Session 5:

Java Server Pages (JSP)

Contents

- Introduction to JSP
- First Java Server Page Example
- Life cycle of JSP page
- Dynamic contents generation techniques in JSP
  - Scripting elements
  - Directives
  - Beans
  - Expression language
  - Custom tag libraries

What is JSP Page?

- A text-based document capable of returning both static and dynamic content to a client browser
- Static content and dynamic content can be intermixed
- Static content
  - HTML, XML, Text
- Dynamic content
  - Java code
  - Displaying properties of JavaBeans
  - Invoking business logic defined in Custom tags

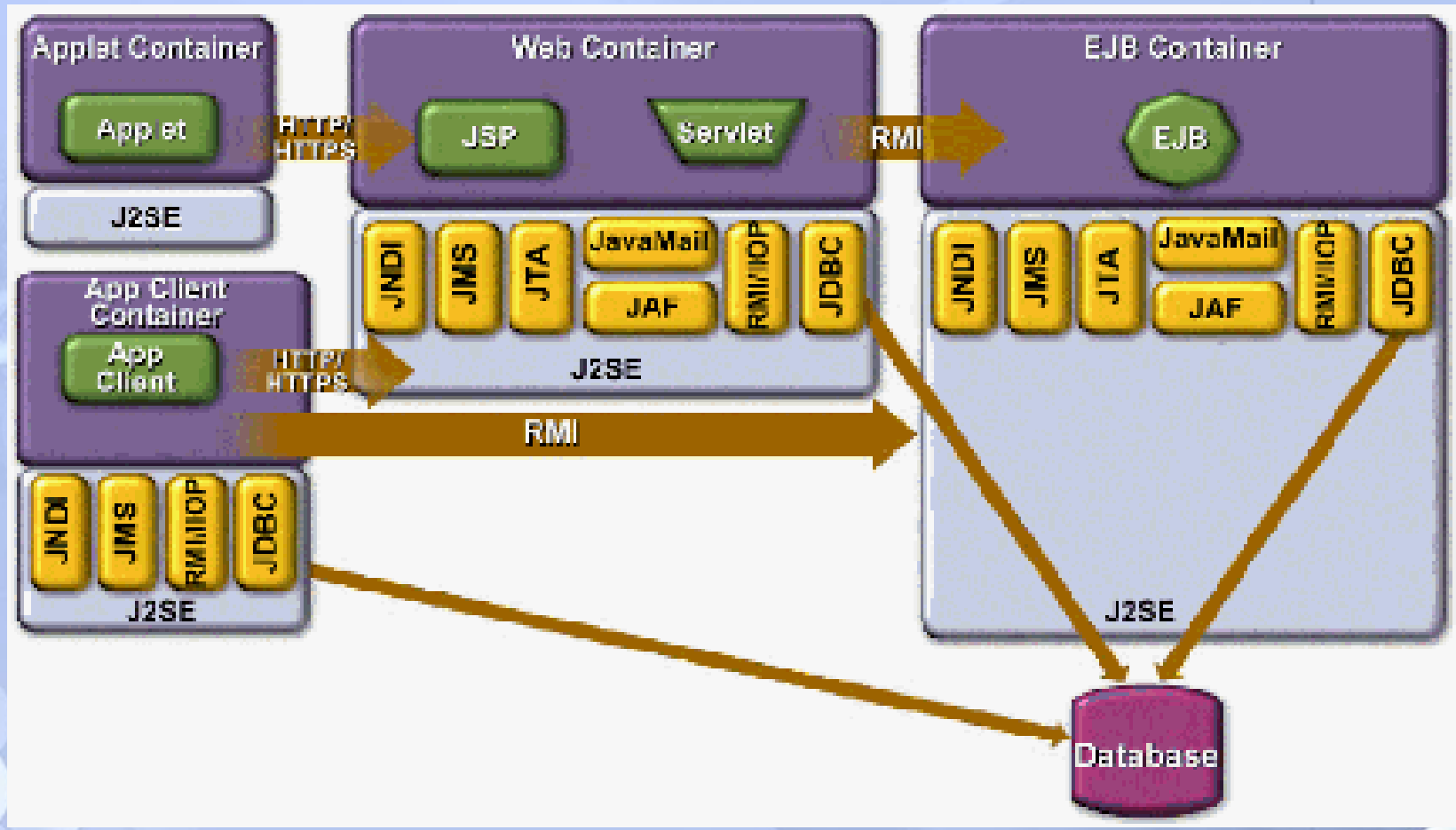
Static & Dynamic Contents

- Static contents
  - Typically static HTML page
  - Same display for everyone
- Dynamic contents
  - Contents is dynamically generated based on conditions
  - Conditions could be
    - User identity
    - Time of the day
    - User entered values through forms and selections
  - Examples
    - E-Trade webpage customized just for you, my Yahoo

