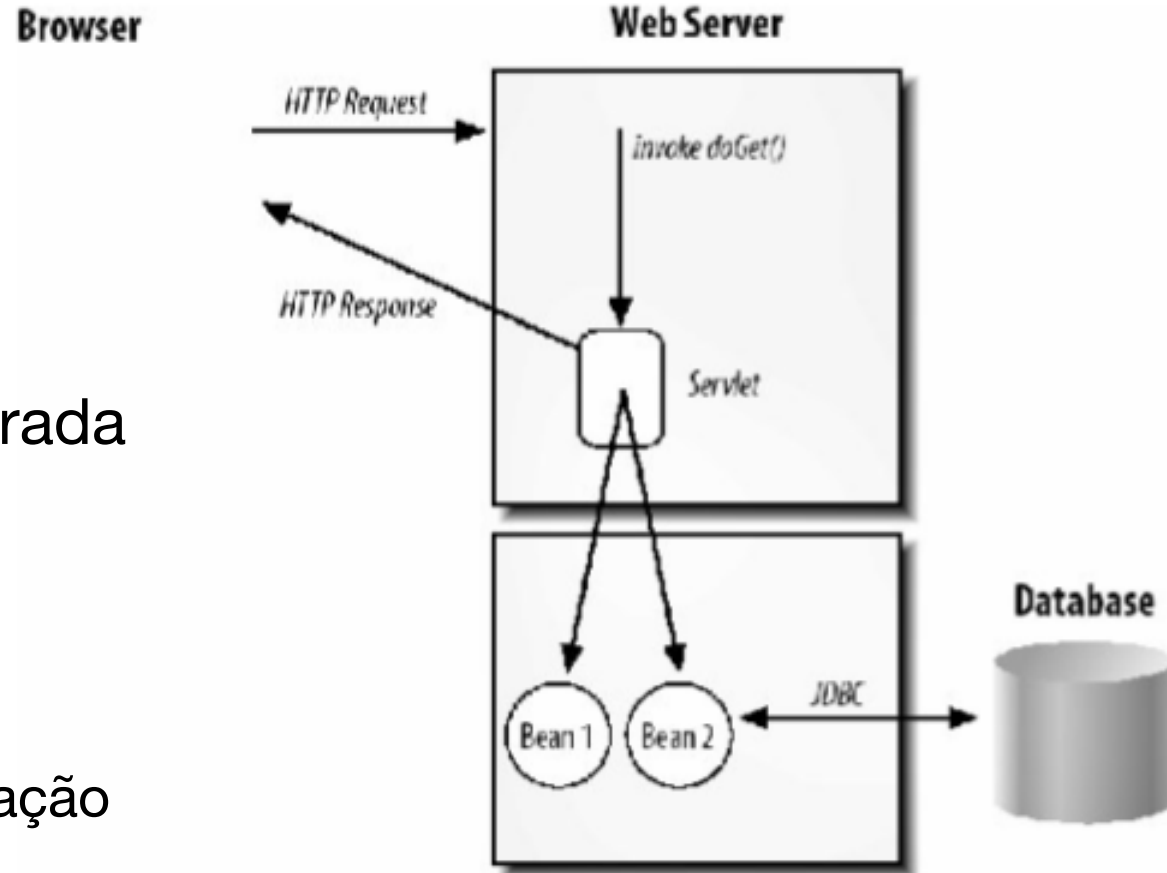
- 25/26 respostas
- Excepto um grupo:
 - 0.975 - 1.025 — no limiar de alterar notas
- Próxima etapa
 - Primeira quinzena de Junho

Módulo 12

PROG. WEB. 3 - JAVASERVER PAGES

Servlets

- Tecnologia Java
- Geração de HTML *on the fly* no servidor
- Nova vista (página) gerada a cada pedido
 - Vista é programada imperativamente, não desenhada
 - Não encorajam separação entre apresentação e negócio/conteúdo

