Programming Servlets

Presentation by Rama Nagesh Mukkamala
IWS II

Overview

- A Basic Servlet
  * Handling a Request
  * SingleThreadModel
  * ServletRequest
  * ServletResponse
  * ServletConfig
  * ServletContext
  * GenericServlet

- Running & Hosting Servlets

- HTTP Servlets
  * HttpServletRequest
  * HttpServletResponse
  * HttpSession
  * Cookies
  * MultiThreadedServlets
  * Servlets & Applets
Overview

- Listeners
- Filters
- Designing Servlets
- Debugging Servlets
- Performance Tuning

A Basic Servlet

Server associates servlets with
URL
A category of URLs (like .jsp files)

Any Servlet will have to implement the Servlet Interface
Methods listed in Table7.1.rtf

First User Access of that URL triggers
  - Instantiation of servlet object
  - Notification of servlet of its loading (Message to init)
  - Execution of the service() method

Subsequent accesses triggers
  - Just the execution of service() method on the same servlet object.


A Basic Servlet

The package that contains the Servlet also defines

* ServletException
  * UnavailableException

    Thrown if init() fails
    - Temporary
    - Permanent

UnavailableException Methods are listed in
Table7_2.doc

Handling a Request

Service method is the interface for working with the servlet.

When a request arrives for a particular servlet
  the Web server encapsulates the request and creates response objects
  before calling the service method.

The code found in sample1.rtf illustrates the classic Hello world servlet.

The servlet designer can use this service method as an interface
to accomplish any complex task that java supports.
SINGLE THREAD MODEL

To mark a servlet as a single Thread Model, implement the SingleThreadModel.

This interface does not have any methods but it is rather a typing mechanism.

When this is chosen the server guarantees that service method is never called from more than one thread.

Issues that may influence this decision,

  * Does it make sense to use MultiThreads or not
  * Does servlet alter information that may not be thread safe.
  * Performance implications

Marking a servlet as SingleThreadModel is easier on the program, but it may be limiting in flexibility, design and performance.

ServletRequest

Call to service() includes an object that implements the ServletRequest interface.

This request object comes with the encapsulated information about:

  * client/server environment
  * raw access to the client request
  * processed versions of the client request

All the methods for ServletRequest are listed in Table 7.3.doc

Servlets designed

  - to handle HTML forms will use getParameterValue(), getParameterValues() methods
  - to handle text strings will use getReader()
  - to handle binary input will use getInputStream()
ServletResponse
Server provides the servlet with an object that implements the ServletResponse interface in each call to the service.

This object provides methods for sending a processed response to the user using MIME-typed message.

The methods provided by ServletException are listed in Table7.5.doc

It is imperative that we set the content type & or locale before preparing or composing any output to the client.

ServletConfig
init() is passed an object that implements ServletConfig interface.
This ServletConfig object is designed to give information to servlet about the server
* Access to the configuration parameters defined on installation
* Access to ServletContext
The methods of the ServletConfig interface are listed in Table7.6.doc

ServletContext
• Provides information about servlet environment
• Accessed through ServletConfig obj.’s getServletContext() 
• Bears one-to-one relation with Web Appl, providing access to WebApp’s resources 
• The methods available in this interface are listed in Table7.7.doc

Generic Servlet
• Provides a quick way to create a servlet
• Implements Servlet Interface
• Provides many convenient methods as listed in Table7.8.doc
• Many of these methods are covers for ServletConfig & Context objs.
• They function by simply forwarding messages to respective objs.
Running & Hosting Servlets

- Earlier each server had own mechanisms for installing and running servlets
- Now Sun standardized this opn. & servlets belong to a Web Application.
- Web App. Is a grouping mechanism that contain resources like
  * Servlets
  * Java Server Pages
  * Utility Classes
  * Static docs. (html, images, sounds etc)
  * Client-side applets, beans, and classes
  * Descriptive meta information that describes above elements
- A Container is an application that provides the context in which servlets and other web app components run. It provides one-to-one correspondence between the WebApp and a ServletContext.
- Web App provides its resources in a hierarchy of directories.
- A special non public directory called WEB-INF exists in this hierarchy.