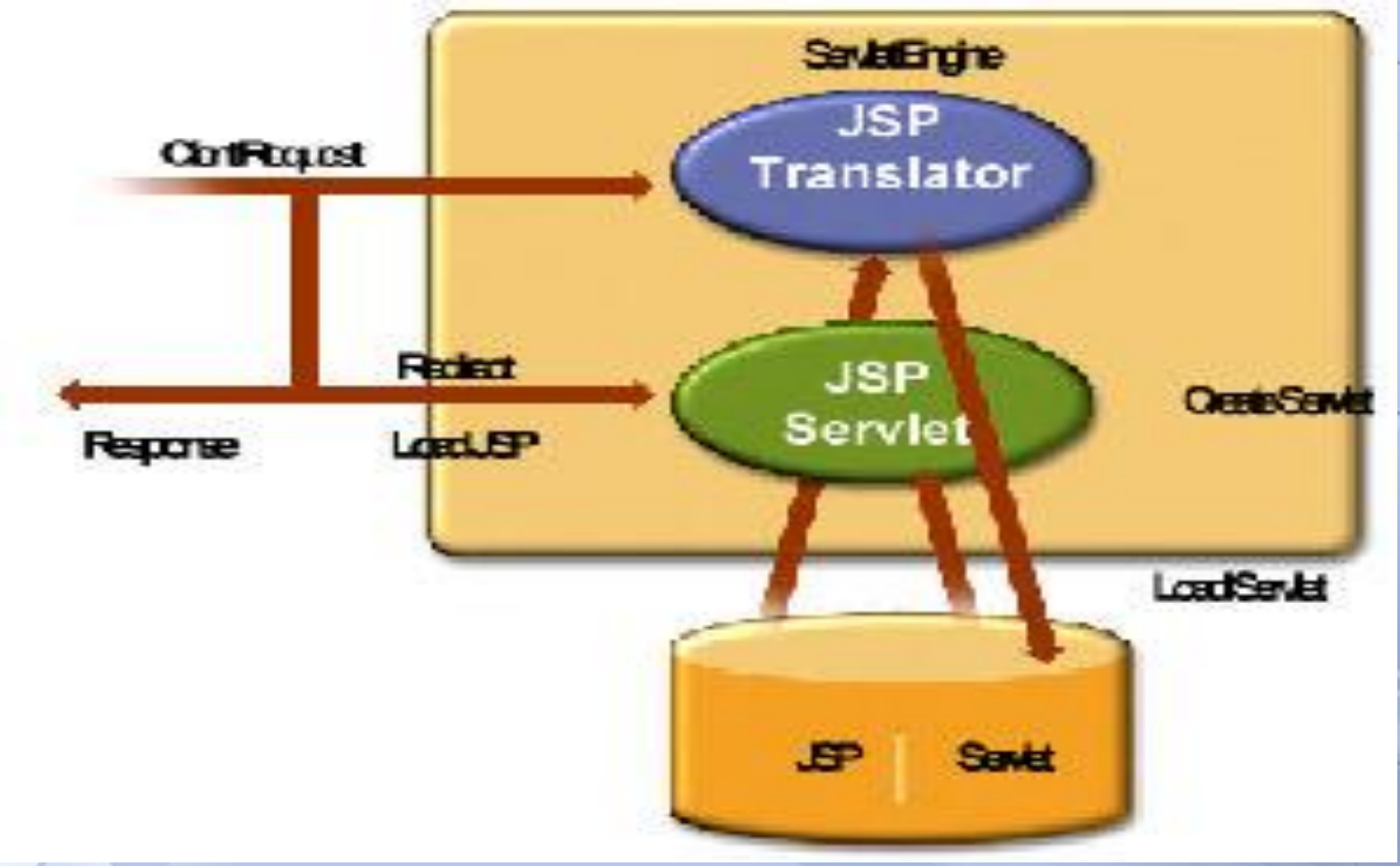
A Simple JSP Page

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

JSP & Servlet as Web Components



JSP Architecture



JSP Page Lifecycle Phases

- Translation phase
- Compile phase
- Execution phase

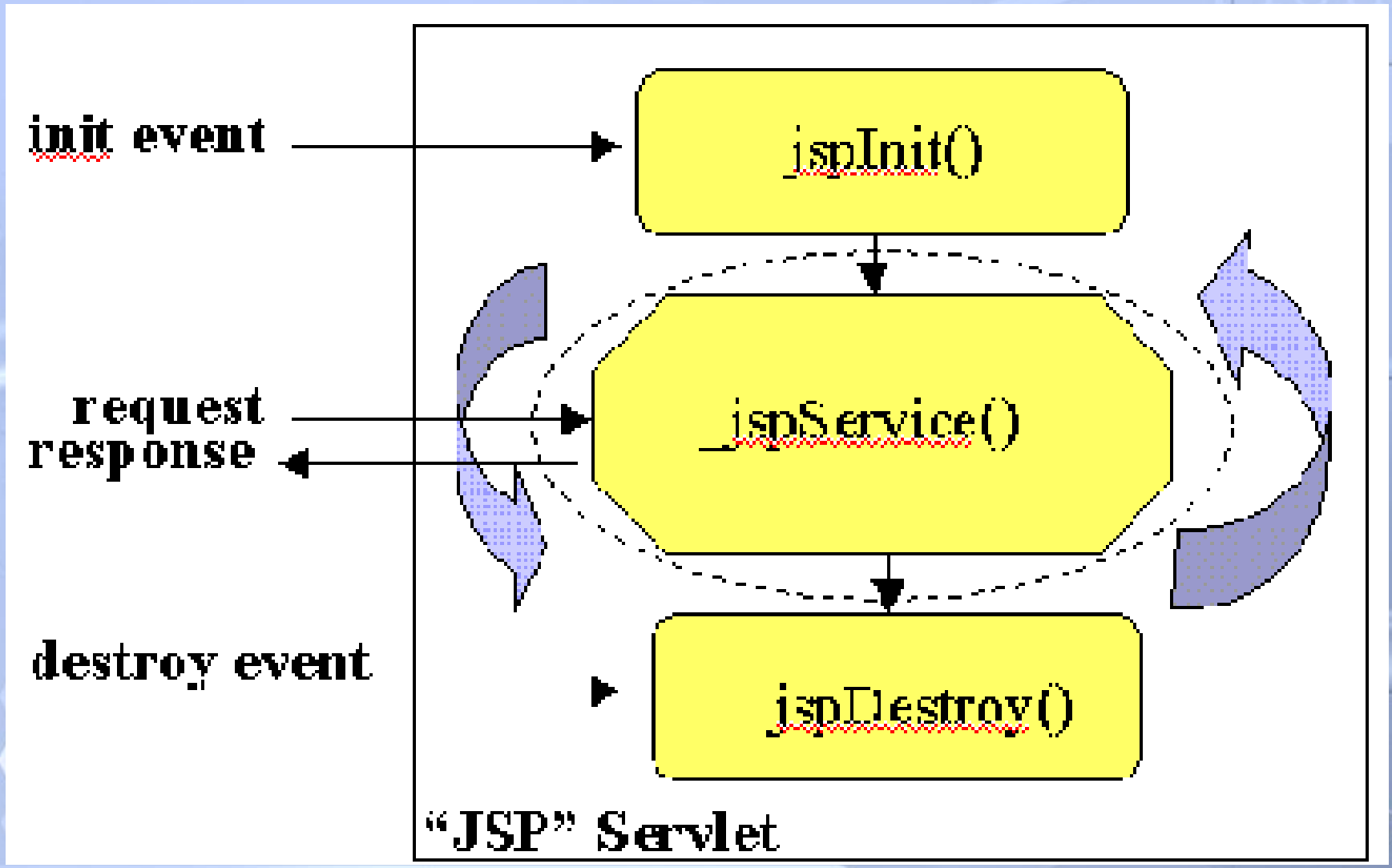
Translation/Compilation Phase

- JSP files get translated into servlet source code, which is then compiled
- Done by the container automatically
- The first time JSP page is accessed after it is deployed (or modified and redeployed)

Translation/Compilation Phase

- Static Template data is transformed into code that will emit data into the stream
- JSP elements are treated differently
  - Directives are used to control how Web container translates and executes JSP page
  - Scripting elements are inserted into JSP page's servlet class
  - Elements of the form <jsp:xxx .../> are converted into method calls to JavaBeans components

JSP Lifecycle Methods during Execution Phase



JSP vs. Servlet

JSP

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

Servlet

- HTML code in Java
- Not easy to author

Key components of JSP

- Scripting Elements
- Directives
- Actions
- Tag libraries

❖ **for the purpose of dynamic content generation**

Scriptlet

- **Scriptlet**
  - Also called “Scripting Elements”
  - Enable programmers to insert Java code in JSPs
  - Performs request processing
    - Interacts with page elements and other components to implement dynamic pages

