第02课：快速实战 Spring Boot

什么是 Spring Boot

Spring 在官方首页这样介绍：

> BUILD ANYTHING . Spring Boot is designed to get you up and running as quickly as possible, with minimal upfront configuration of Spring. Spring Boot takes an opinionated view of building production ready applications.

解释一下：Spring Boot 可以构建一切。Spring Boot 设计之初就是为了最少的配置，最快的速度来启动和运行 Spring 项目。Spring Boot 使用特定的配置来构建生产就绪型的项目。

使用 Spring Boot 有什么好处

其实就是简单、快速、方便！如果搭建一个 Spring Web 项目的时候需要怎么做呢？

- 配置 web.xml，加载 Spring 和 Spring MVC
- 配置数据库连接、配置 Spring 事务
- 加载配置文件的读取，开启注解
- 配置日志文件
- …
- 配置完成之后部署 Tomcat 调试
- …

现在非常流行微服务，如果我这个项目仅仅只是需要发送一个邮件，如果我的项目仅仅是生产一个积分；我都需要这样折腾一遍!

但是如果使用 Spring Boot 呢？很简单，仅仅只需要三步就可以快速的搭建起一个 Web 项目！

使用 Spring Boot 到底有多爽，用下面这幅图来表达：

惊呆了！！

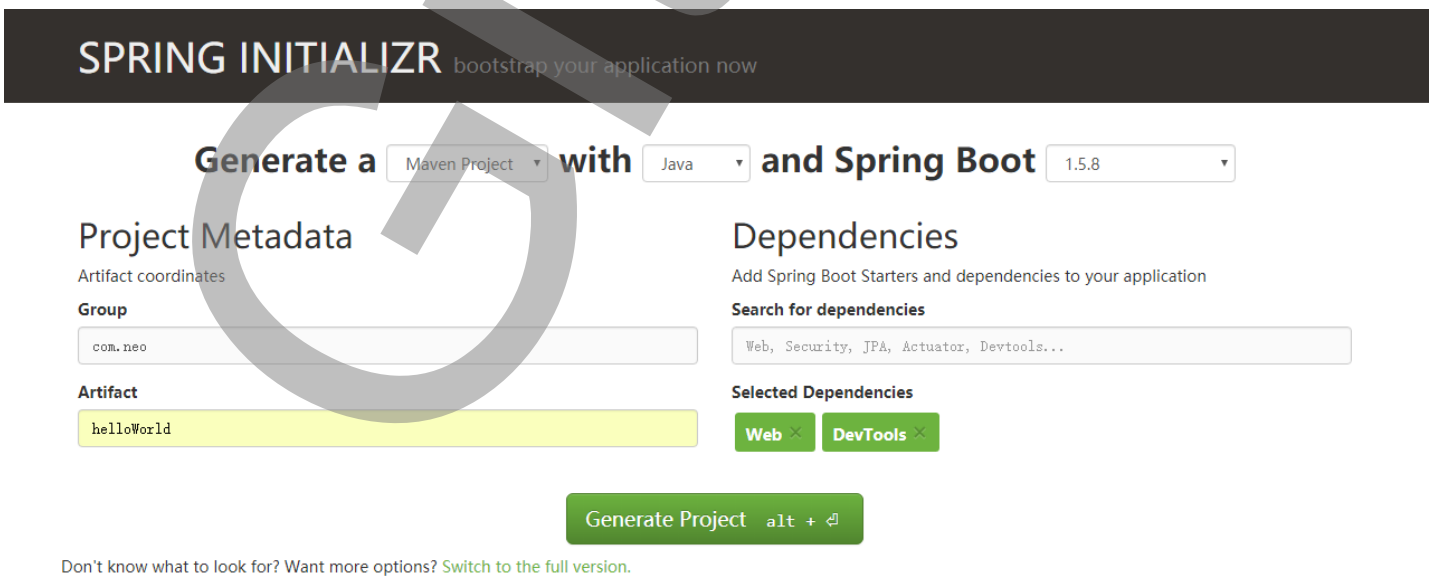
快速入门

说了那么多，手痒痒的很，马上来一发试试!

