Apache Tomcat 7: Katzenleben

Frank Pientka

Apache Tomcat zählt seit über 10 Jahren zu den am meisten verwendeten Webservern im Java-Bereich. Da die aktuelle Version über 5 Jahren nach Erscheinen seiner Vorgängerversionen erscheint, darf man gespannt sein, was die7er Version an Neuerungen [7] und Verbesserungen mit sich bringt. Im Vordergrund steht hierbei die Unterstützung des Servlet3.0-Standards.

Die Mutter aller Webcontainer

Die Geburtsstunde vom Jakarta Tomcat schlug 1999 mit der Version 3, die auf Basis von Sun übergebenen Servlet-Container-Referenzimplementierung JServ beruhte. Es folgten mit 4, 5 und 6 noch einige Versionen, die die jeweiligen aktuellen Java und Servlet-Versionen unterstützten. Im Nachhinein war es ein großer Schritt von Sun, damit die ersten Teile von Java unter eine Open-Source-Lizenz zu stellen. Als kleiner Bruder des erfolgreichen Apache http-Webservers hat Tomcat in der Java-Webentwicklergemeinde eine große Verbreitung gefunden, was sich nicht nur in mehr als 10 Millionen Downloads niederschlägt. Dazu trug sicher sein einfaches und robustes Installations- und Konfigurationskonzept bei. Letztendlich gingen aus dem Tomcat-Projekt das automatische BUILD-Werkzeug ANT und die Apache Taglibs hervor, die inzwischen eigenständige Projekte sind. Seine Popularität führt dazu, dass viele Webframeworks mit ihm getestet werden und er in einigen „ausgewachsenen“ Java EE-Anwendungsservern, wie JBoss und Geronimo, verwendet wird. Durch die frühe Unterstützung neuer Standards, wie Servlet 3.0 und OSGi, hat der Webserver Jetty Tomcat überholt. Wir werden hier schauen, was sich in diesen Punkten beim Tomcat 7 getan hat.

Modularität und Einfachheit

Geringer Platzbedarf und einfache Konfigurierbarkeit in wenigen XML-Dateien sind ausgewiesene Stärken von Tomcat. Mit wenigen Änderungen kann die Konfiguration von der Server-Installation getrennt werden, um so entweder mehrere Tomcat-Instanzen mit einer Installation oder mit unterschiedlichen Tomcat-Unterversionen zu betreiben. Mit der neuen Tomcat 7-Version, können die meisten älteren Konfigurationen und Anwendungen ohne Anpassungen weiter betrieben werden, da diese Version abwärts kompatibel ist [2]. Die Konfigurationsmöglichkeiten, Schnittstellen und Rollen des Tomcats sind im folgenden Bild als grobe Orientierung dargestellt.