Directive

- **Directive**
  - Message to JSP container
    - i.e., program that compiles/executes JSPs
  - Enable programmers to specify
    - Page settings
    - Content to include from other resources
    - Custom tag libraries used in the JSP

Action

- **Action**
  - Predefined JSP tags that encapsulate functionality
  - Often performed based on information from client request
  - Can be used to create Java objects for use in scriptlets

Custom Tag Library

- **Custom Tag Library**
  - JSP's tag extension mechanism
  - Enables programmers to define new tags
    - Tags encapsulate complex functionality
  - Tags can manipulate JSP content

JSP Scripting Elements

JSP Scripting Elements

- Lets you insert Java code into the servlet that will be generated from JSP page
- Minimize the usage of JSP scripting elements in your JSP pages if possible
- There are three forms
  - **Expressions: <%= Expressions %>**
  - **Scriptlets: <% Code %>**
  - **Declarations: <%! Declarations %>**
  - **Comments**
    - JSP comments (delimited by **<%--** and **--%>**)
    - XHTML comments (delimited by **<!--** and **-->**)
    - Java's comments (delimited by // and /* and */)

Expressions

- During execution phase
  - Expression is evaluated and converted into a String
  - The String is then Inserted into the servlet's output stream directly
  - Results in something like out.println(expression)
  - Can use predefined variables (implicit objects) within expression

