Web Development using Java, JSP, and Web Services

JSP Web Development

Lecture #5 2008

1. Requests
   - CGI
   - JSP
   - Cookies
   - Sessions
2. Responses
   - MIME Types
   - JSP
3. Client-Side Components
   - ActiveX Components
   - .NET Components
   - Java Applets

Request Methods

- GET - data in URL-encoded name/value-pairs
- POST - data encoded in HTTP request body
- PUT - data stream in HTTP request body

Common Gateway Interface (CGI)

- The original way to send dynamic data to web servers
- Data delivered URL-encoded in specific parameters
- Scripts located and invoked by the web server on demand
- Data parsing was difficult and error-prone
- Serious security issues

URL Encoding

- A way to escape data in text form
- Reserved and un-recognized characters are hex-coded
- Used to encode data-carrying URLs

Character	URL Encoding
A-Z, a-z, 0-9	not escaped
$-_.+!*'(),	not escaped
others	hex-coded
e.g., (whitespace)	%20

CGI variables

- The original way to send dynamic data to web servers
- Data delivered URL-encoded in specific parameters
- Still available to web servers
- Contains useful metadata
  (such as client IP, username, length of data stream etc)

The Request Object

- Represents the HTTP request
- Contains all info in the request
- Exposes an API for traversing request data
- Performs URL-decoding of data
- Provides uniform ways to read data regardless of method

Reading Parameters

- request.getParameter("name")
- request.getParameterValues("name")
- request.getParameterNames()
- request.getParameterMap()

Reading Parameters

- Always check for null
- Always provide a default value (as appropriate)

Parameter	getParameter() returns
not present in request	null
contains empty string	empty string

Model View Controller

- Model represents data (JSP or Servlet)
- View represents interfaces (JSP)
- Controller manages state and control flow (Servlet)

Code Reuse

- Declare reusable functions in JSP fragments
- Use include directives to import them into JSP
- Very useful for small utility functions

Raw Data

- Can access (PUT) data streams directly
- request.getReader() - gets a character reader
- request.getInputStream() - get a binary input stream
- NOTE: request data no longer available via getParameter()
- Usually used for file upload scenarios
- Apache Jakarta common library contains utilities for accessing raw data streams
- Plenty of third party file upload Servlets available

Filtering Data

- Data validation necessary
- Always check data size
- Always check for illegal characters
- Consider site impact of invalid data

```
<    Escape to &lt;    (XML, XHTML, HTML)
>    Escape to &gt;    (     - " -      )
&    Escape to &amp;   (     - " -      )
"    Escape to &quot;  (     - " -      )
'    Escape to \'      (   databases    )
```

Cookies

- HTTP connections may be closed at any time
- Need a way to identify requests from the same client
- Cookies are small text fragments sent in headers
- Cookies are stored on client computer file systems and are used to identify site visitors (track users)

Browsers & Cookies

- Maximal cookie size 4096 bytes
- Max 20 cookies per site
- Max 300 cookies in total
- Stored on client computer
- Cookies can be blocked by browsers
- Cookies can be removed from client computer
- Cookies can be altered
- Don't store sensitive data in cookies
- Use cookies, but don't depend on them

Sessions

- Data contexts stored on the web server and shared between requests from the same user
- Can store anything (POJO)
- Can be serialized to databases or short-lived
- Need a way to identify session for new requests
  - Cookies
  - URL rewrites
  - hidden form fields
- The JSP session API hides the session identification

Responses

- May be sent in one chunk
- May be sent incrementally

MIME Types

- Instructs the client how to interpret data returned
- Defaults to text/html is JSP
- Usually used for binary data
- NOTE: MIME type must be set before any data is sent

MIME Type Examples

MIME type	data type
text/css	HTML cascading style sheet
text/html	HTML (JSP default)
text/plain	plain text
application/pdf	PDF document
application/postscript	Postscript document
application/zip	ZIP file
application/octet-stream	binary data
image/gif	GIF image
image/jpeg	JPEG image
image/png	PNG image
audio/x-wav	Microsoft Windows sound
video/mpeg	MPEG video
video/quicktime	Quicktime video

