Tomcat 5 New Features

ApacheCon US 2003
Session MO10 – 11/17/2003 – 16:00-17:00

Craig R. McClanahan
Senior Staff Engineer
Sun Microsystems, Inc.

Slides:
http://www.apache.org/~craigmcc/

Agenda

- Introduction
- New Servlet 2.4 Features
- HTTP Connectors
- JMX Based Architecture
- Security Related Issues
- Miscellaneous Changes
- Summary

Introduction

- In three years, Tomcat has become a very successful base platform for building Java-based web applications
- Code is used as the basis for the reference implementation of J2EE 1.4
- We're going to focus on the changes in the *Catalina* servlet container between Tomcat 4 and Tomcat 5

New Servlet 2.4 Features

Servlet and Tomcat Versions

- Tomcat major version numbers correspond to version changes in the underlying Servlet and JSP specifications
  - Tomcat 3.x – Servlet 2.2 / JSP 1.1
  - Tomcat 4.x – Servlet 2.3 / JSP 1.2
  - Tomcat 5.x – Servlet 2.4 / JSP 2.0
- Servlet 2.4 is primarily a maintenance release
- Nevertheless, some interesting changes

Servlet 2.4 – XML Schema

- Better validation of *web.xml* elements
  - Data types
  - Primary key uniqueness
- Reusable elements across J2EE specifications
- Arbitrary ordering of elements within <web-app>
- DTDs still supported for backwards compatibility

Servlet 2.4 – Listeners

- Request lifecycle listeners
  - Complete the event handling model started in Servlet 2.3 (application and session)
- ServletRequestListener
  - Beginning and end of request lifetime
- ServletRequestAttributeListener
  - Add, update, remove request attributes

Servlet 2.4 – Filters

- Servlet 2.3 – Filters only invoked on initial request
  - Not on RequestDispatcher.forward() or RequestDispatcher.include()
- Servlet 2.4 – Filters may be requested to be invoked on any combination of request, include, or forward
- For backwards compatibility, request-only is the default

Servlet 2.4 – Deprecation

- The *SingleThreadModel* interface has been deprecated
  - But is still supported for backwards compatibility
- Original intent was to make writing threadsafe servlets easier
- But it never dealt adequeately with all of the relevant issues
  - Session attributes
  - Static variables

Servlet 2.4 – Miscellaneous

- New ServletRequest Methods:
  - getLocalAddr(), getLocalName(), getLocalPort()
- New ServletResponse Methods:
  - setCharacterEncoding(), getContentType()
- Configurable locale—character encoding mappings
- Roughly 50 other clarifications and fixes

HTTP Connectors

HTTP in Tomcat 4

- Original architecture centered around three interfaces:
  - org.apache.catalina.Connector
  - org.apache.catalina.Request
  - org.apache.catalina.Response
- With some HTTP-specific extensions:
  - org.apache.catalina.HttpRequest
  - org.apache.catalina.HttpResponse

HTTP in Tomcat 4

- Pure-Java implementation in the org.apache.catalina.connector.http package
  - HttpConnector – implements Connector
  - HttpProcessor – per-thread request processor
  - HttpRequestImpl – implements HttpRequest
  - HttpResponseImpl – implements HttpResponse
- Tomcat 4.1 replaced this connector by Coyote
  - Still supported for backwards compatibility

HTTP in Tomcat 4

- Design goals of Coyote
  - Focus on higer performance
  - Independent of underlying servlet container architecture
    - Supports Tomcat 3.3, 4.x, and 5.x
  - Afford opportunity to refactor location of certain functionality to leverage webserver connectors

HTTP in Tomcat 4

- Besides a pure-Java HTTP stack, Tomcat has historically supported connections to a native code HTTP server
  - Most common example, of course, is Apache's HTTPD server
  - But connectors for Netscape (NSAPI), Microsoft (ISAPI) and other webservers also exist
- Most popular connector was (and is) *mod_jk*
  - Includes basic load balancing support

