



Integrating Servlets and JSP: The MVC Architecture

Core Servlets & JSP book: www.coreservlets.com
More Servlets & JSP book: www.moreservlets.com
Servlet and JSP Training Courses: courses.coreservlets.com

Slides © Marty Hall, <http://www.coreservlets.com>, book © Sun Microsystems Press

2

Agenda

- Reasons to combine servlets and JSP
- Approach to integration
- Dispatching requests
- Storing data for later retrieval
- Example 1:
an on-line travel agent
- Example 2:
an on-line boat store
- Including requests:
showing raw servlet and JSP output

3

Uses of JSP Constructs

Simple
Application



Complex
Application

- Scripting elements calling servlet code directly
- Scripting elements calling servlet code indirectly (by means of utility classes)
- Beans
- Custom tags
- **Servlet/JSP combo (MVC), with beans and possibly custom tags**

4

MVC Architecture

www.coreservlets.com

Why Combine Servlets & JSP?

- **Typical picture: use JSP to make it easier to develop and maintain the HTML content**
 - For simple dynamic code, call servlet code from scripting elements
 - For slightly more complex applications, use custom classes called from scripting elements
 - For moderately complex applications, use beans and custom tags
- **But, that's not enough**
 - For complex processing, starting with JSP is awkward
 - Despite the ease of separating the real code into separate classes, beans, and custom tags, the assumption behind JSP is that a *single* page gives a *single* basic look

5

MVC Architecture

www.coreservlets.com

Possibilities for Handling a Single Request

- **Servlet only**
 - Output is a binary type. E.g.: an image
 - No output. E.g.: you are doing forwarding or redirection as in Search Engine example.
 - Format/layout of page is highly variable. E.g.: portal.
- **JSP only**
 - Output is mostly character data. E.g.: HTML
 - Format/layout mostly fixed.
- **Combination**
 - A single request will result in multiple substantially different-looking results.
 - Complicated data processing, but relatively fixed layout.
- **These apply to a *single* request**
 - You still use both servlets and JSP within your *overall* application.

Approach

- **Joint servlet/JSP process:**
 - Original request is answered by a servlet
 - Servlet processes request data, does database lookup, business logic, etc.
 - Results are placed in beans
 - Request is forwarded to a JSP page to format result
 - Different JSP pages can be used to handle different types of presentation
- **Often called the "MVC" (Model View Controller) or "Model 2" approach to JSP**
- **Formalized in Apache Struts Framework**
 - <http://jakarta.apache.org/struts/>

Implementing MVC

- **The important thing is the idea**
 - Syntax not complicated
- **We already know**
 - How to extract previously-stored data in a JSP page
 - Use `jsp:useBean` with the `scope` attribute
- **Two pieces of syntax we don't yet know**
 - How does a servlet invoke a JSP page?
 - How does a servlet store data where it can be retrieved by
 - `jsp:useBean` with `scope="request"`
 - `jsp:useBean` with `scope="session"`
 - `jsp:useBean` with `scope="application"`

Dispatching Requests from Servlets to JSP Pages

- **First, call the `getRequestDispatcher` method of `ServletContext`**
 - Supply URL relative to server or Web application root
 - Example
 - ```
String url = "/presentations/presentation1.jsp";
RequestDispatcher dispatcher =
 getServletContext().getRequestDispatcher(url);
```
- **Second**
  - Call `forward` to completely transfer control to destination page (no communication with client in between, as there is with `response.sendRedirect`)
    - This is the normal approach with MVC
  - Call `include` to insert output of destination page and then continue on

# Forwarding Requests: Example Code

```
public void doGet(HttpServletRequest request,
 HttpServletResponse response)
 throws ServletException, IOException {
 String operation = request.getParameter("operation");
 if (operation == null) {
 operation = "unknown";
 }
 if (operation.equals("operation1")) {
 gotoPage("/operations/presentation1.jsp",
 request, response);
 } else if (operation.equals("operation2")) {
 gotoPage("/operations/presentation2.jsp",
 request, response);
 } else {
 gotoPage("/operations/unknownRequestHandler.jsp",
 request, response);
 }
}
```

