黑客派

SpringBoot 入门

作者：Javen

原文链接：https://hacpai.com/article/1494493052132

来源网站：黑客派

<h2 id="简介">简介</h2>

在您第1次接触和学习Spring框架的时候，是否因为其繁杂的配置而退却了？在你第n次使用Sprig框架的时候，是否觉得一堆反复黏贴的配置有一些厌烦？那么您就不妨来试试使用Spring Boot来让更易上手，更简单快捷地构建Spring应用！
Spring Boot让我们的Spring应用变的更轻量化。比如：你可以仅仅依靠一个Java类来运行一个Sring引用。你也可以打包你的应用为jar并通过使用java -jar来运行你的Spring Web应用。
Spring Boot的主要优点：

 为所有Spring开发者更快的入门
 开箱即用，提供各种默认配置来简化项目配置
 内嵌式容器简化Web项目
 没有冗余代码生成和XML配置的要求

<hr>
<h2 id="快速入门">快速入门</h2>
本章主要目标完成Spring Boot基础项目的构建，并且实现一个简单的Http请求处理，通过这个子对Spring Boot有一个初步的了解，并体验其结构简单、开发快速的特性。
<h2 id="系统要求：">系统要求：</h2>

 Java 7及以上
 Spring Framework 4.1.5及以上

本文采用<code>Java 1.8.0_73</code>、<code>Spring Boot 1.3.2</code>调试过。
<h2 id="使用Maven构建项目">使用Maven构建项目</h2>

 通过<code>SPRING INITIALIZR</code>工具产生基础项目

   访问：<code>http://start.spring.io/</code>
   选择构建工具<code>Maven Project</code>、Spring Boot版本<code>1.3.2</code>以一些工程基本信息，可参考下图所示 

SPRING INITIALIZR

点击<code>Generate Project</code>下载项目压缩包

解压项目包，并用IDE以<code>Maven</code>项目导入，以<code>IntelliJ IDEA 14</cod
>为例：

菜单中选择<code>File</code>–><code>New</code>–><code>Project from Exi
ting Sources...</code>
选择解压后的项目文件夹，点击<code>OK</code>
点击<code>Import project from external model</code>并选择<code>Maven</code>
点击<code>Next</code>到底为止。
若你的环境有多个版本的JDK，注意到选择<code>Java SDK</code>的时候请选择<code>Ja
a 7</code>以上的版本

<h2 id="项目结构解析"><a href="https://link.hacpai.com/forward?goto=http%3A%2F%2Fblog
didispace.com%2Fspring-boot-learning-1%2F%23%E9%A1%B9%E7%9B%AE%E7%BB%93%E
%9E%84%E8%A7%A3%E6%9E%90" class="headerlink" title="项目结构解析" target="_blank" re
="nofollow ugc">项目结构解析</h2>
<a href="https://link.hacpai.com/forward?goto=http%3A%2F%2Fblog.didispace.com%2F
ontent%2Fimages%2F2016%2F02%2Fchapter1-2.png" title="项目结构" class="gallery-item" ta
get="_blank" rel="nofollow ugc"><img src="https://static.hacpai.com/images/img-loading.s
g" alt="项目结构" data-src="http://blog.didispace.com/content/images/2016/02/chapter1-2.p
g">项目结构
通过上面步骤完成了基础项目的创建，如上图所示，Spring Boot的基础结构共三个文件（具体
径根据用户生成项目时填写的Group所有差异）：

<code>src/main/java</code>下的程序入口：<code>Chapter1Application</code>
<code>src/main/resources</code>下的配置文件：<code>application.properties</code

