

# Spring Boot简介

## 本节目标

1.了解Spring Boot发展背景 2.了解Spring Boot的特点 3.掌握基于Spring Boot项目构建

## 1. Spring Boot背景

多年以来，Spring IO 平台饱受非议的一点就是大量的XML配置以及复杂的依赖管理。在2013年的SpringOne 2GX会议上，Pivotal的CTO Adrian Colyer回应了这些批评，并且特别提到该平台将来的目标之一就是实现免XML配置的开发体验。[Boot](#) 所实现的功能超出了这个任务的描述，开发人员不仅不再需要编写XML，而且在一些场景中甚至不需要编写繁琐的import语句。在对外公开的beta版本刚刚发布之时，Boot描述了如何使用该框架在140个字符内实现可运行的web应用，从而获得了极大的关注度，该样例发表在Twitter上。

演示：通过[在线方式创建Spring Boot项目](#)，展示Spring Boot的新开发体验。

## 2. 什么是Spring Boot



- Spring Boot是由Pivotal团队提供的全新框架，Spring Boot并不是要成为Spring IO平台里面众多“Foundation”层项目的替代者。Spring Boot的目标不在于为已解决的问题域提供新的解决方案，而是**为平台带来另一种开发体验**，从而简化对这些已有技术的使用。

- 该框架使用了特定的方式(**继承Starter或者依赖Starter**, 约定优先于配置)来进行配置, 从而使开发人员不再需要定义样板化的配置。通过这种方式, Boot致力于在蓬勃发展的快速应用开发领域 (rapid application development) 成为领导者。
- Spring Boot是基于Spring 4进行设计, 继承了原有Spring框架的优秀基因。它并不是一个框架, 从根本上讲, 它就是一些库的集合, maven或者gradle项目导入相应依赖即可使用Spring Boot, 而且无需自行管理这些库的版本。

### 3. 为什么使用Spring Boot

- Spring Boot是为简化Spring项目配置而生, 使用它使得jar依赖管理以及应用编译和部署更为简单
- Spring Boot提供自动化配置, 使用Spring Boot只需编写必要的代码和配置必须的属性
- 使用Spring Boot, 只需20行左右的代码即可生成一个基本的Spring Web应用, 并且内置了tomcat, 构建的fatJar包通过java -jar就可以直接运行
- 如下特性使得Spring Boot非常契合微服务的概念, 可以结合Spring Boot与Spring Cloud和Docker技术来构建微服务并部署到云端:
  - 一个可执行jar即为一个独立服务
    - 很容易加载到容器, 每个服务可以在自己的容器 (例如docker) 中运行
  - 通过一个脚本就可以实现配置与部署, 很适合云端部署, 并且自动扩展也更容易

### 4. Spring Boot有哪些特性



#### 4.1 自动管理依赖

spring-boot-\*.jar包已对一些功能性jar包进行了集成, 示例如下:

- spring-boot-starter 核心Spring Boot starter, 包括自动配置支持, 日志和YAML
- spring-boot-starter-actuator 生产准备的特性, 用于帮你监控和管理应用
- spring-boot-starter-web 对全栈web开发的支持, 包括Tomcat和spring-webmvc
- spring-boot-starter-aop 对面向切面编程的支持, 包括 spring-aop和AspectJ
- spring-boot-starter-data-jdbc 对JDBC数据库的支持
- spring-boot-starter-security 对 spring-security 的支持
- spring-boot-starter-ws 支持Spring Web Services
- spring-boot-starter-data-redis 支持Redis键值存储数据库, 包括spring-redis
- spring-boot-starter-test 支持常规的测试依赖, 包括JUnit, spring-test等模块

#### 4.2 独立运行

Spring Boot默认将应用打包成一个可执行的jar包文件, 构建成功后使用java -jar命令即可运行应用。或者在应用项目的主程序中运行main函数即可, 不需要依赖Tomcat, Jetty, Undertow等外部的应用服务器。其中内置的servlet Container:

| Name         | Servlet Version |
|--------------|-----------------|
| Tomcat 8.5   | 3.1             |
| Jetty 9.4    | 3.1             |
| Undertow 1.4 | 3.1             |

此外，你仍然可以部署Spring Boot项目到任何兼容Servlet3.0+的容器。

## 4.3 自动配置

- Spring Boot尝试根据你添加的jar依赖自动配置你的应用。例如：如果H2在类路径中，并且你没有手动配置任何数据库连接的bean，则Spring Boot会自动配置一个内存数据库。

