



- ANDROID的
- 核心JAVA
- 桌面JAVA
- 企业JAVA
- JAVA基础知识
- JVM语言
- 软件开发
- DEVOPS的

» 企业Java » spring » Tomcat vs. Jetty vs. Undertow: Spring Boot嵌入式Servlet容器的比较

关于安迪·贝克



Andy是一名高级软件工程师，在网络，桌面和移动应用程序方面拥有16年的工作经验。他分别拥有弗吉尼亚大学和乔治华盛顿大学的计算机科学学士和硕士学位。他擅长使用Java Web服务，在处理Web应用程序，数据库以及持续集成和部署方面拥有丰富的经验。他目前在金融科技组织担任技术主管，负责支持Java中的移动应用程序服务。



## Tomcat vs. Jetty vs. Undertow: Spring Boot嵌入式Servlet容器的比较

发布者: 安迪·贝克 在 春天 2017年1月26日 0 11498意见



随着微服务的普及，我们看到嵌入式servlet容器的应用程序也越来越流行。Spring boot是一个基于Java的框架，支持应用程序服务。它作为独立jar运行，带有嵌入式servlet容器或作为容器内的WAR文件。

### 想掌握Spring Framework吗？

订阅我们的新闻通讯并立即下载Spring Framework Cookbook！

为了帮助您掌握领先和创新的Java框架，我们编写了一个带有所有主要功能和用例的kick-ass指南！除了在线学习，您可以下载PDF格式的电子书！

电子邮件地址：

注册

在这个例子中，我们将关注具有嵌入式servlet容器的独立jar。该框架支持三种不同类型的嵌入式servlet容器：Tomcat（默认），Jetty和Undertow。我们将比较三者，并查看属性，设置，性能和内存的差异。请记住，此示例正在分析默认配置。有许多方法可以优化性能或内存使用，包括自定义自动配置和组件扫描。

我们使用了Eclipse Neon，Java 8，Maven 3.3.9，Spring 1.4.3，Tomcat 8.5.6，Jetty 9.3.14和Undertow 1.3.24。

### 目录

- 1.设置Spring Boot应用程序
- 汤姆猫
- 3.码头
- 4.承诺
- 5.性能和负载
- 5.1。衡量绩效
- 5.2。测量记忆
- 6.比较
- 7.结论
- 8.下载源代码

## 1.设置Spring Boot应用程序

我们将使用Maven在Eclipse中使用适当的依赖项设置一个新项目。我们将在此示例中使用starter父级，但生产应用程序中的依赖项可能会更改为简化，优化或自定义。

### 1.1设置Spring Boot依赖项

## 表达逻辑

FILEX嵌入式文件系统

使嵌入式物联网开发尽可能简单。

rtos.com

## 分子气相沉积

自动固化烤箱

适用于当今最先进的MEMS半导体工艺应用

yieldengineering.com

通讯

173,459名内部人士已经免费白皮书！

立即加入他们，即可获新新闻的独家访问权限，Android，Scala，Groovy和其解。

电子邮件地址：

在您的区域中接收Java和J

注册

加入我们



每月有访问者和我们被访问网站。因此，我们独特而有应该查看伴计划。

Code Geeks 的客座作家，并磨练

默认的嵌入式servlet容器是Tomcat。此版本的Spring Web 1.4.3引入了Tomcat版本8.5.6。

*的pom.xml*

```
01 <parent>
02   <groupId>org.springframework.boot</groupId>
03   <artifactId>spring-boot-starter-parent</artifactId>
04   <version>1.4.3.RELEASE</version>
05 </parent>
06
07 <dependencies>
08   <!-- TOMCAT -->
09   <dependency>
10     <groupId>org.springframework.boot</groupId>
11     <artifactId>spring-boot-starter-web</artifactId>
12   </dependency>
13 </dependencies>
```

## 1.2设置Spring Boot主应用程序和控制器

要设置Spring Boot应用程序，请

```
@SpringBootApplication
```

在Main类中包含注释。该

```
@SpringBootApplication
```

注释带来

```
@SpringBootConfiguration
```

```
,
@EnableAutoConfiguration
```

和

```
@ComponentScan
```

注释。

*Application.java*

```
1 @SpringBootApplication
2 @ConfigurationProperties
3 public class Application {
4   public static void main(String[] args) {
5     SpringApplication.run(Application.class, args);
6   }
}
```

