使用嵌入式 Jetty 应用服务器托管 Spring Boot Web 应用程序

韩博孙
2019 年 2 月 13 日 麻省理工学院

评价我: 5.00/5 (2 票)

在本教程中，我将向读者展示如何使用 Spring Boot 创建 MVC 应用程序，打包为自托管 WAR 文件，并使用嵌入式 Jetty 应用服务器执行。

下载演示项目 - 6.8 KB

介绍

2018 年 4 月，我写了一篇关于使用 Spring Boot 创建 Web 应用程序的快速教程文章，打包在 WAR 存档中。它与嵌入式 Tomcat 应用程序服务器一起运行。本文将使用相同的代码库，但由嵌入式 Jetty 应用服务器托管。你可能会问，重复同一篇文章有什么意义？重要的是当你完成本教程后，你可以轻松地换出嵌入式 Tomcat 服务器并使用嵌入式 Jetty 服务器。有选择总是比被一个选择困住要好。

快速查看源代码

我没有对孪生文章中的实际源代码进行任何更改。如果您很懒惰并且不想在本教程和其他教程之间切换，请快速浏览本节，然后我们将跳转到有关将嵌入式 Tomcat 服务器替换为嵌入式 Jetty 服务器的好内容。

这是主条目的代码：

爪哇 复制代码

```
package org.hanbo.boot.app;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

@SpringBootApplication
public class App extends SpringBootServletInitializer
{
   @Override
   protected SpringApplicationBuilder configure(SpringApplicationBuilder appBuilder)
   {
      return appBuilder.sources(App.class);
```

```
    }

    public static void main(String[] args) throws Exception
    {
        SpringApplication.run(App.class, args);
    }
}
```

主条目将应用程序设置为作为 Web 应用程序运行。类扩展自`SpringBootServletInitializer`它，允许将应用程序打包到 WAR 存档中。嵌入式服务器将使用 WAR 归档文件结构来运行应用程序。它是托管 Spring MVC 应用程序的理想方法，具有 JSP 模板等。

每个 MVC 应用程序都有一个控制器。这里是：

爪哇 复制代码

```
package org.hanbo.boot.app.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class HelloController
{
    @RequestMapping(value="/meow", method = RequestMethod.GET)
    public ModelAndView hello(
        @RequestParam("sayit")
        String sayit
    )
    {
        ModelAndView retVal = new ModelAndView();
        retVal.setViewName("testme");
        retVal.addObject("mymessage", sayit);
        return retVal;
    }
}
```

这个控制器所做的就是设置回显一些输入的网页。这是一个网址：

复制代码

```
http://localhost:8080/meow?sayit=This+is+pretty+crazy
```

只要应用程序运行，并且当您将 URL 粘贴到浏览器中并按**Enter 时**，您将看到一个页面显示：

复制代码

```
What did you say?

I said: "This is pretty crazy."
```

我需要让 Web 应用程序知道在哪里可以找到 JSP 模板，即名为“ `testme`”的视图名称。我需要让 Web 应用程序知道视图模板文件的文件扩展名。我已经在`application.properties`：

爪哇 复制代码

```
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
```

这看起来很熟悉，对吧？只是一个提示：
`org.springframework.web.servlet.view.InternalResourceViewResolver`。

