رمضان 1429

الإصدار 1.0

Java Server Pages (JSP) Basics

أساسيات JSP

Disclaimer & Acknowledgments إخلاء مسؤولية و عرفان

- Even though Sang Shin and Sameer Tyagi are full-time employees of Sun Microsystems, the contents here are created as their own personal endeavor and thus does not reflect any official stance of Sun Microsystems.

  على الرغم من أن Sang Shin يعمل في شركة SUN فإن محتوى هذه الدروس تم إنجازها بمجهوده الشخصي وبالتالي لا تعبر بأي حال عن الموقف الرسمي للشركة SUN

- Sun Microsystems is not responsible for any inaccuracies in the contents.

  لا تتحمل شركة Sun أي مسؤولية عن الأخطاء الموجودة في محتوى هذه الدروس

- Acknowledgments عرفان
  - Large portion of slides, speaker notes, and example code of this presentation are created from "JSP" section of Java WSDP tutorial written by Stephanie Bodoff of Sun Microsystems

    محتوى هذه الشريحة و الأمثلة مقتبسة من فصل "JSP" للكتاب التعليم الذي كتبه Stephanie Bodoff لصالح Sun Microsystems : "Java WSDP"

  - Many slides are borrowed from "Servlet/JSP" codecamp material authored by Doris Chen of Sun Microsystems
  - Some example codes and contents are borrowed from "Core Servlets and JavaServer Pages" book written by Marty Hall

Revision History تواريخ المراجعات

- 12/24/2002: version 1 created by Sameer Tyagi
- 01/12/2003: version 2 revised by Sang Shin with speaker notes
- 05/13/2003: version 3 revised (Sang Shin)

Sub-topics of JSP
المواضيع الفرعية لJSP

Sub-topics of JSP

المواضيع الفرعية JSP

- JSP Basic — أساسيات JSP
- JavaBeans for JSP
- Custom tags — الوسوم الخاصة
- JSP Standard Tag Library (JSTL)

مكتبة الأوسمة المعيارية لJSP

We will deal with only the first two topics in this session

سنتعرض للموضوعين الأولين فقط في هذه الجلسة

Advanced Topics of Servlet/JSP

مواضيع متقدمة ل Servlet/JSP

- MVC (Model-View-Controller) Pattern نمط MVC
- Model 2 Framework هيكل النموذج2
- Apache Struts
- Java Server Faces (JSF)
- Other frameworks هياكل أخرى

Agenda الرزنامة

- JSP in big picture of Java EE مكانة JSP البارزة في Java EE
- Introduction to JSP مقدمة ل JSP
- Life-cycle of JSP page دورة حياة صفحة JSP
- Steps for developing JSP-based Web application
  خطوات لتطوير تطبيق شبكة مبني على تقنية JSP
- Dynamic contents generation techniques in JSP
  تقنية توليد المحتوى ديناميكيا في JSP
- Invoking Java code using JSP scripting elements
  إستدعاء مصدر جافا باستخدام عناصر لبرنامج نصية ل JSP
- JavaBeans for JSP
- Error handling معالجة الأخطاء

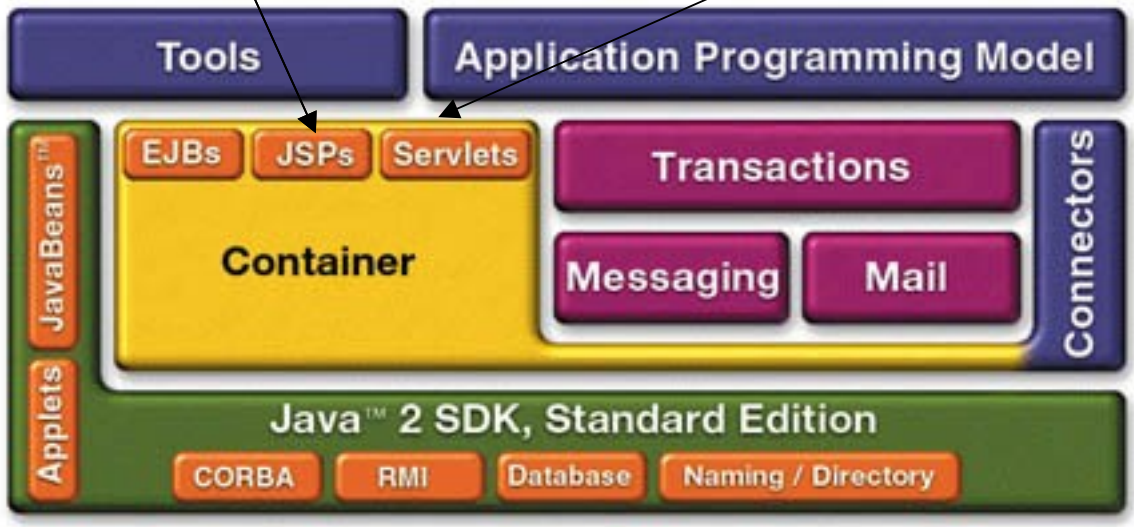
JSP in a Big Picture of Java EE

مكانةJSP البارزة في Java EE

JSP & Servlet in Java EE Architecture

An extensible Web technology that uses template data, custom elements, scripting languages, and server-side Java objects to **return dynamic content to a client**. Typically the template data is HTML or XML elements. The client is often a **Web browser**.

Java Servlet A Java program that extends the functionality of a Web server, generating dynamic content and interacting with Web clients using a **request-response paradigm**.

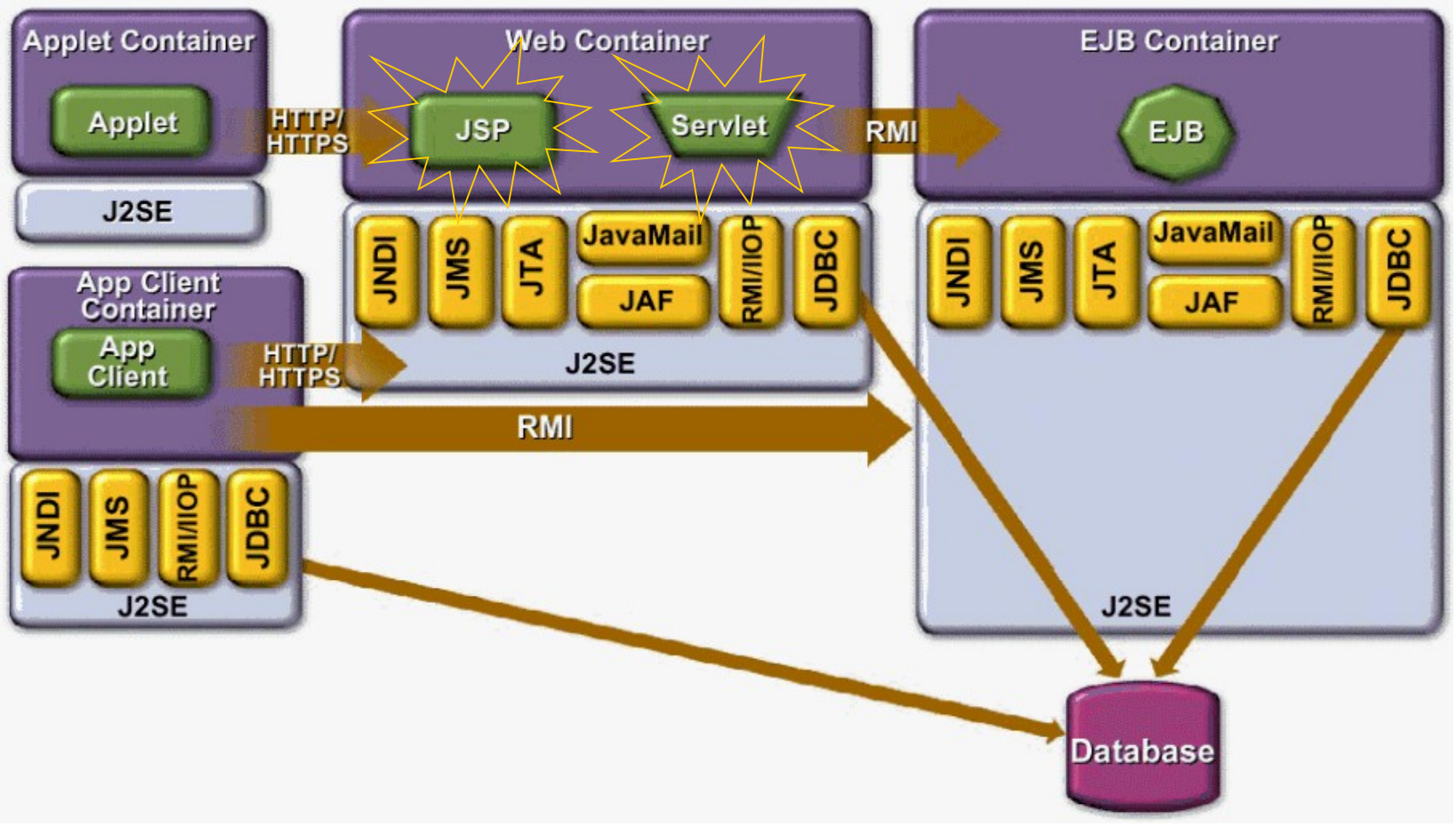


What do you mean by Static & Dynamic Contents?

ماذا نعني بمحتوات ثابت و محتوات ديناميكي؟

- Static contents محتويات ثابتة
  - Typically static HTML page صفحة HTML ثابتة نوعيا
  - Same display for everyone نفس العرض لكل أحد
- Dynamic contents محتويات ديناميكية
  - Contents is dynamically generated based on conditions
    *المحتوى يتولد ديناميكيا متأثرا بشروط معينة
  - Conditions could be الشروط يمكن أن تكون
    - User identity هوية المستخدم
    - Time of the day وقت اليوم
    - User entered values through forms and selections
      مدخلات المستخدم من خلال مجالات الإدخال و إختياراته
  - Examples أمثلة
    - Etrade webpage customized just for you, my Yahoo

JSP & Servlet as Web Components



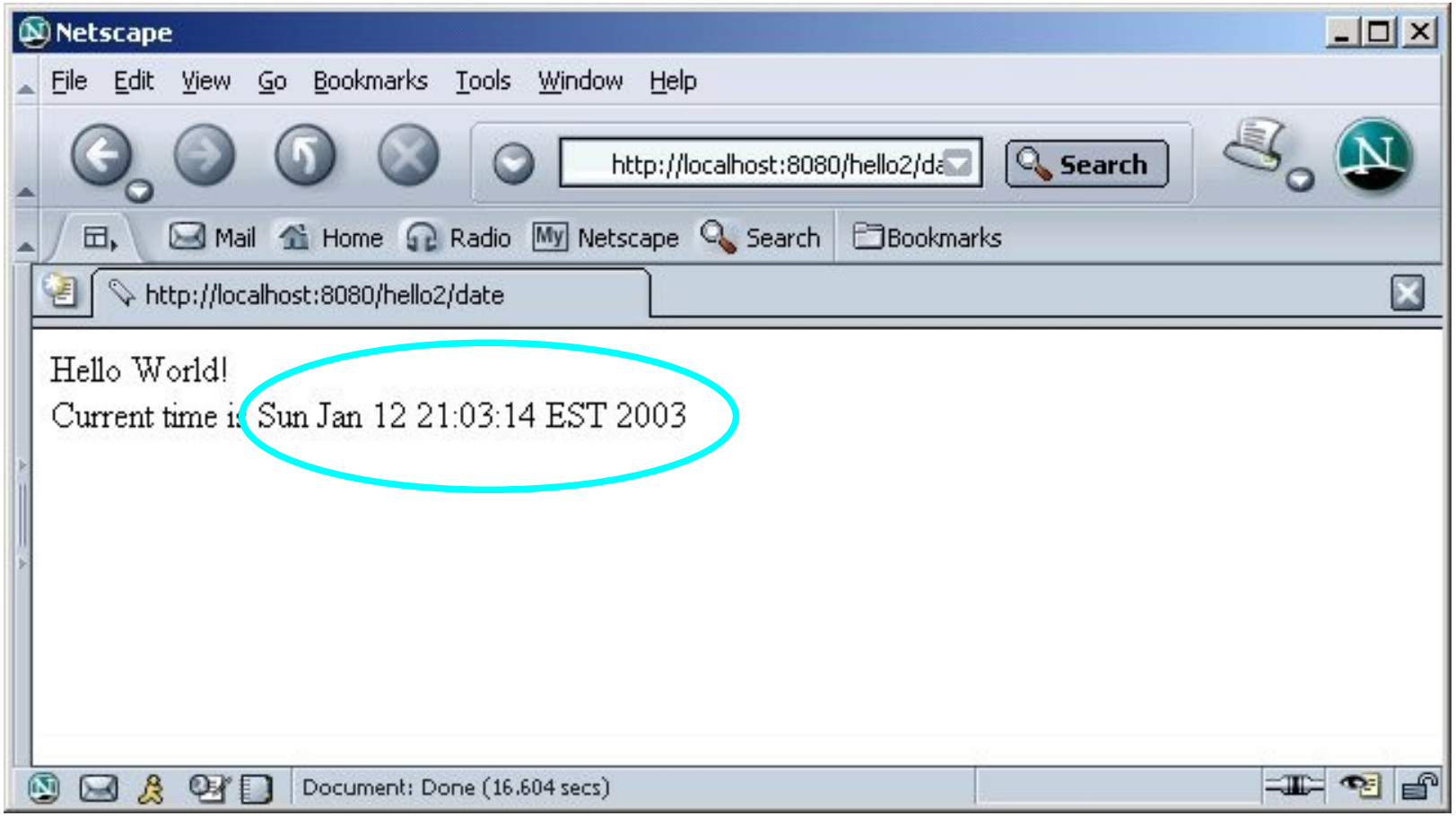
What is JSP Page? ما هي صفحة JSP؟

- A text-based document capable of returning both static and dynamic content to a client browser
  وثيقة نصية قادرة على إرجاع محتوى ثابت و متحرك إلى متصفح الحريف
- Static content and dynamic content can be intermixed
  يمكن مزج المحتوى الثابت مع المحتوى الديناميكي
- Static content محتوى ثابت
  - HTML, XML, Text
- Dynamic content محتوى ديناميكي
  - Java code مصدر جافا
  - Displaying properties of JavaBeans عرض خصائص JavaBeans
  - Invoking business logic defined in Custom tags
    إستدعاء business logic المعرفة في الأوسمة الخاصة