Running & Hosting Servlets ...contd

- WEB-INF is organized as follows
  /WEB-INF/web.xml contains deployment descriptor
  /WEB-INF/classes/* contains servlets and utility classes
  /WEB-INF/lib/*.jar contains archived servlets, beans and other utilities
- The Web App’s class loader loads classes from the WEB-INF/classes first and then from the library JARs in the WEB-INF/lib directory
- The deployment descriptor in WEB-INF/web.xml can contain
  * ServletContext init parameters
  * Session Configuration
  * Servlet / JSP definitions
  * Servlet / JSP mappings
  * MIME – type mappings
  * Welcome File list
  * Error pages
  * Security restrictions
HTTP Servlets

- Are designed to provide access to HTTP-specific information.
- Are used for handling HTTP protocol exclusively
- Can be created by sub-classing HttpServlet in javax.servlet.http package which itself is a specialization of GenericServlet class.
- This specialization adds many methods particularly for variety of request types
- HttpServlet Methods are listed in Table7.10.doc
- Most of these methods will report a HTTP_BAD_REQUEST error by default if it not coded in the servlet’s implementation.
- javax.servlet.http library provides a class HttpUtils to support Http style servlets
- HttpUtils methods are listed in Table7.11.doc

HttpServletRequest

- Implements the ServletRequest interface discussed earlier
- Class defn. adds methods that provide specific HTTP information.
- HttpServletRequest Methods are listed in Table7.12.doc

HttpServletRequest ... contd.

- Also adds functionality for managing Session information which will be discussed further down.
- The sample HttpServlet coded in PrintEnvServlet.java shows how a servlet can access headers and parameters from the request and display them back to the client.
- The html page coded in PrintEnvTester.html can be used to access above servlet.

HttpServletResponse

- Like its counterpart above this object provides HTTP-specific access to the servlet’s response object.
- The additional functionality includes
  - Ability to set set headers and Cookies
  - Encoded URLs
  - Send Error codes
  - Redirect the client to another URL
- The methods of this object are listed in Table7.13.doc
When a servlet is not able to deliver expected contents, it can use error codes / status codes to notify the client.

The important status codes available for this object as static variables are listed in Table7_14.doc. The complete list is available in javax.servlet documentation.

The primary aspects of design and deployment of a HTTP servlet are demonstrated in the sample code for AdRotatorServlet, coded by the authors at AdRotatorServlet.java.

This sample servlet’s salient features are
* It implements Single Thread Model
* Uses Vector as primary data structure to hold image files objs.
* Class variable curIndex is used to keep track of current image to display.
* The vector is populated from the image directory parsing for the extensions .jpeg and .gif., when servlet is initialized in init() method.

A session is a single continuous interaction with a particular user.
Actual implementation of session tracking is server dependent.
* One common way is to allot an ID to each session and create a hash table
  * If you have to handle large no. of open sessions the data may be serialized and stored in a normal disk.
* You as a programmer can access a session via HttpSession obj. and manipulate or query it using the methods in Table7_15.doc.
* Always the current HttpSession is acquired from the HttpServletRequest using the method getSession.
* There are two ways of notifying clients of the session’s existence
  * Using Cookies where the browser supports them
  * Send session ID as part of an encoded URL.
* Use of HttpSession object and its methods are demonstrated by the authors in the sample code entitled SessionInfoServlet.java.
* In the very beginning of the doGet() method, session obj. is queried to see if it is new, using isNew(). Accordingly the sessioninfo.count is set / acquired & incremented, before composing the response obj.
**Cookies**

- Sessions are transient objects meant for temporary usage.
- Cookies are key-value pairs that a servlet can associate with a client on a semi-permanent basis. They can be set to have arbitrary life time, and persist between sessions.
- Though Cookies are easy to work with and program, they have drawbacks like:
  * They are available only in form i.e. strings
  * They are stored on client and take client’s disk space
  * They are passed over the internet, adding to the traffic density
- Cookie obj.'s methods are listed in Table7.16.doc
- In HTTP cookies are part of the header for request and reply.
- To access cookies for a request, use the HttpServletRequest obj’s getCookies()
- To associate cookies with response, use the HttpServletResponse method addCookie().
- The use of the Cookies is demonstrated by the authors in the sample code entitled CookieCounterServlet.java