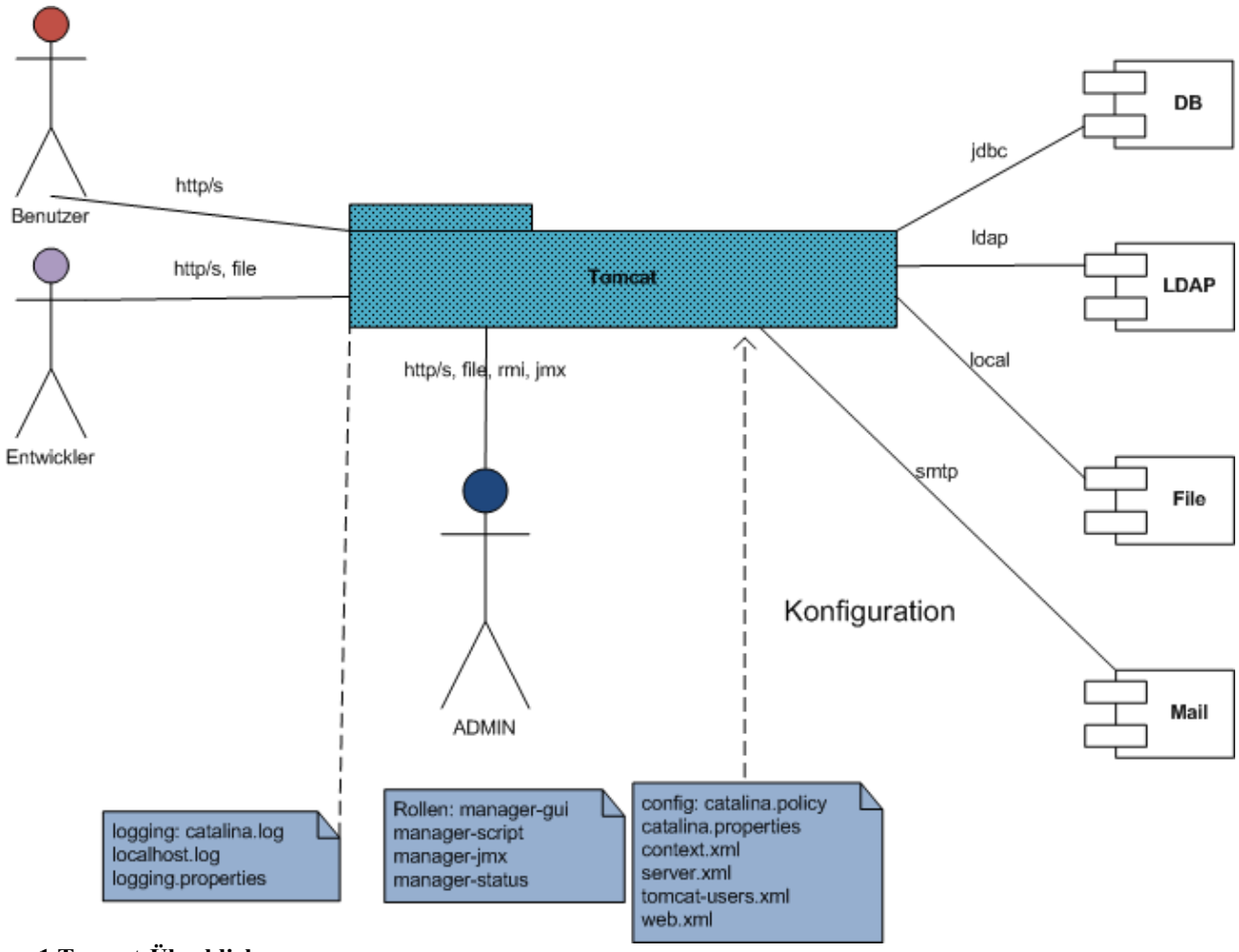

Abbildung 1 Tomcat-Überblick

Bei der Paketierung ist die Dokumentation nicht mehr Teil des Kernpaketes und muss separat installiert werden. Gleiches galt für die Zusatzpakete JMX-Remoting, WebServices und Logging auch bisher schon. Als Java Ablaufumgebung wird jetzt nur noch die Java-Runtime Environment (JRE) 6.0 unterstützt. Das führt zur Aktualisierung des integrierten Eclipse-Compilers auf die Version 3.6 und durch JDBC 4.0 der DBCP 1.4 Bibliothek. Statt DBCP (`factory="org.apache.commons.dbcp.BasicDataSourceFactory"`) kann alternativ der neue optimierte eigene Datenpool (`factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"`) verwendet werden [7].

Besonders die Unterstützung der verbesserten Nutzung und Wiederverwendung von Datenbank-Ressourcen im neuen Tomcat, kommt auch alten Anwendungen zu Gute Die Unterstützung für einige Altlasten, wie j2-Connector oder den Jikes-Kompiler, wurden mit der neuen Version entsorgt.

Aber sicher!

