TM & © 2005 Zinkia Entertainment S.L. Licensed by Granada Ventures Ltd.

Java Server Pages

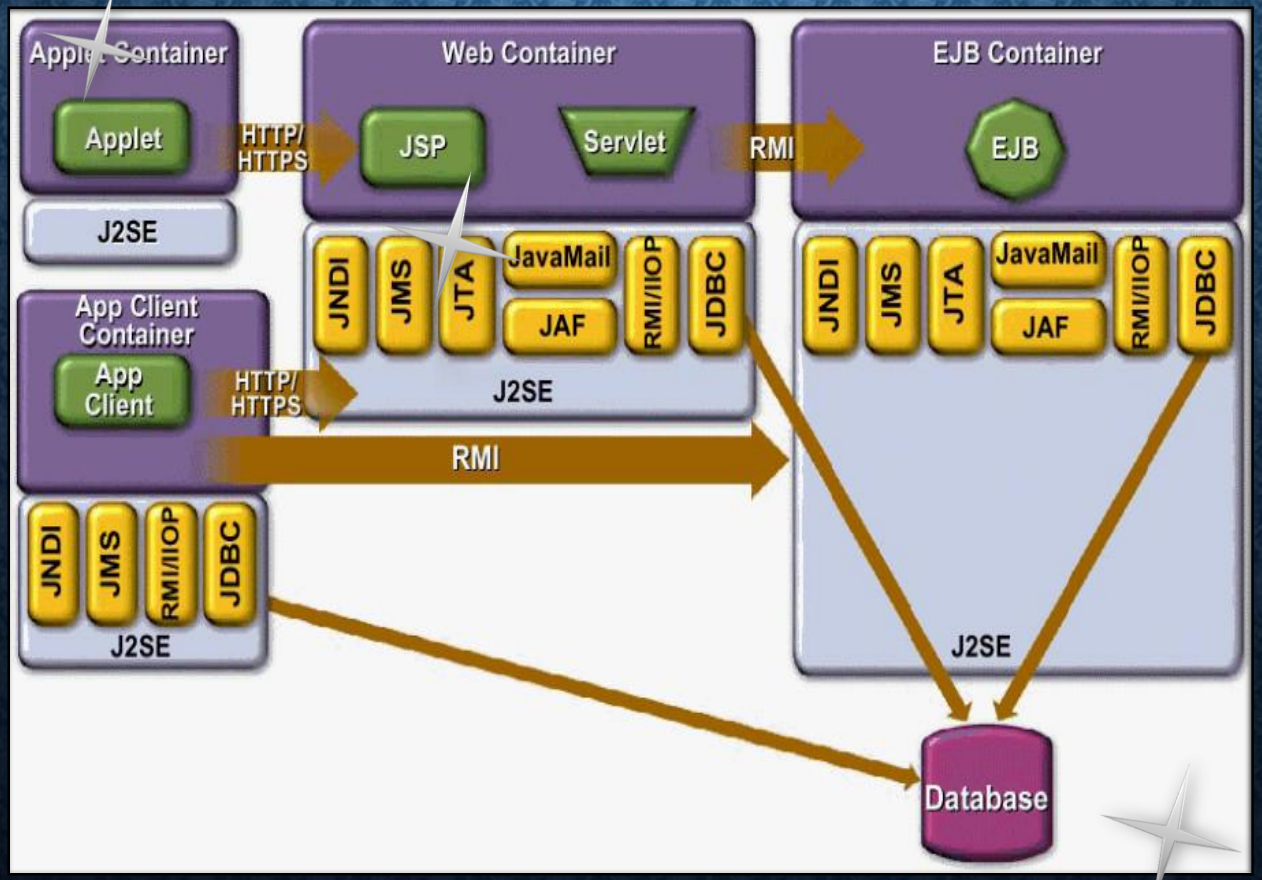
PBW

zinkia
GRANADA International
POCOYO Learning Through Laughter™

```
<html>
        <body>
                <% System.out.println (“
                “);
                %>
                <font color=white>
                        Java Server Pages
                </font>
        </body>
</html>
```