A Simple JSP Page
(Blue: static, Red: Dynamic contents)

```
<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)

Servlets

- HTML code in Java
- Not easy to author

JSP

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

JSP Benefits مميزات JSP

- Content and display logic are separated
- Simplify web application development with JSP, JavaBeans and custom tags
  سهولة تطوير تطبيق الشبكة مع JSP وJavaBeansو الأوسمة الخاصة
- Supports software reuse through the use of components (JavaBeans, Custom tags) دعم إعادة إستخدام البرامج من خلال إستخدام المكونات
- Automatic deployment نشر تلقائي
  - Recompile automatically when changes are made to JSP pages
    إعادة تجميع تلقائي عند حدوث تخييرات في صفحات JSP
- Easier to author web pages أسهل في تصميم صفحات الواب
- Platform-independent إستقلالية المنصة

Why JSP over Servlet? لماذا JSP فوق Servlet؟

- Servlets can do a lot of things, but it is pain to:

  Servlets بإمكانها القيام بالكثير من الأشياء و لكن يشق عليها

  – Use those println() statements to generate HTML page

    إستعمال الدالة ()println لتوليد HTML

  – Maintain that HTML page صيانة صفحات HTML

- No need for compiling, packaging, CLASSPATH setting, لا حاجة إلى التجميع و التحزيم و تعيينCLASSPATH

Do I have to Use JSP over Servlet or vice-versa?

هل أستطيع إستعمال JSP فوق Servlet أو بالعكس؟

- No, you want to use both leveraging the strengths of each technology
  - Servlet's strength is “controlling and dispatching”

    قوة Servlet في التحكم و الإيفاد
  - JSP's strength is “displaying”

    قوة JSP في العرض
- In a typically production environment, both servlet and JSP are used in a so-called MVC (Model-View-Controller) pattern

  في بيئة إنتاج نوعية*, كل من Servlet و JSPيتم إستعمالهما معا فيما يسمى بنمطMVC
  - Servlet handles controller part

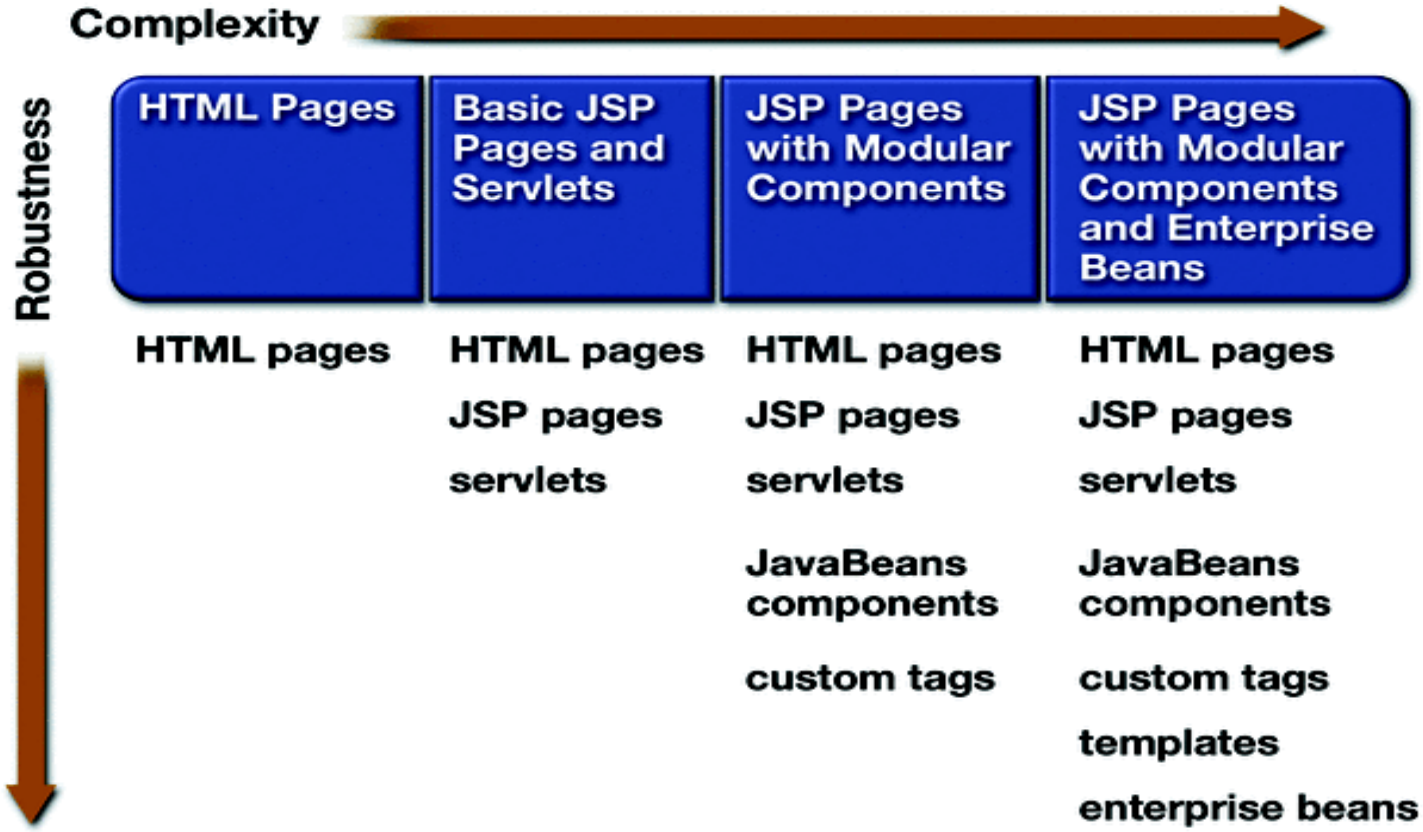
    تعالج جزء التحكم Servlet
  - JSP handles view part

    JSP تعالج جزء العرض

JSP Architecture

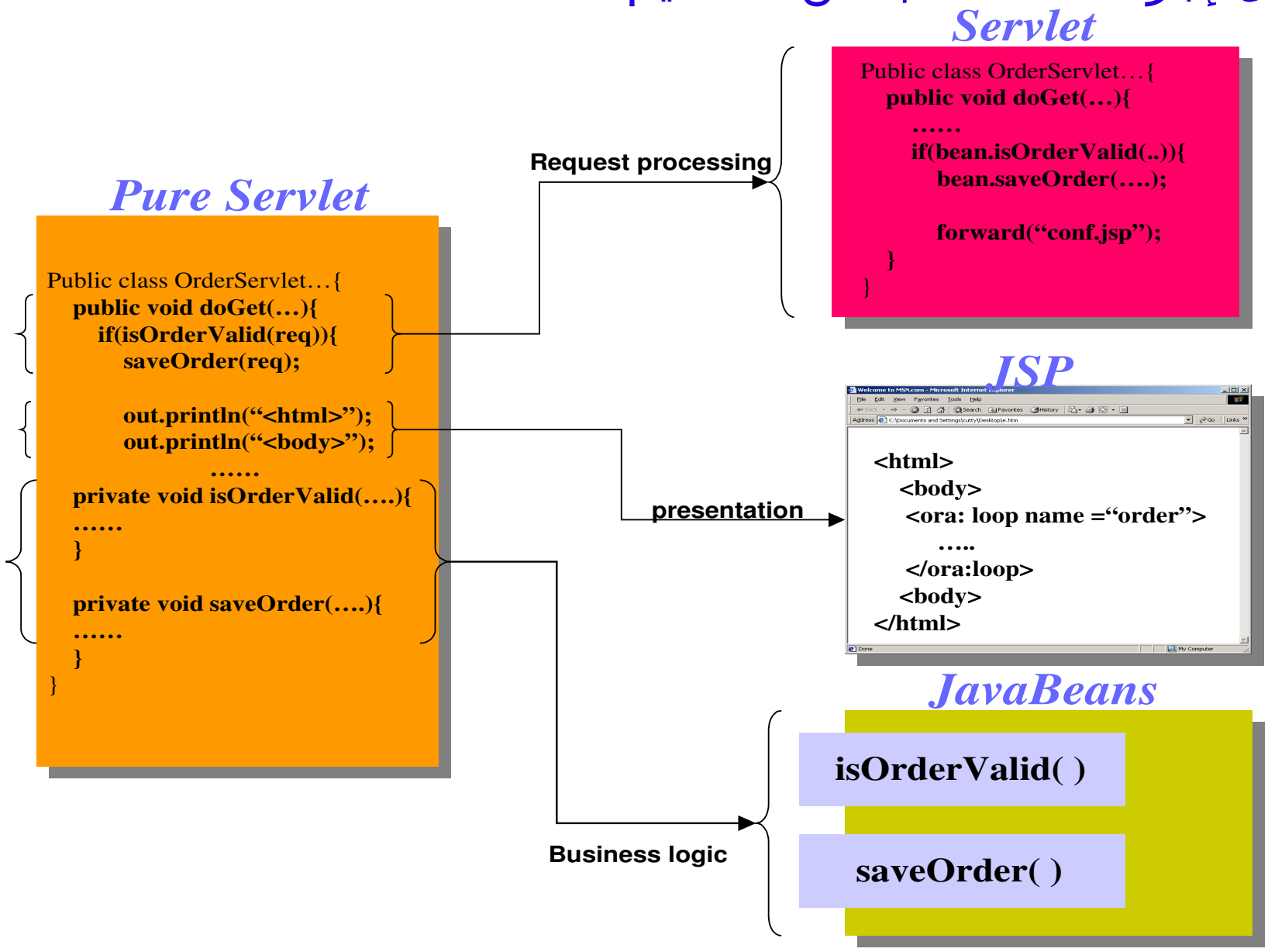
بنية JSP

Web Application Designs تصاميم تطبيق الشبكة



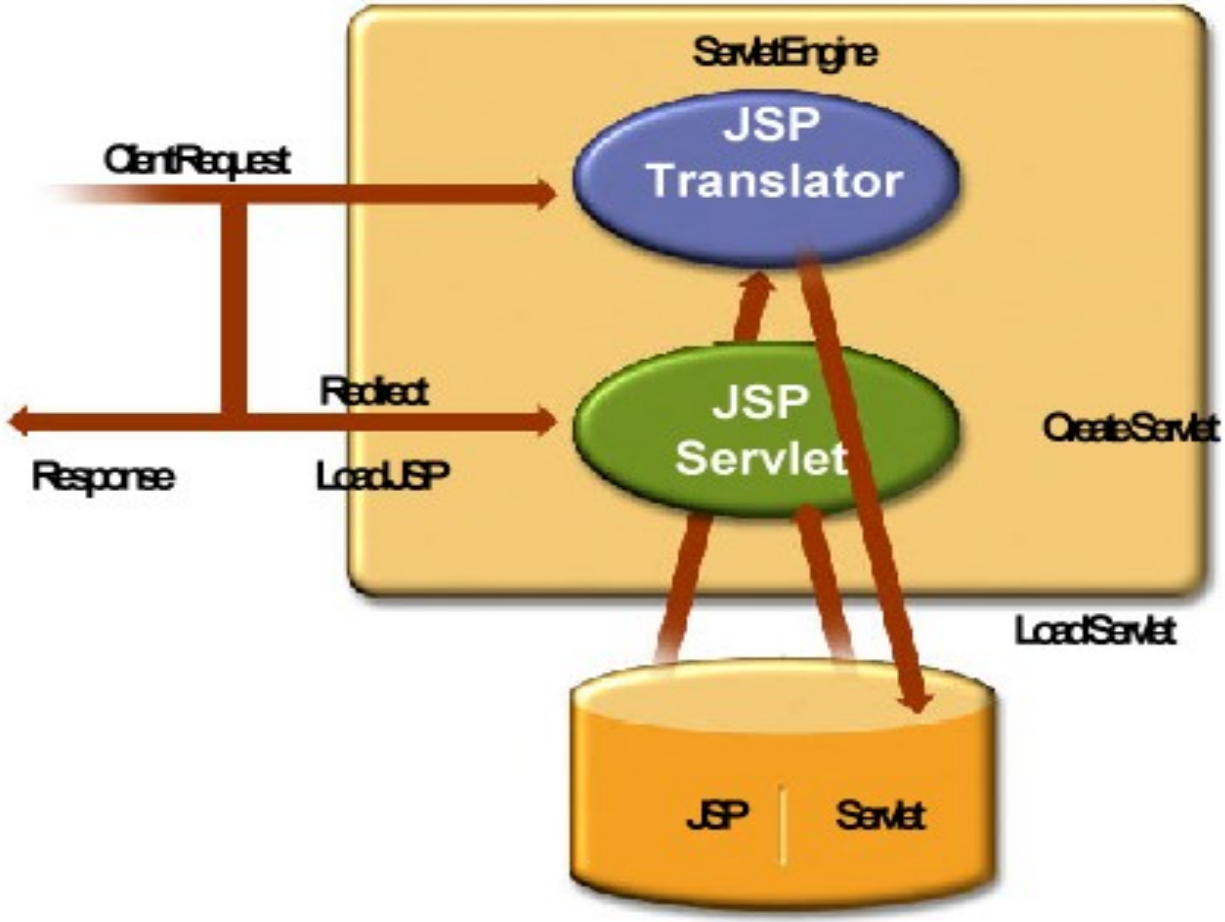
Separate Request processing From Presentation

فصل إجراء ات الطلب عن التقديم



JSP Architecture

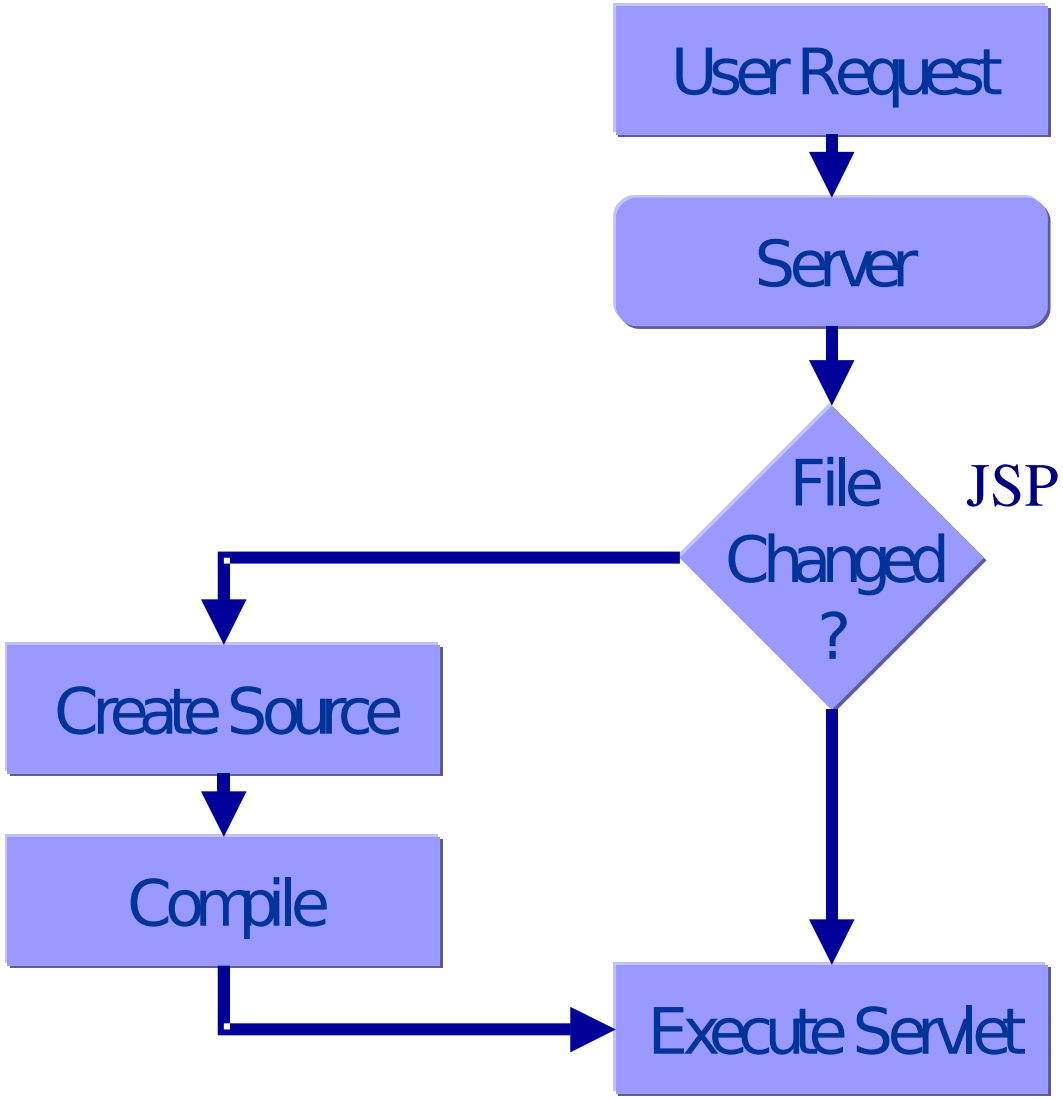
بنية JSP



Life-Cycle of a JSP Page

دورة حياة صفحة JSP

How Does JSP Work? كيف يعمل JSP؟



JSP Page Lifecycle Phases

مراحل دورة حياة صفحة JSP