Im neuen Tomcat wird Sicherheit groß geschrieben. So sind den Verwaltungsanwendungen verschiedene Rollen mit eigene URLs zugeordnet. Um, wie gewohnt, die Tomcat-Status-Seite (http://localhost:8080/manager/status) oder den Tomcat-Webmanager (http://localhost:8080/manager/status?XML=true) aufzurufen müssen Sie einem Benutzer in der Datei `tomcat-users.xml` vorher die Rolle `manager-gui` zuordnen. Für die anderen Verwaltungsschnittstellen gibt es die neuen Rollen `manager-script`, `manager-jmx` (http://localhost:8080/manager/jmxproxy/) und `manager-status`. Nützlich ist noch die Rolle `manager-script`, da Sie so einfach auf der Kommandozeile den Status des Tomcats überprüfen können.

```
curl --url http://localhost:8080/manager/text/list --user tomcat:password
curl --url http://localhost:8080/manager/text/serverinfo --user tomcat:password
```

Um Speicherlöcher [3] für Webanwendungen in der JVM einfacher zu identifizieren und zu beseitigen, enthält die Manager-Anwendung einen Knopf und eine URL http://localhost:8080/manager/html/findleaks bzw. http://localhost:8080/manager/text/findleaks für die Ausführung der Analyse.

Durch die Refaktorierung des Codes wurde die Sicherheit, Anpassbarkeit, Stabilität und Einbettung des Tomcats verbessert. Einige dieser Verbesserungen wurden teilweise in die letzten 6er zurückportiert, so dass auch die Vorgängerversion von diesen profitieren kann. Um Cross-Site-Request-Forgery („Seiten-übergreifende Aufruf-Manipulation, um Daten unberechtigt zu erlangen“, XSRF oder CSRF abgekürzt) zu verhindern, wurde eine neue Filter-Klasse `org.apache.catalina.filters.CsrfPreventionFilter` erstellt und in der Datei `manager\WEB-INF\web.xml` registriert.

Neben dem Sicherheitsmanager mit der erweiterten `catalina.policy`-Datei sind hier vor allem die Connectoren, MBean-Registrierung, Jasper-Integration und der Lebenzzyklus-Manager betroffen. Alle Connectoren nutzen für einen besseren Ressourcenverbrauch die Java-Executor-Threadpools. Einige Tomcat-eigene Filter (Valves) stehen nun alternativ als ServletFilter bereit, die aus Portabilitätsgründen auch für die Zukunft empfohlen werden. Wer noch weitere Infos zum Thema Absicherung von produktiven Tomcat-Servern sucht findet diese unter [9].

Das asynchrone Web

Das durch AJAX-Anwendungen mögliche http-Client-Push-Verfahren Comet konnte vor Servlet 3.0 nur produktspezifisch gelöst werden. Jetzt kann man die asynchrone Webverarbeitung auch mit Tomcat nutzen. Dafür unterstützen alle neuen HTTP- und AJP-Connectoren die asynchrone Anfrageverarbeitung. Für eine portable Comet-Laufzeitumgebung bietet sich das Atmosphere Framework an, das auch Tomcat unterstützt. Für Comet [6] wird bei Tomcat das Chat-Beispiel http://localhost:8080/examples/jsp/chat/ mitgeliefert, zu dem man den non-blocking-Java-Connector `org.apache.coyote.http11.Http11NioProtocol` in der `server.xml`-Datei konfigurieren muss. Bei den Beispielen und der Dokumentation sind die neuen Möglichkeiten nur rudimentär beschrieben. Für die asynchrone Verarbeitung gibt es ein neues Beispiel, das den asynchronen Start, Timeout und Listener eines Servlets am Beispiel einer JSP zeigt. http://localhost:8080/examples/jsp/async/

Hallo Servlet 3.0-Welt

Tomcat 7 ist bereits in den WTP Server-Tools der Eclipse IDE 3.6 (Helios) integriert, die JavaEE6 unterstützen. und kann direkt verwendet werden. Damit können einfach die neuen Möglichkeiten von Servlet 3.0, JSP 2.2 und EL 2.2 ausprobiert werden.