# Forwarding Requests: Example Code (Continued)

```
private void gotoPage(String address,
 HttpServletRequest request,
 HttpServletResponse response)
 throws ServletException, IOException {
 RequestDispatcher dispatcher =
 getServletContext().getRequestDispatcher(address);
 dispatcher.forward(request, response);
}
```

## Reminder: JSP useBean Scope Alternatives

- **request**
  - `<jsp:useBean id="..." class="..." scope="request" />`
- **session**
  - `<jsp:useBean id="..." class="..." scope="session" />`
- **application**
  - `<jsp:useBean id="..." class="..." scope="application" />`
- **page**
  - `<jsp:useBean id="..." class="..." scope="page" />`  
or just  
`<jsp:useBean id="..." class="..." />`
  - This scope is not used in MVC (Model 2) architecture

## Storing Data for Later Use: The Servlet Request

- **Purpose**
  - Storing data that servlet looked up and that JSP page will use only in this request.
- **Servlet syntax to store data**

```
SomeClass value = new SomeClass(...);
request.setAttribute("key", value);
// Use RequestDispatcher to forward to JSP
```
- **JSP syntax to retrieve data**

```
<jsp:useBean
 id="key"
 class="somepackage.SomeClass"
 scope="request" />
```

# Storing Data for Later Use: The Servlet Request (Variation)

- **Purpose**
  - Storing data that servlet looked up and that JSP page will use only in this request.
- **Servlet syntax to store data**
  - Add new request parameters to servlet request

```
String address = "/path/resource.jsp?newParam=value";
RequestDispatcher dispatcher =
 getServletContext().getRequestDispatcher(address);
dispatcher.forward(request, response);
```
- **JSP syntax to retrieve data**
  - No useBean syntax. However, recall that request parameters can be accessed without explicit Java code by means of `jsp:setProperty`.

# Storing Data for Later Use: The Session Object

- **Purpose**
  - Storing data that servlet looked up and that JSP page will use in this request and in later requests from same client.
- **Servlet syntax to store data**

```
SomeClass value = new SomeClass(...);
HttpSession session =
 request.getSession(true);
session.setAttribute("key", value);
// Use RequestDispatcher to forward to JSP
```
- **JSP syntax to retrieve data**

```
<jsp:useBean
 id="key"
 class="somepackage.SomeClass"
 scope="session" />
```

# Variation for Session Tracking

- Use `response.sendRedirect` instead of `RequestDispatcher.forward`
- **Distinctions: with `sendRedirect`:**
  - User sees JSP URL (user sees only servlet URL with `RequestDispatcher.forward`)
  - Two round trips to client (only one with `forward`)
- **Advantage of `sendRedirect`**
  - User can visit JSP page separately
    - User can bookmark JSP page
- **Disadvantage of `sendRedirect`**
  - Since user can visit JSP page without going through servlet first, JSP data might not be available
    - So, JSP page needs code to detect this situation

16

MVC Architecture

www.coreservlets.com

# Storing Data for Later Use: The Servlet Context

- **Purpose**
  - Storing data that servlet looked up and that JSP page will use in this request and in later requests from *any* client.
- **Servlet syntax to store data**

```
synchronized(this) {
 SomeClass value = new SomeClass(...);
 getServletContext().setAttribute("key",
 value);

 // RequestDispatcher forwards to JSP
}
```
- **JSP syntax to retrieve data**

```
<jsp:useBean
 id="key"
 class="somepackage.SomeClass"
 scope="application" />
```