- Translation phase طور الترجمة
- Compile phase طور التجميع
- Execution phase طور التنفيذ

Translation/Compilation Phase

طور الترجمة و التجميع

- JSP files get translated into servlet source code, which is then compiled

ملفات JSP يتم ترجمتها إلى servlet و الذي بدوره يتم تجميعه

- Done by the container automatically تقوم بذلك الحاوية تلقائيا
- The first time JSP page is accessed after it is deployed (or modified and redeployed)

في أول الوقت كان يتم النفاذ إلى صفحة JSPبعد نشرها *

- For JSP page “pageName”, the source code resides
  - <AppServer_HOME>/work/Standard Engine/localhost/context_root/pageName$jsp.java
  - <AppServer_HOME>/work/Standard Engine/localhost/date/index$jsp.java

Translation/Compilation Phase

طور الترجمة و التجميع

- Static Template data is transformed into code that will emit data into the stream
- JSP elements are treated differently

عناصر JSP يتم معالجتها بطريقة مختلفة

  - Directives are used to control how Web container translates and executes JSP page

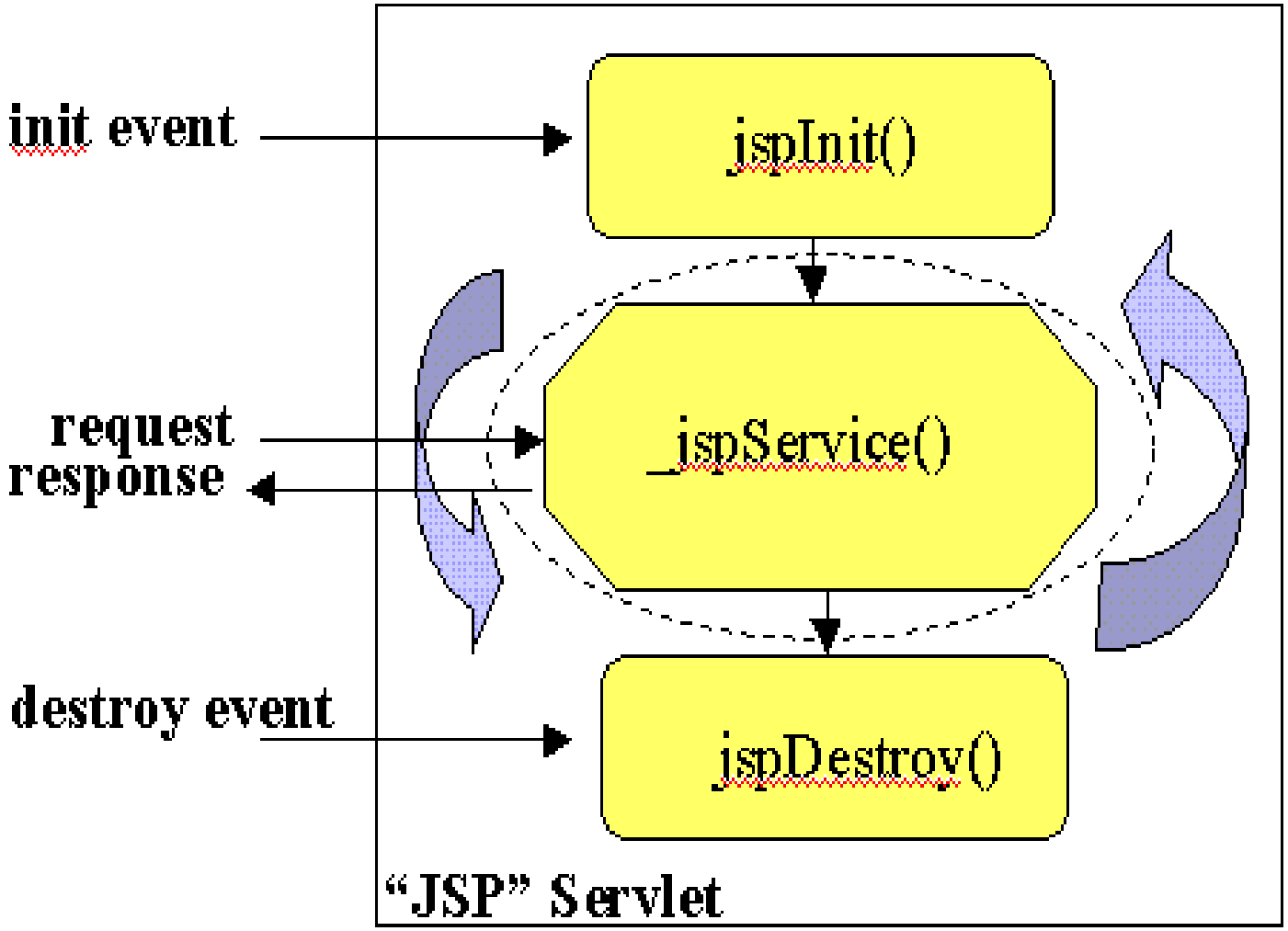
    الأوامر التنفيذية تستخدم للتحكم في كيفية ترجمة حاوية الشبكة و تنفيذها لصفحات JSP
  - Scripting elements are inserted into JSP page's servlet class

    عناصر البرامج النصية يتم إدراجها في صفحات JSPمن خلال أقسام servlet
  - Elements of the form <jsp:xxx .../> are converted into method calls to JavaBeans components

    العناصر التي على شكل<jsp:xxx ... />يتم تحويلها إلى طريقة تقوم بنداء مكونات JavaBeans

JSP Lifecycle Methods during Execution Phase

طرق دورة حياة JSP أثناء مرحلة تنفيذ



Initialization of a JSP Page تمهيد JSP صفحة

- Declare methods for performing the following tasks using JSP declaration mechanism

  إعلان طرق لإنجازالمهمات التالية باستخدام ألية الإعلان لدى JSP

  - Read persistent configuration data قراءة إعدادات البيانات المستمرة
  - Initialize resources تمهيد الموارد
  - Perform any other one-time activities by overriding jspInit() method of JspPage interface

Finalization of a JSP Page

- Declare methods for performing the following tasks using JSP declaration mechanism

  إعلان طرق لإنجازالمهمات التالية باستخدام ألية الإعلان لدى JSP

  - Read persistent configuration data قراءة إعدادات البيانات المستمرة
  - Release resources تمهيد الموارد
  - Perform any other one-time cleanup activities by overriding jspDestroy() method of JspPage interface

Example: initdestroy.jsp