JSP

JSP dan Servlet Sebagai Web Component



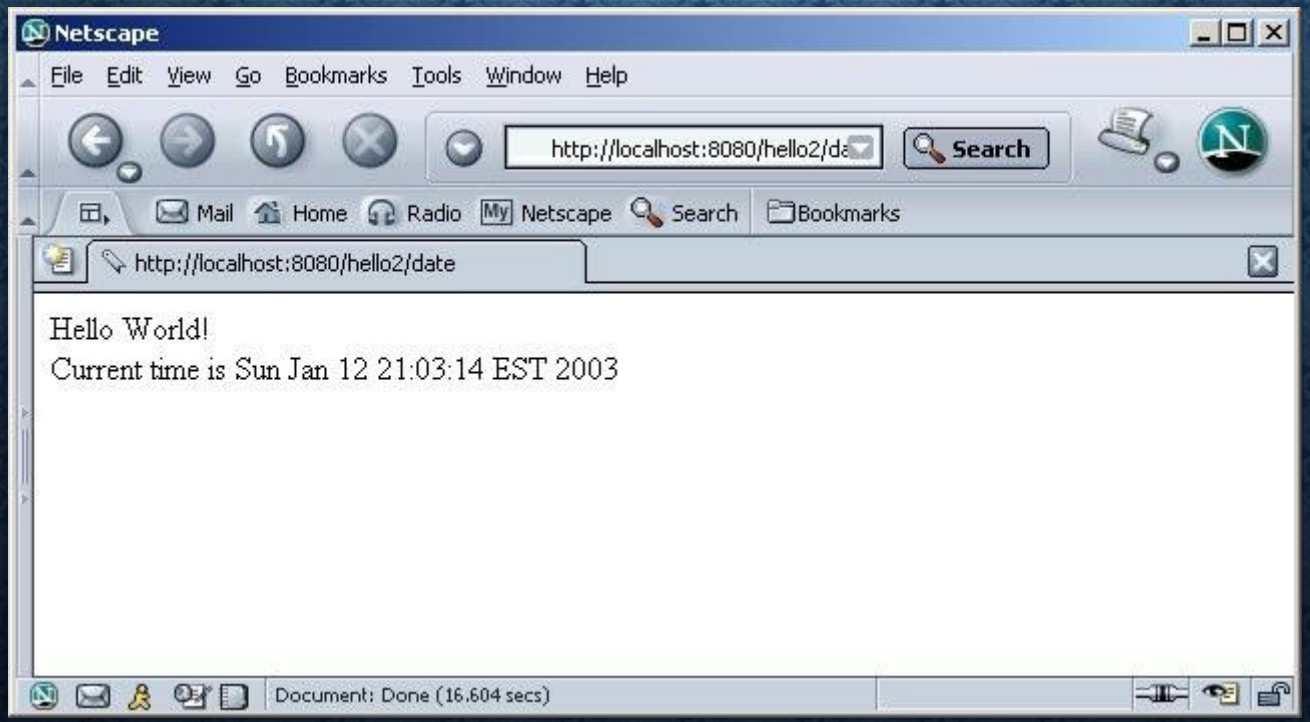
Apa itu JSP Page?

- Dokumen berbasiskan teks:
  yang bisa mengembalikan halaman STATIS atau pun DINAMIS ke browser *client*
- Konten dinamis dan statis dapat dicampur
- Konten statis: HTML, XML, Text
- Konten dinamis:
  - Java Code
  - Tampilan *Properties* dari JavaBeans
  - Meminta *Business Logic* yang didefinisikan dalam *Custom Tag*

Halaman JSP Sederhana
(Putih: Statis | Kuning:Dinamis)

```
<html>
   <body>
        Hello World!<br>
        Current time is <%= new java.util.Date() %>
   </body>
</html>
```

Output


Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/hello2/da
Search
Mail Home Radio Netscape Search Bookmarks
http://localhost:8080/hello2/date
Hello World!
Current time is Sun Jan 12 21:03:14 EST 2003
Document: Done (16.604 secs)

Perbandingan

Servlet

- HTML code in Java
- Not easy to author

JSP

- Java-like code in HTML
- Very easy to author
- Code is compiled into a servlet

Keuntungan JSP

- *Content* dan *Display Logic* dipisah
- Menyederhanakan pengembangan aplikasi web dengan JSP, JavaBeans dan *Custom Tags*
- Mendukung *Software Reuse* melalui penggunaan komponen-komponen (JavaBeans, Custom tags)
- Automatic Deployment
  - kompilasi dilakukan otomatis apabila terjadi perubahan pada JSP
- *Platform-independent*

