Apache Tomcat 7
Servlet API 3.0
(BETA)

Peter Roßbach
pr@objektpark.de

Mein Rucksack

Javamagazin

- Peter Roßbach
  - Freiberuflicher
    IT-Systemarchitekt, Berater, Trainer und Autor
    - Entwickler im Apache Tomcat Projekt
    - Mitglied der Apache Software Foundation
    - Webexperte
    - Autor der TomC@Kolumne im Java Magazin

  - Autor
    - http://tomcat.objektpark.org/
    - Tomcat 4x http://www.tom4.de
    - Java Server und Servlets
    - E-Mail : pr@objektpark.de

  - Fachautor und Speaker auf deutschen Java Konferenzen

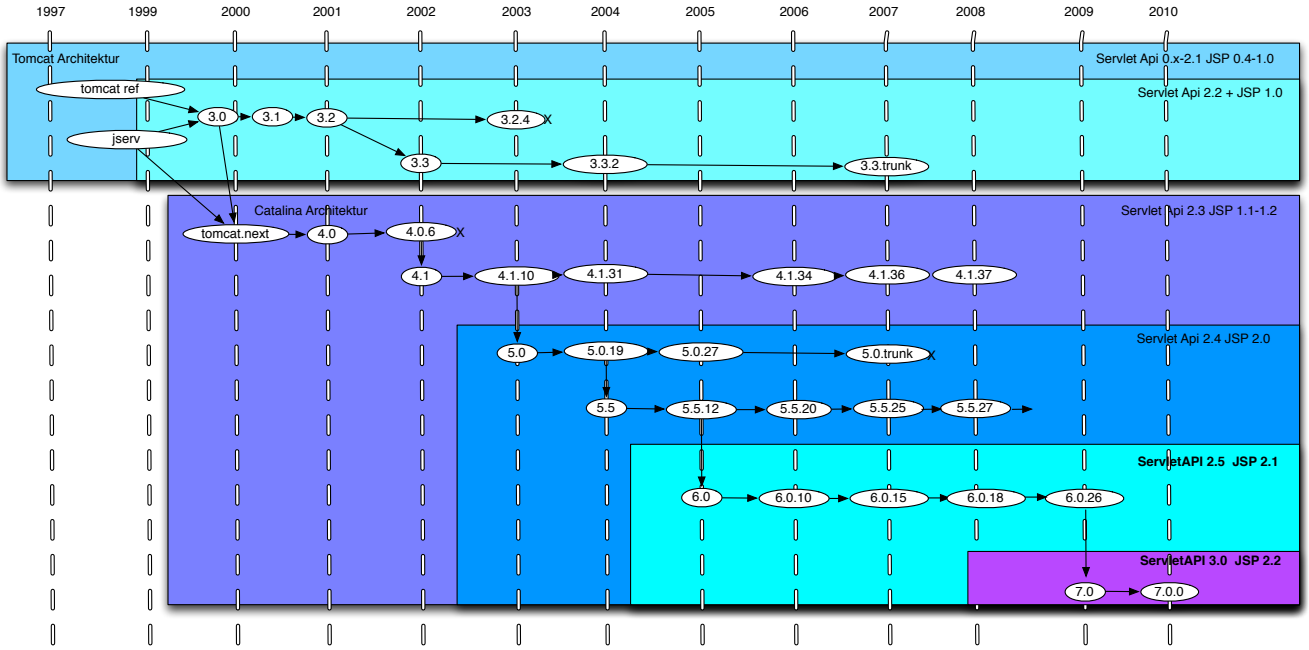
Was gibt es Neues?

- Servlet API 3.0 Support (JSR 315)
- JSP API 2.2 (Expression Language 2.2) JSR 254
- Erste Release des Apache Tomcat 7.0.0 ist fast fertig
  - offizielle Test (TCK) zu 100% erfolgreich
  - RC4 TAG
- Integration JEE 6 Web Profile
  - Geronimo 3
  - OpenEJB 3.1.2
  - MyFaces 2 (Dev)
  - JBOSS 6 und JBoss Web 3
  - Eclipse 3.6 - WTP 2.2

Projekt wächst

- Standard Taglib Projekt (JSTL) ist nun Teilprojekt des Apache Tomcat Projekts
  - http://tomcat.apache.org/taglibs/
  - JSTL 1.0, 1.1 und 1.2
- 20 Entwickler im Apache Tomcat Projekt!
- http://wiki.apache.org/tomcat/PoweredBy
  - 94 Referenz Unternehmen
  - 32 Hosting Provider
- ca. 1 Mio Download pro Monat...

