# 介绍

## 什么是springboot？

<https://www.cnblogs.com/larryzeal/p/5799195.html>

Spring Boot是为了简化Spring应用的创建、运行、调试、部署等而出现的，使用它可以做到专注于Spring应用的开发，而无需过多关注XML的配置。

简单来说，它提供了一堆依赖打包，并已经按照使用习惯解决了依赖问题---习惯大于约定。

Spring Boot默认使用tomcat作为服务器，使用logback提供日志记录。

# Springboot入门案例

<https://github.com/chenjxJava/springboot-learning-example>

## 创建一个helloworld

### 创建一个maven工程

### 修改pom文件

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.4.RELEASE</version> </parent> <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency> </dependencies> |

### 创建一个启动类Application

|  |
| --- |
| public class Application {   public static void main(String[] args) throws Exception {  Object[] clazz = {SampleController.class,RestfulController.class};  SpringApplication.*run*(clazz, args);  } } |

### 编写controller

|  |
| --- |
| @RestController @EnableAutoConfiguration public class SampleController {   @RequestMapping("/hello")  @ResponseBody  String home() {  return "Hello World11!";  } } |

### 运行启动类Application

注：

## Springboot配置文件

### 优先级

./config ------------config文件夹下

./ ------------classpath

在/config文件夹下设置的优先级会高于classpath下

### 配置属性

server.port=8081 --------------设置端口号为8081

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/jpatest

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jackson.serialization.indent\_output=true

## Springboot整合dubbo

### 创建一个服务提供者

1. 创建一个实体类

|  |
| --- |
| */\*\*  \* 城市实体类  \*/* public class City implements Serializable {   private static final long *serialVersionUID* = -1L;  private Long id; //城市编号  private Long provinceId; //省份编号  private String cityName; //城市名称  private String description; //描述   public City() {  }   public City(Long id, Long provinceId, String cityName, String description) {  this.id = id;  this.provinceId = provinceId;  this.cityName = cityName;  this.description = description;  }   public Long getId() {  return id;  }   public void setId(Long id) {  this.id = id;  }   public Long getProvinceId() {  return provinceId;  }   public void setProvinceId(Long provinceId) {  this.provinceId = provinceId;  }   public String getCityName() {  return cityName;  }   public void setCityName(String cityName) {  this.cityName = cityName;  }   public String getDescription() {  return description;  }   public void setDescription(String description) {  this.description = description;  } } |

1. 创建服务层接口

|  |
| --- |
| */\*\*  \* 城市业务 Dubbo 服务层  \*/* public interface CityDubboService {   */\*\*  \* 根据城市名称，查询城市信息  \** ***@param*** *cityName  \*/* City findCityByName(String cityName); } |

C.创建接口实现类

|  |
| --- |
| */\*\*  \* 城市业务 Dubbo 服务层实现层  \*/* // 注册为 Dubbo 服务 @Service(version = "1.0.0") public class CityDubboServiceImpl implements CityDubboService {   public City findCityByName(String cityName) {  return new City(1L,2L,"温岭","是我的故乡");  } } |

D.application.properties配置文件（./config或者./）

## Dubbo 服务提供者配置

spring.dubbo.application.name=provider

spring.dubbo.registry.address=zookeeper://127.0.0.1:2181

spring.dubbo.protocol.name=dubbo

spring.dubbo.protocol.port=20880

spring.dubbo.scan=org.spring.springboot.dubbo