# 什么是Spring boot

Springboot是一个基于spring所有功能的工具框架.能够让一个spring框架的开发过程简化，再简化，能应对非常多的开发场景实现自动配置.

例如：开发web应用，springboot帮你完成了web容器的配置，SpringMvc，Spring的配置

# 特点

### 独立运行的spring容器

Spring开发的项目的运行一般都需要别的容器的支持，比如开发一个web应用，要使用第三方的web容器。

但Springboot可以独立运行，一个main方法加载springboot运行代码

### 内嵌Servlet容器

Springboot为web工程提供了内嵌的web容器，默认使用tomcat(可以配置为jetty和undertow)，并且按照习惯进行默认配置，例如端口号8080 项目访问路径 /.结合第一个特点，web工程就可以不用封装war包访问外部的web容器运行，直接使用jar包运行.

### 简化依赖

Springboot能够实现它的独有的特点，是因为它在Spring基础之上扩展了非常庞大的量的代码.导致要想使用Spring必须依赖大量的资源.这样极其不方便的.所以Springboot为开发者准备来的丰富环境的简化依赖。

例如:我们要开发一个web应用(spring-context，spring-webmvc，jackson，loggings等等)，在springboot只要想开发一个web应用，只需要依赖一个spring-boot-starter-web，实现依赖的传递。

### 自动配置(核心特点)

springboot最核心的特点，就是自动配置.它为大多数开发场景准备对应的配置逻辑.在使用springboot开发时，可以不考虑如何配置这些技术，环境，只需要按照springboot提供的逻辑准备你的技术，环境的各种尚需经

例如:自动完成web容器的配置

简化了端口，程序访问根目录，静态资源访问，前后缀拼接等等配置内容

自动完成了datasource数据源配置

你一旦需要持久层数据源连接数据库，只需要提供对应属性就可以了

# Spring Boot入门(快速开发SSM项目)

### 创建一个maven项目

#### 选择骨架(默认不选)

|  |
| --- |
|  |

#### 填写信息(groupId，ArtifactId)

|  |
| --- |
|  |

#### Finish创建完成

|  |
| --- |
|  |

### 配置Spring相关依赖(pom.xml)

#### 继承spring boot资源

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>cn.shu</groupId>  <artifactId>spring-boot-demo</artifactId>  <version>1.0-SNAPSHOT</version>  <!--继承springboot相关资源-->  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.9.RELEASE</version>  </parent>  </project> |

#### 添加spring web应用的资源

|  |
| --- |
| <!--web应用的依赖-->  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId> spring-boot-starter-web </artifactId>  </dependency> </dependencies> |

### 开发一个Controller类(TestController)

|  |
| --- |
|  |

### 开发启动类(唯一入口)

|  |
| --- |
|  |

### 测试

#### 启动main方法

|  |
| --- |
|  |

#### 网页测试

|  |
| --- |
|  |

# Spring Boot的配置文件

配置文件需要以application命名，支持二种格式，需要放到resource目录下

### application.properties类型

配置端口，注意与yml类型区别

|  |
| --- |
|  |

### application.yml类型

配置端口，注意与properties类型区别

一级添加一个空格，再下一级添加二个空格

|  |
| --- |
|  |

# Spring Boot整合Mybatis

作为springboot框架，为持久层mybatis整合准备自动配置的逻辑，其中dataSource这个对象不需要我们自行配置bean标签，bean注解等内容，是通过DataSourceAutoConfiguration的配置类实现自动配置.我们只需要给它准备必要的四个属性，提供mybatis的依赖就能是实现持久层整合

Mybatis不属于spring的一员，使用配置稍微复杂一些

### 导入相关依赖

|  |
| --- |
| <!--starter JDBC--> <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jdbc</artifactId> </dependency>  <!--mybatis 整合spring boot--> <dependency>  <groupId>org.mybatis.spring.boot</groupId>  <artifactId>mybatis-spring-boot-starter</artifactId>  <version>1.3.0</version> </dependency>  <!--mysql JDBC--> <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId> </dependency>  <!--德鲁伊数据源--> <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  <version>1.1.20</version> </dependency> |

### 配置MyBatis(spring-boot的配置文件中)

|  |
| --- |
| server:  port: 8099 spring:  datasource:  *#数据源类型* type: com.alibaba.druid.pool.DruidDataSource  url: jdbc:mysql:///microtest  username: root  password: admin  driver-class-name: com.mysql.jdbc.Driver *#mybatis* mybatis:  *#mapper文件* mapper-locations: classpath:mappers/\*.xml  configuration:  *#关闭二级缓存* cache-enabled: false  *#开启驼峰命名* map-underscore-to-camel-case: true |

### 配置mapper扫描包

在我们的入口类中，添加@MapperScan注解，指向我们的maper接口所在包即可

|  |
| --- |
|  |

# 打包独立运行(jar)

工程的静态资源需要放到 resource->static下，没有web-inf

作为一个web应用，使用springboot开发完毕之后，无需按照war包形式打包放到第三方web容器运行.只需要在一个JDK环境下，运行java命令，来使当前项目jar包生效。

### 添加打包插件(pom.xml)

|  |
| --- |
| <build>  <plugins>  <plugin>  <!--添加一个对当前工程打包使用的maven插件-->  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build> |

### 可指定打包名称

|  |
| --- |
| <build>  <finalName>myblog</finalName>  <plugins>  <plugin>  <!--添加一个对当前工程打包使用的maven插件-->  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build> |

### 运行

JDK环境运行：

|  |
| --- |
| java -jar jar包名称 |