```
<%@ page import="database.*" %>
<%@ page errorPage="errorpage.jsp" %>
<%--  Declare initialization and finalization methods using JSP
   declaration --%>
<%!

  private BookDBAO bookDBAO;
  public void jspInit() {

    // retrieve database access object, which was set once per
    web application
    bookDBAO =
      (BookDBAO)getServletContext().getAttribute("bookDB");
    if (bookDBAO == null)
        System.out.println("Couldn't get database.");
  }

  public void jspDestroy() {
    bookDBAO = null;
  }
%>
```

Steps for Developing JSP-based Web Application

خطوات تطوير تطبيق شبكة مبنية على JSP

Web Application Development and Deployment Steps خطوات تطوير تطبيق شبكة مبنية على JSP

1. Write (and compile) the Web component code (Servlet or JSP) and helper classes referenced by the web component code
   كتابة(و تجميع) مصدر مكون الشبكة(JSP أو Servlet) و أقسام المساعد المفهرسة من قبل الشيفرة المصدرية لمكون الشبكة
2. Create any static resources (for example, images or HTML pages)
   إنشاء أي موارد ثابتة(مثل الصور و صفحات HTML)
3. Create deployment descriptor (web.xml) إنشاء واصف النشر
4. Build the Web application (*.war file or deployment-ready directory)
   بناء تطبيق الشبكة(ملف war أو مجلد جاهز للنشر)
5. Install or deploy the web application into a Web container
   تثبيت أو نشر تطبيق الشبكة عن طريق حاوية الشبكة
   - Clients (Browsers) are now ready to access them via URL
     الحرفاء(المتصفحون) هم الأن جاهزون للنفاذ من خلال المسار

1. Write and compile the Web component code

كتابة و تجميع مصدر مكون الشبكة

- Create development tree structure إنشاء بنية شجرة التطوير
- Write either servlet code and/or JSP pages along with related helper code

  كتابة إما مصدر servletأو/و صفحات JSPإلى جانب مصدر المساعد المرتبط
- Create build.xml for Ant-based build (and other application life-cycle management) process

  إنشاء build.xml للبناء إعتمادا على Ant(و إدارة دورة حياة التطبيق)

Development Tree Structure

بنية شجرة التطوير

- Keep Web application source separate from compiled files
  الحفاظ على فصل المصدر البرمجي لتطبيق الشبكة عن ملفات التجميع
  - facilitate iterative development تسهيل التطوير التكراري
- Root directory الدليل الجذر
  - build.xml: Ant build file ملف بناء Ant
  - context.xml: Optional application configuration file
    ملف خياري لتشكيلات التطبيق
  - src: Java source of servlets and JavaBeans components
    المصدر البرمجي للservlets و مكونات JavaBeans
  - web: JSP pages and HTML pages, images
    صفحات JSP وصفحات HTML و صور

Example: Hello1 Example Tree Structure (before “ant build” command)

- hello1 directory (from Java EE 1.4 tutorial)
  - web directory
    - duke.waving.gif
    - index.jsp
    - response.jsp
    - WEB-INF directory
      - web.xml
  - build.xml
  - (it does not have src directory since this does not use any Java classes as utility classes)

إنه لا يحتوي مجلد srcلأنه لا يستخدم أي أقسام جافا كأقسام وسائل

2. Create any static resources إنشاء أي مورد ثابت

- HTML pages صفحات HTML
  - Custom pages
  - Login pages
  - Error pages
- Image files that are used by HTML pages or JSP pages ملفات الصور التي تستخدم من قبل صفحات HTMLو صفحات JSP
  - Example: duke.waving.gif

3. Create deployment descriptor (web.xml)

إنشاء واصف النشر

- Deployment descriptor contains deployment time runtime instructions to the Web container

  واصف النشر يحتوي على وقت نشر تعليمات زمن التشغيل* لحاوية الشبكة

  - URN that the client uses to access the web component

    URN التي يستخدمها الحريف للنفاذ إلى مكون الشبكة

- Every web application has to have it

  كل تطبيقات الشبكة في حاجة إليها

web.xml for Hello1

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app PUBLIC '-//Sun Microsystems, Inc.//DTD Web
  Application 2.3//EN' 'http://java.sun.com/dtd/web-app_2_3.dtd'>

<web-app>
  <display-name>Hello2</display-name>
  <description>no description</description>
  <servlet>
    <servlet-name>greeting</servlet-name>
    <display-name>greeting</display-name>
    <description>no description</description>
    <jsp-file>/greeting.jsp</jsp-file>   <!-- what gets called -->
  </servlet>
  <servlet-mapping>
    <servlet-name>greeting</servlet-name>
    <url-pattern>/greeting</url-pattern>  <!-- URL from browser --
  >
  </servlet-mapping>
</web-app>
```

4. Build the Web application

بناء تطبيق الشبكة

- Either *.WAR file or unpacked form of *.WAR file

  إما عن طريق ملف WAR أو ملف WARالغير محزم

- Build process is made of عملية البناء تعتمد على
  - create build directory (if it is not present) and its subdirectories

    إنشاء مجلد build(إذا كان غير موجود) و جميع مجلداته الفرعية
  - copy *.jsp files under build directory نسخ ملفات *.jsp تحت مجلد build
  - compile Java code into build/WEB-INF/classes directory

    تجميع مصدر جافا في مجلد build/WEB-INF/classes
  - copy web.xml file into build/WEB-INF directory

    نسخ ملف web.xml في مجلد build/WEB-INF
  - copy image files into build directory نسخ ملفات الصور في مجلد build

Example: Hello2 Tree Structure (after "ant build" command)

- Hello2
  - web directory
  - build.xml
  - build directory
    - WEB-INF directory
      - classes directory
      - web.xml
    - duke.waving.gif
    - greeting.jsp
    - response.jsp

5. Install or Deploy Web application

تثبيت أو نشر تطبيق الشبكة

- There are 2 different ways to install/deploy Web application هناك طريقتان مختلفتان لتثبيت و نشر تطبيق الشبكة
  - By asking Tomcat Manager via sending a command to it (Tomcat Manager is just another Servlet app that is always running)
    عن طريق سؤال منظم Tomcat من خلال إرسال أوامر له
    (منظم Tomcat ليس إلا تطبيق Servlet يكون دائما في حالة تشغيل
    - ant install ant مثبت
    - ant deploy ant ناشر
  - By manually copying files to Tomcat's webapps directory and then restarting Tomcat
    عن طريق النسخ اليدوي للملفات في مجلد webappsالخاص ب Tomcat ومن
    ثمة إعادة تشغيل Tomcat

5.1 Install or Deploy Web application

تثبيت أو نشر تطبيق الشبكة

- Via Tomcat manager عن طريق منظم Tomcat
  - You need proper credential to perform this

    *أنت في حاجة إلى إعتماد سليم لإنجاز ذلك
    - This is why you need build.properties file with proper userid and password, otherwise you will experience HTTP 401 error

      هذا سبب حاجتك إلى ملف build.properties مع معرف و كلمة مرور صحيحة

      بطريقة أخرى يجب أن تتمرس مع أخطاء HTTP 401
  - ant install for temporary deployment مثبت ant للنشر الوقتي
  - ant deploy for permanent deployment ناشر ant للنشر الدائم
- Manual Method طريقة يدوية
  - Copying *.war file or unpacked directory to <tomcat-install>/webapps/ directory manually and then restart Tomcat

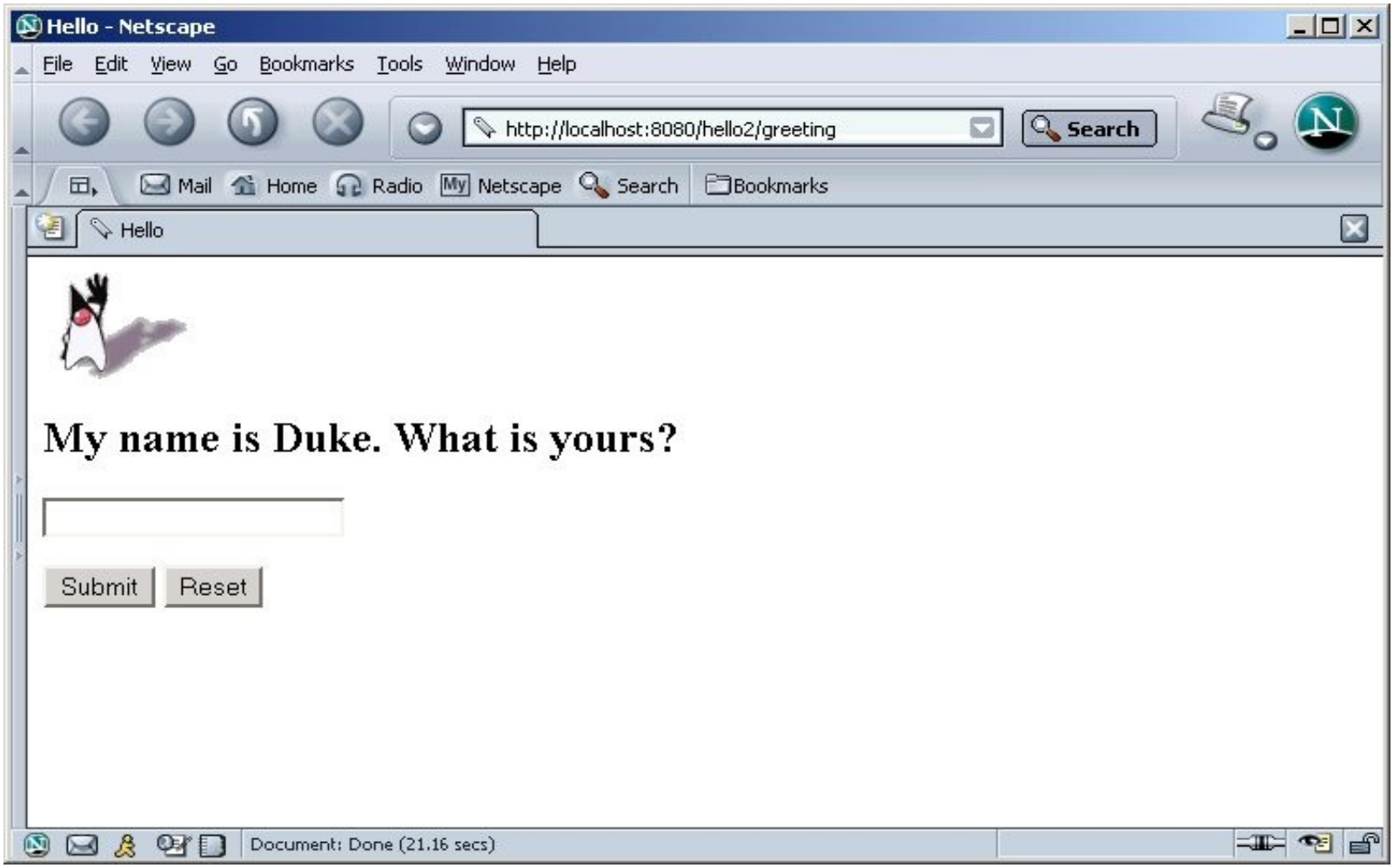
    نسخ ملف war أو المجلد الغير محزم إلى مجلد <tomcat-install>/webapps/

    يدويا و من ثمة إعادة تشغيل Tomcat

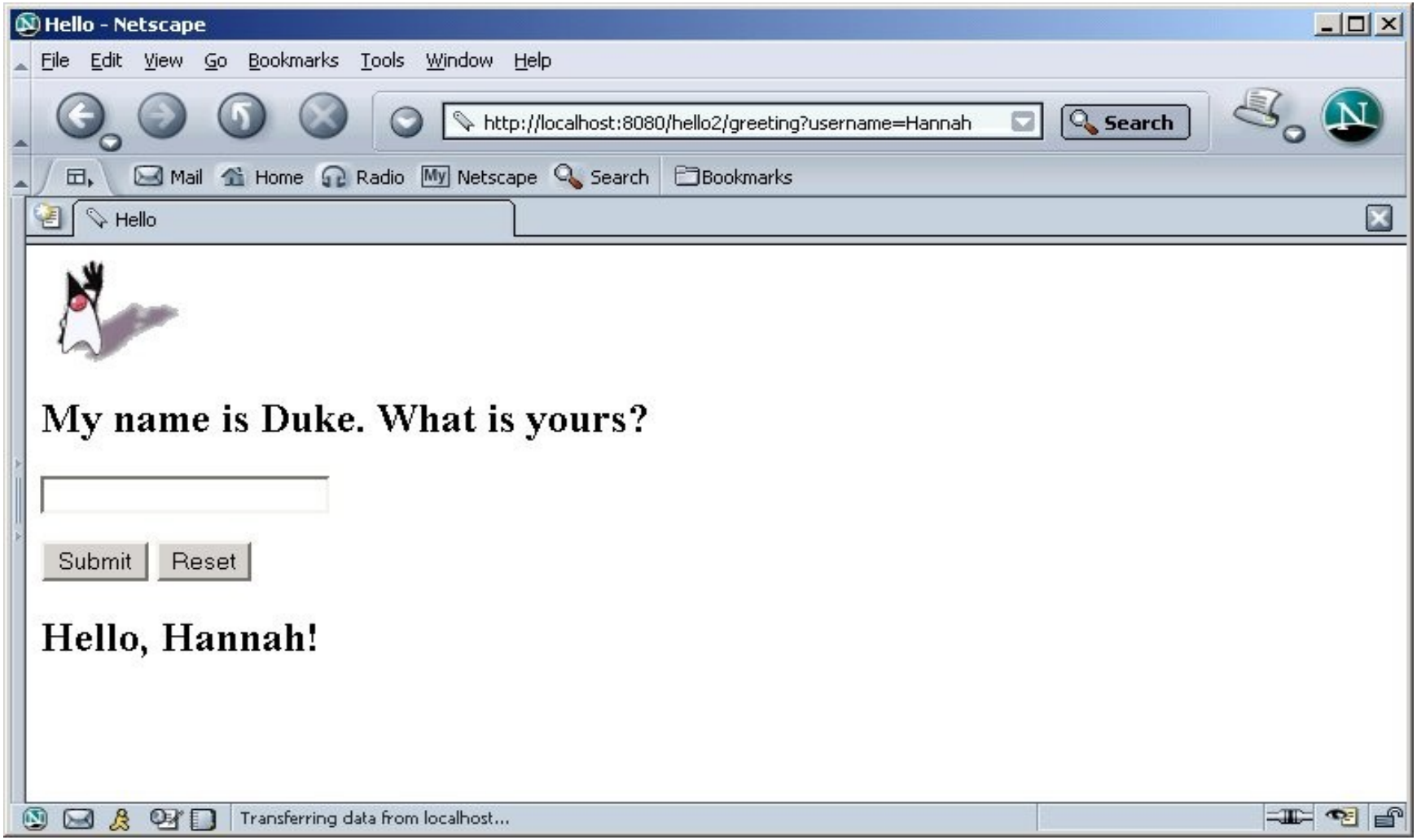
6. Perform Client Access to Web Application

- From a browser, go to URN of the Web application من المتصفح إذهب إلى URN تطبيق الشبكة التالي
  - http://localhost:8080/hello2/greeting

http://localhost:8080/hello2/greeting



response.jsp



Comparing Hello1 Servlet & Hello2 JSP code

مقارنة Hello1 Servlet

و

مصدر Hello2 JSP

GreetingServlet.java (1)

```
import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * This is a simple example of an HTTP Servlet.  It responds to the GET
 * method of the HTTP protocol.
 */
public class GreetingServlet extends HttpServlet {

    public void doGet (HttpServletRequest request,
                       HttpServletResponse response)
                       throws ServletException, IOException
 {

      response.setContentType("text/html");
      response.setBufferSize(8192);
      PrintWriter out = response.getWriter();

      // then write the data of the response
      out.println("<html>" +
                  "<head><title>Hello</title></head>");
```