Eclipse 3.6 (Helios) unterstützt bereits die Entwicklung von Java EE 6-Anwendungen. Deshalb kann auch in der Web Tools Plattform 3.2 eine Tomcat 7 Laufzeitumgebung verwendet werden. Nachdem Sie diese zusammen mit einem JDK 6 konfiguriert haben, können Sie ihr erstes dynamisches Webprojekt mit einem Hallo-Welt-Servlet erstellen. Für ein zügiges Arbeiten mit Eclipse und Tomcat sollten Sie auch in den aktuellen Versionen einige Einstellungen vornehmen. Wenn Sie die Warnung bzgl. der fehlenden `serialVersionUID`-Variable stört können Sie das unter Window, Preferences, Java, Compiler, Errors/Warnings bei der "Serializable class without ..." auf "Ignore" setzen.

Da einige Einstellungen von Tomcat für die Entwicklung mit Eclipse hinderlich sind, sollten Sie in der conf/server.xml-Datei vom Tomcat den Port von 8080 auf 80 setzen (<Connector port="80"). In der conf/context.xml -Datei setzen Sie zum Wiederholten Laden von Servlet <Context reloadable="true" privileged="true">. Damit Sie auch ohne Einsteigsseite die Datei direkt aufrufen können ändern Sie den Parameterwert von listings des DefaultServlets in der conf/web.xml-Datei auf true.

```
<servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
      <init-param>
        <param-name>debug</param-name>
        <param-value>0</param-value>
      </init-param>
      <init-param>
        <param-name>listings</param-name>
        <param-value>true</param-value>

      </init-param>
```

In Servlet 3.0 können viele Konfigurationen, die sonst in der web.xml-Datei der Webanwendung vorgenommen werden, durch Annotationen geschehen. Ein Hallo Servlet 3.0-Welt-Beispiel mit Default-Parameterwerten, das über http://localhost:8080/webapp/hello aufgerufen wird, sieht dann, wie folgt aus.

```
WebServlet(value="/hello",
                initParams = {
                  @WebInitParam(name="greeting", value="Hello "),
                  @WebInitParam(name="to", value=" Servlet world 3.0!")
                })
public class HelloServlet extends GenericServlet {

  public void doGet (ServletRequest req, ServletResponse res)
    throws IOException, ServletException
  {
    PrintWriter out = res.getWriter();
    out.println(getInitParameter("greeting"));
    out.println(getInitParameter("to"));
  }
}
```

Um ein UploadServlet zu erstellen, mussten vor Servlet 3.0 externe Fileupload-Servlet-Bibliotheken verwendet werden. Nun wird ein Servlet mit der @MultipartConfig-Annotation markiert, die die Attribute location, maxFileSize, maxRequestSize und fileSizeThreshold haben kann

```
@MultipartConfig
public class UploadServlet extends HttpServlet {}
```

Um Klassenkonflikte mit Anwendungen zu vermeiden und die Migration zu vereinfachen, die die Fileupload-Bibliotheken von `commons-fileupload 1.2.2` und `commons-io` 1.4 direkt verwenden, wurden diese im Tomcat mit dem Paketnamen in `org.apache.tomcat.util.http.fileupload` umbenannt.

Durch die Unterstützung der Servlet-3.0-Spezifikation haben sich die Konfiguration von Cookies (über die neue Klasse `SessionCookieConfig`) und die asynchrone Kommunikation geändert. Für die asynchrone Bearbeitung von Requests mit dem Comet-Prozessor und der asyncSupported=true-Annotation haben die Projektbeteiligten den Paketnamen geändert.

```
@WebServlet(urlPatterns = "/AsyncServlet", asyncSupported=true)
```

Die Konfiguration der Webanwendung kann jetzt nicht nur in der zentralen web.xml erfolgen, sondern in mehreren *web-fragment.xml*-Dateien. So wird die Wiederverwendbarkeit von Bibliotheken erhöht und die Modularisierung derer Konfiguration verbessert.

