Tomcat 7 & Servlet 3
Mark Thomas
April 2009
Who am I?

- Apache Tomcat committer
- Resolved 1,500+ Tomcat bugs
- Apache Tomcat PMC member
- Member of the Apache Software Foundation
- Member of the ASF security committee
- Created the Tomcat security pages
- Senior Software Engineer and Consultant at SpringSource
Agenda

• Tomcat versions vs Servlet & JSP specification versions
• New features for Tomcat 7
• Specification timeline and process
• New features / changes in Servlet 3.0
  – Asynchronous processing
  – Dynamic configuration
  – Web fragments
  – Annotations
  – Programmatic login
  – Session cookie configuration
  – Other possible changes
• Current status of Tomcat 7 development
## Tomcat and specification versions

<table>
<thead>
<tr>
<th>Tomcat version</th>
<th>Servlet version</th>
<th>JSP version</th>
<th>JDK version</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.0.x</td>
<td>3.0</td>
<td>2.1?</td>
<td>1.6+</td>
</tr>
<tr>
<td>6.0.x</td>
<td>2.5</td>
<td>2.1</td>
<td>1.5+</td>
</tr>
<tr>
<td>5.0.x / 5.5.x</td>
<td>2.4</td>
<td>2.0</td>
<td>1.4+</td>
</tr>
<tr>
<td>4.1.x</td>
<td>2.3</td>
<td>1.2</td>
<td>1.3+</td>
</tr>
<tr>
<td>3.x</td>
<td>2.2</td>
<td>1.2</td>
<td>1.2+ (?)</td>
</tr>
</tbody>
</table>
New for Tomcat 7

- Servlet 3.0 support
- Cluster communication via UDP
- Significantly improved JMX support - GSOC
- Replace Valves with Filters - GSOC
- Bayeux
- Numerous smaller improvements
- Code clean up
  - Remove unused stuff
  - Resolve inconsistencies
Servlet 3.0 timeline

- Early draft review – May 2008
- Public review – December 2008
- Final release – planned for June 2009
- Final release probably September 2009
  - Lots of changes since public review
  - JEE needs more time
  - Likely to be another public review
Asynchronous processing

• One of the major improvements
• Most containers already have this in some form
• Tomcat offers the CometProcessor interface
• What is it?
  – Decouple container request thread from ServletRequest/ServletResponse objects
• What is it NOT?
  – Non blocking servlet IO implementation
  – This was briefly discussed
  – Backwards compatibility challenges
  – Very complex programming model
Asynchronous processing

doFilter(Request req, Response res, FilterChain chain) {
    // pre filter actions
    chain.doFilter(req, res);
    // post filter action
}
// recycle request/response objects

service(Request req, Response res) {
    // read request
    // write response
}
// recycle request/response objects (no filter)
Asynchronous processing

• Backwards compatibility
• Servlet/Filters are non asynchronous by default
  – Asynchronous processing requires explicit support in code
  – Currently done using annotation
  – Still looking at other ways of enabling

```java
@WebFilter(asyncSupported=true)
public class MyFilter {
}

@WebServlet(asyncSupported=true)
public class MyServlet {
}
```
Asynchronous processing

- Starting

```java
interface javax.servlet.ServletRequest {

    AsyncContext startAsync();

    AsyncContext startAsync(Request, Response);

}

// throws IllegalStateException if
// isAsyncSupported() returns false
```
Asynchronous processing

- `javax.servlet.AsyncContext`

- Similarities to CometEvent in Tomcat
  - Wraps request/response objects
  - Can dispatch the request to a URL
  - Can request a container thread to execute a task
  - Can notify container that the request is complete
Asynchronous processing

- Example

```java
service(Request req, Response res) {
    AsyncContext actx = req.startAsync();
    Runnable runnable = new Runnable() {
        public void run() {
            Message m = jmsTemplate.receive();
            res.write(m);
            req.complete();
        }
    };
    executor.submit(runnable);
}
```
Asynchronous processing

- Defensive programming

```java
service(Request req, Response res) {
    if (req.isAsyncSupported() &&
        !req.isAsyncStarted()) {
        AsyncContext actx = req.getAsyncContext();
        req.startAsync();
        ...
    } else {
        ...
    }
}
```
Asynchronous processing