17

MVC Architecture

www.coreservlets.com

# Relative URLs in JSP Pages

- **Issue:**
  - Forwarding with a request dispatcher is transparent to the client. *Original* URL is only URL browser knows about.
- **Why does this matter?**
  - What will browser do with tags like the following:  
<IMG SRC="foo.gif" ...>  
<LINK REL=STYLESHEET  
    HREF="JSP-Styles.css"  
    TYPE="text/css">  
<A HREF="bar.jsp">...</A>
  - Answer: browser treats them as relative to *servlet URL*
- **Simplest solution:**
  - Use URLs that begin with a slash

18

MVC Architecture

www.coreservlets.com

# MVC Example 1: An On-Line Travel Agent

The screenshot displays two browser windows. The left window, titled "Online Travel Quick Search - Microsoft Internet Explorer", shows a search form with the following fields: Email address (joe@somehost.com), Password (XXXXXXXX), Origin (Baltimore), Destination (Los Angeles), Start date (MM/DD/YY) (3/20/00), and End date (MM/DD/YY) (3/25/00). Below the form are icons for a plane, a car, a hotel, and a passport, with buttons for "Book Flight", "Rent Car", "Find Hotel", and "Edit Account". A link "Not yet a member? Get a free account here." is also visible. The right window, titled "Best Available Flights - Microsoft Internet Explorer", shows the results for "Joe Hacker". It lists three flight options: "Java Airways Flight 1522 (\$455.95)", "Servlet Express Flight 2622 (\$505.95)", and "Geek Airlines Flight 3.14159 (\$675.00)". Each option includes "Outgoing" and "Return" flight details. At the bottom, there is a table of frequent flyer numbers and a "Book It!" button.

| Airline      | Frequent Flyer Number |
|--------------|-----------------------|
| Java Airways | 321-9299-J            |
| United       | 442-2212-U            |
| Southwest    | 1A345                 |

Credit Card: JavaSmartCard (XXXX-XXXX-XXXX-3120)  
Hold for 24 Hours  
Book It!

19

MVC Architecture

www.coreservlets.com

# MVC Example 1: An On-Line Travel Agent

- **All requests include**
  - Email address, password, trip origin, trip destination, start date, and end date
- **Original request answered by servlet**
  - Looks up real name, address, credit card information, frequent flyer data, etc., using email address and password as key. *Data stored in session object.*
- **Depending on what button user pressed, request forwarded to:**
  - Page showing available flights, times, and costs
  - Page showing available hotels, features, and costs
  - Rental car info, edit customer data, error handler

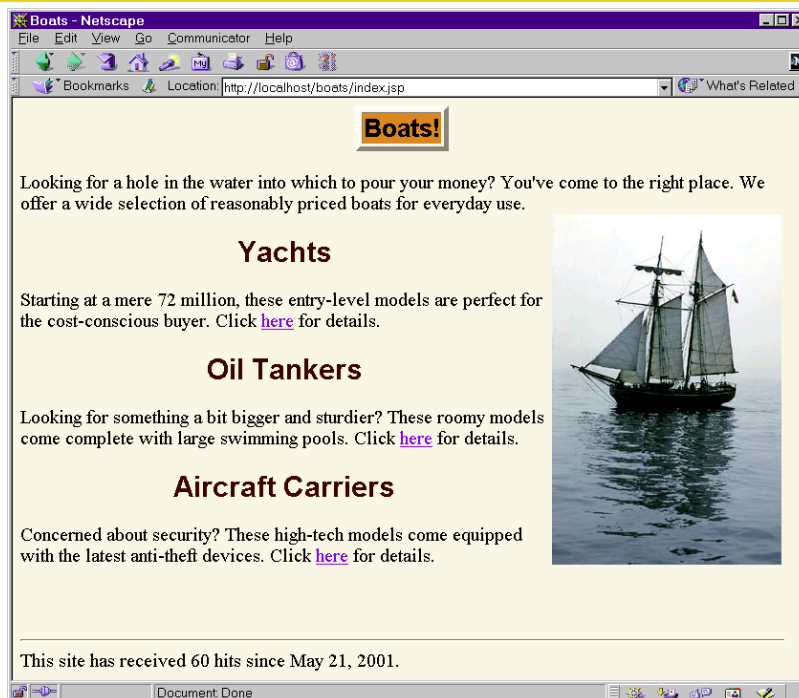
# An On-Line Travel Agent: Servlet Code

```
public void doPost(HttpServletRequest request,
 HttpServletResponse response)
 ...// Store data in TravelCustomer bean called "customer"
 HttpSession session = request.getSession(true);
 session.setAttribute("customer", customer);
 if (request.getParameter("flights") != null) {
 gotoPage("/travel/BookFlights.jsp",
 request, response);
 } else if ...
}
private void gotoPage(String address,
 HttpServletRequest request,
 HttpServletResponse response)
 throws ServletException, IOException {
 RequestDispatcher dispatcher =
 getServletContext().getRequestDispatcher(address);
 dispatcher.forward(request, response);
}
```