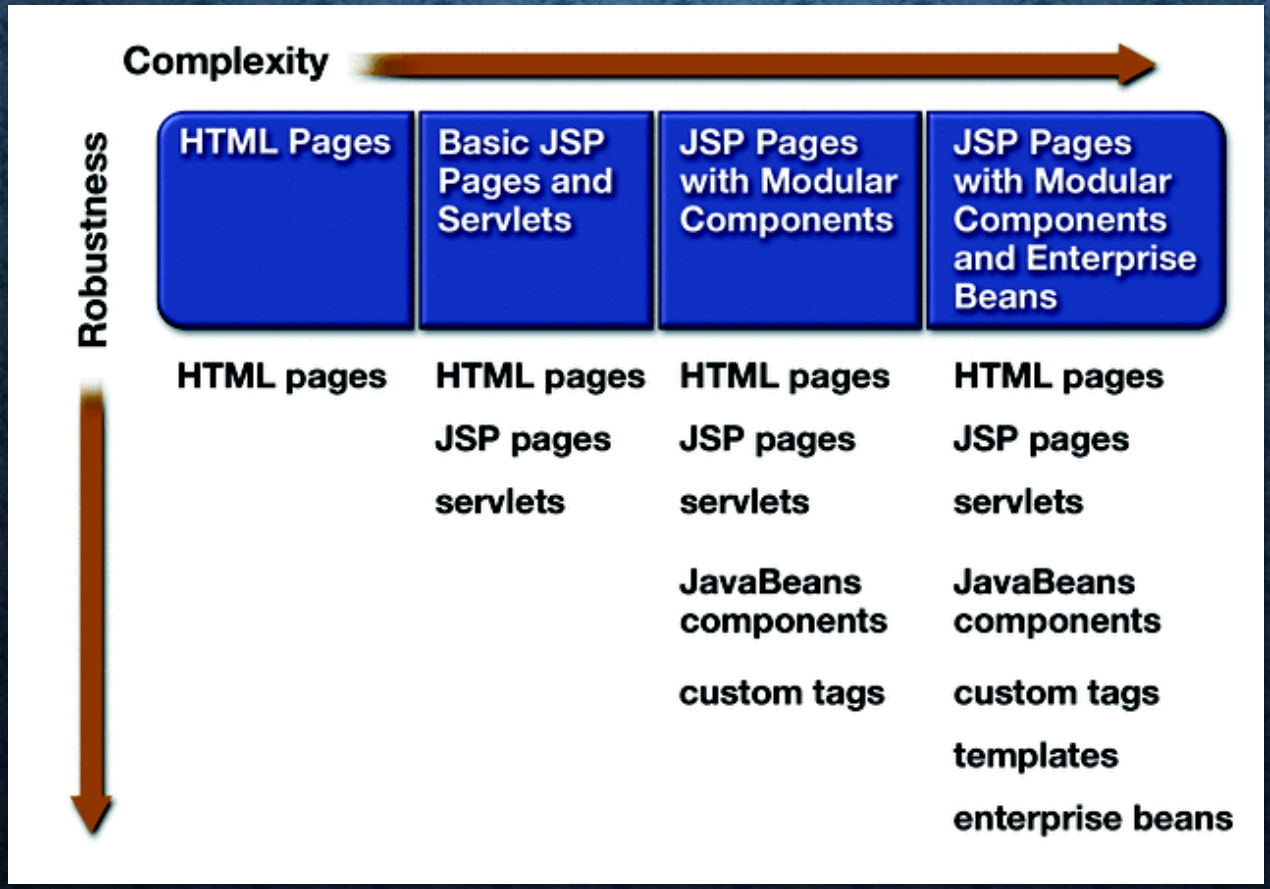
Kenapa JSP diletakkan di atas Servlet

- Servlets bisa melakukan banyak hal , tetapi sulit untuk:
  - Menggunakan println() untuk meN-*generate* halaman HTML
  - Me-*maintain* halaman HTML
- Tidak butuh di-*compile*
  tidak butuh diatur CLASSPATH dan *Packaging*

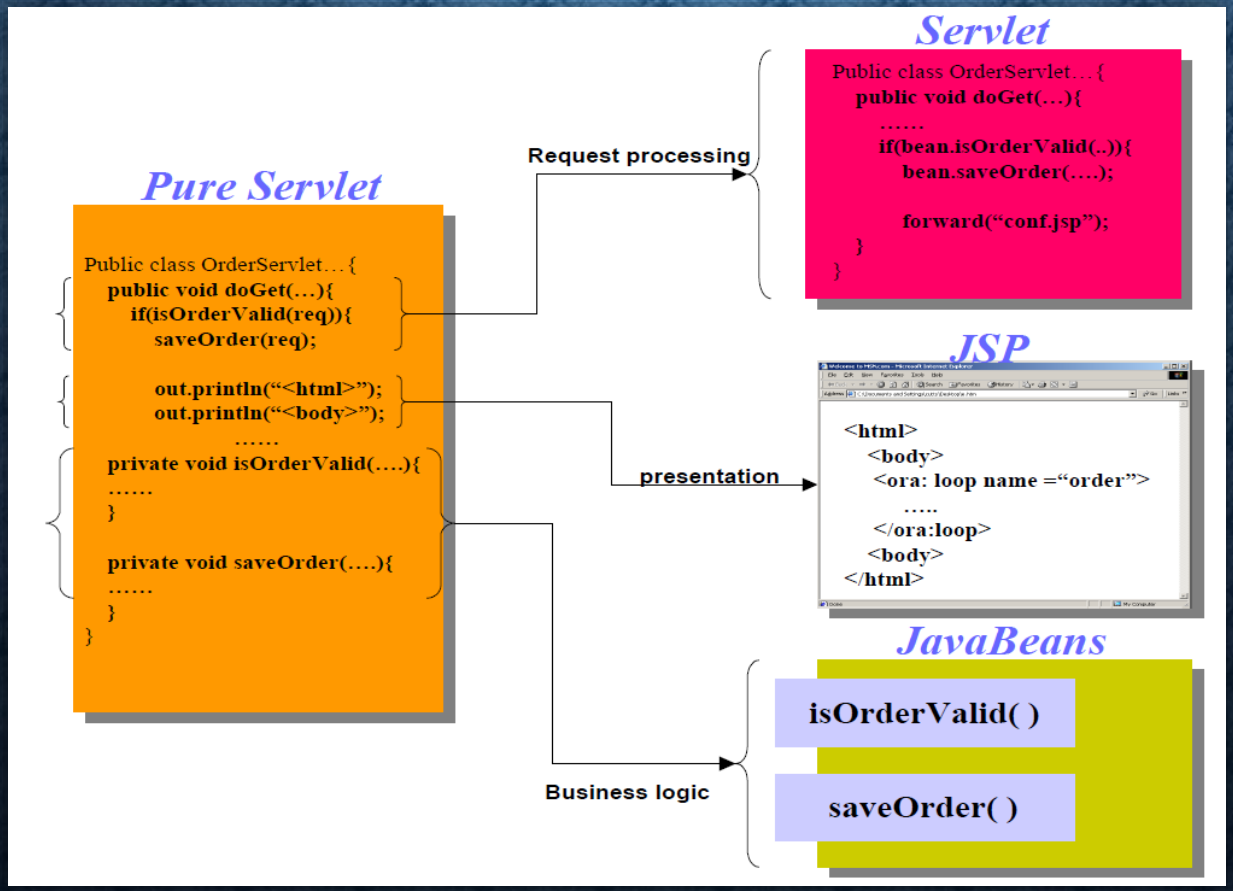
Pilih yang mana: JSP di atas Servlet atau sebaliknya ?