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
</dependency>
```

- 使用@EnableAutoConfiguration或者@SpringBootApplication注解，配合@Configuration注解类，即可达到自动配置的目的。
- Spring Boot的这种自动配置是非侵入式的，你可以定义自己的配置或bean来替代自动配置的内容。

## 4.4 外部化配置

Spring Boot可以使用properties文件，YAML文件，环境变量，命令行参数等来外部化配置。属性值可以使用@Value注解直接注入到bean中，并通过Spring的Environment抽象或经过@ConfigurationProperties注解绑定到结构化对象来访问。如下是对接第三方支付的配置实例：

```
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;

@ConfigurationProperties(prefix = "app.pay")
@Component
public class PayProperties {

    private WxPay wxPay = new WxPay();

    /**
     * 微信支付配置
     */
    public static class WxPay {

        /**
         * 应用ID
         */
    }
}
```



```
server.port=80
server.servlet.session.timeout=30m
```

## 4.6 准生产的应用监控

Spring Boot提供了基于HTTP, SSH, Telnet对运行时的项目进行监控。

## 5. 创建SpringBoot项目

创建Spring Boot项目的基本流程：



### 5.1 准备工作

- JDK 8
- 创建Maven项目
- 目录结构如下

```
E:.\
|
|_src
| |_main
| | |_java
| | | |_com
| | | | |_bittech
| | | | |_boot
| | |_resources
| | |_static //静态资源目录
| |_test
| |_java
```

### 5.2 添加依赖

本课程选择Spring Boot 2.x的第一个发布版本 `2.0.2.RELEASE` 进行课程讲解, 如下在pom.xml中添加spring-boot 依赖。

```
<!-- 项目的pom定义继承自SpringBoot的父pom -->
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.0.2.RELEASE</version>
</parent>
```

```
<!-- web项目，添加spring-boot-starter-web依赖即可，版本号由父pom已经定义好了，此处省略 -->
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
</dependencies>
```

## 5.3 编写简单示例

- 编写一个主函数构建Spring容器

```
package com.bittech.boot;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ExampleApplication {
    public static void main(String[] args) {
        SpringApplication.run(ExampleApplication.class, args);
    }
}
```

- 编写一个控制器

```
package com.bittech.boot.control;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(value = "/")
public class ExampleController {

    @RequestMapping(value = "")
    public String index() {
        return "Hello Spring Boot World ";
    }
}
```

- 编写一个控制器

在src/main/resources目录下的static目录下放置静态资源。

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Spring Boot</title>
</head>
<body>
<h1>Hello Spring Boot</h1>
</body>
</html>

```

## 5.4 运行示例

- 运行程序,控制台输出



```

Run: ExampleApplication x
[Spring Boot] (v2.0.2.RELEASE)
2018-05-15 16:00:10.640 INFO 6608 --- [main] com.bittech.boot.ExampleApplication : Starting ExampleApplication on DESKTOP-KFN3QUK with PID 6608 (E:\workspace\java\spring-boot\ExampleApplication\bin)
2018-05-15 16:00:10.643 INFO 6608 --- [main] com.bittech.boot.ExampleApplication : No active profile set, falling back to default profiles: default
2018-05-15 16:00:10.693 INFO 6608 --- [main] ConfigServletWebServerApplicationContext : Refreshing org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext
2018-05-15 16:00:11.897 INFO 6608 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2018-05-15 16:00:11.926 INFO 6608 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2018-05-15 16:00:11.927 INFO 6608 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache Tomcat/8.5.31
2018-05-15 16:00:11.939 INFO 6608 --- [ost-startStop-1] o.a.catalina.core.AprLifecycleListener : The APR based Apache Tomcat Native library which allows optimal performance in production environments is not available
2018-05-15 16:00:12.074 INFO 6608 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2018-05-15 16:00:12.075 INFO 6608 --- [ost-startStop-1] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 1385 ms
2018-05-15 16:00:12.294 INFO 6608 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean : Servlet dispatcherServlet mapped to [/]
2018-05-15 16:00:12.299 INFO 6608 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'characterEncodingFilter' to: [/]
2018-05-15 16:00:12.300 INFO 6608 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'hiddenHttpMethodFilter' to: [/]
2018-05-15 16:00:12.300 INFO 6608 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'httpPutFormContentFilter' to: [/]
2018-05-15 16:00:12.300 INFO 6608 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'requestContextFilter' to: [/]
2018-05-15 16:00:12.461 INFO 6608 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerMethod]
2018-05-15 16:00:12.655 INFO 6608 --- [main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext
2018-05-15 16:00:12.714 INFO 6608 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "[]" onto public java.lang.String com.bittech.boot.controller.ExampleController.hello()
2018-05-15 16:00:12.716 INFO 6608 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "[/error]" onto public org.springframework.http.ResponseEntity<java.lang.String> com.bittech.boot.controller.ExampleController.error()
2018-05-15 16:00:12.717 INFO 6608 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "[/error],produces=[text/html]" onto public org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerMethod com.bittech.boot.controller.ExampleController.error()
2018-05-15 16:00:12.734 INFO 6608 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/**] onto handler of type [class org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerMethod]
2018-05-15 16:00:12.734 INFO 6608 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerMethod]
2018-05-15 16:00:12.843 INFO 6608 --- [main] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
2018-05-15 16:00:12.877 INFO 6608 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2018-05-15 16:00:12.881 INFO 6608 --- [main] com.bittech.boot.ExampleApplication : Started ExampleApplication in 2.512 seconds (JVM running for 2.936s)
2018-05-15 16:02:15.932 INFO 6608 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring FrameworkServlet 'dispatcherServlet'
2018-05-15 16:02:15.932 INFO 6608 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization started
2018-05-15 16:02:15.948 INFO 6608 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization completed in 16 ms