History



Web Container API's

Servlet	JSP	Tomcat	Java
3.0	2.2	7.0.x	1.6
2.5	2.1	6.0.x	1.5
2.4	2.0	5.0.x & 5.5.x	1.4
2.3	1.2	4.0.x & 4.1.x	1.3
2.2	1.1	3.2.x & 3.3.x	1.2?

Servlet API 3.0

- Modularität der Anwendung und des Containers verbessern
  - Web-Fragement
- Asynchrone Kommunikation
  - Server Push
  - Entkoppelung vom Container Thread
- File Upload direkt im Container
- Skallierbarkeit erhöhen
- Session und Cookie Handling verbessert

JSP 2.2 API

- Wenige Änderung zu JSP 2.1
- Update für JSF auf Expression Language 2.2
  - Kleine Anpassungen
- Grössere Änderungen sind nicht zu erwarten!
- Migrationsprobleme sind also nur im geringen Masse zu erwarten.

Apache Tomcat 7 Basis

- Neues Lifecycle API der Komponenten
- Alle Connectoren haben den Async Support
  - Effizient nur mit NIO oder APR Connectoren
- Alle Connectoren nutzen nun die Executor ThreadPools
- MemoryLeakDetection für Anwendungsredeployment
- Einige Packages oder Komponenten sind verschoben
- SSL Reneogition Fixes (tcnative)
- Module für eigenen JDBC Pool und Bayeux Implementierung
- Unit Testen mit Tomcat
  - org.apache.catalina.startup.TomcatBaseTest
  - org.apache.catalina.startup.Tomcat

Servlet API 3.0

- Programmatischen und modulare Erweiterbarkeit
- Vereinfachung des Deployments
- Async Servlet Support
- Kleinere Erweiterungen ...
  - File Upload
  - Session Tracking

Neue Möglichkeiten API 3.0

- Integration von Web-Frameworks ohne weitere Konfiguration in der Datei web.xml.
  - Weitere Annotationen
  - Erweiterung des ServletContext
  - Modularisierung der web.xml: „Ein Framework bzw. JAR kann ein Stück der gesamten web.xml Konfiguration enthalten.“
    - META-INF/web.xml in jedem Jar (WEB-INF/lib/*.jar) möglich
    - web-fragment Element definieren Servlets, Filters und Listeners

Erweiterbarkeit

- Mit dem API des ServletContext können Servlets, Filter, Listener während des Startups hinzugefügt und gelöscht werden.
- Nutzung von Annotations zur Deklaration von Web-Anwendung
  - Servlets, Filter und Listener
  - Servlet- und Filter-Mapping

Annotations

```
@WebServlet(“/myServlet”)
public class MySampleServlet extends HttpServlet {
   public void doGet(HttpServletRequest req,
   HttpServletResponse res) {}
}
```

```
@WebFilter(”/myServlet/*“)
public class MyFilter extends Filter{
   public void doFilter(req,res) {}
}
```


ServletFilter

```
import javax.servlet.FilterChain;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.annotation.FilterMapping;
import javax.servlet.http.annotation.WebFilter;

@WebFilter("/myservlet/*")
public class SetCharacterEncodingFilter implements Filter
{
....
    public void doFilter (HttpServletRequest request, HttpServletResponse
response, FilterChain chain)
    throws java.io.IOException, javax.servlet.ServletException
    {
        // Conditionally select and set the character encoding to be used
        if (ignore || (request.getCharacterEncoding() == null)) {
            String encoding = selectEncoding(request);
            if (encoding != null)
                request.setCharacterEncoding(encoding);
        }

        chain.doFilter(request, response);
    }
}
```

