

Jetty

o pequeno grande
servidor Java para web

Julio Viegas

Arquiteto de software/sistemas - SPC Brasil
Instrutor Java - Fontoura Education/Sun Microsystems

Perfil profissional

- +10 anos na area de tecnologia.
- Desenvolvedor, Instrutor, Mentor.
- SCJP 5, SCJA 1, SCEA 1 e 5, SCDJWS 1.4 e 5, SCWCD 1.4, SCBCD 5, Sun T3(Sun Certified Java Instructor).
- Atualmente atua como arquiteto de software e sistemas no SPC Brasil e como Instrutor Sun Microsystems/Fontoura.
- Hobista em eletronica analogica e digital. MCUs 8051 e Renesas M16C. Viciado em Unix, Wii, Twitter e blogueiro ocasional. Colaborador em diversas comunidades relacionadas a desenvolvimento de tecnologia.
- Mais em julioviogas.com.

Agenda

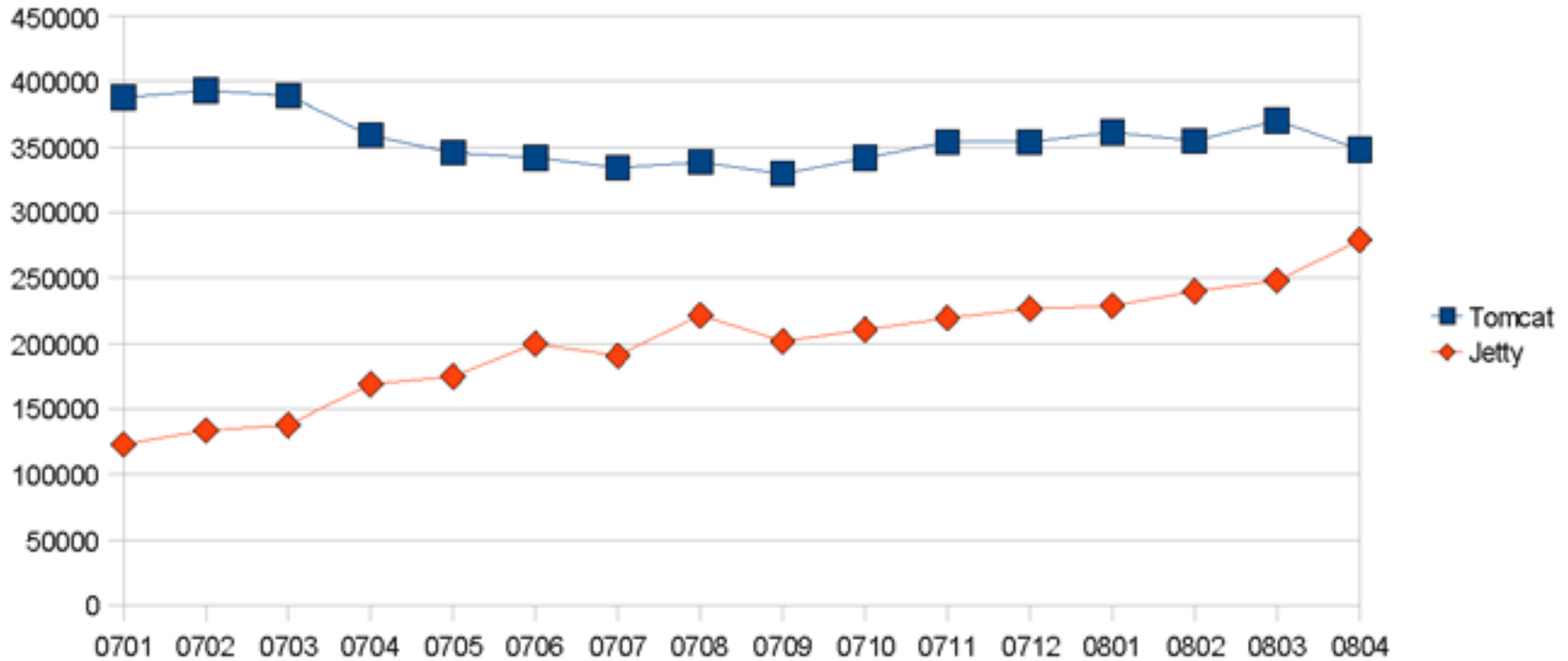
- O projeto Jetty
- Funcionalidades
- Arquitetura básica do container
- Instalando e executando o Jetty
- Configuração
- Realizando deploy
- i-Jetty
- Continuations/Comet
- Cluster com Terracotta
- Extras e referências adicionais

O projeto Jetty

- Um container servlet 2.5, APENAS ISSO!
- Desenvolvido por um time estavel desde 1995.
- Comunidade vibrante e de alto nivel!
- Foi o container web padrao do JBoss.
- Grande suporte a Web 2.0: AJAX e Web Assincrona.
- Adotado pelo Google App Engine e GWT, Yahoo Zimbra, Eclipse Foundation...
- Embutido em varios outros produtos: Cisco, Hadoop, Android...
- Suporte comercial fornecido por Webtide
- Hack friendly!