GreetingServlet.java (2)

```
    // then write the data of the response
    out.println("<body  bgcolor=\"#ffffff\">" +
       "<img src=\"duke.waving.gif\">" +
       "<h2>Hello, my name is Duke. What's yours?</h2>" +
       "<form method=\"get\">" +
       "<input type=\"text\" name=\"username\" size=\"25\">" +
       "<p></p>" +
       "<input type=\"submit\" value=\"Submit\">" +
       "<input type=\"reset\" value=\"Reset\">" +
       "</form>");

    String username = request.getParameter("username");

    // dispatch to another web resource
    if ( username != null && username.length() > 0 ) {
          RequestDispatcher dispatcher =
             getServletContext().getRequestDispatcher("/response");

          if (dispatcher != null)
          dispatcher.include(request, response);

    }
    out.println("</body></html>");
    out.close();
}
```

GreetingServlet.java (3)

```
    public String getServletInfo() {
        return "The Hello servlet says hello.";

    }
}
```

greeting.jsp

```
<html>
<head><title>Hello</title></head>
<body bgcolor="white">
<img src="duke.waving.gif">
<h2>My name is Duke. What is yours?</h2>

<form method="get">
<input type="text" name="username" size="25">
<p></p>
<input type="submit" value="Submit">
<input type="reset" value="Reset">
</form>

<%
    String username = request.getParameter("username");
    if ( username != null && username.length() > 0 ) {
%>
    <%@include file="response.jsp" %>
<%
    }
%>
</body>
</html>
```

ResponseServlet.java

```
import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

// This is a simple example of an HTTP Servlet.  It responds to the GET
// method of the HTTP protocol.
public class ResponseServlet extends HttpServlet {

    public void doGet (HttpServletRequest request,
                       HttpServletResponse response)
                       throws ServletException, IOException{
        PrintWriter out = response.getWriter();

        // then write the data of the response
        String username = request.getParameter("username");
        if ( username != null && username.length() > 0 )
          out.println("<h2>Hello, " + username + "!</h2>");
    }

    public String getServletInfo() {
        return "The Response servlet says hello.";
    }
}
```

response.jsp

```
<h2><font color="black">Hello, <%=username%>!</font></h2>
```

JSP “is” Servlet!

JSP هي Servlet!

JSP is “Servlet” JSP هي Servlet

- JSP pages get translated into servlet

  صفحات JSP يتم ترجمتها إلى servlet

  - Tomcat translates greeting.jsp into greeting$jsp.java

    Tomcat تترجم greeting.jsp إلى greeting$jsp.java

- Scriptlet (Java code) within JSP page ends up being inserted into jspService() method of resulting servlet
- Implicit objects for servlet are also available to JSP page designers, JavaBeans developers, custom tag designers

  الكائنات الضمنية ل servletهي أيضا متوفرة لمصممي صفحة JSPو مطوري JavaBeans و مصممي الأوسمة الخاصة

greeting$jsp.java (1)

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import org.apache.jasper.runtime.*;


public class greeting$jsp extends HttpJspBase {


  static {
  }
  public greeting$jsp( ) {
  }

  private static boolean _jspx_inited = false;

  public final void _jspx_init() throws
   org.apache.jasper.runtime.JspException {
  }
```

greeting$jsp.java (2)

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse  response)
              throws java.io.IOException, ServletException {

  JspFactory _jspxFactory = null;
  PageContext pageContext = null;
  HttpSession session = null;
  ServletContext application = null;
  ServletConfig config = null;
  JspWriter out = null;
  Object page = this;
  String  _value = null;
```

greeting$jsp.java (3)

```
try {

  if (_jspx_inited == false) {
    synchronized (this) {
      if (_jspx_inited == false) {
        _jspx_init();
        _jspx_inited = true;
      }
    }
  }
  _jspxFactory = JspFactory.getDefaultFactory();
  response.setContentType("text/html;charset=ISO-8859-1");
  pageContext = _jspxFactory.getPageContext(this, request,
                  response,"", true, 8192, true);

  application = pageContext.getServletContext();
  config = pageContext.getServletConfig();
  session = pageContext.getSession();
  out = pageContext.getOut();
```

greeting$jsp.java (4)

```
// HTML // begin [file="/greeting.jsp";from=(38,4);to=(53,0)]

  out.write("\n\n<html>\n<head><title>Hello</title></head>\n<body
bgcolor=\"white\">\n<img src=\"duke.waving.gif\"> \n<h2>My name is Duke.
What is yours?</h2>\n\n<form method=\"get\">\n<input type=\"text\"
name=\"username\" size=\"25\">\n<p></p>\n<input type=\"submit\"
value=\"Submit\">\n<input type=\"reset\"
value=\"Reset\">\n</form>\n\n");


// end
// begin [file="/greeting.jsp";from=(53,2);to=(56,0)]

    String username = request.getParameter("username");
     if ( username != null && username.length() > 0 ) {
// end
// HTML // begin [file="/greeting.jsp";from=(56,2);to=(57,4)]
  out.write("\n    ");

// end
// HTML // begin [file="/response.jsp";from=(38,4);to=(40,31)]
  out.write("\n\n<h2><font color=\"black\">Hello, ");

// end
// begin [file="/response.jsp";from=(40,34);to=(40,42)]
  out.print(username);
// end
```

greeting$jsp.java (5)

```
      // HTML // begin [file="/response.jsp";from=(40,44);to=(55,0)]
        out.write("!</font></h2>\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");

      // end
      // HTML // begin [file="/greeting.jsp";from=(57,37);to=(58,0)]
        out.write("\n");

      // end
      // begin [file="/greeting.jsp";from=(58,2);to=(60,0)]

            }
      // end
      // HTML // begin [file="/greeting.jsp";from=(60,2);to=(63,0)]
        out.write("\n</body>\n</html>\n");

      // end
    } catch (Throwable t) {
      if (out != null && out.getBufferSize() != 0) out.clearBuffer();
      if (pageContext != null) pageContext.handlePageException(t);
    } finally {
      if (_jspxFactory != null)
        _jspxFactory.releasePageContext(pageContext);
    }
  }
}
```

Dynamic Content Generation Techniques in JSP

تقنية توليد المحتوى ديناميكيا في JSP

Dynamic Contents Generation Techniques with JSP Technology تقنية توليد المحتوى ديناميكيا مع تقنيةJSP

- Various techniques can be chosen depending on the following factors

  هناك العديد من التقنيات التي يمكن إختيارها بغض النظر عن العوامل التالية

  - size and complexity of the project حجم و درجة تعقيد المشروع
  - Requirements on re usability of code, maintainability, degree of robustness متطلبات في قابلية إعادة إستعمال المصدر البرمجي وقابلية الصيانة و درجة القوة*

- Simple to incrementally complex techniques are available

  إمكانية الإنتقال تدريجيا من التقنيات البسيطة إلى المعقدة*

Dynamic Contents Generation Techniques with JSP

تقنية توليد المحتوى ديناميكيا معJSP

a) Call Java code directly within JSP

نداء مباشر لمصدر جافا من خلال JSP

b) Call Java code indirectly within JSP

نداء غير مباشرلمصدر جافا من خلال JSP

c) Use JavaBeans within JSP إستخدم JavaBeans من خلال JSP

d) Develop and use your own custom tags

طور و استخدم أوسمتك الشخصية

e) Leverage 3rd-party custom tags or JSTL (JSP Standard Tag Library)

f) Follow MVC design pattern إتبع نمط تصميم MVC

g) Leverage proven Model2 frameworks

(a) Call Java code directly

- Place all Java code in JSP page

  ضع جميع مصادر جافا المصدرية في صفحة JSP

- Suitable only for a very simple Web application
  - hard to maintain مشقة في الصيانة
  - hard to reuse code مشقة في إعادة إستعمال المصدر البرمجي
  - hard to understand for web page authors
- Not recommended for relatively sophisticated Web applications
  - weak separation between contents and presentation

(b) Call Java code indirectly نداء غي

- Develop separate utility classes تطوير أقسام نافعة منقسمة
- Insert into JSP page only the Java code needed to invoke the utility classes

  إدراج في صفحة JSP مصدر جافا البرمجي الذي نحتاج إليه فقط لاستدعاء الأقسام النافعة
- Better separation of contents generation from presentation logic than the previous method

  * فصل أحسن من الطريقة السابقة بين المحتويات المتولدة من presentation logic
- Better re usability and maintainability than the previous method

  قابلية أكبر من الطريقة السابقة لإعادة الإستعمال و الصيانة
- Still weak separation between contents and presentation, however

  و مع ذلك الفصل بين المحتويات و presentation يبقى ضعيفا

(c) Use JavaBeans إستخدام JavaBeans

- Develop utility classes in the form of JavaBeans
  تطوير أقسام نافعة على شكل JavaBeans
- Leverage built-in JSP facility of creating JavaBeans instances, getting and setting JavaBeans properties
  - Use JSP element syntax إستخدم صيغة عنصر JSP
- Easier to use for web page authors
  سهل الإستخدام لمؤلفي صفحات الواب
- Better re usability and maintainability than the previous method قابلية أفضل لإعادة الإستخدام و الصيانة من الطريقة السابقة

(d) Develop and Use Custom Tags

تطوير و استخدام الأوسمة الخاصة

- Develop sophisticated components called custom tags

  تطوير مكونات متطورة تسمى أوسمة خاصة

  - Custom tags are specifically designed for JSP

    الأوسمة الخاصة مصممة خصيصا* ل JSP

- More powerful than JavaBeans component

  أكثر قوة من مكون JavaBeans

  - More than just getter and setter methods

    أكثر من أن تكون عبارة عن طرق الحصول على القيمة و ضبطها

- re usability, maintainability, robustness

  القابلية لإعادة الإستخدام و الصيانة مع القوة

- Development of custom tags are more difficult than creating JavaBeans, however

  و مع ذلك فإن تطوير الأوسمة الخاصة أكثر صعوبة من إنشاء JavaBeans

(e) Use 3rd-party Custom tags or JSTL

- There are many open source and commercial custom tags available هناك الكثير من الأوسمة الخاصة المفتوحة المصدر و التجارية
  - Apache Struts
- JSTL (JSP Standard Tag Library) standardize the set of custom tags that should be available over Java EE platform at a minimum

  JSTL تقوم بجعل مجموعة الأوسمة الخاصة معيارية و أن تكون متوفرة على منصة Java EE على الأقل

  - As a developer or deployer, you can be assured that a standard set of custom tags are already present in Java EE compliant platform (J2EE 1.3 and after) كمطور أو ناشر يمكن أن يضمن لك وجود مجموعة من الأوسمة الخاصة في طبقة منصة Java EE(J2EE 1.3 ما بعدها

(f) Design/Use MVC Design Pattern

إستخدام تصميم أسلوب تصميم **MVC**

- Follow MVC design pattern تتبع أسلوب تصميم MVC
  - Model using some model technologies
    نموذج يقوم باستخدام بعض النماذج التقنية
  - View using JSP طبقة العرض باستخدام JSP
  - Controller using Servlet طبقة التحكم باستخدام Servlet
- Creating and maintaining your own MVC framework is highly discourage, however

  و مع ذلك فأن إنشاء و صيانة هيكل MVC* الخاص بك مثبط للعزيمة

(g) Use Proven MVC Model2 Frameworks

- There are many to choose from هناك الكثير يمكنك أن تختار منهم
  - Apache Struts
  - JavaServer Faces (JSF)
  - Other frameworks: Echo, Tapestry, WebWorks, Wicket

Invoking Java Code with JSP Scripting Elements

إستدعاء مصدر جافا باستخدام عناصر لبرنامج نصية ل JSP

JSP Scripting Elements عناصر لبرنامج نصية ل JSP

- Lets you insert Java code into the servlet that will be generated from JSP page تمكنك من إدراج مصدر جافا مصدري عن طريق Servlet و التي تتولد بدورها من صفحة JSP
- Minimize the usage of JSP scripting elements in your JSP pages if possible حدد من إستعمالك لعناصر البرمجية النصية قدر الإمكان ل JSP
- There are three forms هناك ثلاثة أشكال
  - Expressions: <%= Expressions %>
  - Scriptlets: <% Code %>
  - Declarations: <%! Declarations %>

Expressions التعابير

- During execution phase أثناء مرحلة التيفيذ
  - Expression is evaluated and converted into a String
    التعابيير يقع تقييمها و تحويلها إلى مقاطع
  - The String is then Inserted into the servlet's output stream directly و من ثمة يتم إدراجها مباشرة في الخرج المتدفق
  - Results in something like out.println(expression)
    ينتج شئ مثل out.println(expression)
  - Can use predefined variables (implicit objects) within expression
    يمكنك إستعمال متغيرات محددة مسبقا (كائنات ضمنية) من خلال التعابير
- Format الصيغة
  - <%= Expression %> or
  - <jsp:expression>Expression</jsp:expression>
  - Semi-colons are not allowed for expressions

Example: Expressions

مثال: التعابير

- Display current time using Date class

  عرض الوقت الحالي باستعمال قسم Date

  - Current time: <%= new java.util.Date() %>
- Display random number using Math class

  عرض رقم عشوائي باستعمال قسم Math

  - Random number: <%= Math.random() %>
- Use implicit objects

  إستخدام الكائنات الضمنية

  - Your hostname: <%= request.getRemoteHost() %>
  - Your parameter: <%= request. getParameter("yourParameter") %>
  - Server: <%= application.getServerInfo() %>
  - Session ID: <%= session.getId() %>