Annotations web.xml

- @WebServlet(name,urlPatterns,value,load-on-startup,description,smallIcon,largeIcon,asyncSupported)
- @WebFilter(display-name,filter-name,description,smallIcon,largeIcon,urlPatterns,servlet-names, dispatcher-types,value, asyncSupported)
  - dispatcher-types ERROR,FORWARD,INCLUDE,REQUEST,ASYNC
  - @WebFilter(value = "/param", filterName = "paramFilter",
    dispatcherTypes = {
    DispatcherType.ERROR, DispatcherType.ASYNC },
    initParams = {
    @WebInitParam(name = "message", value = "Servlet says: ") })
- @WebInitParam(description,name,value)
- @WebListener(description)

weitere Beispiele

```
@WebServlet(name= "MyServlet", urlPatterns={"/foo", "/bar"})
public class SampleUsingAnnotationAttributes extends HttpServlet{

    public void doGet(HttpServletRequest req, HttpServletResponse
    res) {
    }
}
```

```
@WebServlet(value="/foo",
   initParams = {
      @WebInitParam(name="debug",value="true")
   }
)
public class SampleUsingAnnotationAttributes extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse
    res) {
    }
}
```

Annotation Overwrite

- Deklaration in der web.xml überschreibt Konfiguration in der Annotation
  - nur urlPatterns der web.xml gelten, sonst die der Annotation
  - initParameter werden durch Deskriptor überschrieben oder durch Annonation ergänzt
  - Ergänzung der Werte durch die Annotation, wenn nicht im Deskriptor explizit definiert.

Disable Servlet

```
<servlet>
    <servlet-name>param</servlet-name>
    <servlet-class>annotation.ParamServlet</servlet-class>
    <init-param>
        <param-name>foo</param-name>
        <param-value>hello</param-value>
    </init-param>
    <init-param>
        <param-name>bar</param-name>
        <param-value>peter</param-value>
    </init-param>

    <!-- Disable this servlet -->
    <enabled>false</enabled>

</servlet>
```

Listener

```
@WebListener(description="init fine tricks")
public class MyListener implements ServletContextListener {
   public void contextInialized(ServletContextEvent event)
   {
   ...
   }
   public void contextDestroy(ServletContextEvent event)
   {
   ...
   }
}
```


ServletContext.addServlet

```
@WebListener
public class MyListener implements ServletContextListener{
  public void contextInitialized (ServletContextEvent sce) {
       ServletContext servletContext =  event.getServletContext();
       ServletRegistration reg =
            servletContext.addServlet("Hello",
                 HelloApiServlet.class);
       reg.addMapping("/hello2");
       Map<String,String> initParameters = new
       HashMap<String,String>();
       initParameters.put("foo", "bar");
       reg.setInitParameters(initParameters);
  }
  ...
}
```

ServletContext.addFilter

```
@WebListener
public class MyListener implements ServletContextListener {
    public void contextInitialized (ServletContextEvent sce) {
        ServletContext servletContext =
            event.getServletContext();

        FilterRegistration freg =
            servletContext.addFilter("helloFilter",
                                     ParamFilter.class);

        freg.addMappingForUrlPatterns(
                EnumSet.of(DispatcherType.REQUEST,
        DispatcherType.ASYNC), true, "/hello2", "/param");

    }
}
```


ServletContext

Method Summary	
FilterRegistration.Dynamic	**addFilter**(java.lang.String filterName, java.lang.Class<? extends Filter> filterClass) Adds the filter with the given name and class type to this servlet context.
FilterRegistration.Dynamic	**addFilter**(java.lang.String filterName, Filter filter)
FilterRegistration.Dynamic	**addFilter**(java.lang.String filterName, java.lang.String className) Adds the filter with the given name and class name to this servlet context.
ServletRegistration.Dynamic	**addServlet**(java.lang.String servletName, java.lang.Class<? extends Servlet> servletClass)
ServletRegistration.Dynamic	**addServlet**(java.lang.String servletName, Servlet servlet)
ServletRegistration.Dynamic	**addServlet**(java.lang.String servletName, java.lang.String className)
<T extends Filter> T	**createFilter**(java.lang.Class<T> c) Instantiates the given Filter class and performs any required resource injection into the new Filter instance before returning it.
<T extends Servlet> T	**createServlet**(java.lang.Class<T> c) Instantiates the given Servlet class and performs any required resource injection into the new Servlet instance before returning it.
FilterRegistration	**findFilterRegistration**(java.lang.String filterName) Gets the FilterRegistration corresponding to the filter with the given filterName.
ServletRegistration	**findServletRegistration**(java.lang.String servletName) Gets the ServletRegistration corresponding to the servlet with the given servletName.