<code>src/test/</code>下的测试入口：<code>Chapter1ApplicationTests</code>

生成的<code>Chapter1Application</code>和<code>Chapter1ApplicationTests</code
类都可以直接运行来启动当前创建的项目，由于目前该项目未配合任何数据访问或Web模块，程序会
加载完Spring之后结束运行。
<h2 id="引入Web模块"><a href="https://link.hacpai.com/forward?goto=http%3A%2F%2Fblo
.didispace.com%2Fspring-boot-learning-1%2F%23%E5%BC%95%E5%85%A5Web%E6%A8%A
%E5%9D%97" class="headerlink" title="引入Web模块" target="_blank" rel="nofollow ugc"><
a>引入Web模块</h2>
当前的<code>pom.xml</code>内容如下，仅引入了两个模块：

<code>spring-boot-starter</code>：核心模块，包括自动配置支持、日志和YAML
<code>spring-boot-starter-test</code>：测试模块，包括JUnit、Hamcrest、Mockito</li

<table>

```
<tbody>
 <tr>
  <td>
   <div>
    1
   </div>
   <div>
    2
   </div>
   <div>
    3
   </div>
   <div>
    4
   </div>
   <div>
    5
   </div>
   <div>
    6
   </div>
   <div>
    7
   </div>
   <div>
    8
   </div>
   <div>
    9
   </div>
   <div>
    10
   </div>
   <div>
    11
   </div>
   <div>
    12
   </div> </td>
  <td>
   <div>
    <span>&lt;<span>dependencies</span>&gt;</span>
   </div>
   <div>
    <span>&lt;<span>dependency</span>&gt;</span>
   </div>
   <div>
    <span>&lt;<span>groupId</span>&gt;</span>org.springframework.boot
    <span>&lt;/<span>groupId</span>&gt;</span>
   </div>
   <div>
    <span>&lt;<span>artifactId</span>&gt;</span>spring-boot-starter
    <span>&lt;/<span>artifactId</span>&gt;</span>
   </div>
```

</dependency>

<dependency>

<groupId>org.springframework.boot
</groupId>

<artifactId>spring-boot-starter-test
</artifactId>

<scope>test
</scope>

</dependency>

</dependencies>
 </td>
</tr>
</tbody>
</table>
引入Web模块，需添加<code>spring-boot-starter-web</code>模块：
<table>
<tbody>
<tr>
<td>

1

2

3

4
 </td>
<td>

<dependency>

<groupId>org.springframework.boot
</groupId>

```
    </div>
    <div>
     <span>&lt;<span>artifactId</span>&gt;</span>spring-boot-starter-web
     <span>&lt;/<span>artifactId</span>&gt;</span>
    </div>
    <div>
     <span>&lt;/<span>dependency</span>&gt;</span>
    </div> </td>
  </tr>
 </tbody>
</table>
<h2 id="编写HelloWorld服务"><a href="https://link.hacpai.com/forward?goto=http%3A%2F2Fblog.didispace.com%2Fspring-boot-learning-1%2F%23%E7%BC%96%E5%86%99HelloWord%E6%9C%8D%E5%8A%A1" class="headerlink" title="编写HelloWorld服务" target="_blank" rl="nofollow ugc"></a>编写HelloWorld服务</h2>
<ul>
 <li>创建<code>package</code>命名为<code>com.didispace.web</code>（根据实际情况改）</li>
 <li>创建<code>HelloController</code>类，内容如下</li>
</ul>
<table>
 <tbody>
  <tr>
   <td>
    <div>
     1
    </div>
    <div>
     2
    </div>
    <div>
     3
    </div>
    <div>
     4
    </div>
    <div>
     5
    </div>
    <div>
     6
    </div>
    <div>
     7
    </div>
    <div>
     8
    </div>
    <div>
     9
    </div> </td>
   <td>
    <div>
     <span>@RestController</span>
```

```
    </div>
    <div>
     <span>public</span>
     <span><span>class</span> <span>HelloController</span> </span>{
    </div>
    <div>
       
    </div>
    <div>
     <span>@RequestMapping</span>(
     <span>"/hello"</span>)
    </div>
    <div>
     <span><span>public</span> String <span>index</span><span>()</span> </span>{
    </div>
    <div>
     <span>return</span>
     <span>"Hello World"</span>;
    </div>
    <div>
      }
    </div>
    <div>
       
    </div>
    <div>
      }
    </div> </td>
  </tr>
 </tbody>
</table>
<ul>
 <li>启动主程序，打开浏览器访问<code>http://localhost:8080/hello</code>，可以看到页面输
<code>Hello World</code></li>
</ul>
<h2 id="编写单元测试用例"><a href="https://link.hacpai.com/forward?goto=http%3A%2F%2
blog.didispace.com%2Fspring-boot-learning-1%2F%23%E7%BC%96%E5%86%99%E5%8D%9
%E5%85%83%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B" class="headerlink" title="
写单元测试用例" target="_blank" rel="nofollow ugc"></a>编写单元测试用例</h2>
<p>打开的<code>src/test/</code>下的测试入口<code>Chapter1ApplicationTests</code>
。下面编写一个简单的单元测试来模拟http请求，具体如下：</p>
<table>
 <tbody>
  <tr>
   <td>
    <div>
     1
    </div>
    <div>
     2
    </div>
    <div>
     3
    </div>
```