最后，视图模板 JSP 文件名为“ *testme.jsp* ”：

jsp 复制代码

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html lang="en">
<head>
   <c:url value="/assets/css/index.css" var="jstlCss" />
   <link href="${jstlCss}" rel="stylesheet" />
</head>
<body>
   <p>What did you say?</p>
   <p>I said: <span class="text-underline">"${mymessage}."</span></p>
   <script type="text/javascript" src="/assets/js/test.js">
   </script>
</body>
</html>
```

这个 Web 应用程序还包含一些静态内容，可以在*src/main/resources/static/**目录中找到。按照惯例，Spring Boot 在此处查找静态内容。

不管怎样，我稍后会描述如何测试这个应用程序。是时候展示好东西了。要使应用程序与嵌入式 Jetty 服务器一起成功运行，我必须进行的唯一更改是在 maven POM 文件中。

Jetty 的 Maven POM 文件

Maven POM 文件用于依赖管理、构建和 WAR 打包。默认情况下，它会将嵌入式Tomcat服务器打包在WAR存档中，即WAR存档中的所有jar依赖项。现在如果我需要用内嵌的 Jetty 服务器替换内嵌的 Tomcat 服务器，我需要首先排除所有的 Tomcat 内嵌服务器 jar 并添加内嵌的 Jetty 服务器 jar。因为 Jetty 不是 Spring Boot 使用的默认应用程序容器，并且使用 Spring MVC，为此 Spring 应用程序启用了 JSP/JSTL。在当时，这一切都非常压倒性。这就是为什么我记录下来以节省每个人的麻烦，我喜欢这个挑战。现在我将告诉你这是如何做到的。

我做的第一件事是排除 Tomcat 服务器 jars。这是我如何做到的：

XML 复制代码

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
   <exclusions>
      <exclusion>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-tomcat</artifactId>
      </exclusion>
   </exclusions>
</dependency>
```

Artifact "`spring-boot-starter-web`" 是包含了 web 应用程序所需的所有 jar 文件，其中也包含了 Tomcat。这就是为什么我需要有一个小部分来排除工件“spring-boot-starter-tomcat”。这不是我自己想出来的。您可以在 Baeldung 提供的教程之一中找到它。接下来是添加自托管 Jetty 依赖项。这里是：

XML 复制代码

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

再说一次，这不是我自己想出来的。如果你猜，是的，来自 Baeldung 提供的同一教程。然而，这就是我能从 Baeldung 得到的。我刚刚完成的两个动作仅支持基本的 Web 应用程序功能。如果您像这样运行应用程序而没有额外的 jars，该应用程序将无法运行。它对我这里的小应用程序没有帮助，因为我需要 JSP/JSTL 支持，而且我需要将应用程序打包为 WAR 文件。所以通过一些研究，我想出了一个可行的解决方案。它是如此简单，简直让我大吃一惊。此解决方案是添加另外三个工件，如下所示：

XML 复制代码

```
<dependency>
   <groupId>javax.servlet</groupId>
   <artifactId>jstl</artifactId>
</dependency>
<dependency>
   <groupId>org.eclipse.jetty</groupId>
   <artifactId>apache-jsp</artifactId>
</dependency>
<dependency>
   <groupId>org.eclipse.jetty</groupId>
   <artifactId>apache-jstl</artifactId>
</dependency>
```

最重要的是最后两个工件，一旦你把它放在 POM 文件中，它就会为嵌入式 Jetty 服务器带来所有必要的 jar 文件。在继续之前，请仔细查看整个 maven POM 文件。在其中，您可能会找到将文件打包为war 的方式，并使用Spring Boot 构建插件来构建应用程序。现在是运行应用程序的时候了。

测试应用程序

在构建和运行应用程序之前，请进入静态 Web 内容文件夹并将所有*.sj文件重命名为*.js文件。

要构建应用程序，首先转到项目的基本目录（POM 文件所在的位置），然后使用以下 Maven 构建命令：

复制代码

```
mvn clean install
```

在命令行提示符下**回车**后，会下载所有的依赖，需要一定的时间，耐心等待。构建成功后，您只需一个命令即可运行应用程序。假设你仍然在项目的基目录下，输入命令：

复制代码

```
java -jar target\boot-war-1.0.0.war
```

命令运行的时候，会吐出很多废话，找这样的段：

缩小▲ 复制代码

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.5.RELEASE)

2019-02-11 23:06:59.055  INFO 12404 --- [           main] org.hanbo.boot.app.App
                   : Starting App v1.0.0 on U3DTEST-PC with PID 12404 (C:\Users\
u3dadmin\workspace-mars8\SpringBootJettyWar\target\boot-war-1.0.0.war started by
 u3dadmin in C:\Users\u3dadmin\workspace-mars8\SpringBootJettyWar)
2019-02-11 23:06:59.132  INFO 12404 --- [           main] org.hanbo.boot.app.App
                   : No active profile set, falling back to default profiles: de
fault
2019-02-11 23:06:59.584  INFO 12404 --- [           main] ConfigServletWebServer
ApplicationContext : Refreshing org.springframework.boot.web.servlet.context.Ann
otationConfigServletWebServerApplicationContext@6aceb1a5: startup date [Mon Feb
11 23:06:59 EST 2019]; root of context hierarchy
2019-02-11 23:07:04.389  INFO 12404 --- [           main] o.s.b.w.e.j.JettyServl
etWebServerFactory : Server initialized with port: 8080

....