Annotations 3.0

- Voraussetzung in WEB-INF/web.xml
  - <metadata-complete>false<metadata-complete>
- Es müssen alle Klassen in WEB-INF/classes und WEB-INF/lib/*.jar untersucht werden.
- Erweiterung im Tomcat
  - JarScanner Konfiguration im Context
    - <Context> <JarScanner className="org.apache.tomcat.util.scan.StandardJarScanner" scanClassPath= "true" /> </Context>
  - Deny System.Property
    - -Dtomcat.util.scan.DefaultJarScanner.jarsToSkip= "logging-api.jar,admin.jar"

Servlet API 3.0: web-fragment

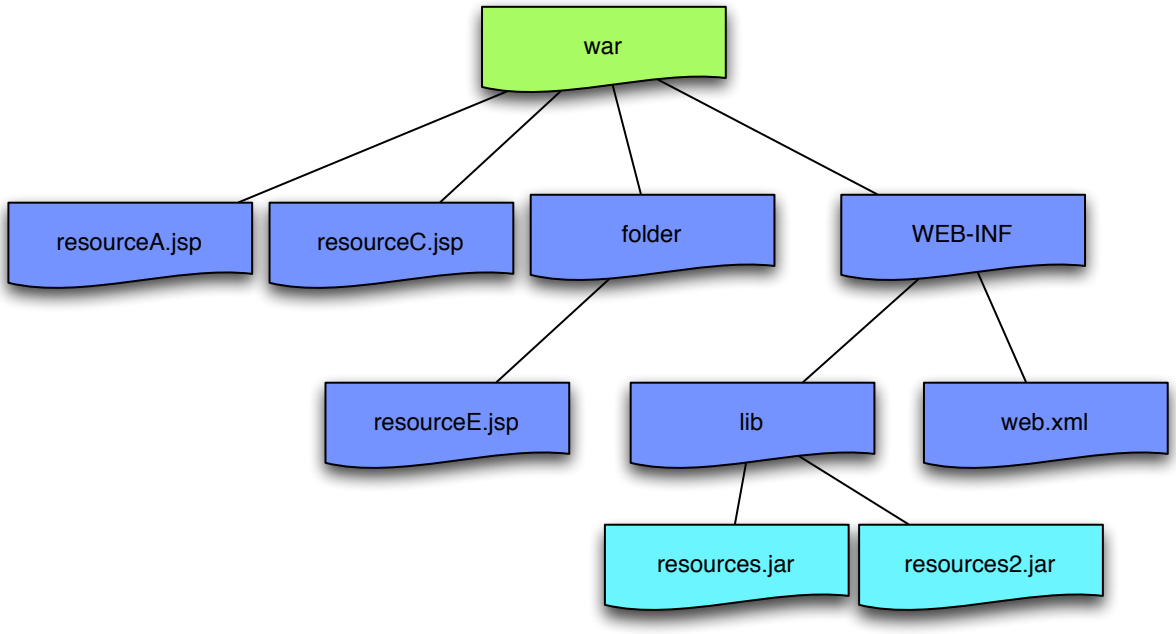
- jar:/WEB-INF/lib/**hello.jar!/META-INF/web-fragment.xml**

```
<web-fragment>
  <servlet>
    <servlet-name>welcome</servlet-name>
    <servlet-class>
      WelcomeServlet
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>welcome</servlet-name>
    <url-pattern>/Welcome</url-pattern>
  </servlet-mapping>
...
</web-fragment>
```