- Kita perlu menerapkan dan memanfaatkan kelebihan dari masing-masing:
  - Kekuatan Servlets adalah “controlling dan dispatching”
  - Kekuatan JSP adalah “displaying”
- Secara umum dalam lingkungan produksi, keduanya (JSP dan Servlet) digunakan dalam sesuatu yang disebut Pola MVC (Model View Controller)
  - Servlet menangani bagian CONTROLLING
  - JSP menangani bagian VIEW

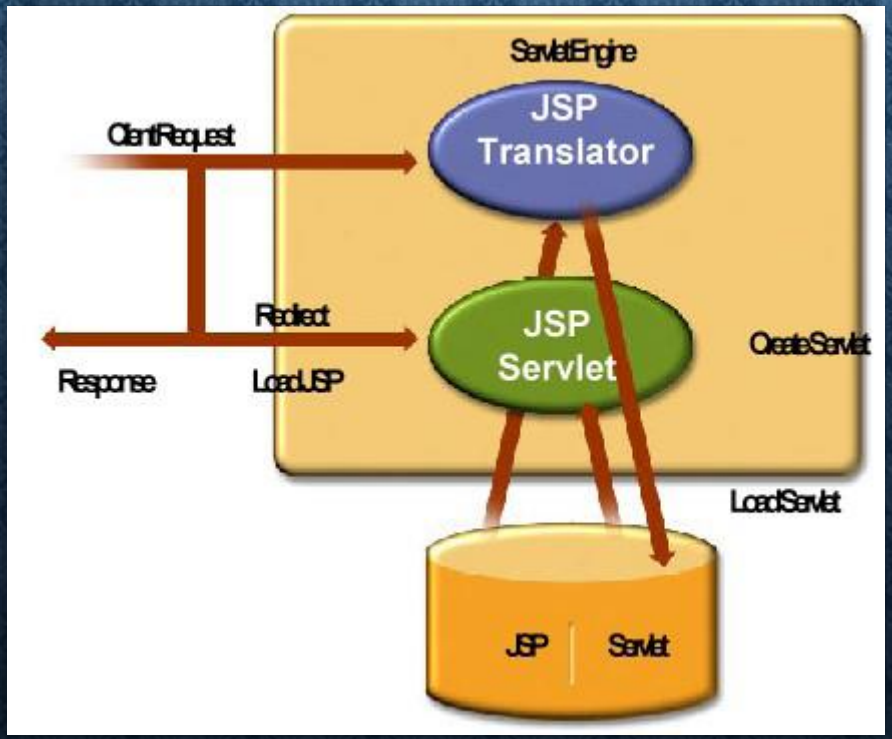
Desain Aplikasi Web



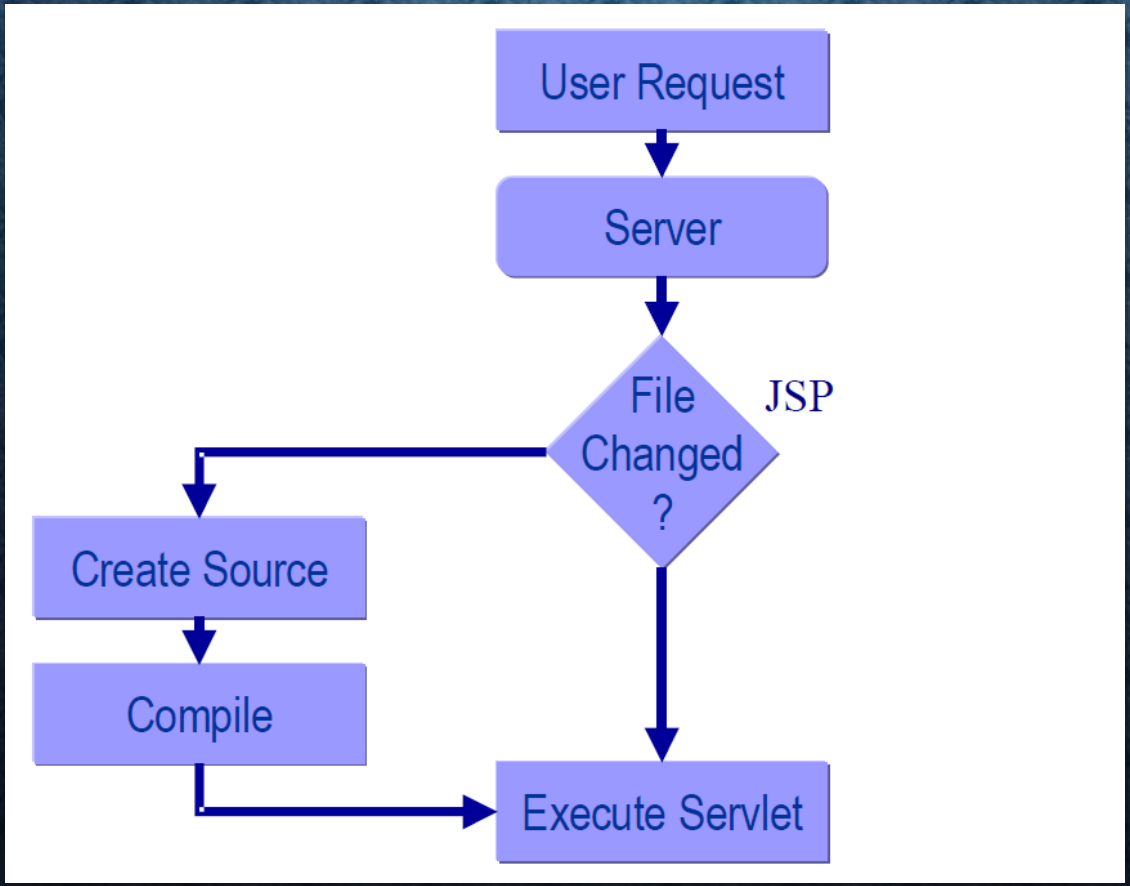
Memisahkan proses *request* dari presentasi