您可以选择取消此注释并

```
@SpringBootConfiguration
```

单独添加或添加到允许您自定义配置的另一类。该

```
@ComponentScan
```

会扫描你的喜欢的项目的应用程序

```
@Controller
```

，你需要安装一个RESTful服务。以下控制器将从HTTP GET请求返回一个简单的"Hello World"字符串。我们还在捆绑示例中包含了另一个返回复杂对象类型的端点映射。

*SampleController.java*

```
01 @Controller
02 public class SampleController {
03
04   @Autowired
05   private ResourceLoader resourceLoader;
06
07   @RequestMapping("/")
08   @ResponseBody
09   public String home() {
10     return "Hello World!";
11   }
}
```

## 1.3关键配置参数

所有嵌入式servlet容器的默认属性都相同。要考虑的一些最重要的属性是用于配置启动信息的属性，如端口和应用程序名称，TSL，访问日志，压缩等等。

例如，要配置SSL，请将以下内容添加到application.properties的键值对。

*application.properties*

```
1 server.port=8443
2 server.ssl.key-store=classpath:keystore.jks
3 server.ssl.key-store-password=secret
4 server.ssl.key-password=another-secret
```

表达逻辑

rto.s.com

FILEX嵌入式文件系统使嵌入式物联网开发尽可能简单。

打开

## 1.4如何查找其他参数

要探索Spring启动应用程序的参数，可以将Spring执行器依赖关系和

```
@ConfigurationProperties
```

注释添加到Main类。然后

```
/configprops
```

， 您访问应用程序上的端点以获取可用属性的列表。

*Application.java*

```
1 @SpringBootApplication
2 @ConfigurationProperties
3 public class Application {
```

*的pom.xml*

```
1 <dependency>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-starter-actuator</artifactId>
4 </dependency>
```

```
1 http://localhost:8080/jcg/service/configprops
```

## 1.5更改Embedded Servlet容器的版本

嵌入式servlet容器版本在pom的以下父依赖项中定义。您可以通过显式包含依赖项并在pom中标识新版本来更改嵌入式servlet容器的版本。我们将在下面的示例中向您展示如何。

*的pom.xml*

```
1 <dependency>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-dependencies</artifactId>
4   <version>1.3.7.RELEASE</version>
5 </dependency>
```

## 汤姆猫

由于Tomcat是默认的嵌入式servlet容器，因此您无需对默认实现执行任何操作即可使用Tomcat。您可以更改正在使用的Tomcat版本或更改

pom.xml

或

application.properties

文件中的属性。

## 2.2更改Tomcat的版本

*的pom.xml*

```
01 <properties><tomcat.version>8.5.6</tomcat.version></properties>
02
03 <dependency>
04   <groupId>org.apache.tomcat.embed</groupId>
05   <artifactId>tomcat-embed-core</artifactId>
06   <version>${tomcat.version}</version>
07 </dependency>
08 <dependency>
09   <groupId>org.apache.tomcat.embed</groupId>
10   <artifactId>tomcat-embed-el</artifactId>
11   <version>${tomcat.version}</version>
12 </dependency>
13 <dependency>
14   <groupId>org.apache.tomcat.embed</groupId>
15   <artifactId>tomcat-embed-websocket</artifactId>
16   <version>${tomcat.version}</version>
17 </dependency>
```

## 3.码头

要将嵌入式servlet容器更改为Jetty，您需要编辑pom文件以删除Tomcat依赖项并添加Jetty。

### 3.1更改为Jetty （版本9.3.14）

*的pom.xml*

```
01 <dependency>
02   <groupId>org.springframework.boot</groupId>
03   <artifactId>spring-boot-starter-web</artifactId>
04   <exclusions>
05     <exclusion>
06       <groupId>org.springframework.boot</groupId>
07       <artifactId>spring-boot-starter-tomcat</artifactId>
08     </exclusion>
09   </exclusions>
```

```
10 </dependency>
11 <dependency>
12   <groupId>org.springframework.boot</groupId>
13   <artifactId>spring-boot-starter-jetty</artifactId>
14 </dependency>
```