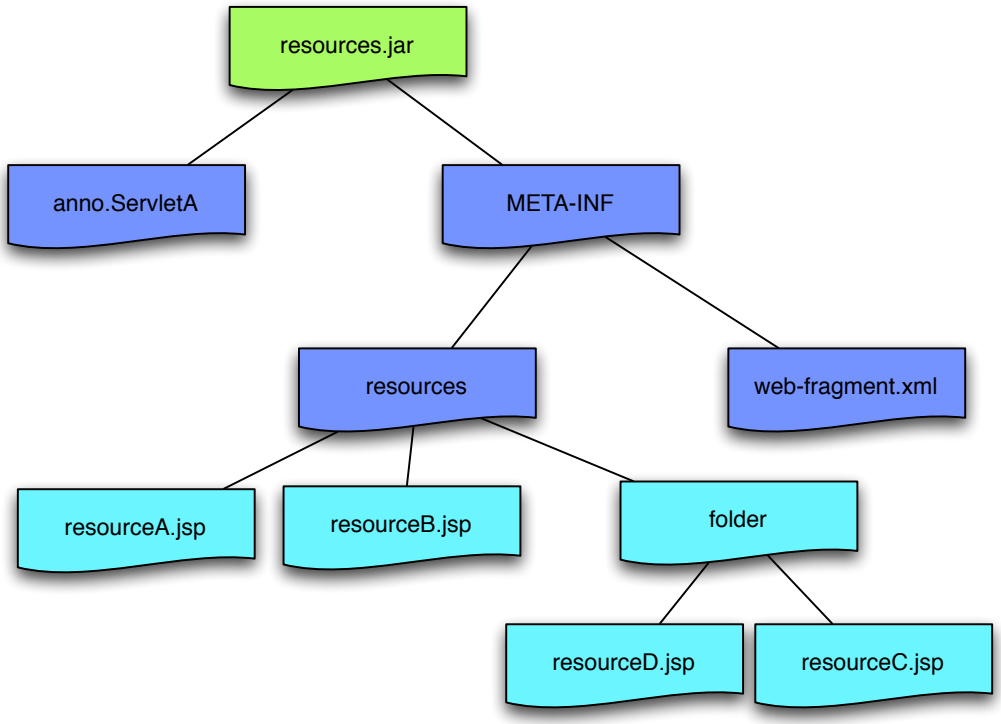
Ressoure Loading

- Anwendung modularisieren
  - WEB-INF/lib/>fragments>.jar
    - Servlets, Filter, Listener
    - Konfigurationen: web-fragments.xml
    - Ressourcen

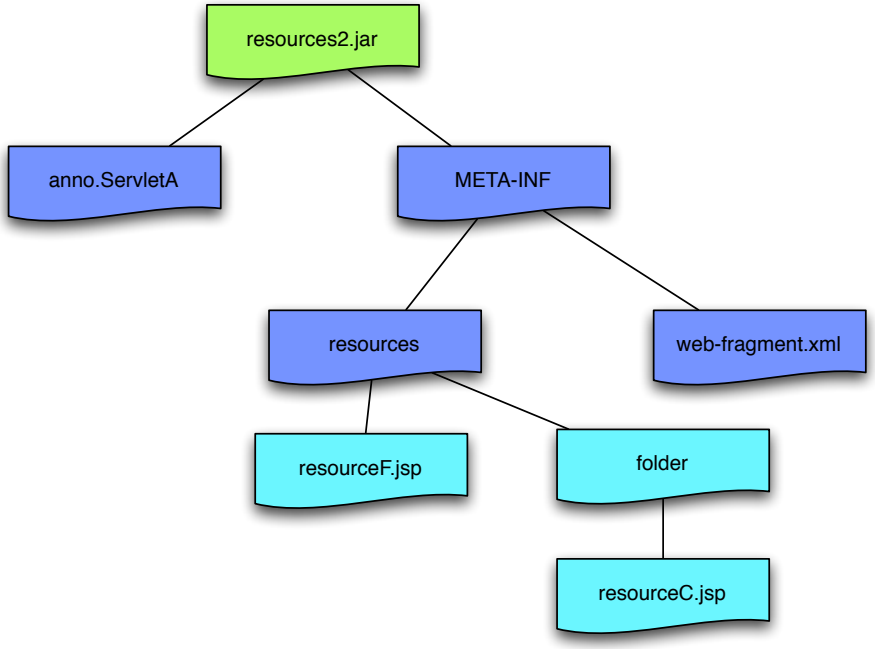
War



resources.jar



resource2.jar



Module ausserhalb der WebApp

```
<Context
   aliases="/content=${catalina.base}/content">

   <JarScanner scanClassPath="true" />

   <Loader
className="org.apache.catalina.loader.VirtualWebappLoader"
virtualClasspath="${catalina.base}/admin/lib/hello2.jar" />

</Context>
```


Ladereihenfolge

- Reihenfolge
  1. Zuerst werden die Informationen der WEB-INF/ web.xml geladen
  2. Dann werden die web-fragmente.xml geladen
     - absolute oder relative Ordering!
  3. Dann werden die Annotations ausgewertet
- ServletContextListener Event Handling (1. oder 3.)
  - addServlets and addFilters + Mappings
- Tomcat lädt noch conf/web.xml!

Ordnung...