Scriptlets

- Used to insert arbitrary Java code into servlet's jspService() method

  يستخدم لإدراج مصدر جافا برمجي كيفي عن طريق طريقة jspService()

- Can do things expressions alone cannot do

  يمكن فعل أشياء باستخدام التعابير لا يمكن القيام بها لوحدنا

  - setting response headers and status codes

    تعيين إجابة ترويسات و حالة المصادر البرمجية

  - writing to a server log الكتابة إلى سجل الخادم
  - updating database تحديث قاعدة البيانات
  - executing code that contains loops, conditionals

    المصادر البرمجية التنفيذية التي تحتوي على حلقات و شروط

- Can use predefined variables (implicit objects)

  يمكن إستخدام متغيرات محددة مسبقا(كائنات ضمنية)

- Format: الصيغة
  - <% Java code %> or
  - <jsp:scriptlet> Java code</jsp:scriptlet>

Example: Scriptlets

- Display query string عرض مقطع إستعلامي

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

- Settng response type تعيين نوع الإجابة

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

Example: Scriptlet with 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
  }
%>
```

Example: Scriptlet Result

Example: JSP page fragment

- Suppose we have the following JSP page fragment نفترض أنه عندنا المقطع التالي من صفحة JSP
  - <H2> sangHTML </H2>
  - <%= sangExpression() %>
  - <% sangScriptletCode(); %>

Example: Resulting Servlet Code

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
                        throws ServletException, IOException {
    response.setContentType("text/html");
    HttpSession session = request.getSession(true);
    JSPWriter out = response.getWriter();

    // Static HTML fragment is sent to output stream  in "as is" form
    out.println("<H2>sangHTML</H2>");

    // Expression is converted into String and then sent to output
    out.println(sangExpression());

    // Scriptlet is inserted as Java code within _jspService()
    sangScriptletCode();
    ...
}
```

Declarations إعلانات

- Used to define variables or methods that get inserted into the main body of servlet class تستخدم لتعريف المتغيرات أو الطرق التي تم إدراجها في الجسم الرئيسي لقسم servlet
  - Outside of _jspSevice() method
  - Implicit objects are not accessible to declarations الكائنات الضمنية غير متاحة للإعلانات
- Usually used with Expressions or Scriptlets عادة ما تستخدم مع التعابير و Scriptlets
- For initialization and cleanup in JSP pages, use declarations to override jspInit() and jspDestroy() methods لتمهيد و تنظيف صفحات JSP تستخدم الإعلانات لتجاوز طرق jspInit() و jspDestroy()
- Format: الصيغة
  - <%! method or variable declaration code %>
  - <jsp:declaration> method or variable declaration code </jsp:declaration>

Example: JSP Page fragment

مثال: مقطق من صفحة JSP

```
<H1>Some heading</H1>
<%!
  private String randomHeading() {
      return(“<H2>” + Math.random() + “</H2>”);
  }
%>
<%= randomHeading() %>
```

Example: Resulting 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());
    ...
  }
  ...
}
```

Example: Declaration

```
<%!
  private BookDBAO bookDBAO;

  public void jspInit() {
    ...
  }
  public void jspDestroy() {
    ...
  }
%>
```

Why XML Syntax? لماذا صيغة XML ؟

- From JSP 1.2
- Examples
  - <jsp:expression>Expression</jsp:expression>
  - <jsp:scriptlet> Java code</jsp:scriptlet>
  - <jsp:declaration> declaration code </jsp:declaration>
- You can leverage
  - XML validation (via XML schema)
  - Many other XML tools
    - editor
    - transformer
    - Java APIs

Including and Forwarding to Other Web Resource

تضمين و إحالة إلى مورد شبكة أخر

Including Contents in a JSP Page

تضمين المحتويات في صفحة JSP

- Two mechanisms for including another Web resource in a JSP page

  أليتان لتضمين مورد شبكة أخر في صفحة JSP

  - include directive — تضمين أمر توجيهي
  - jsp:include element

Include Directive تضمين أمر توجيهي

- Is processed when the JSP page is translated into a servlet class

  يتم تنفيذه عند ترجمة صفحة JSP إلى قسم servlet

- Effect of the directive is to insert the text contained in another file--either static content or another JSP page--in the including JSP page

  تأثير الأمر التوجيهي هو في إدراج النص الموجود في ملف أخر -إما محتوى ثابت أو صفحة JSPأخرى-- في صفحة JSP المضمنة

- Used to include banner content, copyright information, or any chunk of content that you might want to reuse in another page

  تستخدم في تضمين محتوى اللافتة معلومات حقوق الملكية أو أي محتوى قد ترغب في اعادة استخدامه في صفحة اخرى*

- Syntax and Example
  - <%@ include file="filename" %>
  - <%@ include file="banner.jsp" %>

jsp:include Element

- Is processed when a JSP page is executed

  يتم إنجازه عند تنفيذ صفحة JSP

- Allows you to include either a static or dynamic resource in a JSP file

  يسمح لك بتضمين إما مورد ثابت أو ديناميكي في ملف JSP

  - static: its content is inserted into the calling JSP file

    الثابت: محتوياته أدرجت في ملف JSP الذي تم إنجازه

  - dynamic: the request is sent to the included resource, the included page is executed, and then the result is included in the response from the calling JSP page

    الديناميكي : الطلب أرسل إلى المورد المضمن و الصفحة, المضمنة نفذت و من
    * ثمة ضمنت النتيجة في الإجابة من قبل صفحةJSPالتي تم إستدعاءها

- Syntax and example

  - <jsp:include page="includedPage" />
  - <jsp:include page="date.jsp"/>

Which One to Use it?

- Use include directive if the file changes rarely

  إستخدم أمر توجيهي مضمن إذا كان الملف نادرا ما يتغير

  - It is faster than jsp:include إنه أكثر قوة من jsp:include
- Use jsp:include for content that changes often

  إستخدم jsp:include للمحتويات التي غالبا ما تتغير
- Use jsp:include if which page to include cannot be decided until the main page is requested

  إستخدم jsp:include إذا كانت الصفحة التي ستضمن لا يمكن أن تقرر قبل أن يتم طلب الصفحة الرئيسية*

Forwarding to another Web component

إحالة إلى مكون شبكة أخر

- Same mechanism as in Servlet مثل الألية التي تحدث في Servlet
- Syntax الصيغة
  - <jsp:forward page="/main.jsp" />
- Original request object is provided to the target page via jsp:parameter element

طلب الكائن الأصلي يتم توفيره لصفحة المطلوبة من خلال عنصر jsp:parameter