Arsitektur JSP



Bagaimana JSP bekerja ?



Pengembangan Aplikasi JSP

Development and Deployment Steps

1. Tulis dan compile *Web component code*

- Membuat *development tree structure*
- Menuliskan baik *servlet code* dan/atau *JSP pages* beserta *helper code* yang berkaitan
- Membuat build.xml untuk proses *Ant-based build* (dan manajemen *life-cycle* aplikasi lain)

Development Tree Structure

- Menjaga *source* aplikasi web terpisah dari *compiled file*
  - Memfasilitasi pengembangan iteratif
- Root directory
  - Build.xml: Ant-build file
  - Context.xml: file konfigurasi aplikasi opsional
  - Src: Java source dari servlets dan komponen JavaBeans
  - Web: Halaman JSP dan halaman HTML , gambar

2. Create any static resources

- HTML pages
  - Custom pages
  - Login pages
  - Error pages
- Gambar yang digunakan oleh halaman HTML atau JSP

Create deployment descriptor (web.xml)

- *Deployment descriptor* terdiri atas *deployment time runtime instructions* ke *Web container*
- Setiap aplikasi web harus memiliki web.xml

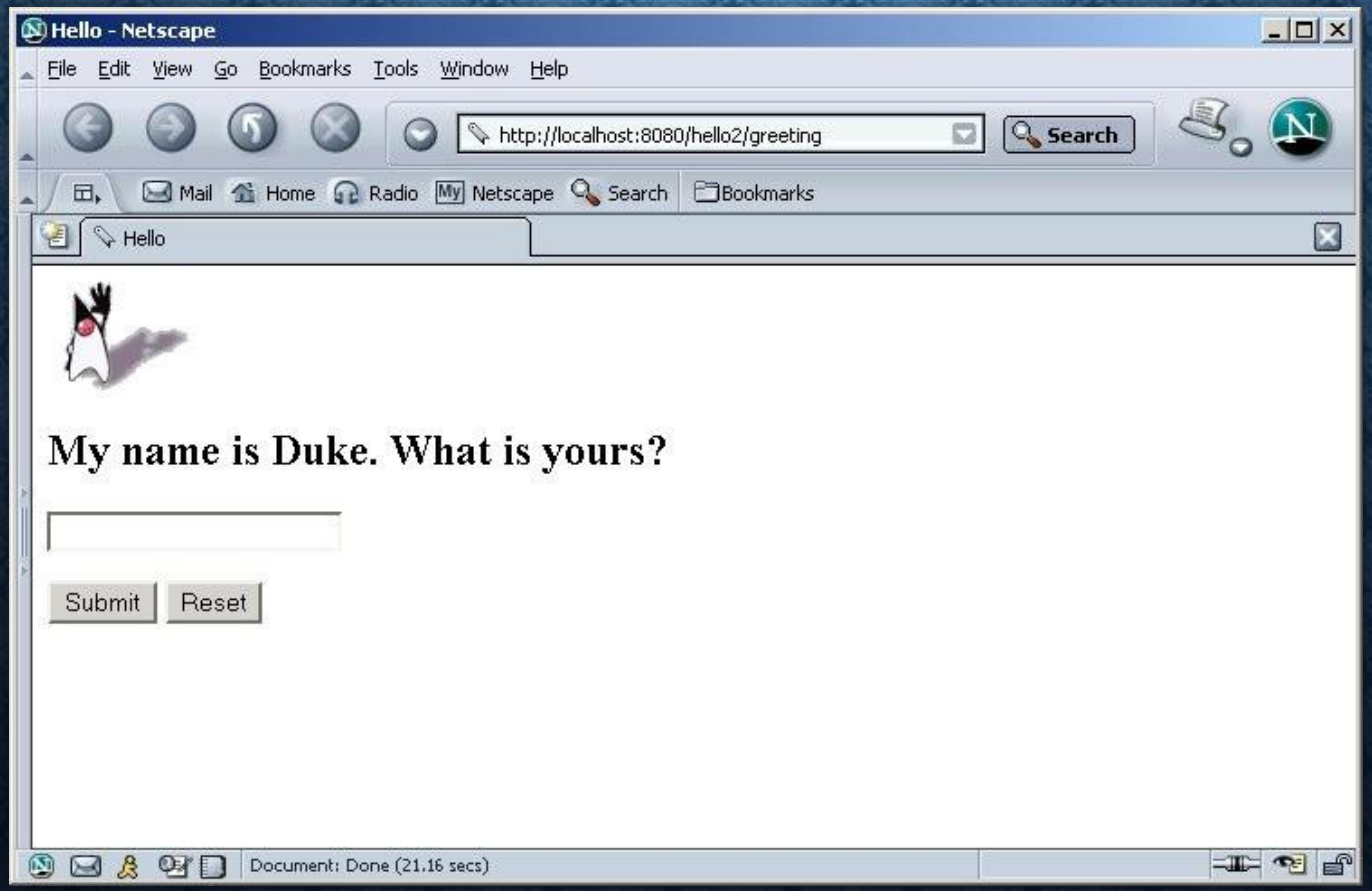
4. Build the Web application

- *Build process* dibuat dari:
  - Membuat direktori build (jika belum ada) dan subdirektorinya
  - MeN-*copy* file JSP ke dalam direktori build
  - Mengompilasi Java code ke dalam direktori build/WEB-INF/classes
  - MeN-*copy* web.xml ke dalam direktori build/WEB-INF
  - Memasukkan gambar yang dibutuhkan ke dalam direktori build

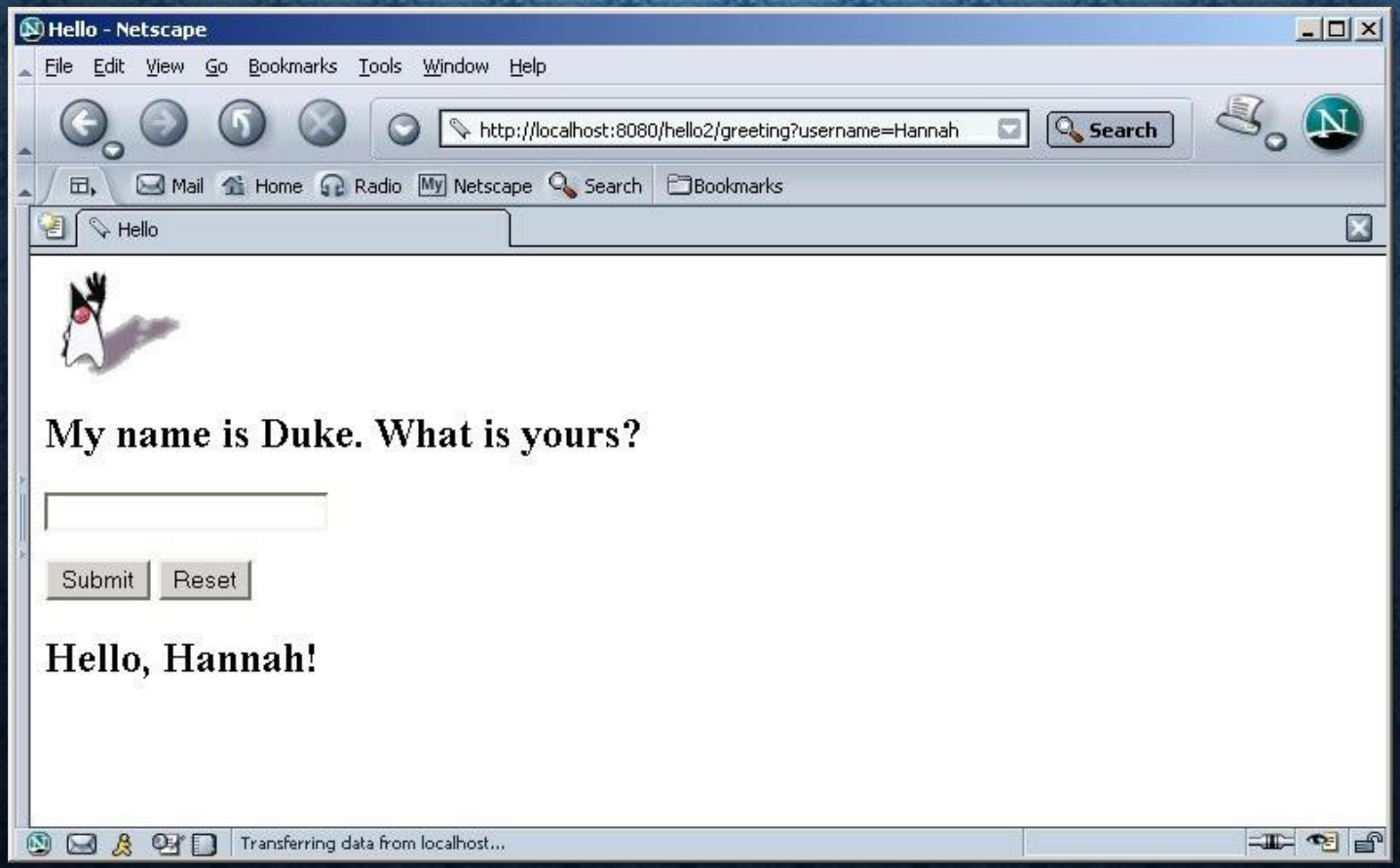
5. Install or Deploy Web application

- Ada 2 cara berbeda untuk menanamkan aplikasi web (*deploy*)
  - Dengan minta tolong Tomcat Manager: dengan mengirimkan sebuah perintah kepadanya (Tomcat manager sebenarnya hanyalah aplikasi servlet lain yang selalu *running*)
  - Secara manual meN-*copy* file-file ke direktori webapps milik Tomcat dan kemudian me-*Restart* Tomcat