- Absolute Ordnung
  - web.xml => <absolute-ordering>
- Relative Ordnung
  - web-fragment.xml => <ordering>

```
<web-app>
  <name>MyApp</name>
  <absolute-ordering>
    <name>MyFragment3</name>
    <name>MyFragment2</name>
  </absolute-ordering>
  ...
</web-app>
```

```
<web-fragment>
  <name>MyWebFragment1</name>
  <ordering> <!-- Ingore as absolute ordering exists -->
    <before>MyWebFragment2</before>
    <after></others></after>
  </ordering>
</web-fragment>
```


Weitere Resourcen laden...

```
<web-app>
  <name>MyApp</name>
  <absolute-ordering>
    <name>MyFragment3</name>
    <name>MyFragment2</name>
    <others/>
  </absolute-ordering>
  ...
</web-app>
```

Konflikte

- Es wird im Prinzip ein grosser Merge durchgeführt.
- Bei Konflikten gewinnt der Eintrag in der web.xml.
- Konflikte zwischen Elemente verschiedener web-Fragments führen zu einem Fehler und die Anwendung ist nicht verfügbar!
- Es gibt additive Elemente wie <context-param> und <init-param>!
- Tomcat log: <Context logEffectiveWebXml="true" />
  - Erzeugt eine web.xml mit dem Resultat!

metadata-complete

API	metadata-complete	Annotations bzw. Fragments
2,5	true	NEIN
2,5	false	JA
3,0	true	NEIN
3,0	false	JA

3.0 <web-app> <**metadata-complete>**false<metadata-complete>...

2.5 <web-app **metadata-complete=**“false“>

SessionTracking

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException
{
    HttpSession session = request.getSession();
    Cart cart = (Cart)session.getValue("cart");
    // ...
    // do logic to get the inventory number and
    // quantity of goods that user wants to add to cart

    cart.addItem(inventoryNumber, quanity);
}
```


Session Markierung

- Bisher:
  - Cookie (JSESSIONID)
  - URL- Paramater (jsessionid)
- ab 3.0 kann Namen der Parameter im Container geändert werden.
- Tomcat 6.0.20
  - -Dorg.apache.catalina.SESSION_COOKIE_NAME=JSESSIONID
  - -Dorg.apache.catalina.SESSION_PARAMETER_NAME=jsessionid
- mod_jk 1.2.28 (worker.lb.session_cookie=JSESSIONID und `worker.lb.session_path=;jsessionid`

SessionCookieConfig

- Voreinstellung der SessionCookie Information
- SessionCookieConfig(
  String domain,
  String path,
  String comment,
  boolean **isHttpOnly**,
  boolean isSecure)

```
@WebListener
public class MyListener implements ServletContextListener {
   public void contextInitialized (ServletContextEvent sce) {
      ServletContext context = sce.getServletContext();
      SessionCookieConfig scc = new SessionCookieConfig(
      „my.domain“,context,“/jup“,true, false) ;
      context.setSessionCookieConfig(scc);

}
```


SessionTrackingModes

- setSessionTrackingModes(java.util.EnumSet<SessionTrackingMode> sessionTrackingModes)
  - SessionTrackingMode.COOKIE (default)
  - SessionTrackingMode.URL (default)
  - SessionTrackingMode.SSL

```
@WebListener
public class MyListener implements ServletContextListener {
   public void contextInitialized (ServletContextEvent sce) {
      ServletContext context = sce.getServletContext();
      context.setSessionTrackingMode(EnumSet.of(SessionTrac
      kingMode.COOKIE, SessionTrackingMode.URL,
      SessionTrackingMode.SSL));

}
```

Async Servlet

Die berücksichtigten Use Cases für Asynchronität sind
-Server Side AJAX Push per Comet
- Asynchrone Web Services und Web Proxies
- Ermöglicht non-blocking Verhalten des Containers bei langen Requests

Erweiterung von ServletRequest um folgende Methoden
addAsyncListener(AsyncListener listener)
addAsyncListener(AsyncListener listener,
ServletRequest req, ServletResponse res)
getAsyncContext()
startAsync() oder startAsync(ServletRequest req, ServletResponse res)

Asynchron IO

- Aktueller Stand entspricht einer Entkoppelung des Container Request Threads von der Bearbeitung von Request und Response
- Was ist der Asynch IO in der Servlet Spec nicht?
  - Eine Non Blocking IO Implementierung
- Warum?
  - Problem mit der Rückwärtskompatibilität
  - Bewusst ein komplexes Programmiermodel vermieden

AsyncSupport