构建项目

（1）访问 http://start.spring.io/。

（2）选择构建工具 Maven Project、Spring Boot 版本 1.5.8 及一些工程基本信息，可参考下图：



（3）单击 Generate Project 按钮并下载项目压缩包。

（4）解压后，单击 Eclipse，Import | Existing Maven Projects | Next | 选择解压后的文件夹 | Finsh 命令，OK Done!

（5）如果使用的是 Idea，单击 File | New | Model from Existing Source.. | 选择解压后的文件夹 | OK 命令，选择 Maven ，一路 Next，OK Done!

如果读者使用的是 Idea 工具，也可以这样：

（1）单击 File | New | Project… 命令，弹出新建项目框。

（2）选择 Spring Initializr 选项，单击 Next 按钮，也会出现上述类似的配置界面，Idea 帮我们做了集成。

（3）填写相关内容后，单击 Next 按钮，选择依赖的包再单击 Next 按钮，最后确定信息无误单击 Finish 按钮。

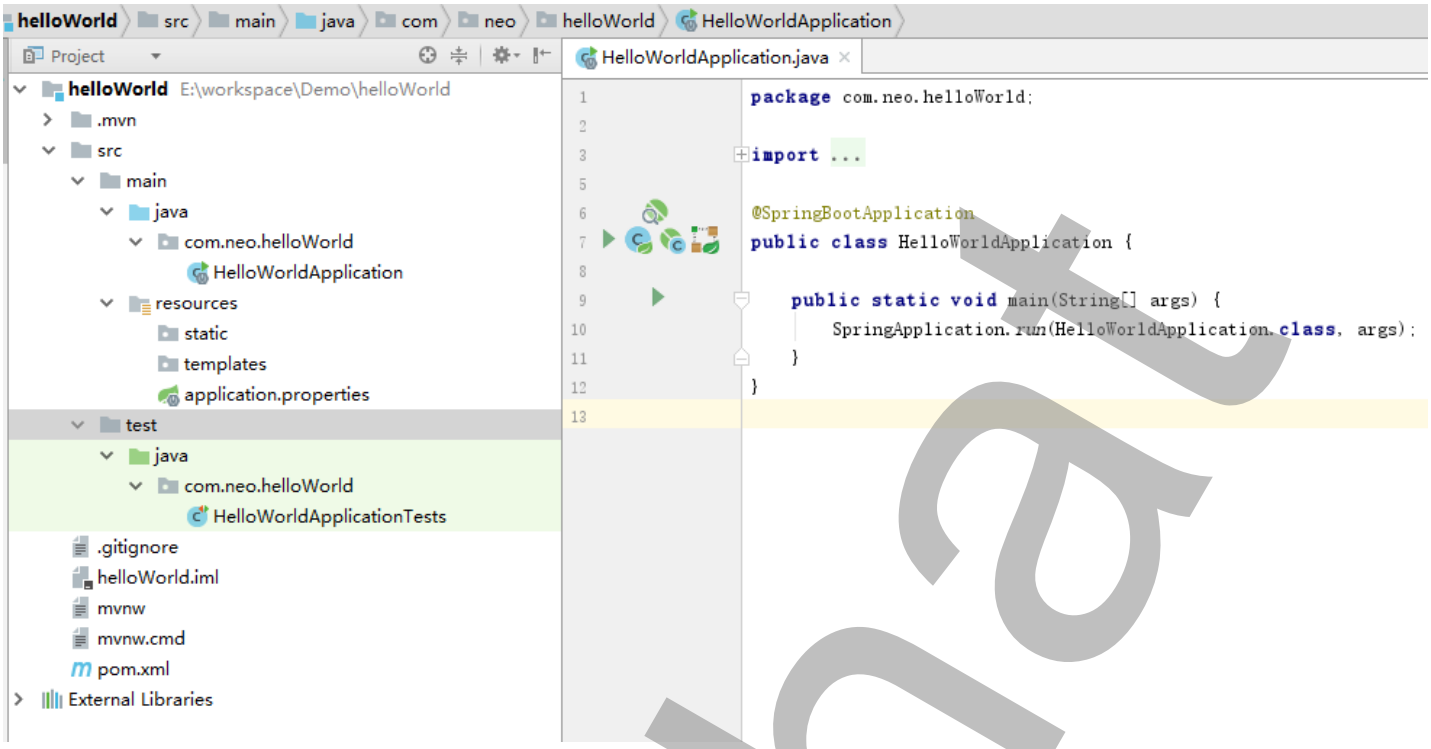
对上面的配置做一个解释：

- 第一个选择框选择创建以 Maven 构建项目，还是以 Gradle 构建项目，这是两种不同的构建方式，其中 Gradel 配置内容更简洁一些，并且包含了 Maven 的使用，不过日常使用 Maven 居多。
- 第二个选择框选择编程语言，现在支持 Java、Kotlin 和 Groovy。
- 第三个选择框选择 Spring Boot 版本，可以看出 Spring Boot 2.0 已经到了第五个里程碑了。在实际使用中，我们会优先使用稳定版本，1.0 的最新稳定版本是 1.5.8，也是我们演示使用的版本。

下面就是项目的配置信息了。

- Group：一般填写公司域名，比如百度公司填 com.baidu，演示使用 com.neo。
- Artifact：可以理解为项目的名称，可以根据实际情况来填，本次演示填写 helloWorld。
- Dependencies：在这块添加我们项目所依赖的 Spring Boot 组件，可以多选。本次选择 Web、Devtools 两个模块。

项目结构介绍



如上图所示，Spring Boot 的基础结构共三个文件：

- src/main/java：程序开发以及主程序入口
- src/main/resources：配置文件
- src/test/java：测试程序

另外，Sping Boot 建议的目录结果如下：

root package 结构：`com.example.myproject`

```
myproject
 +-src
    +- main
         +- java
              +- com.example.myproject
                   +- comm
                   +- domain
                   +- repository
                   +- service
                   +- web
                   +- Application.java
         +- resources
              +- static