```
<jsp:forward page="..." >
  <jsp:param name="param1" value="value1"/>
</jsp:forward>
```

Directives

أوامر توجيهية

Directives أوامر توجيهية

- Directives are messages to the JSP container in order to affect overall structure of the servlet

  الأوامر التوجيهية هي رسائل إلى حاوية JSPبغرض التأثير على البنية العامة لل servlet

- Do not produce output into the current output stream

  لا تنتج خرج من خلال الخرج المتدفق الحالي*

- Syntax الصيغة
  - <%@ directive {attr=value}* %>

Three Types of Directives

ثلاثة أنواع من الأوامر التوجيهية

- page: Communicate page dependent attributes and communicate these to the JSP container
  - <%@ page import="java.util.* %>
- include: Used to include text and/or code at JSP page translation-time
  - <%@ include file="header.html" %>
- Taglib: Indicates a tag library that the JSP container should interpret
  - <%@ taglib uri="mytags" prefix="codecamp" %>

Page Directives

صفحةالأوامر التوجيهية

- Give high-level information about the servlet that results from the JSP page.

  تعطي مستوى عالي من المعلومات حول ال servletالتي تنتج من صفحة JSP

- Control التحكم
  - Which classes are imported
    - <%@ page import="java.util.* %>
  - What MIME type is generated
    - <%@ page contentType="MIME-Type" %>
  - How multithreading is handled
    - <%@ page isThreadSafe="true" %> <%!--Default --%>
    - <%@ page isThreadSafe="false" %>
  - What page handles unexpected errors
    - <%@ page errorPage="errorpage.jsp" %>

Implicit Objects

كائنات ضمنية

- A JSP page has access to certain implicit objects that are always available, without being declared first

  صفحة JSPعندها نفوذ إلى بعض الكائنات الضمنية التي تكون دائما متوفرة من دون أن تكون معلنة من قبل

- Created by container

  أنشأت من قبل الحاوية

- Corresponds to classes defined in Servlet

  تراسل الأقسام المعرفة في Servlet

Implicit Objects

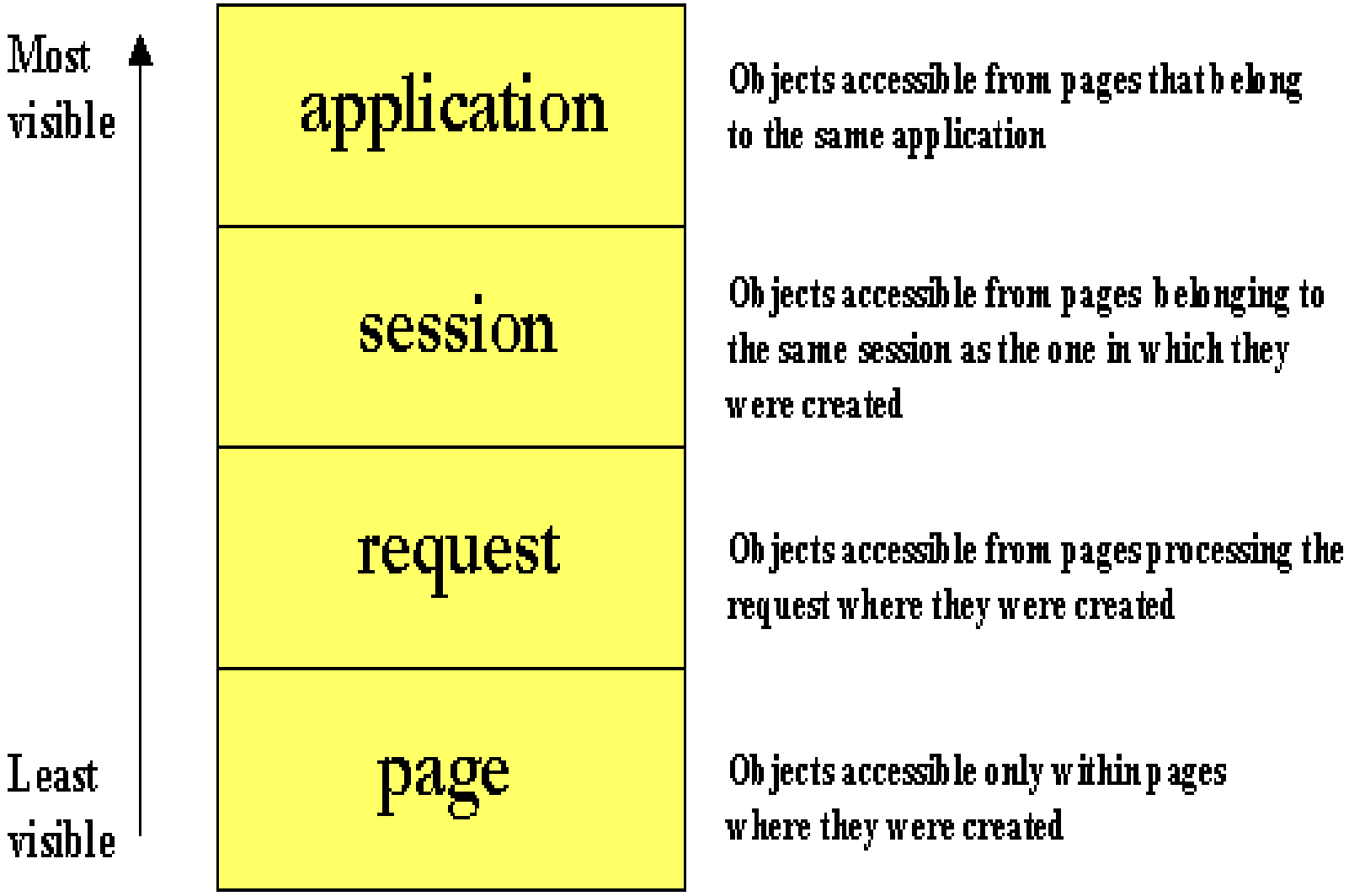
كائنات ضمنية

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

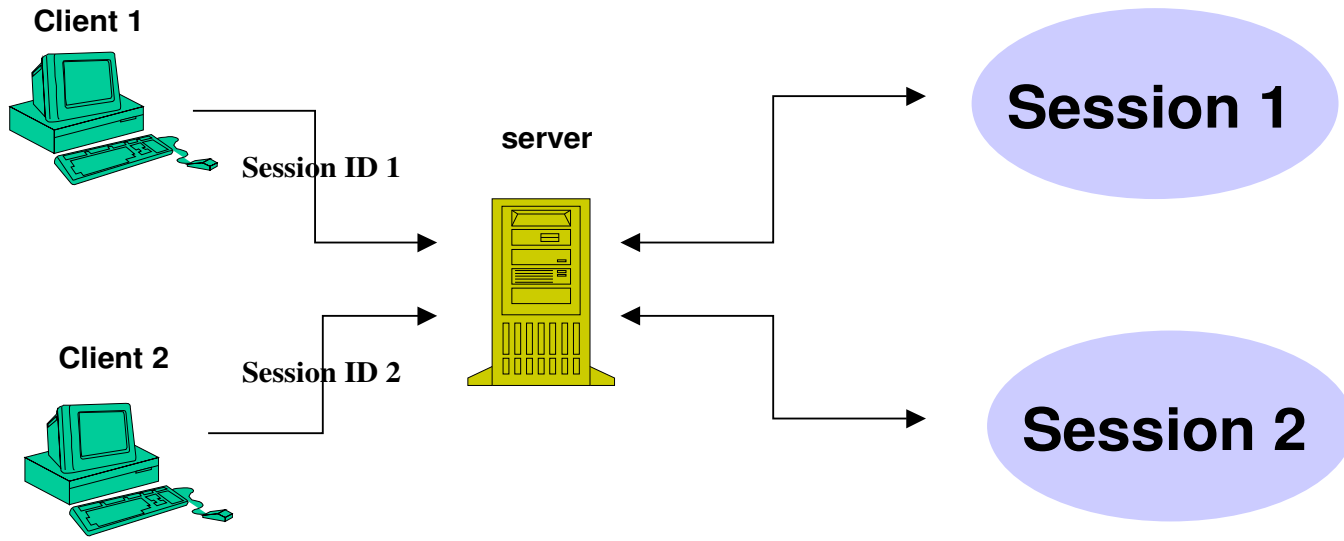
Scope Objects

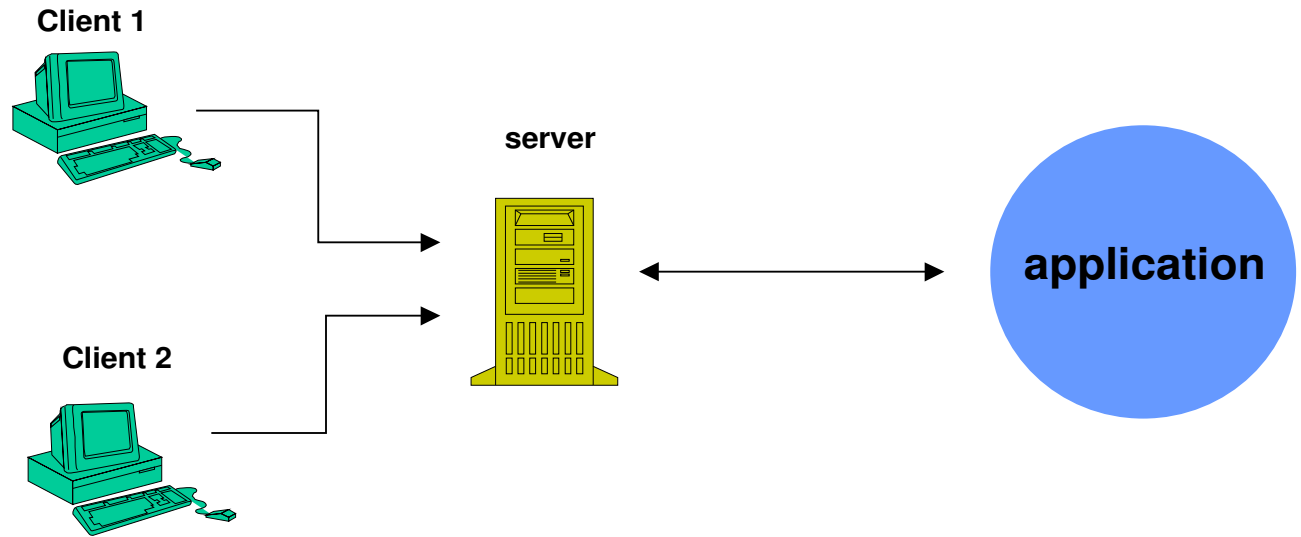
نطاق الكائنات

Different Scope نطاقات مختلفة

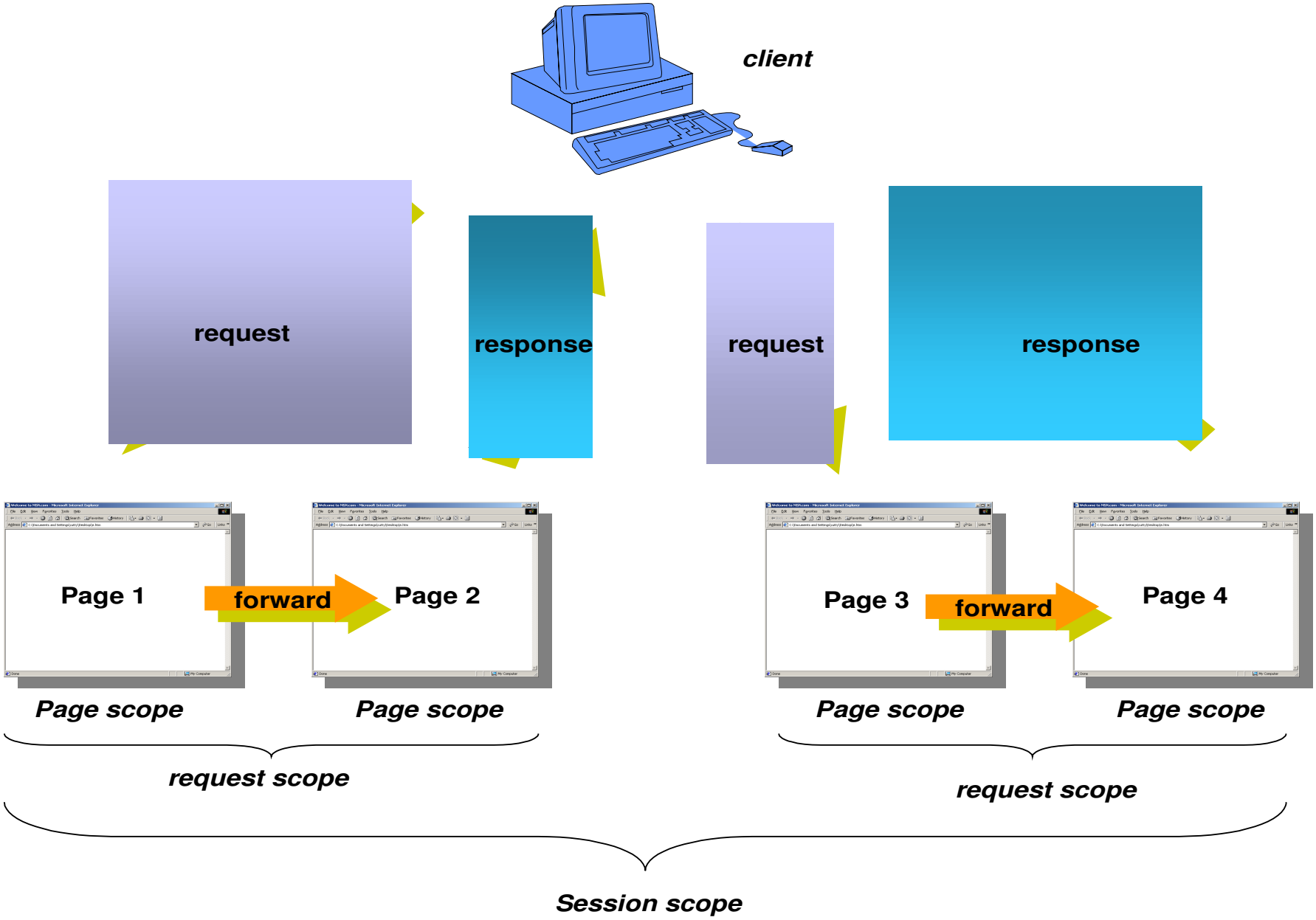


Session, Application Scope الجلسة و نطاق التطبيق





Session, Request, Page Scope الجلسة و الطلب و نطاق الصفحة



JavaBeans Components for JSP

What are JavaBeans? ما هي JavaBeans ؟

- Java classes that can be easily reused and composed together into an application

  أقسام جافا يمكن إعادة إستعمالها بسهولة كما يمكنها أن تتألف معا في تطبيق معين

- Any Java class that follows certain design conventions can be a JavaBeans component

  أي قسم جافا يتبع بعض التصاميم المتفق عليها يمكن أن يكون مكون JavaBeans

  - properties of the class خصائص القسم
  - public methods to get and set properties طرق عامة لوضع و أخذ الخصائص

- Within a JSP page, you can create and initialize beans and get and set the values of their properties

  من خلال صفحة JSP يمكن إنشاء و تهيئة beansو قيم الخصائص باستعمال get و set

- JavaBeans can contain business logic or data base access logic

  JavaBeans يمكن أن يحتوي على business logic أو يمكنه النفاذ المنطقي*إلى البيانات

JavaBeans Design Conventions

التصميم المتفق عليه لJavaBeans

- JavaBeans maintain internal properties

  يقوم JavaBeans بصيانة الخصائص الداخلية

- A property can be

  الخاصية يمكن أن تكون

  - Read/write, read-only, or write-only قراءة/كتابة أو قراءة فقط أو كتابة فقط
  - Simple or indexed
- Properties should be accessed and set via getXxx and setXxx methods

  الخصائص يجب أن تكون يمكن الوصول إليها و تعيينها عن طريق طريقة getXxxو طريقة setXxx

  - PropertyClass getProperty() { ... }
  - PropertyClass setProperty() { ... }
- JavaBeans must have a zero-argument (empty) constructor

  JavaBeans يجب أن يمتلك دالة بانية خالية من المعطيات

Example: JavaBeans

```
public class Currency {
  private Locale locale;
  private double amount;
  public Currency() {
    locale = null;
    amount = 0.0;
  }
  public void setLocale(Locale l) {
    locale = l;
  }
  public void setAmount(double a) {
    amount = a;
  }
  public String getFormat() {
     NumberFormat nf =
     NumberFormat.getCurrencyInstance(locale);
     return nf.format(amount);
  }
}
```

Why Use JavaBeans in JSP Page?

لماذا نستخدم JavaBeans في صفحة JSP؟

- A JSP page can create and use any type of Java programming language object within a declaration or scriptlet like following:

صفحة JSP يمكن أن تنشأ و تستخدم أي نوع من كائنات لغة جافا البرمجية من خلال الإبلاغ أو باستخدام scriptlet كما هو موضح في المثال التالي

```
<%
  ShoppingCart cart = (ShoppingCart)session.getAttribute("cart");
  // If the user has no cart, create a new one
  if (cart == null) {
     cart = new ShoppingCart();
     session.setAttribute("cart", cart);
  }
%>
```

Why Use JavaBeans in JSP Page?

لماذا نستخدم JavaBeans في صفحة JSP؟

- JSP pages can use JSP elements to create and access the object that conforms to JavaBeans conventions

صفحات JSPيمكن أن تستخدم عناصر JSP لإنشاء والنفاذ إلى الكائن المطابق لاتفاق JavaBeans

```
<jsp:useBean id="cart" class="cart.ShoppingCart"
  scope="session"/>
```

Create an instance of “ShoppingCart” if none exists, stores it as an attribute of the session scope object, and makes the bean available throughout the session by the identifier “cart”

إنشاء instance من ShoppingCart إذاكان غير موجود, و تخزينه كخاصية لكائن نطاق الجلسة و جعل ال bean متوفرفي كل مكان* في الجلسة عن طريق المعرف “cart”

Compare the Two مقارنة الإثنين

```
<%
  ShoppingCart cart = (ShoppingCart)session.getAttribute("cart");
  // If the user has no cart object as an attribute in Session scope
  // object, then create a new one.  Otherwise, use the existing
  // instance.
  if (cart == null) {
      cart = new ShoppingCart();
      session.setAttribute("cart", cart);
  }
%>
```

versus

```
<jsp:useBean id="cart" class="cart.ShoppingCart"
                    scope="session"/>
```

Why Use JavaBeans in JSP Page?

لماذا نستخدم JavaBeans في صفحة JSP؟

- No need to learn Java programming language for page designers لا حاجة لتعلم لغة جافا البرمجية لمصممي الصفحات
- Stronger separation between content and presentation أقوى فصل بين المحتوى والعرض
- Higher re usability of code قابلية عالية لإعادة إستعمال المصدر البرمجي
- Simpler object sharing through built-in sharing mechanism إشتراك أسهل في الكائن عن طريق ألية الإشتراك في البناء
- Convenient matching between request parameters and object properties مطابقة مريحة بين موسطات الطلب و خصائص الكائن

Creating a JavaBeans

إنشاء JavaBeans

- Declare that the page will use a bean that is stored within and accessible from the specified scope by jsp:useBean element

إبلاغ أن الصفحة ستستخدم bean المخزن عن طريقها و المتاحة من النطاق الخاص عن طريق العنصر jsp:useBean

```
<jsp:useBean id="beanName" class="fully_qualified_classname"
   scope="scope"/>
```

or

```
<jsp:useBean id="beanName" class="fully_qualified_classname"
   scope="scope">
      <jsp:setProperty .../>
</jsp:useBean>
```

Setting JavaBeans Properties

تعيين خصائص JavaBeans

- 2 ways to set a property of a bean طريقتان لتعيين خاصية bean
- Via scriptlet عن طريق scriptlet
  - <% beanName.setPropName(value); %>
- Via JSP:setProperty عن طريق JSP:setProperty
  - <jsp:setProperty name="beanName" property="propName" value="string constant"/>
  - “beanName” must be the same as that specified for the id attribute in a useBean element إسم bean يجب أن يكون نفس الإسم الذي عين في خاصية id في عنصر useBean
  - There must be a setPropName method in the bean لا بد من وجود طريقة setPropName في ال bean

Setting JavaBeans Properties

تعيين خصائص JavaBeans

- jsp:setProperty syntax depends on the source of the property jsp:setProperty مرتبطة بمصدر الخاصية

  صيغة

  - <jsp:setProperty name="beanName" property="propName" value="string constant"/>
  - <jsp:setProperty name="beanName" property="propName" param="paramName"/>
  - <jsp:setProperty name="beanName" property="propName"/>
  - <jsp:setProperty name="beanName" property="*"/>
  - <jsp:setProperty name="beanName" property="propName" value="<%= expression %>"/>

Example: jsp:setProperty with Request parameter

مثال : jsp:setProperty مع موسط الطلب

