# SpringBoot简介

1、SpringBoot特性：

* 为基于Spring的开发提供更快的入门体验
* 开箱即用，没有代码生成，也无需XML配置。同时也可以修改默认值来满足特定的需求。
* 提供了一些大型项目中常见的非功能性特性，如嵌入式服务器、安全、指标，健康检测、外部配置等。
* Spring Boot并不是不对Spring功能上的增强，而是提供了一种快速使用Spring的方式

2、文档项目实例地址：<https://github.com/slowwalkerlcr/SpringBoot_Demo>

学习实例：

码云地址<https://git.oschina.net/didispace/SpringBoot-Learning>

博客：<http://blog.didispace.com/spring-boot-learning-1/>

# SpringBoot 基本Demo

## 创建一个基本的springboot项目

项目实例：

<https://github.com/slowwalkerlcr/SpringBoot_Demo/tree/master/SpringBootBase>

## application properties

* 公共属性

<http://docs.spring.io/spring-boot/docs/1.5.2.RELEASE/reference/htmlsingle/#appendix>

可以自己配置配置参数

Eg：

server.port(默认8080)、

server.context-path（默认/）、

logging.level （日志级别）

server.tomcat.max-threads= （tomcat调优等参数）

* 默认读取application.properties路径

<http://docs.spring.io/spring-boot/docs/1.5.2.RELEASE/reference/htmlsingle/#boot-features-external-config-application-property-files>

1. 当前目录/config目录下
2. 当前目录
3. Classpath下的config目录下
4. Classpath

Application.properties优先级 命令形式＞4>3>2>1

* 自定义application.properties文件名和路径

项目实例：

<https://github.com/slowwalkerlcr/SpringBoot_Demo/tree/master/SpringBootApplication>

1. 使用SpringApplication的setDefaultProperties方法
2. @PropertySource

## 多环境配置

dev、test、prod等，各个环境使用对应的配置文件

## 命令启动

启动脚本可以参考此命令：

Java -jar demo.jar --spring.profiles.active=dev配置文件参数

--spring.config.location指定配置文件路径

--server.port 指定端口

## Application.properties和application.yml

两种不容的配置文件格式

## 引用外部配置

项目实例：

<https://github.com/slowwalkerlcr/SpringBoot_Demo/tree/master/SpringBootApplication>

* 引用外部配置
* 外部配置转Bean
* 随机数功能

<http://docs.spring.io/spring-boot/docs/1.5.2.RELEASE/reference/htmlsingle/#boot-features-external-config-random-values>

## SpringBoot模版使用

项目实例：

src/main/resources目录下

static(存放静态资源文件例如：css、js、图片)

templates(存放模版文件)

1. [Thymeleaf](https://git.oschina.net/didispace/SpringBoot-Learning/commit/70570d5d1f89161d20fad5c1932ff74726a340c0)

依赖：

<dependency>

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

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

</dependency>

1. [freemarker](https://git.oschina.net/didispace/SpringBoot-Learning/commit/7eab9c30bd0eb47a7cb32d87025395c718cfcd4d)
2. [velocity](https://git.oschina.net/didispace/SpringBoot-Learning/commit/8c0b4127cf2e8900211c6f13245a85824eaf7866)

# @importResouce使用

如果在实际项目中，存在特殊场景必须使用xml配置，那么可以通过这个注解来加载自己的xml配置文件

项目实例：

<https://github.com/slowwalkerlcr/SpringBoot_Demo/tree/master/SpringBootDiffProperties>

# SpringBoot 监听器和事件、过滤器等

项目实例：

<https://github.com/slowwalkerlcr/SpringBoot_Demo/tree/master/SpringBootEg>

1. SpringBoot监听器和事件

<http://docs.spring.io/spring-boot/docs/1.5.2.RELEASE/reference/htmlsingle/#boot-features-application-events-and-listeners>

1. ApplicationStartingEvent ：spring boot启动开始时执行的事件
2. ApplicationEnvironmentPreparedEvent ：spring boot 对应Enviroment已经准备完毕，但此时上下文context还没有创建。.
3. ApplicationPreparedEvent ： spring boot上下文context创建完成，但此时spring中的bean是没有完全加载完成的
4. ApplicationReadyEvent ：spring上下文已经加载完城.
5. ApplicationFailedEvent ：启动失败事件