- Format
  - **<%= Expression %>** or
  - **<jsp:expression>Expression</jsp:expression>**

Example: Expressions

- Display current time using Date class
  - Current time: **<%= new java.util.Date() %>**
- Display random number using Math class
  - Random number: **<%= Math.random() %>**
- Use implicit objects
  - Your hostname: **<%= request.getRemoteHost() %>**
  - Your parameter: **<%= request.getParameter("yourParameter") %>**
  - Server: **<%= application.getServerInfo() %>**
  - Session ID: **<%= session.getId() %>**

Example: Expressions

```
<html>
<head>
<title>Insert title here</title>
</head>
<body>
    Current time: <%= new java.util.Date() %> <br/>
    Random number: <%= Math.random() %><br/>
    Your hostname: <%= request.getRemoteHost() %><br/>
    Your parameter: <%= request.getParameter("yourParameter") %><br/>
    Server: <%= application.getServerInfo() %><br/>
    Session ID: <%= session.getId() %><br/>

</body>
</html>
```

Result

Current time: Sun Dec 14 20:25:03 MMT 2008
Random number: 0.0276646597271944 5
Your hostname: 0:0:0:0:0:0:0:1
Your parameter: null
Server: Apache Tomcat/6.0.14
Session ID: 89DF7E44C249C832D352C623EB76501F

Scriptlets

- Used to insert arbitrary Java code into servlet's jspService() method
- Can do things expressions alone cannot do
  - setting response headers and status codes
  - writing to a server log
  - updating database
  - executing code that contains loops, conditionals
- Can use predefined variables (implicit objects)
- Format:
  - **<% Java code %>** or
  - **<jsp:scriptlet> Java code</jsp:scriptlet>**

Example: Scriptlets

- Display query string

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

- Settng response type

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

Example: Scriptlet with Loop

```
<TABLE>
    <%
        int n=10;
        for ( int i = 0; i < n; i++ ) {
     %>
      <TR>
          <TD>Number</TD>
          <TD><%= i+1 %></TD>
      </TR>
      <%
          }
      %>
</TABLE>
```

Declarations

- Used to define variables or methods that get inserted into the main body of servlet class
  - Outside of _jspSevice() method
  - Implicit objects are not accessible to declarations
- Usually used with Expressions or Scriptlets
- For initialization and cleanup in JSP pages, use declarations to override jspInit() and jspDestroy() methods
- Format:
  - **<%! method or variable declaration code %>**
  - **<jsp:declaration> method or variable declaration code </jsp:declaration>**

Example: JSP Page fragment

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

<%= randomHeading() %>
```

Example: Declaration

```
<%!
    private BookDBAO bookDBAO;

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

Example : welcome.jsp

```
<?xml version = "1.0"?>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

    <!-- welcome.jsp -->
    <!-- JSP that processes a "get" request containing data. -->

    <html xmlns = "http://www.w3.org/1999/xhtml">

     <!-- head section of document -->
     <head>
       <title>Processing "get" requests with data</title>
     </head>
```

welcome.jsp (cont.)

```
<!-- body section of document -->
    <body>
        <% // begin scriptlet
            String name = request.getParameter( "firstName" );
            if ( name != null ) {
        %> <%-- end scriptlet to insert fixed template data --%>
            <h1>
                Hello <%= name %>, <br />
                Welcome to JavaServer Pages!
            </h1>
        <% // continue scriptlet
            }  // end if
            else {
        %> <%-- end scriptlet to insert fixed template data --%>
```

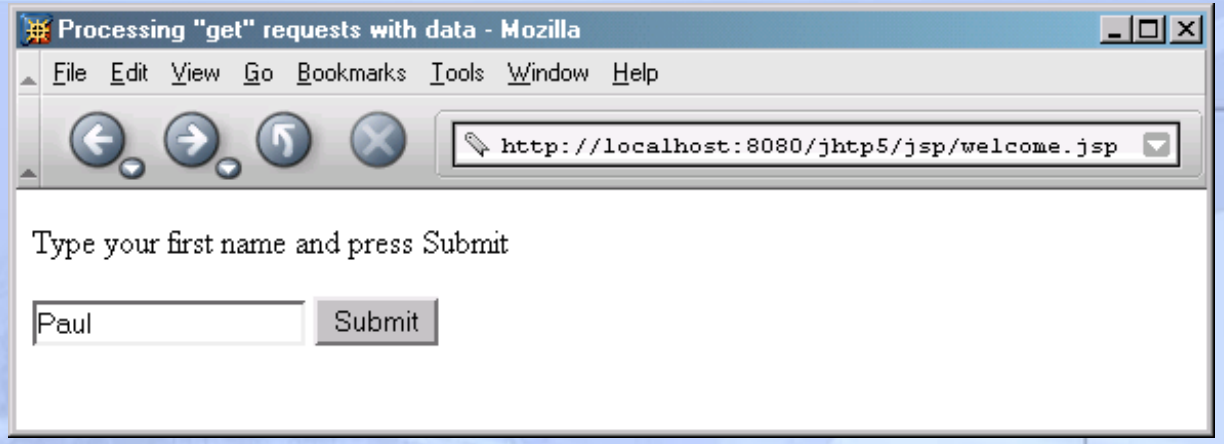
welcome.jsp (cont.)

```
        <form action = "welcome.jsp" method = "get">
            <p>Type your first name and press Submit</p>
            <p><input type = "text" name = "firstName" />
              <input type = "submit" value = "Submit" />
            </p>
          </form>

    <% // continue scriptlet

       }  // end else
    %> <%-- end scriptlet --%>
  </body>

</html>  <!-- end XHTML document -->
```


