Introduction to Apache Tomcat

Tim Funk

November 2010

Who am I?

- Tomcat committer for over 8 years
- Day job: programmer at Armstrong World Industries.

Disclaimers

- Opinions - Mine
- Accuracy in Servlet spec docs

Topics

- Apache Tomcat Background / Overview
- Servlet 3.0 Updates
- Similarities to Apache Tomcat 6
- Apache Tomcat 7 features

Introduction

- Apache Tomcat is a Servlet Container
  - Servlet Spec 3.0 (JSR 315)
  - JSP Spec 2.2 (JSR 245)
  - EL 2.2
- Not a full J2EE app server
  - See Geronimo, Glassfish, Weblogic, JBoss, etc
  - Missing many J2EE features
  - This is not a bad thing

Servlet Container 101

- In the beginning
  - Static files
- Static … useful .. Not useful enough
  - cgi
- CGI “problems”
  - Each cgi is own process (startup/teardown)
  - Sharing memory
- Servlet containers
  - One Process
  - Request is run in a thread
  - Threads can be pooled
  - One process – “easier” to pool resources

History

- First public version was 3 donated by Sun
- Apache Tomcat 4 - Catalina architecture
- http://tomcat.apache.org/whichversion.html

Servlet/JSP Spec	Apache Tomcat version	Actual release revision	Minimum Java Version
3.0/2.2	7.0.x	7.0.4 (beta)	1.6
2.5/2.1	6.0.x	6.0.29	1.5
2.4/2.0	5.5.x	5.5.31	1.4
2.3/1.2	4.1.x (archived)	4.1.40 (archived)	1.3
2.2/1.1	3.3.x (archived)	3.3.2 (archived)	1.1

1 minute tour of webapp creation

Simple Servlet

```
package bacon;

import javax.servlet.*;
import javax.servlet.http.*;

public class MMMM extends HttpServlet {
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
                  throws ServletException, java.io.IOException {
      response.setContentType("text/plain");
      response.getWriter().println("sweet sweet bacon");
    }
}
```

Map it

In WEB-INF/web.xml

```
 <servlet>
    <servlet-name>ilovebacon</servlet-name>
    <servlet-class>bacon.MMMM</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>ilovebacon</servlet-name>
    <url-pattern>*.bacon</url-pattern>
  </servlet-mapping>
```

deploy it

Copy the webapp into

`$CATALINA_HOME/webapps/drool`

Test it …

`http://localhost:8080/drool/man-crush.bacon`

Result:

`sweet sweet bacon`

And for fun:

`http://localhost:8080/drool/i-love-to-eat.bacon`

Simple JSP

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
   prefix="c"%>
<% request.setAttribute("now", new java.util.Date())%>
Hello world the time is ${now}.
<c:if test='${(now.time/1000)%2==0}'>Even second</c:if>
---
```

Result

```
Hello world the time is Fri Nov 5 08:10:22 EDT 2010.
Even second
```

Simple Filter

```
package bacon;
import javax.servlet.*;
import javax.servlet.http.*;

public class BaconFilter implements Filter {
  public void init(FilterConfig config) throws ServletException { }
  public void destroy() { }

  public void doFilter(ServletRequest req, ServletResponse resp,
   FilterChain chain) throws ServletException, java.io.IOException
  {
    HttpServletResponse response = (HttpServletResponse)resp;
    response.setHeader("X-You-Need", "more bacon");
    chain.doFilter(req, resp);
  }
}
```

Map it

In WEB-INF/web.xml

```
<filter>
   <filter-name>tasty</filter-name>
   <filter-class>bacon.BaconFilter</filter-class>
 </filter>

 <filter-mapping>
   <filter-name>tasty</filter-name>
   <servlet-name>ilovebacon</servlet-name>
 </filter-mapping>
```

And … the rest

- Listeners for the creation and attribute replacement for
  - ServletContext
  - Session
  - Request
- RequestDispatcher
- Hand wave here the rest of the spec – remember: this is just an intro

Servlet 3.0 – webapp config

- Before – web.xml
- After
  - web.xml
  - Annotations
  - META-INF/web-fragment.xml
  - Programmatic via ServletContext
- Potential security nightmare

Servlet 3.0 – webapp config

@WebServlet

```
package bacon;
import javax.servlet.*;
import javax.servlet.http.*;

@WebServlet(name="ilovebacon", urlPatterns={"*.bacon"})
public class MMMM extends HttpServlet {
    // pretend all that code was still here
}
```

Servlet 3.0 – webapp config

@WebFilter

```
package bacon;
import javax.servlet.*;
import javax.servlet.http.*;

@WebFilter(“*.bacon”, urlPatterns={"*.bacon"})
public class BaconFilter implements Filter {
    // pretend all that code was still here
}
```

Servlet 3.0 – webapp config

Other annotations

- @WebInitParam
- @WebListener
- @MultipartConfig
- @ServletSecurity
  - <security-constraint>

web.xml fragments

- Allows to you to have a snippet of XML injected into web.xml
- Place into META-INF/web-fragment.xml of any jar in the webapp lib dir
- The spec has many rules deciding which fragment is first

web.xml fragments

