

CS290i - Lecture 6

Servlets, Webmacro, JSP

Scalable Internet Services and Systems, Winter 2002

Thorsten von Eicken
Department of Computer Science
University of California at Santa Barbara

Java Servlets

■ What are servlets ?

- Java objects which extend the functionality of an HTTP server
- Comparable to Netscape's NSAPI, Microsoft's ISAPI, or Apache Modules
- Platform independent
- Server Independent

■ Servlets vs. CGI

- Servlets do not require a new process for each request
- Servlets stay loaded between requests
- Same servlet can handle many concurrent requests
- Servlets can be *sandboxed* to reduce danger on server

2

Hello World! Servlet

```
1: import java.io.*;
2: import javax.servlet.*;
3: import javax.servlet.http.*;
4:
5: public class HelloClientServlet extends HttpServlet
6: {
7:     protected void doGet(HttpServletRequest req,
8:                           HttpServletResponse res)
9:         throws ServletException, IOException
10:    {
11:        res.setContentType("text/html");
12:        PrintWriter out = res.getWriter();
13:        out.println("<HTML><HEAD><TITLE>Hello Client!</TITLE>" +
14:                  "</HEAD><BODY>Hello Client!</BODY></HTML>");
15:        out.close();
16:    }
17:
18:    public String getServletInfo()
19:    {
20:        return "HelloClientServlet 1.0 by Stefan Zeiger";
21:    }
22: }
```

3

Servlet Lifecycle

■ Lifecycle

- Constructor (no args)
- Init(ServletConfig config)
 - ♦ Init and retrieve any config info
 - ♦ Called only once
- Service(ServletRequest req, ServletResponse res)
 - ♦ Called for each request, must be thread safe
 - ♦ Not HTTP specific
- Destroy()
 - ♦ To load a new version or shutdown
 - ♦ Must be thread-safe (requests may still be running)



■ HttpServlet

- Service dispatches to doXXX
 - ♦ (doGet, doPost, doPut, doDelete, ...)
 - ♦ HEAD -> doGet and discard output
- HttpServletRequest, HttpServletResponse

4

HTTP Request Object

- Encapsulates all info from client
- Access to request headers
 - ◆ String `getRequestURI()`;
 - ◆ Enumeration `getHeaderNames()`;
 - ◆ String `getHeader(String headerName)`;
 - ◆ `HttpSession getSession()`;
 - ◆ `Cookie[] getCookies()`;
- Access to `InputStream` or `Reader`
 - ◆ `Reader`: use for text, handles Unicode
 - ◆ `Stream`: use for binary data
- Access to CGI-like info
 - ◆ String `getRemoteAddr()`;
- Access to form data

5

HTTP Response Object

- Encapsulates all communication to client
- Access to response headers
 - ◆ `setContentType(String type)`;
 - ◆ `setContentLength(int length)`;
 - ◆ `setStatus(int statusCode)`;
- Access to `OutputStream` or `Writer`
 - ◆ `PrintWriter getWriter()`;
- Access to setting cookies
 - ◆ `addCookie(Cookie c)`;
- Convenience methods for redirects, errors, etc.
 - ◆ `sendError(int statusCode)`;
 - ◆ `sendRedirect(String url)`;

6

Session Tracking

■ Session:

- Series of requests made by a client during a time period
- Tracked & abstracted by unified layer
- Use cookies, URL rewriting, or SSL

■ Mechanics

- `HttpSession s = req.getSession(true)`;
 - ◆ Get hashtable (Dictionary) containing session state
 - ◆ "true" -> create new session if necessary
 - ◆ May change response headers (e.g. cookie)
- Boolean `s.isNew()`;
- `s.putValue(String key, Object value)`;
- Object `s.getValue(String key)`;
- `Res.encodeUrl(String url)` -> add session ID

7

More...

■ Inter-servlet communications

- Redirect to another servlet
- Include another servlet's response
- Access static documents
- Sharing data through `ServletContext`

■ Logging

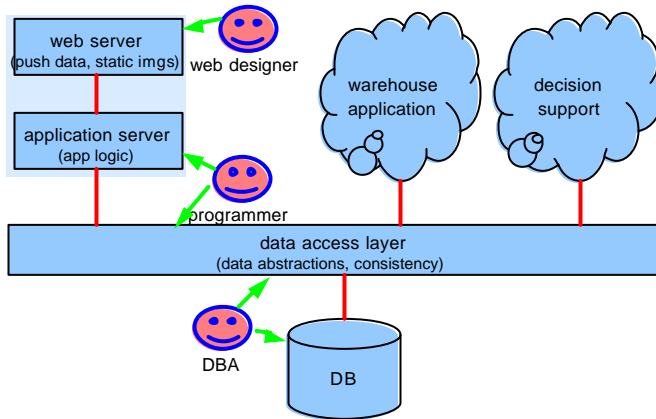
- `Log(String msg)`;
- `Log(String msg, Throwable t)`;

■ Versions

- We will use the Servlet 2.0 API

8

System Structure



9

Separation of work

- Web site designer
 - ◆ Designs overall look & feel of web site
 - ◆ What pages contain, how to get from one to another
 - ◆ Codes HTML & creates GIFs
 - ◆ Doesn't do Java, doesn't compile, install, ...
 - ◆ Doesn't want to talk to a programmer for every change
- Programmer
 - ◆ Develops engine driving web site operation
 - ◆ What needs to happen for each request
 - ◆ Where to get the data for a response
 - ◆ Writes Java, deals with SQL
 - ◆ Doesn't do HTML, can't draw GIFs
 - ◆ Doesn't want to search for code among pages of HTML
- DBA (DataBase Administrator)
 - ◆ Designs structure of database
 - ◆ Tells programmers what queries to use

10

Model/View/Controller

■ Model:

- Back-end database, "hard-core code"
- Business logic (catalog pages, shopping cart, etc.)
- (Often two distinct layers)

■ View:

- Look & feel of site, mostly design, little code
- Page layout & content

■ Controller:

- Deciding which view to show (language, input validation, ...)
- Tracking users, keeping session data
- Logging

11

JSP - JavaServer Pages

Welcome back **Thorster!**
Your account balance is
\$1,000,000.00. How would
like to spend it?