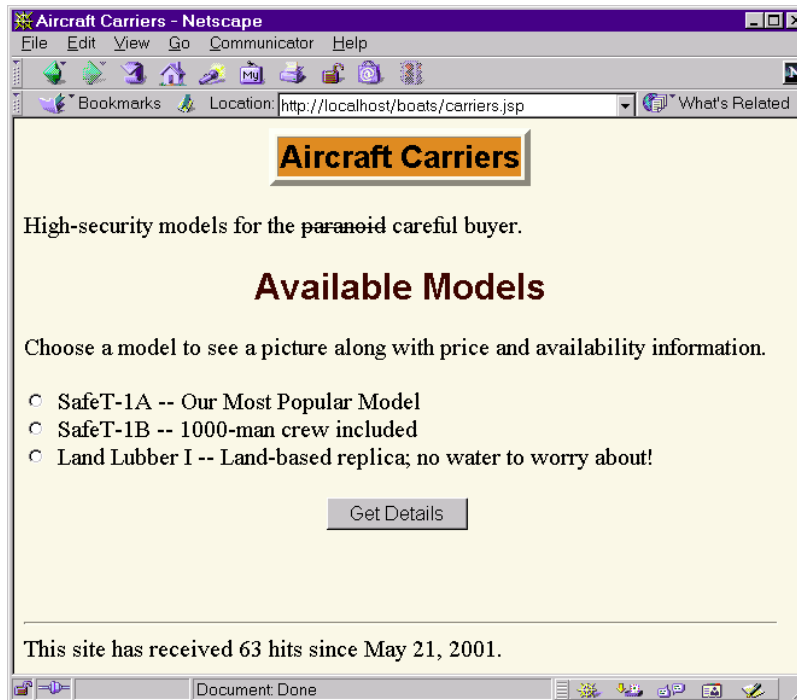
# An On-Line Travel Agent: JSP Code (Flight Page)

```
<BODY>
<H1>Best Available Flights</H1>
<CENTER>
<jsp:useBean id="customer"
 class="coreservlets.TravelCustomer"
 scope="session" />
Finding flights for
<jsp:getProperty name="customer"
 property="fullName" />
<P>
<jsp:getProperty name="customer" property="flights" />
...
```