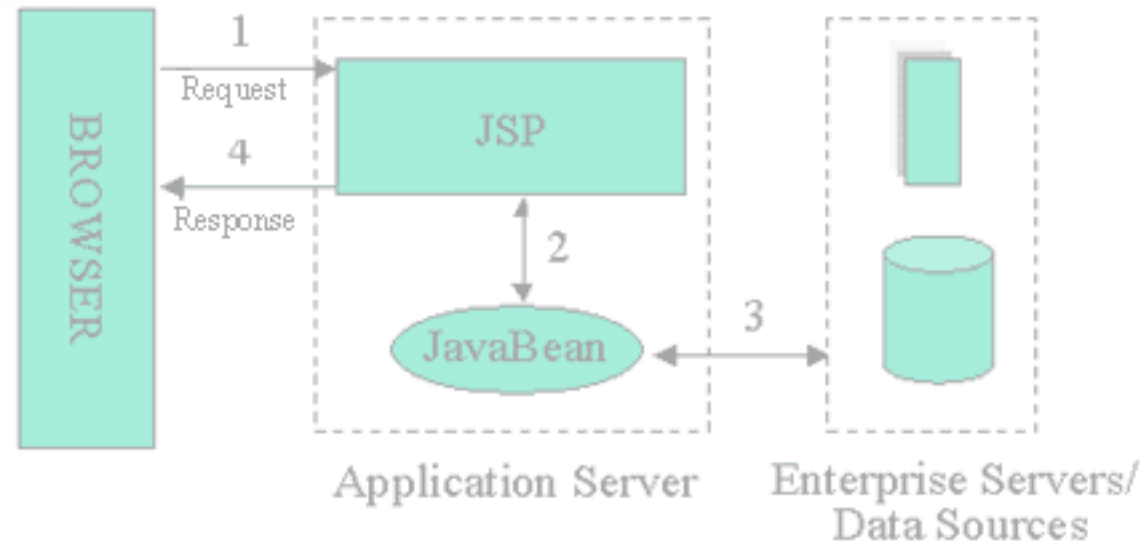


JSP (Java Server Pages)

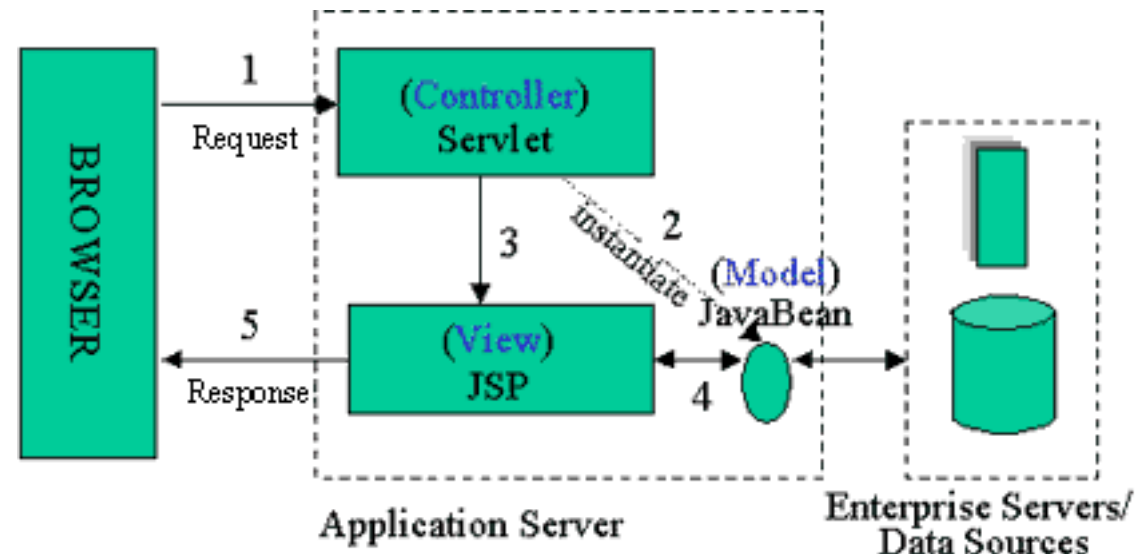
- Um salto de abstracção em relação às Servlets
- Ficheiros .jsp compilados para Java (servlets)
 - ou directamente para bytecode
 - ou interpretados *on the fly*...
- Permitem incluir código Java no HTML
- Código Java compilado e executado para gerar HTML
- Necessário Web server com um Java Servlet container
 - Apache Tomcat, (JBoss) Undertow, Eclipse Jetty

JSP: Arquitecturas

- Model 1 architecture



- Model 2 architecture (MVC-like)