```
@WebFilter(value="/hello", asyncSupported=true)
public class myFilter implements Filter {}
```

```
@WebServlet(value="/hello",asyncSupported=true)
public class myServlet extend HttpServlet {}
```


AsyncSupport

```
interface javax.servlet.ServletRequest {

 AsyncContext startAsync();

 AsyncContext startAsync(Request,Response);

}
```

throws IllegalStateException wenn isAsyncSupported() returns false

Async Beispiel

```
@WebServlet(value = "/simple", asyncSupported = true)
public class SimpleDispatch extends HttpServlet {
    protected void service(final HttpServletRequest req,
            final HttpServletResponse resp) throws ServletException,
            IOException {
        if (null != req.getAttribute("dispatch")) {
            resp.getWriter().write(
                    "Simple request start at" + req.getAttribute("dispatch")
                            + " and dispatch worked at"
                            + System.currentTimeMillis() + "\n");
            req.getAsyncContext().complete();
        } else {
            resp.setContentType("text/plain");
            final AsyncContext actx = req.startAsync();
            actx.setTimeout(Long.MAX_VALUE);
            Runnable run = new Runnable() {
                public void run() {
                    try {
                        req.setAttribute("dispatch", System
                                        .currentTimeMillis());
                        Thread.currentThread().setName("Simple-Thread");
                        Thread.sleep(2 * 1000);
                        actx.dispatch();
                    } catch (Exception x) {}
                }
            };
            Thread t = new Thread(run);
            t.start();
        }
    }
}
```


Besser....

```
service(Request req, Response res) {
  if (req.isAsyncSupported()) {
    AsyncContext actx =
      req.isAsyncStarted()?
        req.getAsyncContext():
        req.startAsync();

    ...
  } else {
    ...
  }
}
```

Listener von ServletRequest

- Registriere einen AsyncListener für Complete und Timeout
  - void addAsyncListener(AsyncListener listener)
  - void addAsyncListener(AsyncListener listener, ServletRequest servletRequest, ServletResponse servletResponse)
    - Bestückung mit Wrapper
  - req.setAsyncTimeout(long timeout)

```
interface javax.servlet.AsyncListener {

  void onComplete(AsyncEvent event)

   void onTimeout(AsyncEvent event)
...

}
```


dispatch

```
interface javax.servlet.AsyncContext {

 void dispatch();

 void dispatch(String path);

 void dispatch(ServletContext ctx,
          String path)
...
}
```

Es wird ein neuer Container Thread gestartet!

Path muss mit ‚/' starten und ist relative zum ServletContext

Beispiel Forward

```
service(Request req, Response res) {
 final AsyncContext acontext = req.startAsync();
 Runnable runnable = new Runnable() {
   public void run() {
     Message m = jmsListener.receive();
     req.setAttribute("quote",m);
     //Neuer Container Thread wird gestartet
     acontext.dispatch("/stock/quote.jsp");
   }
 };
 executor.submit(runnable);
// Ende des bestehenden Container Threads
// Kein Recycle oder Close des Request/Response!
}
```


Async Dispatch Attribute

- javax.servlet.async.request_uri
- javax.servlet. async.context_path
- javax.servlet. async.servlet_path
- javax.servlet. async.path_info
- javax.servlet. async.query_string

AsyncContext

Method Summary	
void	**complete**() Completes the asynchronous operation that was started on the request that was used to initialze this AsyncContext, closing the response that was used to initialize this AsyncContext.
void	**dispatch**() Dispatches the request and response objects of this AsyncContext to the servlet container.
void	**dispatch**(ServletContext context, java.lang.String path) Dispatches the request and response objects of this AsyncContext to the given path scoped to the given context.
void	**dispatch**(java.lang.String path) Dispatches the request and response objects of this AsyncContext to the given path.
ServletRequest	**getRequest**() Gets the request that was used to initialize this AsyncContext by calling ServletRequest.startAsync() or ServletRequest.startAsync(ServletRequest, ServletResponse).
ServletResponse	**getResponse**() Gets the response that was used to initialize this AsyncContext by calling ServletRequest.startAsync() or ServletRequest.startAsync(ServletRequest, ServletResponse).
boolean	**hasOriginalRequestAndResponse**() Checks if this AsyncContext was initialized with the original request and response objects by calling ServletRequest.startAsync(), or if it was initialized with wrapped request and/or response objects using ServletRequest.startAsync(ServletRequest, ServletResponse).
void	**start**(java.lang.Runnable run) The container dispatches a thread to run the specified Runnable in the ServletContext that initialized this AsyncContext.

File Upload

- Anfrage ist vom Type „multipart/form-data“
- Neue Methoden am HttpServletRequest
  - public Iterable<Part> getParts()
  - public Part getPart(String name)
- Annotation MultipartConfig erforderlich