```

- 访问浏览器

- http://127.0.0.1:8080/



- http://127.0.0.1:8080/index.html



# Hello Spring Boot



## 5.5 构建可执行程序

- pom.xml中添加 `spring-boot-maven-plugin` 插件

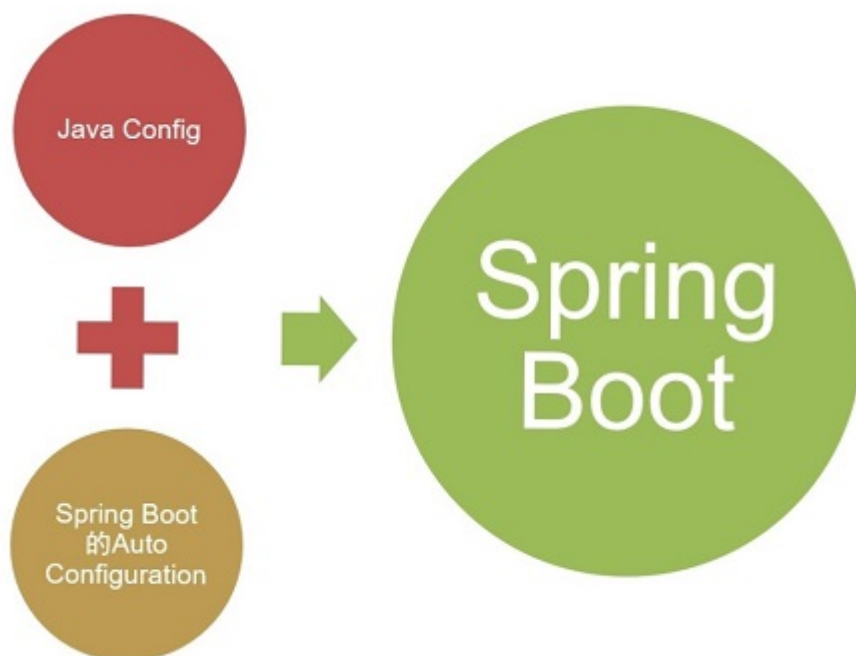
```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <executions>
        <execution>
          <goals>
            <goal>repackage</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

- 执行 `mvn package` 命令，构建可执行jar
- 切换目录到 `target` 下，执行 `jar -jar spring-boot-1.0.0.jar` 运行程序

## 6. 为什么SpringBoot如此简洁

Spring Boot成为可能的原因：

- Spring Framework的Java Config配置方式的支持和Spring Boot提供的自动配置



- Spring IO生态体系中各类框架和库的支持
- Spring Core核心的IoC容器和AOP实现
- 开源社区丰富的工具链支持Spring Boot的发展



# 总结

| 知识块          | 知识点              | 分类  | 掌握程度 |
|--------------|------------------|-----|------|
| SpringBoot   | 1.SpringBoot特点   | 理解性 | 了解   |
| SpringBoot项目 | 1.SpringBoot项目构建 | 实战型 | 掌握   |

# 课后作业

- 创建一个基于Maven管理的Spring Boot的Web项目