**Multithreaded Servlets**

- Are those that allow the service() method to be called by more than one thread at the same time, ensuring that any data accessed in the call is thread safe.
- Two basic approaches are:
  * Design the service to be independent of the other calls. Typical of servlets that work an algorithmic task without returning data from any shared source, and working entirely with local variables.
  * When use of shared resources like files or databases is inevitable, use synchronization to protect the integrity of the shared resources.
- The authors have demonstrated the use of FileLock obj to ensure that files are protected, from multiple user access.
- They have also coded a sample servlet entitled FileLockingServlet
- Another approach for handling shared resources is to use a cache. Ideal for situations where such resources are accessed only for reading and not for changing
Servlet and Applets

- Though Servlets are a powerful mechanism to serve dynamic web pages, Applets still continue to hold their place as they can run on most browsers making them a portable client choice.
- Total application is often split into applets/servlet pairs for reasons like
  * performance
  * minimizing network traffic
  * centralizing business logic
  * portability/accessibility
- Applets can use std. HTTP to communicate with servlet.
- Servlets designed to talk to applets can optimize communication traffic by using text/binary data instead of HTML
- All client centric processing like presentation and validation of forms etc should be offloaded to applets, while all business logic / data centric processing should be handled in servlets for optimum results

Listeners

- Listeners are objects that receive notification events from the servlet context
- Version 2.3 of Servlet spec. supports four types of listener interfaces
  * javax.servlet.ServletContextListener
  * javax.servlet.ServletContextAttributesListener
  * HttpSessionAttributesListener
  * HttpSessionListener
- Listeners can implement any or all of the above, and are designed to listen to changes in the context / its attributes, a session / its attributes
- ServletContextListener defines two methods:
  * public void contextInitialized(ServletContextEvent sce);
  * public void contextDestroyed(ServletContextEvent sce);
  ServletContextEvent contains method
  public ServletContext getServletContext() 
- ServletContextAttributeListener has three methods
  attributeAdded, attributeRemoved, and attributeReplaced, all of which take
  ServletContextAttributeEvent obj as the only parameter, which in turn methods
  String getName() and Object getValue
- To integrate these listeners in the Web App. Include the listener tags in deployment descriptor
Filters

- Filters are s/w code that modify or adapt the Http Request / Response to suit the requirements of a particular web resource in the appl.
- Are meant to customize requests and responses by wrapping the real ones
- Possible uses of filters include
  * Authentication
  * Logging and auditing
  * Image Conversion
  * Data Compression
  * Encryption
  * Triggering Resource access events etc
- Filters provide powerful means for creating general constraints in a Web Appl. Without having to implement repetitive code in every servlet
- To create them use the Filters Interface’s doFilter method.

Designing Servlets

- First analyze your application and break it up into modules
- Assign the Servlets only to truly dynamic pages, i.e. pages that need to change with each request
- Servlets should be assigned by page / task types not for every page.
- Have the service method call other methods as needed based on request type.
- There are a number of simple thumb rules to follow; Dos and Don’ts.
  * Do not assume anything. Double check all necessary assumptions
  * Validate all data
  * Do not run Web server as root.
  * Follow need to know basis, Do not give away more than necessary
  * Try to use only proven encryption algorithms.
  * Avoid native code and use 100 % java in servlets
  * Use timeouts where ever possible
- After listing 16 such rules the authors give a warning that this list is incomplete and would probably never be.
Debugging Servlets

Issues at hand are

- To find suitable debuggers
- Servlets may not be subject to same load and stress as in real time server
- Portability
- Approaches to Debugging

Log style Debugging is time consuming. Logging is done purely for administrative purposes and not debugging.

- Till good debugging tools are developed Log style debugging fills the gap
- Visit [www.javaworld.com](http://www.javaworld.com) to find tools that might help in debugging servlets
- The book provides one example code implementation DebugLog to demonstrate this approach of log_style.

Performance Tuning

- Usually before judging performance it should be gauged
- To do this the authors have provided a class called LoadTester.
- The purpose of LoadTester is to create a load on the server and measure the average response time. This in their own words is a simple book example. You may look at jmeter at [http://java.apache.org](http://java.apache.org).
- In case the total response time is too high, the first step is to figure out why.
- Consider bottle necks like
  * Opening network connections, one too many
  * Opening and Closing files
  * Allocating memory
  * Poorly written Algorithms
    - Use Caching as far possible
    - Use buffered readers and streams
    - Use Simple algorithms first
- Do not try to optimize the whole thing
- Extending, remember the fastest servlet is the one you don’t have to write
ServletContext

ServletContext
ServletContext