2019-02-11 23:07:09.103  INFO 12404 --- [           main] o.e.jetty.server.handl
er.ContextHandler  : Started o.s.b.w.e.j.JettyEmbeddedWebAppContext@6293abcc{app
lication,/,[org.springframework.boot.web.embedded.jetty.JettyServletWebServerFac
```

```
tory$LoaderHidingResource@7995092a],AVAILABLE}
2019-02-11 23:07:09.106  INFO 12404 --- [           main] org.eclipse.jetty.serv
er.Server          : Started @16076ms

....

2019-02-11 23:07:10.454  INFO 12404 --- [           main] o.e.j.s.h.ContextHandl
er.application     : Initializing Spring FrameworkServlet 'dispatcherServlet'
2019-02-11 23:07:10.454  INFO 12404 --- [           main] o.s.web.servlet.Dispat
cherServlet        : FrameworkServlet 'dispatcherServlet': initialization starte
d
2019-02-11 23:07:10.479  INFO 12404 --- [           main] o.s.web.servlet.Dispat
cherServlet        : FrameworkServlet 'dispatcherServlet': initialization comple
ted in 20 ms
2019-02-11 23:07:10.844  INFO 12404 --- [           main] o.e.jetty.server.Abstr
actConnector       : Started ServerConnector@1bd4fdd{HTTP/1.1,[http/1.1]}{0.0.0.
0:8080}
2019-02-11 23:07:10.849  INFO 12404 --- [           main] o.s.b.web.embedded.jet
ty.JettyWebServer  : Jetty started on port(s) 8080 (http/1.1) with context path
'/'
2019-02-11 23:07:10.854  INFO 12404 --- [           main] org.hanbo.boot.app.App
                   : Started App in 15.3 seconds (JVM running for 17.828)
```

第一段是应用程序运行的开始。在最后的某个地方，您可以看到嵌入式 Jetty 服务器初始化。在输出的中间，有更多的 Jetty 服务器正在启动。最后，我们看到应用程序在 8080 端口上运行并准备接受用户请求。如果您没有看到任何异常输出，并且看到最后一行，则您已成功启动应用程序。

要测试应用程序，只需将以下链接复制并粘贴到浏览器：

复制代码

```
http://localhost:8080/meow?sayit=This+is+pretty+crazy
```

然后你会看到一个简单的页面显示：

复制代码

```
What did you say?

I said: "This is pretty crazy."
```

或者尝试一些新的东西：

复制代码

```
http://localhost:8080/meow?sayit=Hello+World
```

然后你会看到一个简单的页面显示：

复制代码

```
What did you say?

I said: "Hello World."
```

概括

这是一个关于 Spring Boot 的非常直接的教程。我拿了去年提交的一篇文章，去掉了嵌入式Tomcat服务器的使用，取而代之的是嵌入式Jetty服务器。这是相当简单的，所有需要做的就是排除神器“ `spring-boot-starter-tomcat` ”。然后为嵌入式 Jetty 服务器添加等效的工件。要为 Spring MVC 支持 JSP/JSTL，必须为 JSP/JSTL 添加至少两个与 Jetty 相关的工件。一旦您知道如何执行此操作，您就可以使用相同的方法替换我 2018 年 Spring Boot 教程示例项目中的所有嵌入式 Tomcat 服务器。享受吧！

历史

- 02/11/2019 - 初稿

执照

分享

关于作者

韩博孙

组长 The Judge Group

美国

手表该会员

没有提供传记

评论和讨论

添加评论或问题 电子邮件提醒 Search Comments

-- 本论坛暂无消息 --