## 4. 承诺

要将嵌入式servlet容器更改为Undertow，您需要编辑pom文件以删除Tomcat依赖项并添加Undertow。

### 4.1 更改为Undertow（最终版本1.3.24）

请注意，弹簧启动启动器中包含的下载版本不正确，参见1.3.25。您需要将其更改为1.3.24.Final，以便在本文撰写时使用。

*的pom.xml*

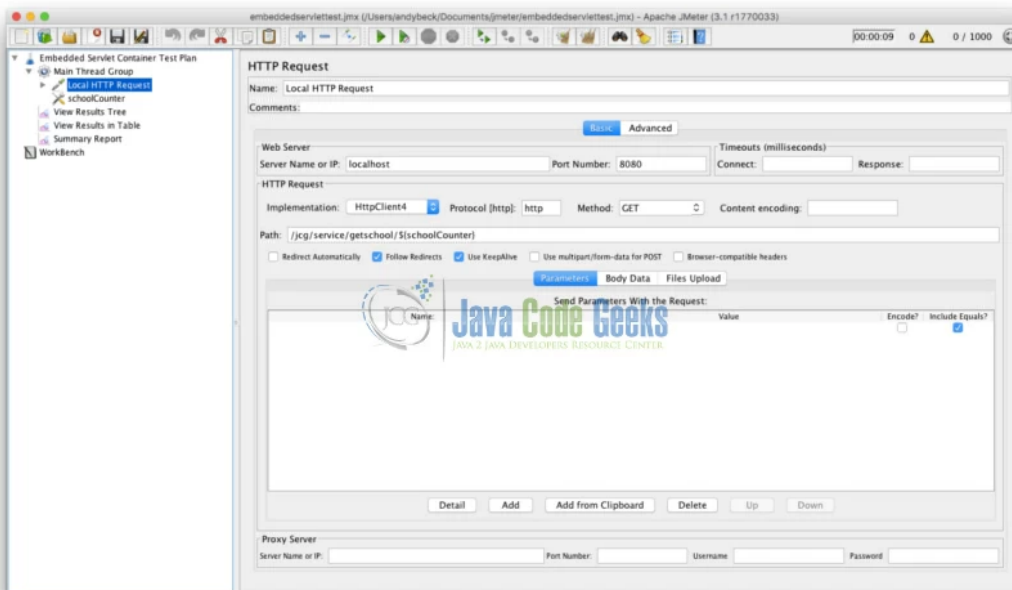
```
01 <dependency>
02   <groupId>org.springframework.boot</groupId>
03   <artifactId>spring-boot-starter-web</artifactId>
04   <exclusions>
05     <exclusion>
06       <groupId>org.springframework.boot</groupId>
07       <artifactId>spring-boot-starter-tomcat</artifactId>
08     </exclusion>
09   </exclusions>
10 </dependency>
11 <dependency>
12   <groupId>org.springframework.boot</groupId>
13   <artifactId>spring-boot-starter-undertow</artifactId>
14 </dependency>
15 <dependency>
16   <groupId>io.undertow</groupId>
17   <artifactId>undertow-core</artifactId>
18   <version>1.3.24.Final</version>
19 </dependency>
20 <dependency>
21   <groupId>io.undertow</groupId>
22   <artifactId>undertow-servlet</artifactId>
23   <version>1.3.24.Final</version>
24 </dependency>
```

## 5. 性能和负载

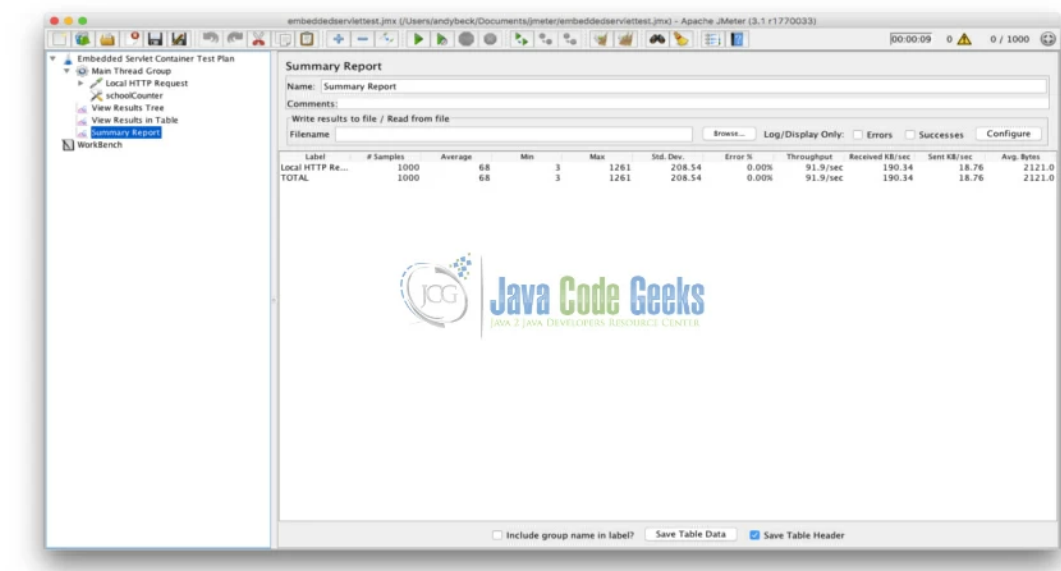
在这个例子中，我们将分析HTTP请求的性能和所有三个嵌入式servlet容器启动时的内存占用。我们使用JMeter通过模拟负载来测量性能，使用JVisualVM来查看内存占用量。