```
<web-fragment>
 <servlet>
    <servlet-name>ilovebacon</servlet-name>
    <servlet-class>bacon.MMMM</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ilovebacon</servlet-name>
    <url-pattern>*.bacon</url-pattern>
  </servlet-mapping>
  <filter>
    <filter-name>tasty</filter-name>
    <filter-class>bacon.BaconFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>tasty</filter-name>
    <servlet-name>ilovebacon</servlet-name>
  </filter-mapping>
</web-fragment>
```

Servlet 3.0 - Dynamic Config

- New methods for ServletContext
  - addServlet /addServletMapping
  - addFilter / addServletFilterMapping
  - …
- Via ServletContextListener – progammatic servlet/filter declarations/mappings

Servlet 3.0 - Session Tracking

- SessionCookieConfig
  - void setDomain(String domain)
  - void setHttpOnly(boolean httpOnly)
  - void setMaxAge(int maxAge)
  - void setName(String name)
  - void setPath(String path)
  - void setSecure(boolean secure)
  - void setComment(String comment)

Servlet 3.0

- Programmatic login
  - HttpServletRequest.login(…)
  - HttpServletRequest.logout()

Servlet 3.0 – File Upload

- Servlet has @MultipartConfig
- Request is multipart/form-data
- Can now use from HttpServletRequest
  - public Collection<Part> getParts()
  - public Part getPart(String name).

Servlet 3.0

- Async Support
  - See Section 2.3 of the Servlet spec
- httpOnly support for cookies

JSP 2.2

- Clarification of 2.1

EL 2.2

- Method invocations
- Before
  - Method invocations via
    - Static functions
    - Faking a Map
- Now
  - Call a method on the bean (like most template engines)

What is the same?

- Server configuration basics
  - server.xml
  - conf directory
- Directory structure
- Classloaders
- Most internal classes/ interfaces
  - Exceptions : comet, Lifecycle
- JMX support (ok .. It got better)
- Evolution … not revolution

Apache Tomcat Specific features

Invoker - GONE

- http://localhost/servlet/bacon.MMM
  - No longer “easy” to deploy

Embedding

- See Mark's 3:30 talk
- New class org.apache.catalina.startup.Tomcat

Embedding

```
// cwd: output/embed
// $JAVA_HOME/bin/java -cp *.jar bsh.Interpreter  min.bsh
import org.apache.catalina.startup.Tomcat;
import org.apache.catalina.Context;
import javax.servlet.http.*;

Tomcat tomcat = new Tomcat();
Context webapp = tomcat.addWebapp("", "/tmp/bacon");
HttpServlet mmmBacon  = new HttpServlet() {
   public void doGet(HttpServletRequest req,
                     HttpServletResponse resp) {
        resp.getOutputStream().println("I love bacon");
}};
tomcat.addServlet("", "bacon", mmmBacon);
webapp.addServletMapping("/", "bacon");
tomcat.start();
tomcat.getServer().await();
```

Aliases

- Scope: <Context ..
- Example:
  <Context .. aliases='/woot=/tmp,/yeah=/etc'
- Transparent: the servlet api doesn't see the aliases
- Helpful for serving static assets you wish to keep out of original codebase

Manager webapp

- Recurring security complaints fixed
- GET vs POST
- CSRF Protection (Filter)
- Tool use moved to /manager/text
- Finer access control - new roles
  - manager-gui
  - manager-jmx
  - manager-script
  - manager-status

“Cruft”

- Checkstyle / Findbugs / IDE Support etc used to
  - Remove unused code
  - Convert to generics
  - StringBuffer → StringBuilder
- All Connectors share same code
  - Side effect to get async working
  - All use Executors for thread pool

Logging

- **New classes**
  - **VerbatimFormatter**
  - **OneLineFormatter**
  - **AsyncFileHandler**

Security

- **Access Log enabled by default**
- **Session cookie changes on authentication**

ExpiresFilter

- org.apache.catalina.filters.ExpiresFilter
- Provides mod_expires like functionality
- The javadocs are very detailed.

ExpiresFilter

```
<filter>
   <filter-name>ExpiresFilter</filter-name>
   <filter-class>org.apache.catalina.filters.ExpiresFilter</filter-
   class>
   <init-param>
    <param-name>ExpiresByType image</param-name>
    <param-value>access plus 10 minutes</param-value>
   </init-param>
   <init-param>
    <param-name>ExpiresByType text/css</param-name>
    <param-value>access plus 10 minutes</param-value>
   </init-param>
   <init-param>
    <param-name>ExpiresByType text/javascript</param-name>
    <param-value>access plus 10 minutes</param-value>
   </init-param>
</filter>
```

VirtualWebapploader

- Allows a customized classpath
- Easier to emulate a standard webapp without the need for assembly all the webapp dependencies as jars in WEB-INF/lib.
-
```
<Context …>
<Loader
  className=
    "org.apache.catalina.loader.VirtualWebappLoader"
  virtualClasspath=
    "/dir/classes;/somedir/somejar.jar;/somedir/*.jar"
/>
</Context>
```

Memory Leak Prevention

- Manager webapp → “Find leaks” button
- Lots of techniques
  - See JreMemoryLeakPreventionListener
- Many ported back to Apache Tomcat 6.0
- See Mark’s talk @ 10 (next) for more details

Other Misc

- Configtest
  - bin/configtest.sh
- Many Valves now available as Filters
  - Compare org.apache.catalina.filters to org.apache.catalina.valves
- JmxRemoteLifecycleListener – Helper for connecting via JMX through firewall

Need Help?

- FAQ
  - http://wiki.apache.org/tomcat/FAQ
- User's List
  - http://tomcat.apache.org/lists.html
  - Subscribe: users-subscribe@tomcat.apache.org
- Your favorite search engine

Questions?