```
@WebServlet(name="upload")
@MultipartConfig(location="temp",
fileSizeThreshold="8196",
maxFileSize="10485760",
maxRequestSize="10485760")

public class myUploadServlet extend HttpServlet {}
```

Part

Method Summary	
void	**delete**() Deletes the underlying storage for a file item, including deleting any associated temporary disk file.
java.lang.String	**getContentType**() Gets the content type of this part.
java.lang.String	**getHeader**(java.lang.String name) Returns the value of the specified mime header as a `String`.
java.lang.Iterable<java.lang.String>	**getHeaderNames**() Returns an Iterable of all the header names this part contains.
java.lang.Iterable<java.lang.String>	**getHeaders**(java.lang.String name) Returns all the values of the specified Part header as an `Iterable` of `String` objects.
java.io.InputStream	**getInputStream**() Gets the content of this part as an `InputStream`
java.lang.String	**getName**() Gets the name of this part
long	**getSize**() Returns the size of this fille.
void	**write**(java.lang.String fileName) A convenience method to write this uploaded item to disk.

Weiteres

- Memory Leak Detection
- Connectoren nutzen alle die Executor Threadpools
- Module
  - JDBC 4 DataSource Pool
  - Comet Bayeux Implementierung
- AccessLogging von allen Requests und Async

MemoryLeaks

- http://wiki.apache.org/tomcat/MemoryLeakProtection
  - Falsche Nutzung von Threads
  - Falsche Nuzung von ThreadLocals
  - Risiko Shared Libs
- Server Listener: JreMemoryPreventionListener
  - ab Tomcat 6.0.24

Konfiguration

```
<Server port="8005" shutdown="SHUTDOWN">
  <Listener className="org.apache.catalina.core.JasperListener"/>
  <Listener className=
        "org.apache.catalina.core.JreMemoryLeakPreventionListener"/>

  <GlobalNamingResources>
       <Resource name="UserDatabase"
          auth="Container"
          type="org.apache.catalina.UserDatabase"
          factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
          pathname="conf/tomcat-users.xml"
         readonly="true"/>
  </GlobalNamingResources>
  <Service name="Catalina">
    <Connector port="8080"/>
    <Engine name="Catalina" defaultHost="localhost">
         <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
             resourceName="UserDatabase" />
      <Host name="localhost" appBase="webapps" />
    </Engine>
  </Service>
</Server>
```

Beispiele probieren...

- <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
- neue Attribute im Context
  - clearReferencesStatic=true
    - Löschen aller public static's
  - clearReferencesStopThreads=true
    - Löschen aller Threads die in der Anwendung gestartet sind.
  - clearReferencesThreadLocals=true
    - Lösche aller ThreadLocals Entries die vom ClassLoader noch referenziert sind.
  - unloadDelay=2000

Status

- Tomcat 7.0.0 -> Vermutlich Ende Juni 2010
  - TCK passed
  - Kleiner Bugs
- Integration in JEE 6 Web Profile in Geronimo
  - Vermutlich Ende Juli 2010

F&Q

- Peter Roßbach
  - News
    - http://tomcat.objektpark.org/
  - Kommen Sie zum meinen Workshop‘s und Sessions
    - JUGS Linz 2010
    - WJAX 2010
    - Beratung, Schulung, Workshops und Tomcat Support
  - mailto:pr@objektpark.de

Hilfe ist nah...

- LinuxHotel.de -- Lernen in angenehmer Atmosphäre mit Experten
  - Schulungen für Administratoren und Entwickler
  - Alles im Bereich Linux, Netzwerk und OS Services
  - Groovy & Grails (Dierk König & Marc Guillemot)
  - Osgi (Gerd Wütherich)
  - Apache Tomcat (Peter Roßbach)
  - Apache httpd (Michael Wandel & Peter Roßbach)
  - Webcontainer Firewalls (Christian Schneider & Thomas Krautgartner)
  - Java für Architekten (Torsten Friebe)
  - bald mehr...