HTTP in Tomcat 5

- Source code for HTTP and webserver connectors factored into separate codebase
  - *jakarta-tomcat-connectors* CVS respository
  - Original pure-Java HTTP stack completely replaced by Coyote
- Mapping of request to a particular webapp and servlet migrated to connector
- Webserver connector development moving towards *mod_jk2*

JMX Based Architecture

JMX in Tomcat 4

- Tomcat 4 container lifecycle based on JavaBeans composition design patterns
  - Components have zero-args constructor
  - All configuration via JavaBeans properties
- At container startup time, component creation and wiring driven by parsing the *server.xml* file
- Embedded API also provided for programmatic startup
  - org.apache.catalina.startup.Embedded

JMX in Tomcat 4

- Tomcat 4.1 added *optional* JMX support for configuration management
  - Components themselves not JMX aware
  - Configuration adds listeners to all interesting events to make corresponding changes in available Mbeans
- JMX support leveraged in Administration webapp for configuration changes

JMX in Tomcat 5

- Tomcat 5.0 *requires* JMX support
  - Components are JMX aware (including knowing their own object names)
  - Components manage registration and deregistration of MBeans on lifecycle events
- Moving towards all dynamic configuraton changes being made via JMX operations
  - Likely to leverage JMX Remoting (JSR-160)
  - Will enable Tomcat support in JMX-based management consoles

Security Related Issues

Security Architecture

- Tomcat 4 supported an optional startup mode that executes webapps under a Java SecurityManager
  - Fine-grained control over functionality accessible to webapps
  - Includes (by default) protection of Catalina internal classes

Security Audit

- A security audit of the Catalina source code was conducted in September-October 2002
- As a result, several changes were made to enhance the internal security of Catalina classes:
  - Public methods changed to protected or private
  - Facades added in front of critical internal classes
  - Package-level protection added to configurable security policy

Miscellaneous Changes

Manager Webapp Enhance-ments

- HTML version enabled by default
- New information retrieval commands
  - List OS and JVM properties
  - List Global JNDI resources
  - List available security roles
  - Session statistics

Manager Webapp Enhance-ments

- Server Status
  - Memory statistics for the JVM itself
  - Information about thread use in container and connectors
  - Optional detailed status for webapps and servlets
- (Experimental) JMX Proxy Servlet
  - Perform MBean queries using JMX syntax
  - Set new values for MBean attributes
  - Likely to be replaced by JMX Remoting

Deployer

- Tomcat 4 supported dynamic deployment, reload, and undeployment of web applications
  - Via direct manager webapp URL requests
  - Via Ant tasks that wrap these requests
  - Included ability to include a *context descriptor* file
- Tomcat 5 extensions and improvements
  - Precompile JSP pages
  - Validate *web.xml* deployment descriptor

Realms

- Tomcat 4 supported a variety of Realm (user database) implementations:
  - JDBCRealm – Database accessed via JDBC
  - JNDIRealm – Directory server accessed by JNDI (normally the LDAP provider is used)
  - MemoryRealm – In memory database backed by *tomcat-users.xml* file
- Tomcat 5 adds support for DataSource-Realm
  - Database accessed via named JNDI data source

Summary

- Tomcat 5 is an evolutionary improvement on the Tomcat 4 code base
  - Implement new specification features
  - Refactorings to improve organization
  - Refactorings to improve performance
- Servlet 2.4 and JSP 2.0 (along with all the other J2EE 1.4 specs) just passed final ballot in the JCP
- We can likely expect a Tomcat 5.0 general release very soon

Resources

- Tomcat Web Site
  - http://jakarta.apache.org/tomcat/
- Tomcat 5.0 Documentation
  - http://jakarta.apache.org/tomcat/tomcat-5.0-docs/
- Tomcat Mailing Lists at Apache
  - tomcat-dev@jakarta.apache.org
  - tomcat-user@jakarta.apache.org

Resources

- Tomcat Issue Tracking System
  - http://nagoya.apache.org/bugzilla/
- Servlet 2.4 Information
  - http://java.sun.com/products/servlet/
- JSP 2.0 Information
  - http://java.sun.com/products/jsp/
- Free! J2EE 1.4 App Server
  - http://java.sun.com/j2ee/

Questions?