### 5.1 测量性能

在此示例中，我们将分析返回字符串的简单RESTful GET请求和返回复杂JSON对象的更复杂GET请求的性能。JMeter是用于测量三种不同类型容器性能的工具。设置此测试的关键是使用适当的负载建立线程组，计数器动态更新API的输入并报告查看器以显示或汇总结果。对于简单的字符串示例，我们使用了一个包含1000个线程的线程组，这些线程将在序列中循环3次。它还使用了10秒的加速时间。对于复杂的对象示例，我们使用相同的参数但没有循环。



JMeter Tomcat线程组



JMeter Tomcat摘要报告

5.1.1 Tomcat

5.1.1.1简单字符串

标签	#Samples	平均	敏	马克斯	标准。开发。	错误％	吞吐量	收到KB /秒	发送KB /秒	平均。字节
启动	3000	7	1	549	35.78374361	0	293.8583603	55.95935572	55.67238466	195
其他	3000	1	0	45	1.359661682	0	287.8802418	54.82094449	54.53981144	195
其他	3000	1	0	24	1.155032275	0	292.1129503	55.62697785	55.3417113	195

5.1.1.2带动态数据的复杂对象

标签	#Samples	平均	敏	马克斯	标准。开发。	错误％	吞吐量	收到KB /秒	发送KB /秒	平均。字节
启动	1000	114	3	1601	322.8671905	0	97.68486861	202.3335999	19.93763432	2121
其他	1000	3	2	17	1.328216473	0	97.88566954	202.7495167	19.9786181	2121
其他	1000	2	1	16	1.110529603	0	98.52216749	204.0678879	20.10852833	2121
其他	1000	2	1	21	1.344498419	0	98.53187506	204.0879951	20.11050966	2121

5.1.2码头

5.1.2.1简单对象

标签	#Samples	平均	敏	马克斯	标准。开发。	错误％	吞吐量	收到KB /秒	发送KB /秒	平均。字节
启动	3000	7	0	561	40.13705065	0	291.5168594	56.0828333	55.22878	197
其他	3000	1	0	21	1.058925031	0	293.5995302	56.48350338	55.6233485	197
其他	3000	1	0	21	0.926034317	0	294.3485086	56.62759395	55.7652448	197

5.1.2.2带动态数据的复杂对象

标签	#Samples	平均	敏	马克斯	标准。开发。	错误％	吞吐量	收到KB /秒	发送KB /秒	平均。字节
启动	1000	110	3	1397	278.7961107	0	98.13542689	203.3626717	19.93375859	2122
其他	1000	3	2	20	1.500210319	0	98.48335631	204.0836739	20.00443175	2122
其他	1000	3	2	45	2.729377218	0	98.29942003	203.7025091	19.96706969	2122

5.1.3承诺

5.1.3.1简单对象

标签	#Samples	平均	敏	马克斯	标准。开发。	错误％	吞吐量	收到KB /秒	发送KB /秒	平均。字节
启动	3000	6	0	451	31.6188702	0	295.6830278	63.81440346	56.01807363	221
其他	3000	1	0	22	1.255447862	0	292.7400468	63.17924839	55.46051669	221
其他	3000	1	0	18	1.559477975	0	294.3773918	63.53262069	55.77071681	221

5.1.3.2带动态数据的复杂对象



为了测量每个嵌入式servlet容器的内存，我们查看了启动时的内存使用情况。JVisualVM是随Java Development Kit提供的工具，用于可视化Java应用程序的内存和占用空间。我们使用此工具显示三个嵌入式servlet容器中每个容器的初始启动影响。堆大小和线程数是分析此初始占用空间的关键。所有三个容器共有的十个线程包括：JMX服务器连接超时，RMI调度程序，RMI TCP连接（2），RMI TCP接受，附加侦听器，DestroyJavaVM，Signal Dispatcher，Finalizer和Reference Handler。

JVisualVM报告

主题：17直播，22开始

主题：19直播，22开始

主题：17直播，20开始

```
1 Content-Type →application/json;charset=UTF-8
```

```

2 | Date →Mon, 09 Jan 2017 02:23:26 GMT
3 | Transfer-Encoding →chunked
4 | X-Application-Context →JcgSpringBootContainers:# Application index.