# MVC Example 2: An Online Boat Store



# MVC Example 2: An Online Boat Store

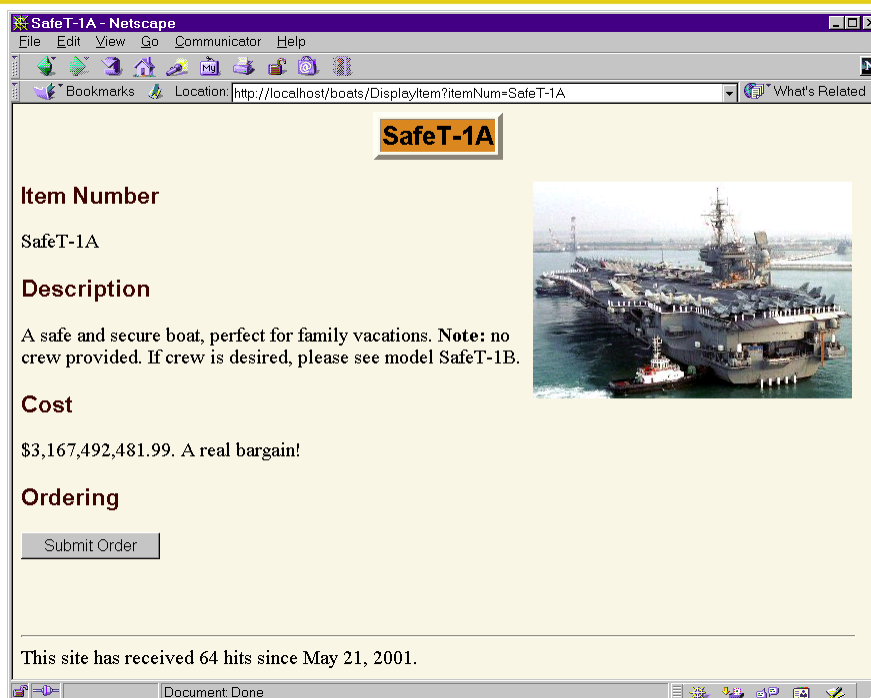


24

MVC Architecture

www.coreservlets.com

# MVC Example 2: An Online Boat Store

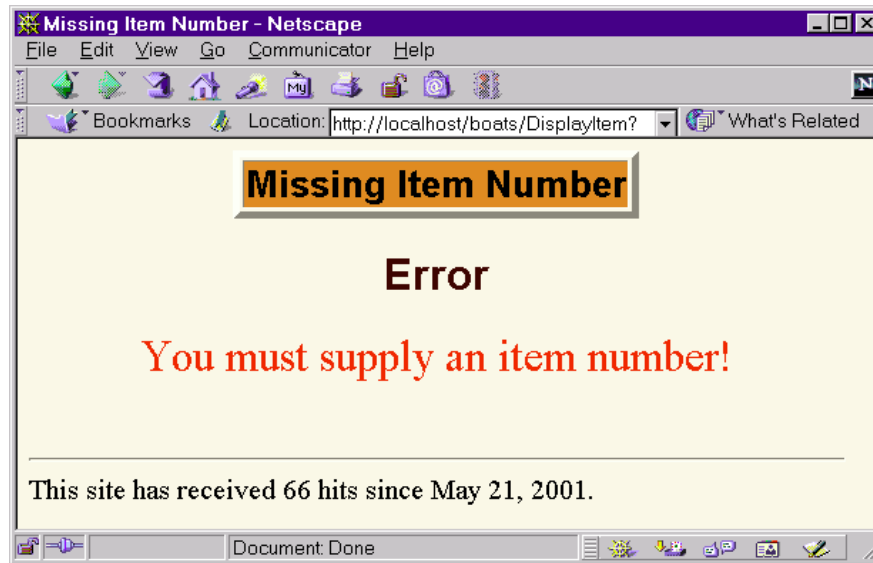


25

MVC Architecture

www.coreservlets.com

# MVC Example 2: An Online Boat Store



26

MVC Architecture

www.coreservlets.com

# MVC Example 2: An Online Boat Store



27

MVC Architecture

www.coreservlets.com

# MVC Example 2: Servlet Code

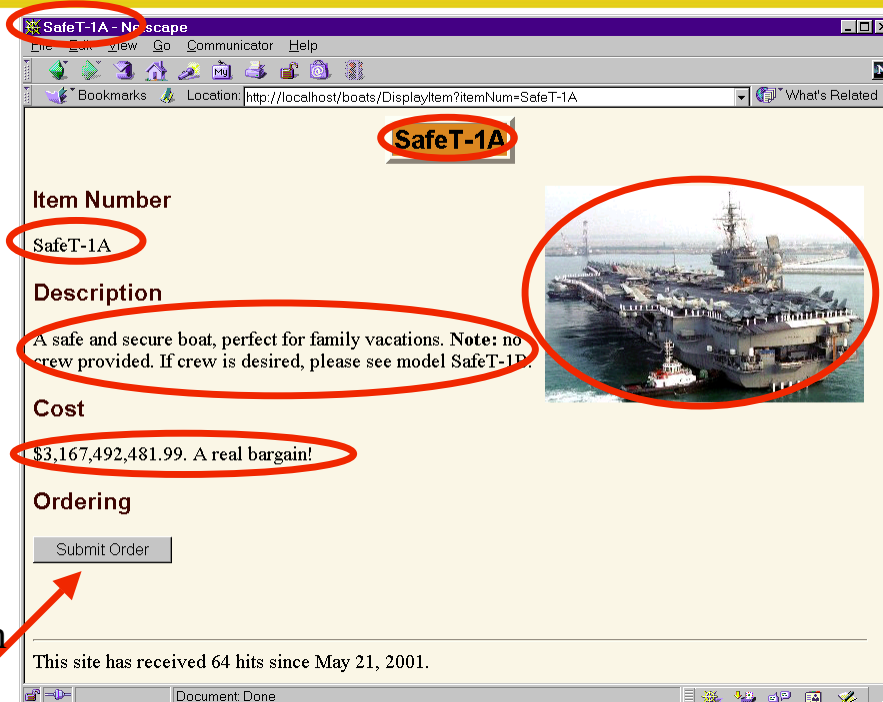
```
public class ShowItem extends HttpServlet {
 public void doGet(HttpServletRequest request,
 HttpServletResponse response)
 throws ServletException, IOException {
 String itemNum = request.getParameter("itemNum");
 String destination;
 if (itemNum == null) {
 destination = "/MissingItem.jsp";
 } else {
 destination = "/ShowItem.jsp";
 ItemTable shipTable = ShipTable.getShipTable();
 SimpleItem item = shipTable.getItem(itemNum);
 request.setAttribute("item", item);
 }
 RequestDispatcher dispatcher =
 getServletContext().getRequestDispatcher(destination);
 dispatcher.forward(request, response);
 }
}
```

28

MVC Architecture

www.coreservlets.com

# MVC Example 2: An Online Boat Store



29

MVC Architecture

www.coreservlets.com

## MVC Example 2: JSP Code (ShowItem.jsp)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
...
<jsp:useBean id="item"
 class="moreservlets.SimpleItem"
 scope="request" />
<TITLE><jsp:getProperty name="item" property="itemNum" />
</TITLE>
...
<TABLE BORDER=5 ALIGN="CENTER">
 <TR><TH CLASS="TITLE">
 <jsp:getProperty name="item" property="itemNum" /></TH></TR>
</TABLE>
<P>
<IMG SRC="<jsp:getProperty name='item' property='imageURL' />"
 ALIGN="RIGHT">

<H3>Item Number</H3>
<jsp:getProperty name="item" property="itemNum" />
<H3>Description</H3>
<jsp:getProperty name="item" property="description" />
```

## MVC Example 2: JSP Code (ShowItem.jsp Cont.)

```
<H3>Cost</H3>
<jsp:getProperty name="item" property="costString" />.
A real bargain!