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20
 </td>
<td>

@RunWith(SpringJUnit4ClassRunner.class)

```
</div>
<div>
 <span>@SpringApplicationConfiguration</span>(classes = MockServletContext.class)
</div>
<div>
 <span>@WebAppConfiguration</span>
</div>
<div>
 <span>public</span>
 <span><span>class</span> <span>Chapter1ApplicationTests</span> </span>{
</div>
<div>
   
</div>
<div>
 <span>private</span> MockMvc mvc;
</div>
<div>
   
</div>
<div>
 <span>@Before</span>
</div>
<div>
 <span><span>public</span> <span>void</span> <span>setUp</span><span>()</spa
> <span>throws</span> Exception </span>{
</div>
<div>
  mvc = MockMvcBuilders.standaloneSetup(
 <span>new</span> HelloController()).build();
</div>
<div>
  }
</div>
<div>
   
</div>
<div>
 <span>@Test</span>
</div>
<div>
 <span><span>public</span> <span>void</span> <span>getHello</span><span>()</s
an> <span>throws</span> Exception </span>{
</div>
<div>
  mvc.perform(MockMvcRequestBuilders.get(
 <span>"/hello"</span>).accept(MediaType.APPLICATION_JSON))
</div>
<div>
  .andExpect(status().isOk())
</div>
<div>
  .andExpect(content().string(equalTo(
 <span>"Hello World"</span>)));
```

```
    </div>
    <div>
     }
    </div>
    <div>
      
    </div>
    <div>
     }
    </div> </td>
  </tr>
 </tbody>
</table>
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<p>使用<code>MockServletContext</code>来构建一个空的<code>WebApplicationContext<
code>，这样我们创建的<code>HelloController</code>就可以在<code>@Before</code>函
中创建并传递到<code>MockMvcBuilders.standaloneSetup（）</code>函数中。</p>
<ul>
 <li>注意引入下面内容，让<code>status</code>、<code>content</code>、<code>equalT
</code>函数可用</li>
</ul>
<table>
 <tbody>
  <tr>
   <td>
    <div>
     1
    </div>
    <div>
     2
    </div>
    <div>
     3
    </div> </td>
   <td>
    <div>
     <span>import</span>
     <span>static</span> org.hamcrest.Matchers.equalTo;
    </div>
    <div>
     <span>import</span>
     <span>static</span> org.springframework.test.web.servlet.result.MockMvcResultMatchers
content;
    </div>
    <div>
     <span>import</span>
```

static org.springframework.test.web.servlet.result.MockMvcResultMatchers status;
 </td>
</tr>
</tbody>
</table>
至此已完成目标，通过Maven构建了一个空白Spring Boot项目，再通过引入web模块实现了一简单的请求处理。