Processing "get" requests with data - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/jsp/welcome.jsp
Type your first name and press Submit
Paul
Submit


Processing "get" requests with data - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/jsp/welcome.jsp?f
Hello Paul,
Welcome to JavaServer Pages!

Directives

Directives

- JSP directives
  - Messages to JSP container
  - Enable programmer to:
    - Specify **page settings**
    - **Include content** from other resources
    - Specify **custom-tag libraries**
  - Syntax

    <%@ directive {attr=value}* %>

Three Types of Directives

Directive	Description
page	Defines page settings for the JSP container to process.
include	Causes the JSP container to perform a translation-time insertion of another resource's content. As the JSP is translated into a servlet and compiled, the referenced file replaces the include directive and is translated as if it were originally part of the JSP.
taglib	Allows programmers to define new tags in the form of *tag libraries*, which can be used to encapsulate functionality and simplify the coding of a JSP.

Three Types of Directives

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

- Give high-level information about the servlet that results from the JSP page.
- Control
  - 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" %>

Page Directives

Attribute	Description
language	The scripting language used in the JSP. Currently, the only valid value is Java.
extends	Specifies the class from which the translated JSP will be inherited. This attribute must be a fully qualified class name.
import	Specifies a comma-separated list of fully qualified type names and/or packages that will be used in the current JSP. When the scripting language is java, the default import list is java.lang.*, javax.servlet.*, javax.servlet.jsp.* and javax.servlet.http.*. If multiple import properties are specified, the package names are placed in a list by the container.

Page Directives

Attribute	Description
session	Specifies whether the page participates in a session. The values for this attribute are true (participates in a session—the default) or false (does not participate in a session). When the page is part of a session, implicit object session is available for use in the page. Otherwise, session is not available, and using session in the scripting code results in a translation-time error.
buffer	Specify the size of output buffer used with the implicit object out. The default buffer size is 8KB.
autoFlush	When set to true(default), 'out' should be flushed automatically when buffer fills.
isThreadSafe	If true(default), it can process multiple request at the same time.

Page Directives

Attribute	Description
info	Specifies the information string about the page. This string is returned by the getServletInfo() method.
errorPage	Any exceptions in the current page that are not caught are sent to the error page for processing. The error page implicit object exception references the original exception.
isErrorPage	Specifies if the current page is an error page that will be invoked in response to an error on another page. true => exception is created **false** (the default), any use of the exception object in the page results in a translation-time error.
contentType	Specifies the MIME type of the data in the response to the client. The default type is text/html.

Action

Standard Actions

- JSP standard actions
  - Provide access to common tasks performed in a JSP
    - **Including content from other resources**
    - **Forwarding requests to other resources**
    - **Interacting with JavaBeans**
  - JSP containers process actions at request time
  - Delimited by <jsp:*action*> and </jsp:*action*>

Standard Actions

Action	Description
`<jsp:include>`	Dynamically includes another resource in a JSP. As the JSP executes, the referenced resource is included and processed.
`<jsp:forward>`	Forwards request processing to another JSP, servlet or static page. This action terminates the current JSP's execution.
`<jsp:plugin>`	Allows a plug-in component to be added to a page in the form of a browser-specific `object` or `embed` HTML element. In the case of a Java applet, this action enables the downloading and installation of the *Java Plug-in*, if it is not already installed on the client computer.
`<jsp:param>`	Used with the `include`, `forward` and `plugin` actions to specify additional name/value pairs of information for use by these actions.
JavaBean Manipulation	
`<jsp:useBean>`	Specifies that the JSP uses a JavaBean instance. This action specifies the scope of the bean and assigns it an ID that scripting components can use to manipulate the bean.
`<jsp:setProperty>`	Sets a property in the specified JavaBean instance. A special feature of this action is automatic matching of request parameters to bean properties of the same name.
`<jsp:getProperty>`	Gets a property in the specified JavaBean instance and converts the result to a string for output in the response.