<H3>Ordering</H3>
<FORM ACTION="DisplayPurchases">
 <INPUT TYPE="HIDDEN" NAME="itemNum"
 VALUE="<jsp:getProperty name='item'
 property='itemNum' />">
 <INPUT TYPE="SUBMIT" VALUE="Submit Order">
</FORM>

<%@ taglib uri="/WEB-INF/tlds/count-taglib.tld"
 prefix="boats" %>
<boats:count />
</BODY>
</HTML>
```

## MVC Example 2: Bean Code (SimpleItem.java)

```
public class SimpleItem {
 private String itemNum = "Missing item number";
 private String description = "Missing description";
 private String imageURL = "Missing image URL";
 private double cost;
 private NumberFormat formatter =
 NumberFormat.getCurrencyInstance();

 public SimpleItem(String itemNum,
 String description,
 String imageURL,
 double cost) {
 setItemNum(itemNum);
 setDescription(description);
 setImageURL(imageURL);
 setCost(cost);
 }

 public SimpleItem() {}
}
```

32

MVC Architecture

www.coreservlets.com

## Forwarding Requests from JSP Pages -- jsp:forward

- You usually forward from a servlet to a JSP page, but you can also forward from JSP

```
<% String destination;
 if (Math.random() > 0.5) {
 destination = "/examples/page1.jsp";
 } else {
 destination = "/examples/page2.jsp";
 }
%>
<jsp:forward page="<%= destination %>" />
```

- Question: can you forward from a servlet to another servlet? How do you know?

33

MVC Architecture

www.coreservlets.com

# Including Pages Instead of Forwarding to Them

- **With the forward method of RequestDispatcher:**
  - Control is *permanently* transferred to new page
  - Original page *cannot* generate any output
- **With the include method of RequestDispatcher:**
  - Control is *temporarily* transferred to new page
  - Original page *can* generate output before and after the included page
  - Original servlet does not see the output of the included page (for this, see later topic on servlet/JSP filters)
  - Useful for portals: JSP presents pieces, but pieces arranged in different orders for different users

# A Servlet that Shows Raw Servlet and JSP Output

```
out.println(...
 "<TEXTAREA ROWS=30 COLS=70>");
if ((url == null) || (url.length() == 0)) {
 out.println("No URL specified.");
} else {
 // Attaching data works only in version 2.2.
 String data = request.getParameter("data");
 if ((data != null) && (data.length() > 0)) {
 url = url + "?" + data;
 }
 RequestDispatcher dispatcher =
 getServletContext().getRequestDispatcher(url);
 dispatcher.include(request, response);
}
out.println("</TEXTAREA>\n" +
 ...);
```

# A Servlet that Shows Raw Servlet and JSP Output

The image shows two side-by-side screenshots of a Netscape browser window. The left window is titled "Viewing JSP and Servlet Output - Netscape" and contains a form with the following text: "Enter a relative URL of the form /path/name and, optionally, any attached GET data you want to send. The raw HTML output of the specified URL (usually a JSP page or servlet) will be shown. Caveats: the URL specified cannot contain the string </TEXTAREA>, and attached GET data works only with servlet engines that support version 2.2." Below this text are two input fields: "URL: /JSP/Expressions.jsp" and "GET Data: testParam=some+data". A "Show Output" button is positioned below the input fields. The right window is titled "/JSP/Expressions.jsp - Netscape" and displays the raw HTML output of the JSP page. The HTML code includes a DOCTYPE declaration, a HEAD section with meta tags for title, author, keywords, and description, and a BODY section with an H2 heading "JSP Expressions" and a list of three items: "Current time: Fri Mar 17 10:53:36 EST 2000", "Your hostname: 127.0.0.1", and "Your session ID: To1010mC06704059939219231At". The list item for the form parameter is rendered as "The <CODE>testParam</CODE> form parameter: some data".

36

MVC Architecture

www.coreservlets.com

## Summary

- **Use MVC (Model 2) approach when:**
  - One submission will result in more than one basic look
  - Several pages have substantial common processing
- **Architecture**
  - A servlet answers the original request
  - Servlet does the real processing & stores results in beans
    - Beans stored in HttpServletRequest, HttpSession, or ServletContext
  - Servlet forwards to JSP page via forward method of RequestDispatcher
  - JSP page reads data from beans by means of jsp:useBean with appropriate scope (request, session, or application)

37

MVC Architecture

www.coreservlets.com