```

### 6.1.2码头响应标头

```

1 | Content-Type →application/json;charset=UTF-8
2 | Date →Mon, 09 Jan 2017 02:29:21 GMT
3 | Transfer-Encoding →chunked
4 | X-Application-Context →JcgSpringBootContainers:# Application index.

```

### 6.1.3 Undertow响应标头

```

1 | Connection →keep-alive
2 | Content-Type →application/json;charset=UTF-8
3 | Date →Mon, 09 Jan 2017 02:20:25 GMT
4 | Transfer-Encoding →chunked
5 | X-Application-Context →JcgSpringBootContainers:# Application index.

```

## 7.结论

这些数字表明Undertow在性能和内存使用方面表现最佳。令人鼓舞的是，Undertow正在接受最新的功能并默认持续的连接。这些数字并不表示基于此示例中使用的负载的性能有显著差异，但我认为它们会扩展，如果性能是最重要的因素，Undertow是您应用程序的正确匹配。由于熟悉它的功能，因此认为组织可能偏爱嵌入式servlet容器也是合理的。很多时候，由于包括性能，内存使用和功能在内的应用程序要求，默认设置必须更改。

## 8.下载源代码


在这里，我们比较了可以包含在Spring Boot应用程序中的三种类型的嵌入式servlet容器。

### 下载

您可以在此处下载Eclipse项目：[JcgSpringBootContainers](#)

 (尚未评级)  开始讨论  11498次播放  推文!

## 您想知道如何发展您的技能组成为Java Rockstar吗?

订阅我们的时事通讯即可立即开始Rocking    
 为了帮助您入门，我们免费为您提供最畅销的电子书！

1. JPA迷你书
2. JVM故障排除指南
3. 单元测试的JUnit教程
4. Java Annotations Tutorial
5. Java面试问题
6. 春季面试问题
7. Android UI设计

还有很多 ....

### 电子邮件地址:

☒ 在您的区域中接收Java和开发人员工作警报

[注册](#)





中国最快的VPN  
ExpressVPN.com

中国最可靠的VPN。  
94个国家的快速服务器。24/7实时聊天支持

[打开](#)

喜欢这篇文章？阅读更多来自[JAVA CODE GEEKS](#)



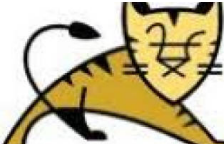
广告



Creating Web Services and a Rest Server with JAX-RS and Jetty



WAR files vs. Java apps with embedded servers



Apache Tomcat Vs Nginx Comparison



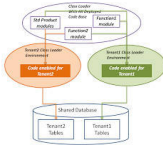
Docker Tutorial for Java Developers



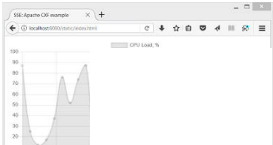
Spring Integration Full Example



Elasticsearch Tutorial




Architecting a Multi-tenant Application



Better SSE, or Events,

### 发表评论

 Start the discussion...

📧 订阅 ▼

#### 知识库

课程

也常被

新闻

资源

教程

#### CODE GEEKS网络

.NET Code Geeks

Java Code Geeks

系统代码极客

网络代码极客

#### 名人堂

Android警报对话框示例

Android OnClickListener示例

如何在Java中将字符转换为字符串，将字符串转换为字符数组

Java继承示例

Java写入文件示例

java.io.FileNotFoundException - 如何解决找不到文件的异常

java.lang.arrayindexoutofboundsexception - 如何处理数组索引超出界限异常

java.lang.NoClassDefFoundError - 如何解决No Class Def Found错误

泽西+杰克逊的JSON示例

Spring JdbcTemplate示例

#### 关于JAVA CODE GEEKS

JCGs (Java Code Geeks) 是一个独立的在线社区，专注于创建最终的Java资源中心；针对技术架构师，技术团队负责人（高级开发人员），项目人员。JCG通过领域专家撰写的每日新闻，文章，教程，评论，公告，代目为Java，SOA，敏捷和电信社区提供服务。

#### 放弃

Java Code Geeks上出现的所有商标和注册商标均为其各自所有者的财产。Corporation在美国和其他国家/地区的商标或注册商标。示例Java Code Corporation连接，也未由Oracle Corporation赞助。