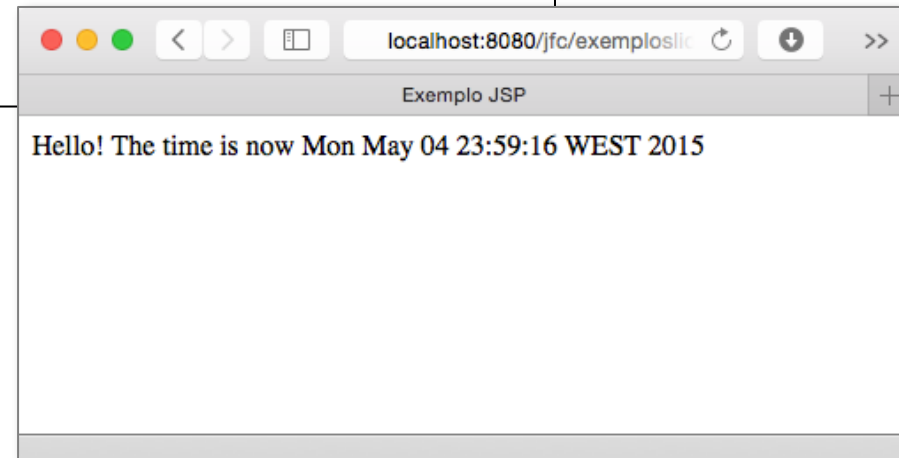


JSP – Expressões

- `<%= expressão Java %>` ou `<jsp:expression> expressão Java </jsp:expression>`

```
<doctype html>
<html>
  <head>
    <title>Exemplo JSP</title>
  </head>
  <body>
    Hello! Time is now <%= new java.util.Date() %>
  </body>
</html>
```

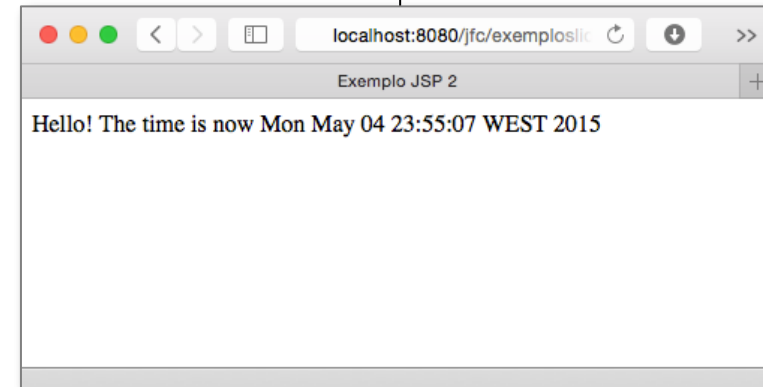
- Expressão avaliada em *run time*



JSP – Directivas

- `<%@ tipo atributo %>` ou `<jsp:directive.tipo atributo />`
 - tipo: page/include/taglib/...

```
<doctype html>
<%@ page import="java.util.Date" %>
<html>
  <head>
    <title>Exemplo JSP 2</title>
  </head>
  <body>
    Hello! Time is now <%= new Date() %>
  </body>
</html>
```



JSP – Directivas (ctd.)

- Directiva page: informação sobre a servlet a gerar
 - import/contentType/pageEncoding/session/...

- Directiva include:

```
<%@ include file="relativeURL" %>
```

```
<jsp:directive.include file="relativeURL" />
```

- Directiva taglib
 - bibliotecas de tags externas (taglibs)

```
<%@ taglib uri="URIForLibrary" prefix="tagPrefix" %>
```

```
<jsp:directive.taglib uri="URIForLibrary"  
                    prefix="tagPrefix" />
```

JSP: Declarações

- `<%! declarações Java (class level) %>`
- `<jsp:declaration> declarações Java (class level) </jsp:declaration>`

```
<html>
  <head><title>Exemplo JSP 3</title></head>
  <body>
    <jsp:declaration>
      Date theDate = new Date();
      Date getDate() {
        System.out.println( " Evaluating date now!" );
        return theDate;
      }
    </jsp:declaration>
    Hello! Time is now <%= getDate() %>
  </body>
</html>
```



Falso!

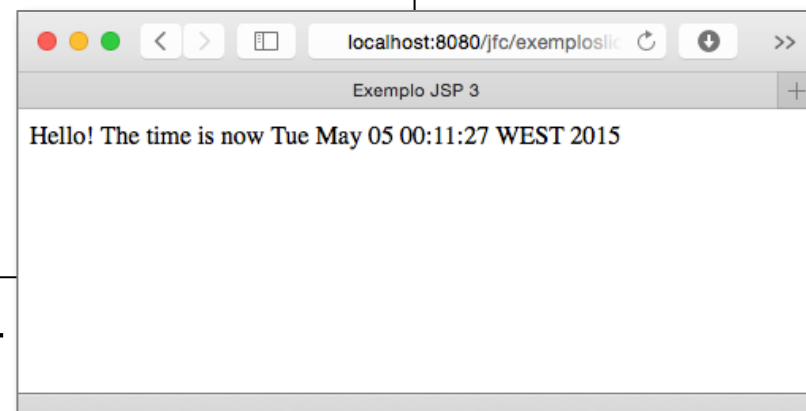
- Mas agora a data não muda!