Durch die Annotationen im Code kann in manchen Fällen die web.xml-Datei sogar komplett entfallen. Als Beispiel wird eine resourceA.jsp-Datei gezeigt, die mit gleichen Namen in der WEB-INF/lib/resources.jar-Datei ist und nicht aus dem Web-Fragment genommen wird, da die Datei in der Webanwendung erste Priorität hat.

http://localhost:8080/webapp-3.0-fragments/resourceA.jsp

Über den ServletContext kann die Konfiguration der Webanwendung dynamische angepasst werden, indem zur Laufzeit Servlets und Filter hinzugefügt werden.

```
interface javax.servlet.ServletContext {
FilterRegistration addFilter(
String filterName, String|Class filterClass);
ServletRegistration addServlet(
String servletName, String|Class servletClass);}
```

Ausblick und Fazit

Die Zahl 7 [4] hat in vielen Kulturen unterschiedliche Bedeutung. Im Chinesischen steht diese für Glück und im Hebräischen für Vollkommenheit. Spektakuläres darf man von einem so ausgereiften Produkt, wie den Apache Tomcat in der 7er Version nicht erwarten. Bei der Einbettbarkeit und der Modularität hat Tomcat Fortschritte gemacht, kann hier jedoch nicht an Jetty heran reichen, der sogar als OSGi-Bundle verwendet werden kann. Eine JavaEE 6-Zertifizierung wird es für Tomcat nur im Zusammenhang mit Geronimo 3 geben. Schwächen hat die aktuelle Tomcat-Version in der Dokumentation der neuen Möglichkeiten, warum wir hier einen kleinen Einblick gegeben haben. Trotzdem wird mit auch mit Tomcat 7 die Webentwicklung mit Java-EE 6, so einfach, wie noch nie. Durch die Verbesserungen in den Bereichen Speicherverbrauch, Modularität und Sicherheit ist der Tomcat 7 besonders für große Anwendungen interessant. Damit ist der neue, wie früher der alte Tomcat auch für die neue Generation von Webanwendung erste Wahl. Auch wenn die meisten Anwender noch die beiden, nach wie vor, unterstützten Vorgängerversionen verwenden, ist Tomcat ein langes Katzenleben zu wünschen. Tomcat hat schon viele seiner Applikationsserver-Konkurrenten und selbst Sun überlebt.

Literatur und Links

[1] Tomcat 7 Dokumentation: http://tomcat.apache.org/tomcat-7.0-doc/index.html
[2] Tomcat 7 Migration: http://tomcat.apache.org/migration.html#6.0.x_to_7.0.x
[3] Aufspüren von Speicherlöchern: http://wiki.apache.org/tomcat/MemoryLeakProtection
[4] Zahl 7: http://de.wikipedia.org/wiki/Sieben
[5] Neues zu Tomcat 7: https://blogs.apache.org/tomcat/category/Tomcat+7
[6] Comet-Unterstützung: http://wiki.apache.org/tomcat/WhatIsComet
[7] Datenbank Verbindungs-Pools mit Tomcat: http://people.apache.org/~fhanik/jdbc-pool/jdbc-pool.html
[8] Webanwendungen mit OSGi modularisieren, Christoph v. Kuepach, Stefan Schedl, JavaSPEKTRUM 03, 2010
[9] Absicherung von Tomcat-Servern: http://tomcat.apache.org/tomcat-7.0-doc/security-manager-howto.html, http://www.owasp.org/index.php/Securing_tomcat

Frank Pientka ist Senior Software Architect bei der Materna GmbH in Dortmund. Er ist zertifizierter SCJP und Gründungsmitglied des iSAQB. Als Autor des Buches zu Apache Geronimo beschäftigt er sich intensiv mit dem Einsatz von Java Open Source Software.
frank.pientka@gmx.de