```
<jsp:setProperty name="bookDB" property="bookId"/>
```

is same as هو نفسه

```
<%
   //Get the identifier of the book to display
   String bookId = request.getParameter("bookId");
   bookDB.setBookId(bookId);
   ...
%>
```

Example: jsp:setProperty with Expression

مثال : jsp:setProperty مع التعابير

```
<jsp:useBean id="currency" class="util.Currency"
   scope="session">
   <jsp:setProperty name="currency" property="locale"
                        value="<%= request.getLocale() %>"/>
</jsp:useBean>

<jsp:setProperty name="currency" property="amount"
 value="<%=cart.getTotal()%>"/>
```

Getting JavaBeans Properties

للوصول إلى خصائص JavaBeans

- 2 different ways to get a property of a bean

  طريقتان مختلفتان للحصول على خاصية ال bean

  - Convert the value of the property into a String and insert the value into the current implicit “out” object

    تحويل قيمة الخاصية إلى مقطع و إدراج القيمة في الكائن الضمني الحالي

  - Retrieve the value of a property without converting it to a String and inserting it into the “out” object

    إسترجاع قيمة الخاصية بدون تحويله إلى مقطع و إدراجه في كائن “out”

Getting JavaBeans Properties & and Convert into String and insert into out

الوصول إلى خصائص JavaBeansو تحويلها إلى مقطع ثم إدراجها في الخرج

- 2 ways طريقتان
  - via scriptlet
    - <%= beanName.getPropName() %>
  - via JSP:setProperty
    - <jsp:getProperty name="beanName" property="propName"/>
- Requirements متطلبات
  - “beanName” must be the same as that specified for the id attribute in a useBean element إسم beanيجب أن يكون نفس الإسم الذي عين في خاصية idفي عنصر Bean
  - There must be a “getPropName()” method in a bean يجب أن يكون هناك طريقة “getPropName()” في bean

Getting JavaBeans Properties without Converting it to String

الوصول إلى خصائص JavaBeans من دون تحويلها إلى مقطع

- Must use a scriptlet يجب إستعمال scriptlet
- Format الصيغة

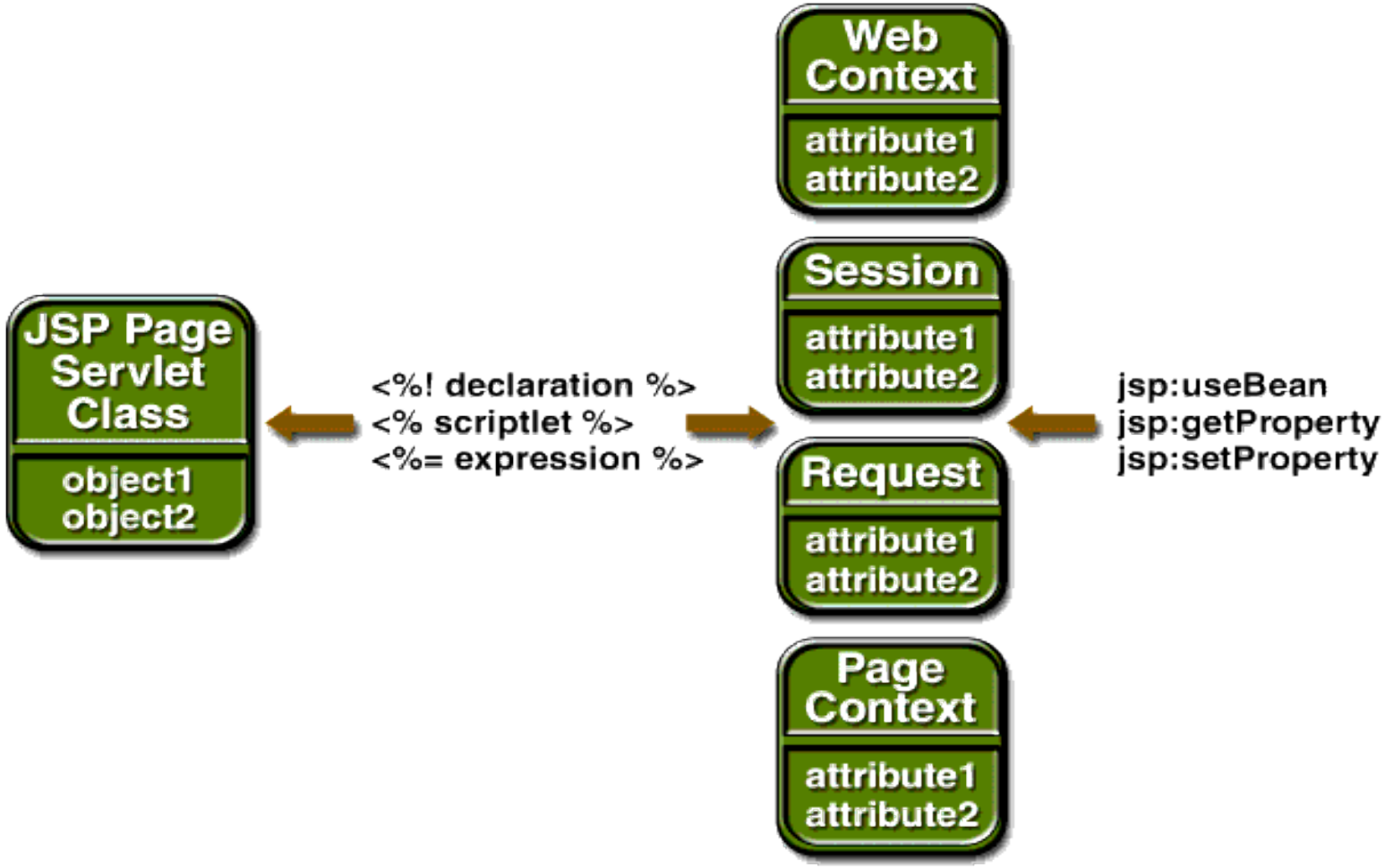
```
<% Object o = beanName.getPropName(); %>
```

- Example مثال

```
<%
  // Print a summary of the shopping cart
  int num = cart.getNumberOfItems();
  if (num > 0) {
%>
```

Accessing Objects from JSP Page

النفاذ إلى الكائنات من خلال صفحة JSP



Error Handling

معالجة الأخطاء

Creating An Exception Error Page

إنشاء صفحة الخطأ* الاستثنائية

- Determine the exception thrown تحديد الإستثناء*
- In each of your JSP, include the name of the error page
  في أي صفحة JSP من صفحاتك ضمن إسم صفحة الخطأ
  - <%@ page errorPage="errorpage.jsp" %>
- Develop an error page, it should include
  أنجز صفحة الخطأ التي يجب تضمينها
  - <%@ page isErrorPage="true" %>
- In the error page, use the exception reference to display exception information
  في صفحة الخطأ إستعمل مرجع للإستثناء* لعرض معلومات الإستثناء
  - <%= exception.toString() %>

Example: initdestroy.jsp

```
<%@ page import="database.*" %>
<%@ page errorPage="errorpage.jsp" %>
<%!

  private BookDBAO bookDBAO;
  public void jspInit() {

    // retrieve database access object, which was set once per
    web application
    bookDBAO =
      (BookDBAO)getServletContext().getAttribute("bookDB");
    if (bookDBAO == null)
        System.out.println("Couldn't get database.");
  }

  public void jspDestroy() {
    bookDBAO = null;
  }
%>
```

Example: errorpage.jsp

```
<%@ page isErrorPage="true" %>
<%@ page import="java.util.*" %>
<%
  ResourceBundle messages =
   (ResourceBundle)session.getAttribute("messages");
  if (messages == null) {
    Locale locale=null;
    String language = request.getParameter("language");

    if (language != null) {
      if (language.equals("English")) {
       locale=new Locale("en", "");
      } else {
       locale=new Locale("es", "");
      }
    } else
      locale=new Locale("en", "");

    messages = ResourceBundle.getBundle("BookStoreMessages",
   locale);
    session.setAttribute("messages", messages);
  }
%> ...
```

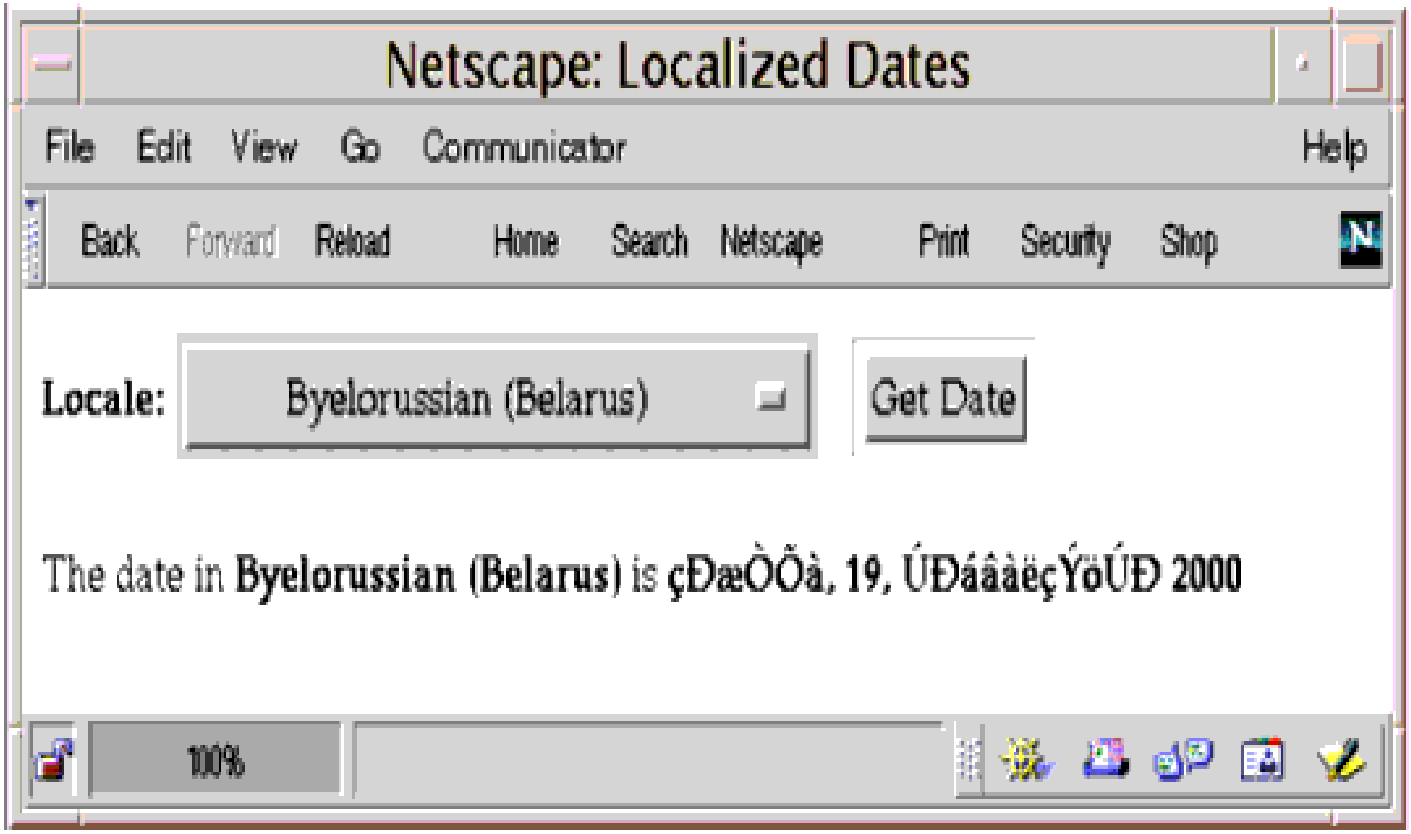
Example: errorpage.jsp

```
... (continued)
<html>
<head>
<title><%=messages.getString("ServerError")%></title>
</head>
<body bgcolor="white">
<h3>
<%=messages.getString("ServerError")%>
</h3>
<p>
<%= exception.getMessage() %>
</body>
</html>
```

Date Example

مثال التاريخ

Date Example Output



Date Example

```
<%@ page import="java.util.*" %>
<%@ page import="MyLocales" %>
<%@ page contentType="text/html; charset=ISO-8859-5" %>
<html>
<head><title>Localized Dates</title></head>
<body bgcolor="white">
<jsp:useBean id="locales" scope="application" class="MyLocales"/>
<form name="localeForm" action="index.jsp" method="post">
<b>Locale:</b>
```

Date Example مثال التاريخ

```
<select name=locale>
<%
 Iterator i = locales.getLocaleNames().iterator();
 String selectedLocale = request.getParameter("locale");
        while (i.hasNext()) {
              String locale = (String)i.next();
              if (selectedLocale != null && selectedLocale.equals(locale) )
   { %>
    <option selected><%=locale%></option>
<%            } else { %>
    <option><%=locale%></option>
<%            }
 }
%>
</select>
<input type="submit" name="Submit" value="Get Date">
</form>
```

Date Example مثال التاريخ

```
<p>
<jsp:include page="date.jsp" flush="true" />
</body>
</html>
```

Passion!