JSP – Scriptlets

- `<% bloco de código Java %>`
- `<jsp:scriptlet> bloco de código Java </jsp:scriptlet>`

```
<doctype html>
<%@ page import="java.util.Date" %>
<html>
  <head><title>Exemplo JSP 3</title></head>
  <body>
    <% System.out.println( "Evaluating date now!" );
      Date date = new Date();
    %>
    Hello! The time is now <%= date %>
  </body>
</html>
```

- Por si só uma Scriptlet não produz HTML
 - consideradas má ideia...



JSP – Gerar HTML

```
<body>
    <%! public int factorial(int n) {
        int fact = 1;
        for (int i = 1; i <= n; i++) { fact *= i; }
        return fact; }
    %>
    <% out.println("<table border=1>");
        for ( int i = 1; i <= 5; i++ ) {
            out.println("<tr>");
            out.println("<td>"+i+"</td>");
            out.println("<td>"+factorial(i)+"</td>");
            out.println("</tr>");
        }
        out.println("</table>");
    %>
</body>
```

1	1
2	2
3	6
4	24
5	120

2	150
4	54

- o mesmo que uma servlet?!

JSP – Gerar HTML

```
<body>
  <%! public int factorial(int n) {
    int fact = 1;
    for (int i = 1; i <= n; i++) { fact *= i; }
    return fact; }
  %>
  <table border=1>
    <% for ( int i = 1; i <= 5; i++ ) { %>
      <tr>
        <td><%= i %></td>
        <td><%= factorial(i) %></td>
      </tr>
    <% } %>
  </table>
</body>
```

1	1
2	2
3	6
4	24
5	120

2	150
4	54

- Alternativa a utilizar out é misturar HTML e Java

Java scriptlets are obsolete

- **Readability** – There are already enough languages in web page programming (HTML, JavaScript and CSS). Keep the Java code in Java classes where it belongs.
- **Separation of Concerns** – Presentation logic and business logic should not be mixed. Easier to take advantage of the Java bean paradigm.
- **Reusability** – Scriptlets break OOP. They cannot be extended or encapsulated.
- **Maintainability** – Easier to refactor Java classes than Java code in JSP pages.

JSP – JSTL (JSP Standard Tag Library)

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<jsp:useBean id="fB" scope="session" class="com.slides.Factorial"/>
...
<table border=1>
  <c:forEach varStatus="i" begin="1" end="5">
    <c:set target="{fB}" property="fact" value="{i.index}" />
    <tr>
      <td>{i.index}</td>
      <td>{fB.fact}</td>
    </tr>
  </c:forEach>
</table>
<c:out value="{fB.fact}" />
```

1	1
2	2
3	6
4	24
5	120

2	150
---	-----

```
package com.slides;
import java.io.Serializable;
public class Factorial implements Serializable {
  private int fact;
  public Factorial () { this.fact = 0; }
  public void setFact(int n) {
    this.fact = 1;
    for (int i = 1; i <= n; i++) { this.fact *= i; }
  }
  public int getFact() { return this.fact; }
}
```

- Solução recomendada

JSP/JSTL – include: directiva vs. acção

- Directiva JSP (`<%@ include file="..." %>`)
 - inclusão estática do texto
 - um pouco melhor em termos de desempenho
- Acção JSP (`<jsp:include page="..." />`)
 - inclusão dinâmica do resultado (recompilação automática)
 - classes (servlets) mais pequenas
- JSTL (`<c:import url="..." />`)
 - mais flexível (fonte externas, permite guardar resultado em variável)

JSP – Expression Language (EL)

- Sintaxe base: `${expr}`
- Acesso simplificado a propriedades Beans e a outros objectos (request, session, application, etc.)

<code>.</code> (ponto)	aceder a uma propriedade de um objecto ou a uma entrada num Map
<code>[]</code>	aceder a um array ou List
<code>empty</code>	testar se variável está vazia

- Avaliação de expressões

<code>+, -, *, /</code> , etc.	Operações aritméticas
<code>< (lt)</code> , <code>> (gt)</code> , <code><= (le)</code> , etc.	Operações de comparação.
<code>! (not)&& (and)</code> , <code> (or)</code> , etc.	Operações lógicas

- Acesso simplificado a funções (definidas em tag libraries)
`${name_space:função(params)}`