Including Contents in a JSP Page

- Two mechanisms for including another Web resource in a JSP page
  - **include** directive
  - **jsp:include** element/action

include Directive

- is processed when the JSP page is translated into a servlet class
- 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
- Used to include banner content, copyright information, or any chunk of content that you might want to reuse in another page
- Syntax and Example
  - **<%@ include file="filename" %>**
  - **<%@ include file="banner.jsp" %>**

jsp:include Element

- is processed when a JSP page is executed
- Allows you to include either a static or dynamic resource in a JSP file
  - static: its content is inserted into the calling JSP file
  - 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
- 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
  - **It is faster than jsp:include**
- Use **jsp:include** for content that changes often
- Use **jsp:include** if which page to include cannot be decided until the main page is requested

Forwarding to another Web component

- Same mechanism as in Servlet
- Syntax

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

- Original request object is provided to the target page via jsp:parameter element

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

<jsp:useBean> Action

- <jsp:useBean> action
  - Enables JSP to manipulate Java object
    - Creates Java object or locates an existing object for use in JSP

<jsp:useBean> Action

Attribute	Description
`id`	The name used to manipulate the Java object with actions `<jsp:setProperty>` and `<jsp:getProperty>`. A variable of this name is also declared for use in JSP scripting elements. The name specified here is case sensitive.
`scope`	The scope in which the Java object is accessible—`page`, `request`, `session` or `application`. The default scope is `page`.
`class`	The fully qualified class name of the Java object.
`beanName`	The name of a bean that can be used with method `instantiate` of class `java.beans.Beans` to load a JavaBean into memory.
`type`	The type of the JavaBean. This can be the same type as the `class` attribute, a superclass of that type or an interface implemented by that type. The default value is the same as for attribute `class`. A `ClassCastException` occurs if the Java object is not of the type specified with attribute `type`.

<jsp:useBean> Action (cont.)

Attribute	Description
name	The ID of the JavaBean for which a property (or properties) will be set.
property	The name of the property to set. Specifying "*" for this attribute causes the JSP to match the request parameters to the properties of the bean. For each request parameter that matches (i.e., the name of the request parameter is identical to the bean's property name), the corresponding property in the bean is set to the value of the parameter. If the value of the request parameter is "", the property value in the bean remains unchanged.
param	If request parameter names do not match bean property names, this attribute can be used to specify which request parameter should be used to obtain the value for a specific bean property. This attribute is optional. If this attribute is omitted, the request parameter names must match bean property names.
value	The value to assign to a bean property. The value typically is the result of a JSP expression. This attribute is particularly useful for setting bean properties that cannot be set using request parameters. This attribute is optional. If this attribute is omitted, the JavaBean property must be of a type that can be set using request parameters.

Fig. 25.16 Attributes of the <jsp:setProperty> action.

Example

```
package mypack;
public class Student {
    private String name;
    private int rollno;

    public Student(){}
    public Student(String name, int rollno) {
        this.name = name;
        this.rollno = rollno; }
    public String getName() { return name; }
    public void setName(String name) {this.name = name; }
    public int getRollno() { return rollno; }
    public void setRollno(int rollno) { this.rollno = rollno; }
}
```

Example JSP

```
<html>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<body>
    <jsp:useBean id="s" class="mypack.Student" scope="session">
        <jsp:setProperty name="s" property="name" value="ma ma"/>
        <jsp:setProperty name="s" property="rollno" value="1"/>
    </jsp:useBean>
    <jsp:getProperty name="s" property="name"/>
    <jsp:getProperty name="s" property="rollno"/>
</body>
</html>
```

Implicit Objects

Implicit Objects

- A JSP page has access to certain implicit objects that are always available, without being declared first
- Created by container
- Corresponds to classes defined in Servlet

Implicit Objects

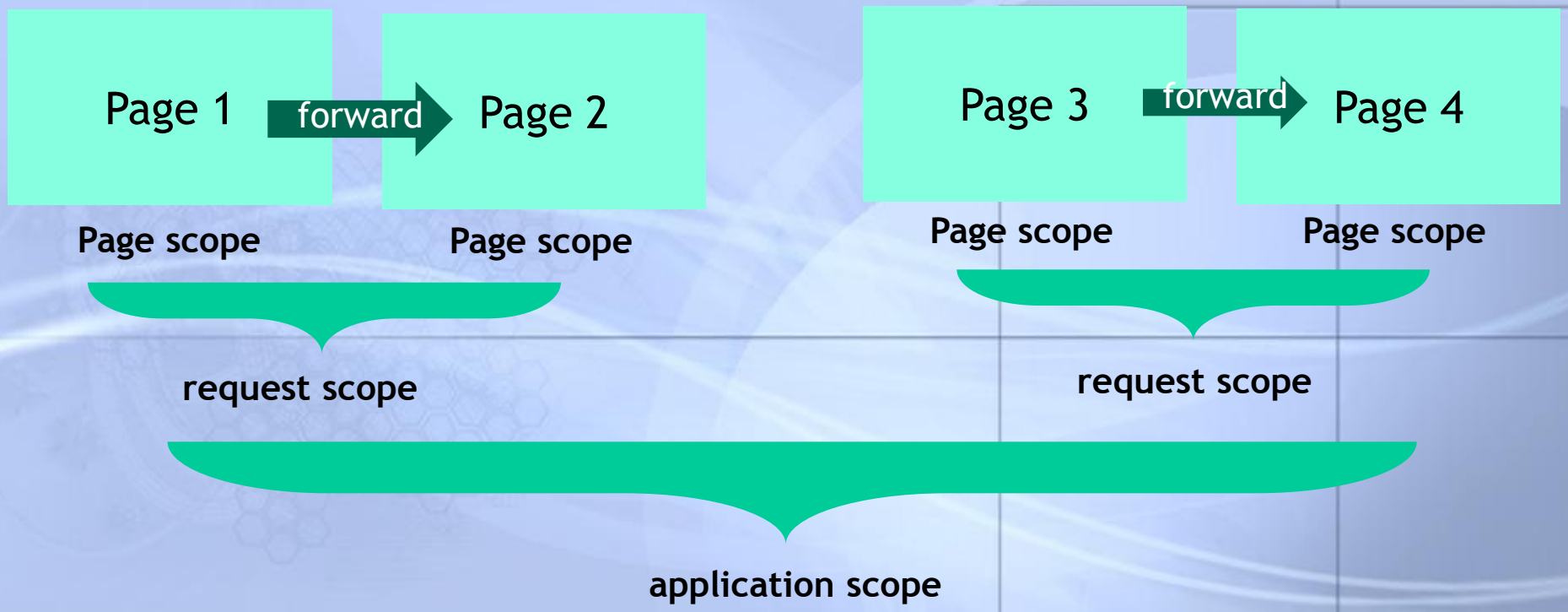
- Provide access to many servlet capabilities within a JSP
- Four scopes
  - **Application scope**
    - Objects owned by the container application
    - Any servlet or JSP can manipulate these objects
  - **Session scope**
    - Objects exist for duration of client's browsing session
    - Objects go out of scope when client terminates session or when session timeout occurs
  - **Request scope**
    - Objects exist for duration of client request
    - Objects go out of scope after response sent to client
  - **Page scope**
    - Objects that exist only in page in which they are defined
    - Each page has its own instance of these objects