6. Client mengakses Web Application



response.jsp



JSP Scripting Elements

- Sisipkanlah Java code ke dalam servlet yang kemudian akan di-*generate* dari halaman JSP
- Minimalkan penggunaan JSP scripting elements di dalam halaman JSP jika memungkin
- Ada 3 bentuk
  - Expressions: <%= Expressions %>
  - Scriptlets: <% Code %>
  - Declarations: <%! Declarations %>kan

Expressions

- Ketika fase eksekusi
  - Expression dievaluasi dan dikonversi menjadi sebuah String
  - String kemudian disisipkan ke dalam stream output secara langsung
  - Hasilnya seperti out.println(expresssion)
  - Bisa menggunakan *predefined variable* (objek implisit) di dalam expression-nya
- Format
  - <%= Expression %> atau
  - <jsp:expression>Expression</jsp:expression>
  - Semi-colons (;) are not allowed for expressions

Contoh: Expressions

- Menampilkan waktu sekarang dengan class Date
  - Current time: <%= new java.util.Date() %>
- Menampilkan angka acak dengan class Math
  - Random number: <%= Math.random() %>
- Menggunakan objek implisit
  - Your hostname: <%= request.getRemoteHost() %>
  - Your parameter: <%= request.getParameter("yourParameter") %>
  - Server: <%= application.getServerInfo() %>
  - Session ID: <%= session.getId() %>

Scriptlets

- Digunakan untuk menyisipkan Java code ke dalam method servlet jspService()
- Bisa melakukan sesuatu yang tidak bisa dilakukan sendiri oleh expressions
  - Mengatur *response headers* dan *status codes*
  - Menulis ke server log
  - Mengupdate database
  - Mengeksekusi code yang mengandung *loop* dan *conditional*
- Format
  - <% Java code %> atau
  - <jsp:scriptlet> Java code</jsp:scriptlet>

Contoh: Scriptlet

- Menampilkan *query string*

```
<%
        String queryData = request.getQueryString();
        out.println("Attached GET data: " + queryData);
%>
```

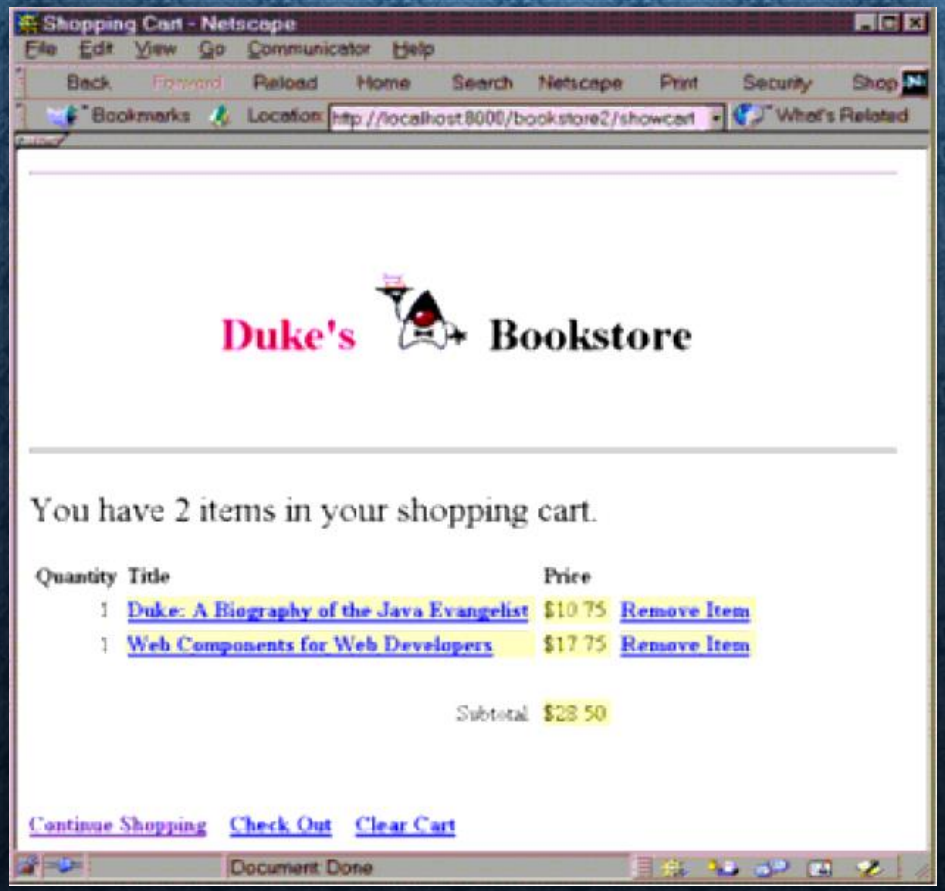
- Mengatur *response type*

```
<% response.setContentType("text/plain"); %>
```