JSP Processing

1. A JSP page is requested
2. Server checks if a Java Servlet for the page exists
3. If no Servlet is found (or newer JSP is detected), the JSP is translated to Java (a Servlet class is created)
4. The Java Servlet is compiled
5. The Java Servlet is invoked and processes the request

JSP Translation

- JSP declarations become
  - Servlet class members
  - Servlet class methods
- JSP scriptlets become
  - local variables in the Servlet `service()` method
- JSP expressions become
  - `out.print()` calls in - " -

Client-Side Components

- Compiled programs that execute on the client
- Downloaded from the server when needed
- Reduces client installation needs
- Description of the component part of the HTML
- Useful for providing advanced user interfaces
- Plenty of third party components available
- Commercial markets for special purpose components

ActiveX Components

- COM-components exposing user interfaces
- Commonly used in ASP environments
- No code validation performed
- Can be written in any MS language
- Distributed as Dynamic Link Libraries (DLLs)

.NET Components

- .NET-components exposing user interfaces
- Commonly used in ASP.NET environments
- Can be used interchangibly with (D)HTML interfaces
- Can be developed using any .NET language
- Strong development environment (Visual Studio)
- Supports code signing
- Distributed as Dynamic Link Libraries (DLLs)

Java Applets

- Java objects extending the Applet class
- Can be used in any HTML environment
- Developed in Java
- Contains a sandbox security model
- Supports code signing
- Distributed as standalone classes or JARs

Java Applets

- Extend `Applet`
  - creates a AWT container
  - functions as a Frame for your applet
- Extend `JApplet`
  - creates a heavy-weight Java Swing container
  - functions as a JFrame for your applet
- Assure thread safe implementations of applets (invoke worker threads via `SwingUtilities.invokeAndWait()`)

Applet Lifecycle

1. `init()`
   - performs applet initialization
   - called after the applet has received its parameters
2. `start()`
   - activates the applet
   - called after init() and whenever an applet page receives focus
3. `stop()`
   - deactivates the applet
   - called when an applet page looses focus
4. `destroy()`
   - deinitializes the applet
   - called when the browser is shut down

Applet Restrictions

- Applets cannot load libraries or define native methods
- An applet cannot ordinarily read or write files on the host that is executing it
- An applet cannot make network connections except to the host that it came from
- An applet cannot start any program on the host that is executing it
- An applet cannot read certain system properties
- Windows that an applet brings up look different than windows that an application brings up

Each browser has a SecurityManager object that implements its security policies. When a SecurityManager detects a violation, it throws a SecurityException. Your applet can catch this SecurityException and react appropriately.

Applet Capabilities

- Applets provide a way to create non-web GUIs to web applications
- Applets can make network connections to the host they came from
  `String host = getCodeBase().getHost();`
- Applets running within a Web browser can easily cause HTML documents to be displayed
  `getAppletContext().showDocument(url,browsertitle)`
- Applets can invoke public methods of other applets on the same page
- Applets that are loaded from the local file system (from a directory in the user's CLASSPATH) have none of the restrictions that applets loaded over the network do

Applet Checklist

- Removed or disable debugging output
- Does the applet stop running when it's offscreen? (should it? it may be application dependent)
- Can the applet stop the annoying behavior? (rethink behavior that can be perceived as annoying)
- Make your applet as flexible as possible
- Make your applet accessible
- Implement the `getParameterInfo()` method
- Implement the `getAppletInfo()` method

Applets as Applications

```
PokerClockApplet applet = new PokerClockApplet();

JFrame frame = new JFrame("Poker Clock Applet");
frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
frame.getContentPane().add(applet,BorderLayout.CENTER);
applet.init();
applet.start();
frame.pack();
frame.setVisible(true);
```

Next Time

- Custom Tag Libraries