Implicit object	Description
Application Scope	
`application`	This `javax.servlet.ServletContext` object represents the container in which the JSP executes.
Page Scope	
`config`	This `javax.servlet.ServletConfig` object represents the JSP configuration options. As with servlets, configuration options can be specified in a Web application descriptor.
`exception`	This `java.lang.Throwable` object represents the exception that is passed to the JSP error page. This object is available only in a JSP error page.
`out`	This `javax.servlet.jsp.JspWriter` object writes text as part of the response to a request. This object is used implicitly with JSP expressions and actions that insert string content in a response.
`page`	This `java.lang.Object` object represents the `this` reference for the current JSP instance.
`pageContext`	This `javax.servlet.jsp.PageContext` object hides the implementation details of the underlying servlet and JSP container and provides JSP programmers with access to the implicit objects discussed in this table.
`response`	This object represents the response to the client and is normally an instance of a class that implements `HttpServletResponse` (package `javax.servlet.http`). If a protocol other than HTTP is used, this object is an instance of a class that implements `javax.servlet.ServletResponse`.
Request Scope	
`request`	This object represents the client request. The object normally is an instance of a class that implements `HttpServletRequest` (package `javax.servlet.http`). If a protocol other than HTTP is used, this object is an instance of a subclass of `javax.servlet.Servlet-Request`.
Session Scope	
`session`	This `javax.servlet.http.HttpSession` object represents the client session information if such a session has been created. This object is available only in pages that participate in a session.

Implicit Objects & their classes

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

Application, Session, Request, Page Scope



Error Handling

Creating An Exception Error Page

- Determine the exception thrown
- In each of your JSP, include the name of the error page
  - ***<%@ page errorPage="errorpage.jsp" %>***
- Develop an error page, it should include
  - **<%@ page isErrorPage="true" %>**
- In the error page, use the exception reference to display exception information
  - **<%= exception.toString() %>**