• Forwarding to a content generator

```java
interface javax.servlet.AsyncContext {
    void dispatch();
    void dispatch(String path);
    void dispatch(ServletContext ctx, String path);
}
// Dispatches to a container thread
```
Asynchronous processing

- Forwarding to a content generator

```java
service(Request req, Response res) {
    final AsyncContext actx = req.startAsync();
    Runnable runnable = new Runnable() {
        public void run() {
            Message m = jmsTemplate.receive();
            req.setAttribute("quote", m);
            actx.dispatch("/stock/quote.jsp");
        }
    };
    executor.submit(runnable);
}
```
Asynchronous processing

• Receiving events

```java
interface javax.servlet.AsyncListener {
    void onComplete(AsyncEvent event);
    void onTimeout(AsyncEvent event);
}
```
Web fragments

- Ability to submit web.xml fragments with JAR packaged libraries
- Can be disabled using
  - `<metadata-complete>true</metadata-complete>`
- META-INF/web-fragment.xml
- Essentially same content as web.xml
Web fragments

- mylib.jar/META-INF/web-fragment.xml

```xml
<web-fragment>
  <servlet>
    <servlet>
      <servlet-name>MyServlet</servlet-name>
      <servlet-class>foo.bar.MyServlet</servlet-class>
    </servlet>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyServlet</servlet-name>
    <url-pattern>*.tsp</url-pattern>
  </servlet-mapping>
</web-fragment>
```
Web fragments

- Ordering of web fragments
  - Absolute ordering
    - web.xml - <absolute-ordering>
  - Relative ordering
    - web-fragment.xml - <ordering>
- Ordering is name based

```xml
<web-fragment>
  <name>MyWebFragment1</name>
  <ordering>
    <after>MyWebFragment2</after>
    <before><others/></before>
  </ordering>
</web-fragment>
```
Dynamic configuration

- Programatically add
  - Servlets
  - Filters
- To a ServletContext
- Can only be done during the ServletContext initialization
  - contextInitialized() method of ServletContextListener

```java
interface javax.servlet.ServletContext {
    FilterRegistration addFilter(
        String filterName, String|Class filterClass);

    ServletRegistration addServlet(
        String servletName, String|Class servletClass);
}
```
Dynamic configuration

- Registration objects

```java
interface Servlet/Filter-Registration{
    setDescription(String);

    setInitParameter(String name, Object value);

    setInitParameters(Map<String, Object> p);

    setAsyncSupported(boolean supported);

    addMappingForUrlPatterns(...);
}
```
Annotations

• New annotations added
  – @WebServlet (must extend HttpServlet)
  – @WebFilter (must implement Filter)
  – @WebInitParam (both servlets/filters)
  – @WebListener
    • ServletContextListener (& attr listener)
    • HttpSessionListener (& attr listener)
    • ServletRequestListener (& attr listener)

• Can be on any class in any jar
  – Providing the class implements the right interfaces
Programmatic login

• New methods

```java
interface HttpServletRequest{

    login(HttpServletRequest resp);

    login(String username, String password);

    logout();
}
```

• Allow a login to happen on a non constrained request
• Sensitive to the response being committed
  – In order to set a session cookie, when configured
Session cookie configuration

- Configure the session cookie

```java
interface javax.servlet.SessionCookieConfig {
    setName(String name);
    setSecure(boolean isSecure);
    setHttpOnly(boolean isHttpOnly);
    setPath(String path);
    setDomain(String domain);
    setComment(String comment);
}
```
Other possible changes

• Generics
• More deprecation
• Delete deprecated methods???
• File upload
  – is being considered for addition
  – challenge: Servlet 3.0 doesn't have non blocking IO
  – removes the usefulness of having yet another file upload API
Current status

• API changes complete
  – Based on public draft
  – We know this is going to change
• Dynamic configuration complete
  – We know this is going to change
• Session cookie configuration complete
  – We know this is going to change
Current status

• Asynchronous processing
  – Filip has a plan :)

• Web fragments
  – I had a plan
  – Spec changes means more complex implementation required

• Annotations
  – I have a plan
Questions?

• markt@apache.org
• users@tomcat.apache.org
• dev@tomcat.apache.org
• http://people.apache.org/~markt/presentations

• mark.thomas@springsource.com
• http://www.springsource.com/webinars