2．过滤器等（取代web.xml的功能）

JavaConfig形式：

Filter注册：FilterRegistrationBean

Servlet注册：ServletRegistrationBean

Listener注册：ServletListenerRegistrationBean

# SpringBootRedis Demo

项目实例：

<https://github.com/slowwalkerlcr/SpringBoot_Demo/tree/master/SpringBootRedis>

# 条件注解

项目实例：

https://github.com/slowwalkerlcr/SpringBoot\_Demo/tree/master/SpringBootCondtion

1. @ConditionalOnBean:当容器中有指定的Bean的条件下
2. @ConditionalOnClass：当类路径下有指定的类的条件下
3. @ConditionalOnExpression:基于SpEL表达式作为判断条件
4. @ConditionalOnJava:基于JVM版本作为判断条件
5. @ConditionalOnJndi:在JNDI存在的条件下查找指定的位置
6. @ConditionalOnMissingBean:当容器中没有指定Bean的情况下
7. @ConditionalOnMissingClass:当类路径下没有指定的类的条件下
8. @ConditionalOnNotWebApplication:当前项目不是Web项目的条件下
9. @ConditionalOnProperty:指定的属性是否有指定的值
10. @ConditionalOnResource:类路径下是否有指定的资源
11. @ConditionalOnSingleCandidate:当指定的Bean在容器中只有一个，或者在有多个Bean的情况下，用来指定首选的Bean
12. @ConditionalOnWebApplication:当前项目是Web项目的条件下

@ConditionalOnClass：该注解的参数对应的类必须存在，否则不解析该注解修饰的配置类；

@ConditionalOnMissingBean(JdbcOperations.class)

public JdbcTemplate jdbcTemplate() {

return new JdbcTemplate(this.dataSource);

}

如果存在名称为JdbcOperations的bean，则该部分代码直接忽略，这是Spring Boot人性化体现之一，开发者申明的bean会放在第一位。

# SpringBoot四大神器

Starter、actuator、cli、autoconfiguration

## actuator

添加依赖

<dependency>

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

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

</dependency>

主要看shutdown、trace（监控）、mappings和env（环境信息）

HTTP方法 路径 描述 鉴权

GET /autoconfig 查看自动配置的使用情况 true

GET /configprops 查看配置属性，包括默认配置 true

GET /beans 查看bean及其关系列表 true

GET /dump 打印线程栈 true

GET /env 查看所有环境变量 true

GET /env/{name} 查看具体变量值 true

GET /health 查看应用健康指标 false

GET /info 查看应用信息 false

GET /mappings 查看所有url映射 true

GET /metrics 查看应用基本指标 true

GET /metrics/{name} 查看具体指标 true

POST /shutdown 关闭应用 false

GET /trace 查看基本追踪信息 true

关闭服务：curl -X POST localhost:8080/SpringBoot/shutdown，

可以用作停止服务的脚本，而不是用kill 进程的方式

#启用shutdown

endpoints.shutdown.enabled=true

#禁用密码验证

endpoints.shutdown.sensitive=false

## Starter（可以认为是一个组件）

1、SpringBoot的starter主要用来简化依赖用的，相当集成了对应的服务，供他人使用

<http://docs.spring.io/spring-boot/docs/1.5.2.RELEASE/reference/htmlsingle/#using-boot-starter>

2、自定义starter组件

Src/main/resources/META-INF/spring.factories设置，spring类加载器自动将设置的类注入到spring容器中

项目实例：

组件提供方：

<https://github.com/slowwalkerlcr/SpringBoot_Demo/tree/master/springboot-starter-helloworld>

组件使用方

<https://github.com/slowwalkerlcr/SpringBoot_Demo/tree/master/SpringBootStarters>

## CLI

CLI：客户端工具，使用很少

快速创建项目：<http://start.spring.io/> 可以选择自己想要的模块、省掉了jar版本不一致的问题

# SpringBoot 热部署

不需要手动启动，可以参考项目的启动时间，比手动启动时间少得多

**devtools**：是boot的一个热部署工具，当我们修改了classpath下的文件（包括类文件、属性文件、页面等）时，会重新启动应用（由于其采用的双类加载器机制，这个启动会非常快）

* **双类加载器机制**：boot使用了两个类加载器来实现重启（restart）机制：**base类加载器（简称bc）**+**restart类加载器（简称rc）**。
  + bc：用于加载不会改变的jar（eg.第三方依赖的jar）
  + rc：用于加载我们正在开发的jar（eg.整个项目里我们自己编写的类）。当应用重启后，原先的rc被丢掉、重新new一个rc来加载这些修改过的东西，而bc却不需要动一下。这就是devtools重启速度快的原因

参考项目：

<https://github.com/slowwalkerlcr/SpringBoot_Demo/tree/master/SpringBootDevTool>

#关闭热部署

#spring.devtools.restart.enabled=false

# SpringBoot 打包生成规范的目录结构

需求：根据移动的部署规范，本来springboot的maven插件会将项目打成一个jar包，需要将jar方式打包成对应的目录结构（包括启动、停止服务脚本，引用的jar包，配置文件（便于修改））

项目实例：<https://github.com/slowwalkerlcr/SpringBoot_Demo/tree/master/SpringBootZip>

交易业务部应用部署规范：

/bin/ #可执行脚本或程序的目录

/lib/ # 库文件的目录

/conf/ # 应用配置文件的目录

/logs/ # 应用日志的目录

/temp/ # 临时目录

/work/ # 工作目录

/webapps/ # web tomcat项目使用

/startapp.sh # 统一启动脚本

/stopapp.sh # 统一停止脚本

/restartapp.sh # 统一重启动脚本

/smartcnf.ini # 统一配置文件

/smartcnf.jar # 统一配置工具jar包