O projeto Jetty

- Market share em crescimento



Funcionalidades

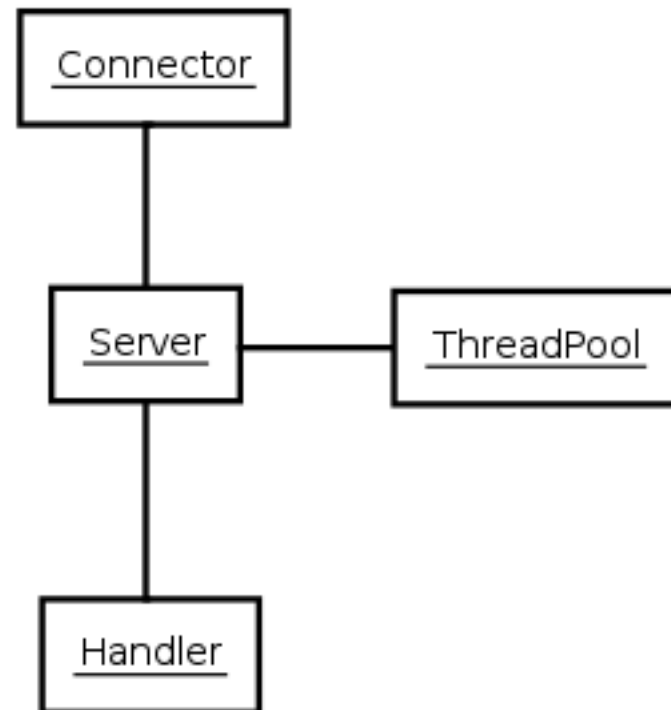
- Pequeno em tamanho e footprint.
- Performático.
- Simples. Baseado em POJO!
- Configuracao: programatica e XML.
- Estavel - sem memory leaks.
- Gratis, aberto e livre.
- Escolha padrao quando o assunto eh Web Assincrona!
- Threadpool simples com suporte a queue.
- DI/loC(antes mesmo do spring popularizar a coisa!).
- Graceful stop & star.

Funcionalidades adicionais

- MultiPartFilter.
- JNDI (para quem ainda precisa disso).
- Modularizacao Maven.
- Integracao com OSGi(bundles a partir da versao 6.1.15).
- QoS Filter: evolucao do DoSFilter.
- SSO via SPI SSORrealm; Implementacao padrao HashSSORrealm.
- Plugin para Eclipse.
- JTS: Atomikos, JOTM...
- Terracotta session cluster(cache com persistencia, grid, cluster lock).
- Adicionais: gzip, ssl, plugin ant, jaas provider...

Arquitetura

- HandlerCollection, ContextHandler, SessionHandler, ResourceHandler, ServletHandler
 - WebInfConfiguration
 - Configuration
 - TagLibConfiguration
- SelectChannelConnector
 - host
 - port
 - forwarded
 - idleMs
 - acceptors
 - queueSize
 - lowResourcesConnections
 - lowResourcesMaxIdleTime
- QueuedThreadPool
 - low/max
 - idleMs



Instalando

- download e unzip(jetty-6.1.xx.zip)!
 - contexts: hotdeploy!
 - etc: config
 - examples
 - extras: integrações opcionais(terracotta,jboss, spring)
 - lib
 - modules: fontes mavenizados
 - patches: "mods" nos fontes
 - webapps/webapps-plus: aplicacoes
 - start.jar
- deb, rpm, windows service wrapper...
- compile via maven

Executando

- Via xml: `java -jar start.jar etc/jetty.xml`
- Via código: `server.start();`
- Teste em `http://localhost:8080/test`
- Graceful stop: `kill -TERM pid`

Configuracao

- XML com DI/loC

```
<Configure id="Server" class="org.mortbay.jetty.Server">  
  <Set name="ThreadPool">  
    <New class="org.mortbay.thread.QueuedThreadPool">  
      <Set name="maxThreads">1000</Set>  
    </New>
```

...

- Programatico

```
Server server = new Server(8080);  
Context root = new Context(server, "/", Context.SESSIONS);  
root.addServlet(new ServletHolder(new HelloServlet("Ciao")), "/*");  
server.start();
```

Realizando deploy

- Estatico: WebAppDeployer

- Diretorio: webapps/\${contexto}
- War: webapps/\${contexto}.war
- Zip: webapps/\${contexto}.war
- Root: webapps/root

- Hot deploy: ContextDeployer

- atualize a configuracao do contexto

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<Configure class="org.mortbay.jetty.webapp.WebAppContext">  
  <Set name="contextPath"><Property name="foo"/></Set>  
  <Set name="war">./webapps/foo.war</Set>  
</Configure>
```

- Demo hello

i-Jetty

- Android
- Demo

Continuations

- Request > Connection > Thread
- AJAX Pooling? NAO!
- Devolva as threads para o ThreadPool!
- HTTP stateless
- HTTP stateful: isso nao existe!
- Comet + Bayeux
- WebSockets?
- Demo pooling vs continuations vs comet

Cluster Terracotta

- Escalabilidade simples
- Cluster: várias JVM como uma única!
- NAM – Network-Attached Memory
- Grid(paradigma Master/Worker com TIM Messaging)
- Grátis e aberto
- O particionamento é pago!

Cluster Terracotta

- DSOs – Distributed Shared Objects = Cluster HEAP
- DMI – Distributed Method Invocation = Distributed Notification/Listener
- Synchronized em cluster: synchronized, wait, notify, java.util.concurrent
- Integrações: Spring(Web Flow e Security), Ehcache, Hibernate, Tomcat, Jetty, Cglib...

Cluster Terracotta

- Demo Terracotta
- Demo Jetty + Terracotta

Perguntas



Obrigado!