## [Spring Boot学习](http://www.cnblogs.com/larryzeal/p/5765945.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。

前提

Spring Boot提供了一系列的依赖包，所以需要构建工具的支持：maven 或 gradle。个人仅熟悉maven，所以下面的内容都是maven相关的。

如果不熟悉maven，请先了解一下。

使用

① 新建一个maven项目。

② pom中parent设为 spring-boot-starter-parent 。建议使用最新的 RELEASE 版本。否则可能需要设置 <repositories/> 和<pluginRepositories/> 。

③ 添加应用需要的starter模块，作为示例，我们仅添加web starter模块。

这里需要解释下starter模块，简单的说，就是一系列的依赖包组合。例如web starter模块，就是包含了Spring Boot预定义的一些Web开发的常用依赖：

○ spring-web, spring-webmvc Spring WebMvc框架

○ tomcat-embed-\* 内嵌Tomcat容器

○ jackson 处理json数据

○ spring-\* Spring框架

○ spring-boot-autoconfigure Spring Boot提供的自动配置功能

换句话说，当你添加了相应的starter模块，就相当于添加了相应的所有必须的依赖包。

　　starter模块的列表及含义，见 [Spring Boot的启动器Starter详解](http://blog.csdn.net/chszs/article/details/50610474) 。

至此，pom内容如下：

<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.larry.spring</groupId>

<artifactId>larry-spring-demo4</artifactId>

<version>0.0.1-SNAPSHOT</version>

<parent>

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

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

<version>1.4.0.RELEASE</version>

</parent>

<dependencies>

<dependency>

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

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

</dependency>

</dependencies>

</project>

保存pom，刷新maven，以便刷新依赖导入。  
基本上，如果没有特别的需要，现在就可以直接写Controller了！！！--特别的需要 是指设置容器、访问端口、路径等。后面再解释。

④ 写一个简单的Controller。--直接拿了 [Spring Boot——开发新一代Spring Java应用](http://www.tianmaying.com/tutorial/spring-boot-overview" \t "http://www.cnblogs.com/larryzeal/p/_blank) 中的示例。

package cn.larry.spring.controller;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.EnableAutoConfiguration;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.ResponseBody;

@Controller

@EnableAutoConfiguration

public class SampleController {

@RequestMapping("/")

@ResponseBody

String home() {

return "Hello World!";

}

public static void main(String[] args) throws Exception {

SpringApplication.run(SampleController.class, args);

}

}

这里有两个新东西：@EnableAutoConfiguration 和 SpringApplication 。

@EnableAutoConfiguration 用于自动配置。简单的说，它会根据你的pom配置（实际上应该是根据具体的依赖）来判断这是一个什么应用，并创建相应的环境。

在上面这个例子中，@EnableAutoConfiguration 会判断出这是一个web应用，所以会创建相应的web环境。

SpringApplication 则是用于从main方法启动Spring应用的类。默认，它会执行以下步骤：

创建一个合适的ApplicationContext实例 （取决于classpath）。

注册一个CommandLinePropertySource，以便将命令行参数作为Spring properties。

刷新application context，加载所有单例beans。

激活所有CommandLineRunner beans。

默认，直接使用SpringApplication 的静态方法run()即可。但也可以创建实例，并自行配置需要的设置。

5. 现在，直接右键启动main方法即可。启动信息（包括关闭信息）

如果使用sts （Spring Tools Suite--没意外的话，后面的博客我会介绍一下），还可以用Spring Application的形式启动，信息不变，但是彩色的

⑥ 根据这个信息，我们可以看出很多东西，不过现在先访问一下吧。

默认访问地址： [http://localhost:8080/](http://localhost:8080/" \t "http://www.cnblogs.com/larryzeal/p/_blank)

按照之前的web项目习惯，你可能会问，怎么没有项目路径？

这就是Spring Boot的默认设置了，将项目路径直接设为根路径。

当然，我们也可以设置自己的项目路径 -- 在classpath下的 application.properties 或者 application.yaml 文件中设置即可。