              +- templates
              +- application.properties
    +- test
 +-pom.xml
```

com.example.myproject 目录下：

- Application.java：建议放到根目录下面，是项目的启动类，Spring Boot 项目只能有一个 main() 方法。
- comm：目录建议放置公共的类，如全局的配置文件、工具类等。
- domain：目录主要用于实体（Entity）与数据访问层（Repository）。
- repository：数据库访问层代码。
- service：该层主要是业务类代码。
- web：该层负责页面访问控制。

resources 目录下：

- static：目录存放 Web 访问的静态资源，如 JS、CSS、图片等。
- templates：目录存放页面模板。
- application.properties：项目的配置信息。

test 目录存放单元测试的代码；pom.xml 用于配置项目依赖包，以及其他配置。

采用默认配置可以省去很多设置，当然也可以根据自己的喜好来进行更改。最后，启动 Application main 方法，至此一个 Java 项目搭建好了！

简单 Web 开发

（1）可以在 Spring Initializr 上面添加，也可以手动在 pom.xml 中添加：

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

pom.xml 文件中默认有两个模块：

- spring-boot-starter：核心模块，包括自动配置支持、日志和 YAML；
- spring-boot-starter-test：测试模块，包括 JUnit、Hamcrest、Mockito。

（2）编写 controller 内容：

```
@RestController
public class HelloWorldController {

    @RequestMapping("/hello")
    public String hello() {
        return "Hello World";
    }

}
```

`@RestController` 的意思就是 controller 里面的方法都以 json 格式输出，不用再配置什么 jackjson 的了！

如果配置为 `@Controller` 就代表着输出为页面内容。

（3）启动主程序，打开浏览器访问 http://localhost:8080/hello，就可以看到以下内容，是不是很简单！

```
Hello World
```

（4）如果我们想传入参数怎么办？

```
@RestController
public class HelloWorldController {

    @RequestMapping("/hello")
    public String index(String name) {
        return "Hello World, " +name;
    }
}
```

重新启动项目，访问 http://localhost:8080/hello?name=neo，返回内容如下：

```
Hello World, neo
```

经过上一个测试发现，修改 controller 内相关代码，就需要重新启动项目才能生效，这样做很麻烦是不是，别着急。Spring Boot 提供了另外一个组件来解决。