Web page

```

<font size=big>Welcome  
back user.getName()!  
</font>  
<br>Your account  
balance is  
$account.getBalance().  
How would like to spend  
it?
  
```

JSP

```

public class WelcomeServlet
  extends HttpServlet
{
  protected void doGet(...)
  {
    HttpSession s = req.getSession();
    User user = s.getValue("user");
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML><HEAD><TIT...");
    out.println("Welcome back ");
    out.println(user.getName());
    ...
    out.close();
  }
}
  
```

Servlet

12

JSP - Summary

■ Embed Java into HTML

- Thanks for ordering
<I><%= request.getParameter("title") %></I>,
we will ship the item tomorrow.

■ Translate JSP pages to Servlets

- Done automatically the first time a page is requested
- HTML text becomes print statements

■ Control over code placement

- <%= expr %>: expression evaluated and placed in output
- <% code %>: scriptlet inserted in service method
- <%! code %>: code inserted in body of servlet class
- <%@ page att="val">: directive for servlet engine
- <%@ include file="url" %>
- <jsp:useBean att="val" />: find or build a JavaBean
- ...

13

WebMacro Example

- Customer order history page
- #set ContentType = "text/html"
<HTML><HEAD><TITLE>\${Customer.Name}</TITLE></HEAD>
<BODY bgcolor='white'>
<h1>\${Customer.Name}: History</h1>
Here's a list of your orders since
\${Customer.Orders.StartDate}:
<table width='70%'>
<th><td>Order Date</td>
<td>Item Requested</td>
<td>Number of Units</td></th>
#foreach \$order in \${Customer.Orders} {
<tr><td>\${order.Date}</td>
<td>\${order.Item.Name}</td>
<td>\${order.Number}</td> </tr>
}
</table></body> </html>

14

Example (cont.)

■ Servlet

- Place objects accessed by template into a hashtable
 - ◆ Context.put("myVar", myData);
- Determine which template to use
 - ◆ return getTemplate("helloWorld.wm");
- WebMacro inspects objects and substitutes values
 - ◆ Uses Java Introspection API
- For example WebMacro might determine:
 - ◆ \${Customer.Name} corresponds to Customer.getName()
 - ◆ \${order.Item.Name} resolves to Order.getItem(String name)
- It is possible to call methods explicitly:
 - ◆ \$Accounting.findAccount(\$Customer)
- WebMacro also supports indexed properties:
 - ◆ Java: public String[] myData.getNames();
 - ◆ WM: #foreach \$name in \$myVar.Names { Hello \$name! }

15

WM HelloWorld

- import org.webmacro.*;
import org.webmacro.servlet.*;
public class HelloWorld extends WMServlet
{
 public Template handle(WebContext context)
 throws HandlerException
 {
 context.put("Hello", "Hello, brave new world!");
 Template view;
 try {
 view = getTemplate("helloWorld.wm");
 } catch (ResourceException e) {
 throw new HandlerException("HelloWorld failed!\n");
 }
 return view;
 }
}
- <html><head><title>Hello, World!</title></head>
#set \$Response.ContentType = "text/html"
WebContext.getResponse().setContentType("text/html")
<h1>WebMacro is working!</h1>
\${Hello}
</body></html>

16

Model/view/controller

■ Model

- Encapsulate back end data; keep it separate from the servlets
 - ◆ Often two distinct layers itself: data and logic
- Java Beans, or similar abstractions

■ View

- JSP: html page with a little Java sprinkled in
- WebMacro: templates with variable substitutions & loops

■ Controller

- Write a servlet that reads the client request, accesses the back end, and fills in the data needed by the template
- WebMacro provides hooks for writing controllers

17

JSP critique

■ Comparing WebMacro to JavaServer Pages

- The Problems with JSP, Jason Hunter

■ It is too tempting to insert Java code in web pages

- A template engine can enforce good design
- WebMacro templates cannot contain Java code
- simple script language aimed at page designers, with access to all of the underlying data

■ Some tasks can only be done using Java code within the page

- Trivial for developers (?), but tricky for web page designers
- WebMacro contains built-in variables covering requests, responses, sessions and cookies

18

JSP critique (cont.)

■ Some simple tasks are made overly difficult

- E.g including files, search paths
- WebMacro makes including headers and footers simple.

■ Looping is unnecessarily difficult

- HTML > JAVA > HTML
- WebMacro's looping directives are less clumsy than JSP-specific tags

■ JSP page syntax errors are cryptic and often useless

- JSP->Java translation causes errors to appear in Java
- WebMacro errors are more meaningful, and a logging capability is provided.

19

JSP critique (cont.)

■ Unless JSPs are pre-compiled, the web server will need to be equipped with a compiler

- Compile on-the-fly is nice, but not necessarily in production
- WebMacro classes are compiled, and templates are pre-parsed

■ JSPs consume both extra hard disk and extra memory space

- Template engines such as WebMacro don't need to duplicate the page data into a second file, so hard drive space is spared.
- Template engines also give the programmer full control over how templates are cached in memory.

20

Template engine critique

- **Where's the spec?**

- Often the implementation is the spec

- **Not widely known**

- Harder to find help & people

- **Performance not well-tuned**