Example

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

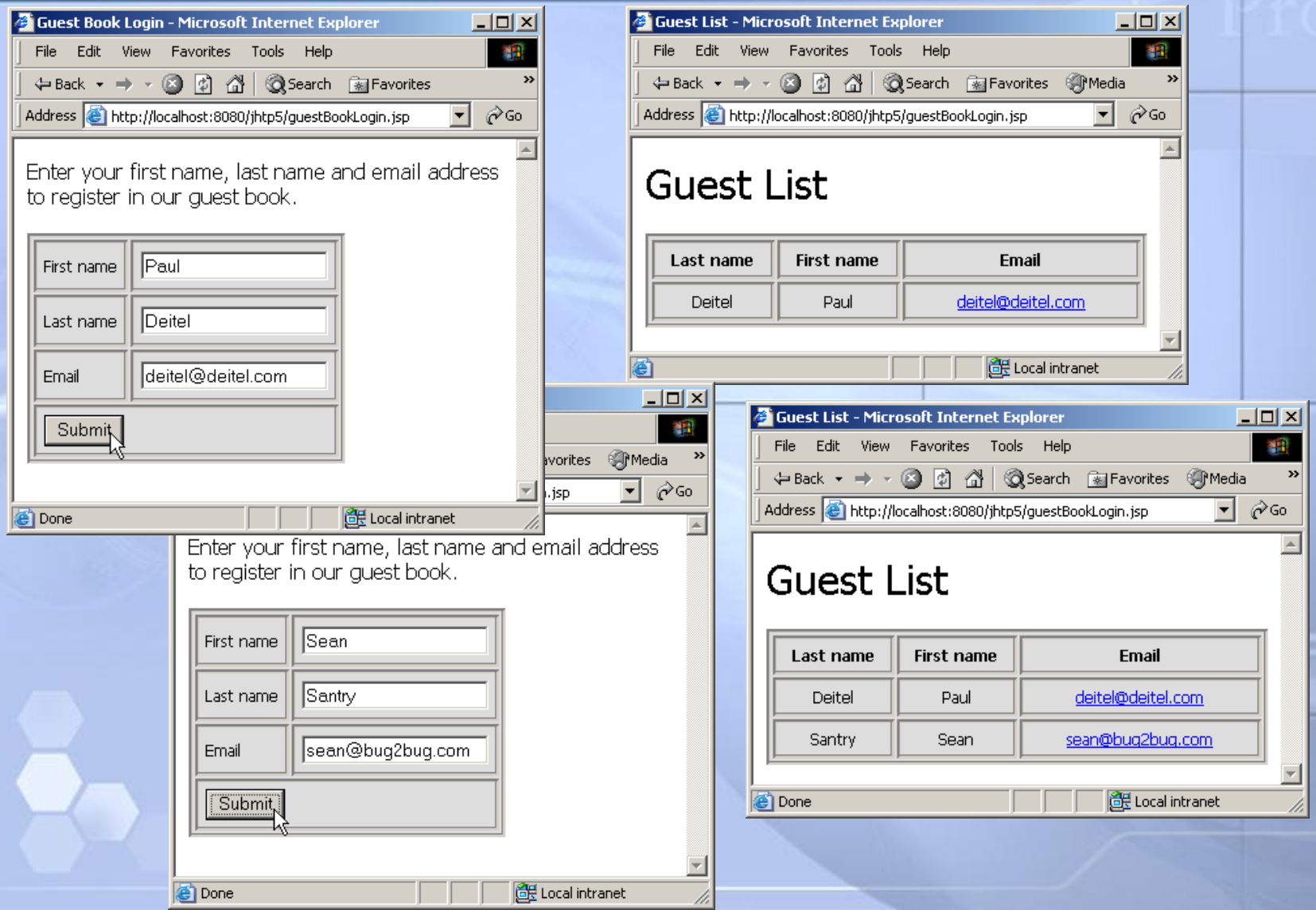
Example: errorpage.jsp

```
<%@ page isErrorPage="true" %>
<%@ page import="java.util.*" %>
…..
```

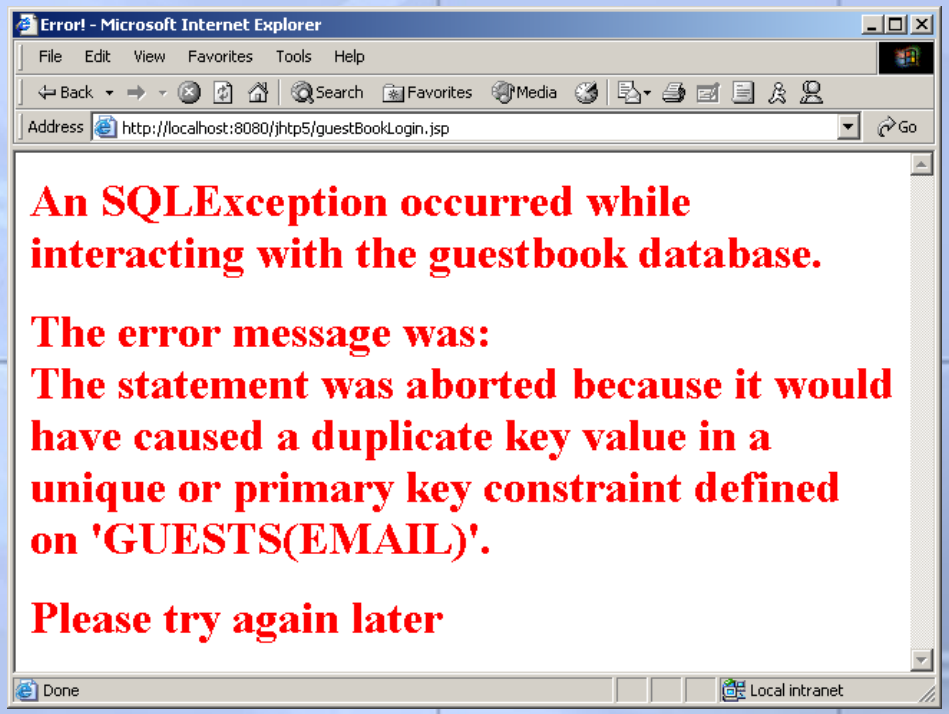
Lets make exercise!

- **Guest Book**
  - Including and using
    - Action
    - JSP page directive
    - JSP error pages
    - Use of JDBC

Case Study: Guest Book (Cont.)



Case Study: Guest Book (Cont.)



Thank You!