热部署

热启动就需要用到我们在一开始引入的另外一个组件：Devtools。它是 Spring Boot 提供的一组开发工具包，其中就包含我们需要的热部署功能。但是在使用这个功能之前还需要再做一些配置。

（1）在 dependency 中添加 optional 属性，并设置为 true：

```
 <dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-Devtools</artifactId>
        <optional>true</optional>
    </dependency>
</dependencies>
```

（2）在 plugin 中配置另外一个属性 fork，并且配置为 true：

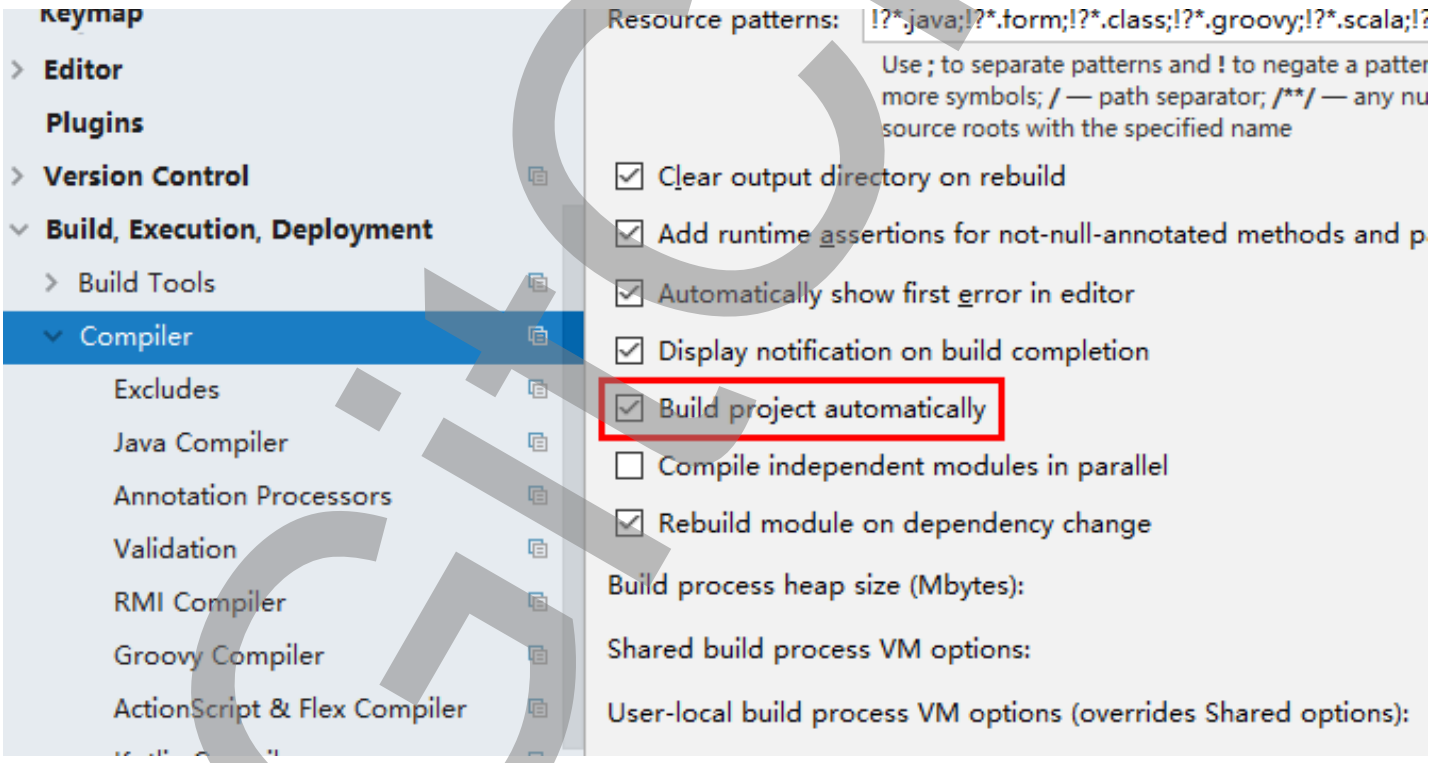
```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <fork>true</fork>
            </configuration>
        </plugin>
</plugins>
</build>
```