Contoh: Scriptlet dengan loop

```
<%
            Iterator i = cart.getItems().iterator();
            while (i.hasNext()) {
            ShoppingCartItem item =
            (ShoppingCartItem)i.next();
            BookDetails bd = (BookDetails)item.getItem();
%>
<tr>
<td align="right" bgcolor="#ffffff">
<%=item.getQuantity()%>
</td>
<td bgcolor="#ffffaa">
<strong><a href="
<%=request.getContextPath()%>/bookdetails?bookId=
<%=bd.getBookId()%>"><%=bd.getTitle()%></a></strong>
</td>
...
<%
            // End of while
            }
%>
```

Contoh: Hasil scriptlet



Declarations

- Digunakan untuk mendefinisikan *variable* atau *method* yang disisipkan ke dalam tubuh utama dari class servlet
- Biasanya digunakan dengan Expressions atau Scriplets
- Untuk inisialisasi dan *cleanup* di dalam halaman JSP, gunakan *declarations* untuk meN-*override* method jspInit() dan jspDestroy()
- Format:
  - <%! method or variable declaration code %>
  - <jsp:declaration> method or variable declaration code </jsp:declaration>

Contoh: JSP Page Fragment

```
<H1>Some heading</H1>

<%!

   private String randomHeading() {

         return("<H2>" + Math.random() + "</H2>");

   }

%>

<%= randomHeading() %>
```

Contoh: menghasilkan servlet code

```
public class xxxx implements HttpJSPPage {
        private String randomHeading() {
                return("<H2>" + Math.random() + "</H2>");
        }
        public void _jspService(    HttpServletRequest request,
                                    HttpServletResponse response)
                                    throws ServletException, IOException {
                response.setContentType("text/html");
                HttpSession session = request.getSession(true);
                JSPWriter out = response.getWriter();
                out.println("<H1>Some heading</H1>");
                out.println(randomHeading());
                ...
        }
        ...
}
```

Contoh: Declarations

```
<%!
        private BookDBAO bookDBAO;
        public void jspInit() {
                ...
        }
        public void jspDestroy() {
                ...
        }
%>
```

Gunakan XML Syntax

- Dari JSP 1.2
- Contoh
  - <jsp:expression>Expression</jsp:expression>
  - <jsp:scriptlet> Java code</jsp:scriptlet>
  - <jsp:declaration> declaration code</jsp:declaration>
- Bisa mengikutsertakan
  - Validasi XML (melalui XML Schema)
  - Banyak XML Tools
    - Editor
    - Transformer
    - Java APIs

MeN-*include* konten di halaman JSP

- 2 mekanisme untuk menyertakan (*include*) *web resource* lain ke dalam sebuah halaman JSP
  - include directive
  - jsp:include element

Include Directive

- Diproses ketika halaman JSP ditranslasikan ke dalam sebuah kelas servlet
- Efek dari directive adalah mampu menyisipkan teks yang terkandung di dalam file lain (baik statis maupun dinamis) ke dalam JSP yang meN-*include*-kan
- Digunakan untuk menyertakan konten banner, informasi *copyright*, atau potongan konten yang ingin Anda tampilkan secara berulang di beberapa halaman
- Sintaks dan Contoh:

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

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

jsp:include Element

- Diproses ketika sebuah halaman JSP dieksekusi
- Mengizinkan kita untuk menyertakan konten (statis atau dinamis) ke dalam sebuah halaman JSP
- Sintaks dan contoh
  - <jsp:include page="includedPage" />
  - <jsp:include page="date.jsp"/>

Kapan kita menggunakannya?

- Gunakan include directive jika file jarang berubah
  - Karena lebih cepat dari jsp:include
- Gunakan jsp:include untuk konten yang sering berubah
- Gunakan jsp:include jika halaman yang ingin disertakan tidak bisa diputuskan sampai halaman utama di-*request*

Forwarding ke Web component lain

- Mekanisme yang sama seperti di servlet
- Sintaks
  - <jsp:forward page="/main.jsp" />
- *Request object* orisinil disediakan ke halaman target melalui elemen jsp:parameter

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

<jsp:param name="param1" value="value1"/>

</jsp:forward>
```

Directives

- Directive adalah pesan ke JSP container agar berdampak kepada struktur keseluruhan dari servlet
- Sintaks
  - <%@ directive {attr=value}* %>

3 Tipe Directives

- Page
  - <%@ page import="java.util.* %>
- Include
  - <%@ include file="header.html" %>
- Taglib
  - <%@ taglib uri="mytags" prefix="codecamp" %>

Implicit Objects

- request (HttpServletRequest)
- response (HttpServletRepsonse)
- session (HttpSession)
- application(ServletContext)
- out (of type JspWriter)
- config (ServletConfig)
- pageContext

Is this the end of slides ?¿

Bersambung…

Nantikan Episode Selanjutnya
Kuliah sendiri-sendiri di rumah