OK，以上两步配置完成，如果读者使用的是 Eclipse，那么恭喜你大功告成了。

如果读者使用的是 Idea 还需要做以下配置。

（3）配置 Idea

选择 File-Settings-Compiler 勾选 `Build project automatically`，低版本 Idea 勾选 `make project automatically`。



使用快捷键：`CTRL + SHIFT + A` 输入 `Registry` 找到选项 `compile.automake.allow.when.app.running` 勾选

全部配置完成后，Idea 就支持热部署了，大家可以试着去改动一下代码就会发现 Spring Boot 会自动重新加

载，再也不需要我们手动点击重新部署了。

为什么 Idea 需要多配置后面这一步呢，因为 Idea 默认不是自动编译的，需要我们手动去配置后才会自动编译，而热部署依赖于项目的自动编译功能。

> 该模块在完整的打包环境下运行的时候会被禁用。如果使用 java -jar 启动应用或者用一个特定的 classloader 启动，它会认为这是一个“生产环境”。

单元测试

单元测试在日常开发中是必不可少的，一个牛逼的程序员，单元测试写得也是杠杠的。下面来看下 Spring Boot 对单元测试又做了哪些支持?

如果我们只想运行一个 hello World，只需要一个注解就可以。在 src/test 目录下新建一个 HelloTests 类，代码如下:

```
public class HelloTest {
    @Test
    public void hello(){
        System.out.println("hello world");
    }
}
```

单击右键“运行”按钮，会发现控制台输出：hello world。仅仅只需要了一个注解。但是如果我们需要测试 web 层的请求呢？Spring Boot 也给出了支持。

以往我们在测试 web 请求的时候，需要手动输入相关参数在页面测试查看效果，或者自己写 post 请求。在 Spring Boot 中，Spring 给出了一个简单的解决方案；使用 mockmvc 进行 web 测试，mockmvc 内置了很多工具类和方法，可以模拟 post、get 请求，并且判断返回的结果是否正确等，也可以利用 `print()` 打印执行结果。

```
@SpringBootTest
public class HelloTest {

    private MockMvc mockMvc;

    @Before
    public void setUp() throws Exception {
        mockMvc = MockMvcBuilders.standaloneSetup(new HelloWorldController()).buil
d();
    }

    @Test
    public void getHello() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.post("/hello?name=neo")).andDo(prin
t());
    }

}
```

在类的上面添加 `@SpringBootTest` ，系统会自动加载 Spring Boot 容器。在日常测试中，我们就可以注入 bean 来做一些局部业务的测试。 `MockMvcRequestBuilders` 可以 post、get 请求，使用 `print()` 方法会将请求和相应的过程都打印出来，如下：

```
MockHttpServletRequest:
      HTTP Method = POST
      Request URI = /hello
       Parameters = {name=[neo]}
          Headers = {}

Handler:
             Type = com.neo.helloWorld.web.HelloWorldController
           Method = public java.lang.String com.neo.helloWorld.web.HelloWorldContr
oller.hello(java.lang.String)

...

MockHttpServletResponse:
           Status = 200
    Error message = null
          Headers = {Content-Type=[text/plain;charset=ISO-8859-1], Content-Length=
[16]}
     Content type = text/plain;charset=ISO-8859-1
             Body = Hello World ,neo
    Forwarded URL = null
   Redirected URL = null
          Cookies = []
```

从返回的 `Body = Hello World ,neo` 可以看出请求成功。

总结

使用 Spring Boot 可以非常方便、快速搭建项目，而不用关心框架之间的兼容性、适用版本等各种问题，我们想使用任何东西，仅仅添加一个配置就可以，所以使用 Sping Boot 非常适合构建微服务。

建议大家使用 Idea 开发 Spring Boot 项目，Eclipse 对 Spring Boot 项目支持并不好，并且使用 Eclipse 偶尔会出现一些诡异的问题，影响初学者的学习。

> 点击这里下载源码 。