# spring boot学习笔记

## 第一章 spring boot环境的搭建

### 一、下载安装JDK，并配置java运行环境变量

参考：<https://www.w3cschool.cn/java/java-environment-setup.html>

### 二、安装eclipse

参考：<https://www.w3cschool.cn/eclipse/eclipse-install.html>

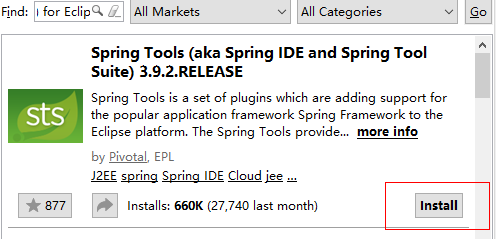
### 三、通过STS插件自动创建spring boot项目

步骤如下：

1**、Eclipse中安装STS插件：**

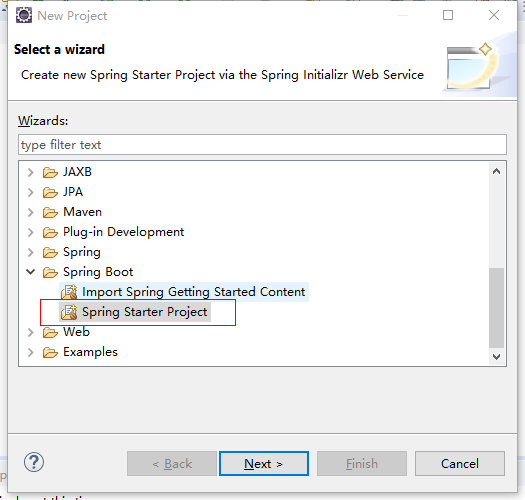
Help -> Eclipse Marketplace...

Search或选择“Popular”标签，选择Spring Tool Suite (STS) for Eclipse插件，选择让install，选择默认，安装完成即可：

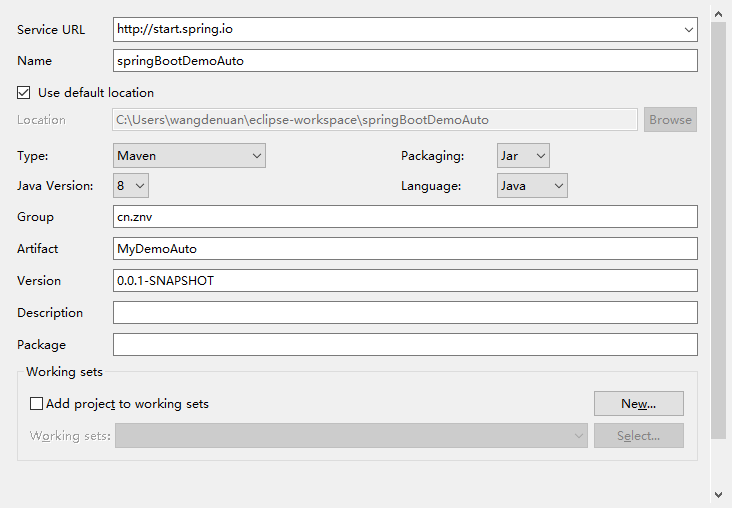


**2、New -> Project...**

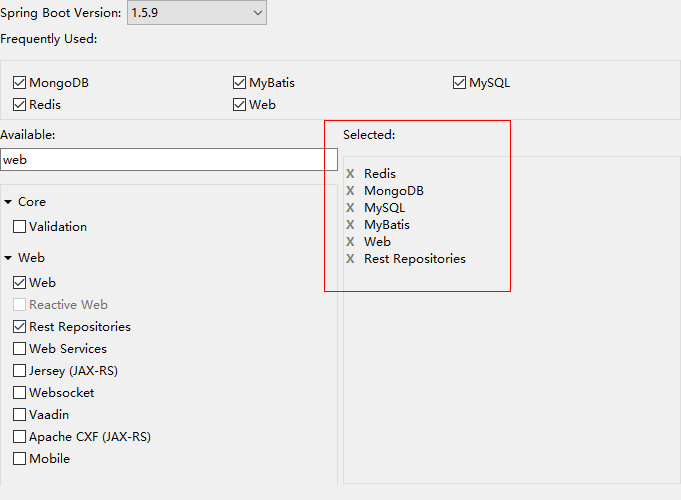
找到Spring目录，选择Spring Starter Project，Next



**3、填写跟项目相关的各种信息，然后Next：**



**4、选择需要的Dependency，然后Next：**



**5、Next，然后Finsh，新项目就创建好了，各个目录如下：**



**6、右键SpringBootDemoAutoApplication中的main方法，Run As -> Spring Boot App，项目就可以启动了。**

***package*** *cn.znv.MyDemoAuto;*

***import*** *org.springframework.boot.SpringApplication;*

***import*** *org.springframework.boot.autoconfigure.SpringBootApplication;*

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

***import*** *org.springframework.web.bind.annotation.RestController;*

*@RestController*

*@SpringBootApplication*

***public******class*** *SpringBootDemoAutoApplication {*

*@RequestMapping("/hello")*

***public*** *String ShowHello() {*

***return*** *"hello";*

*}*

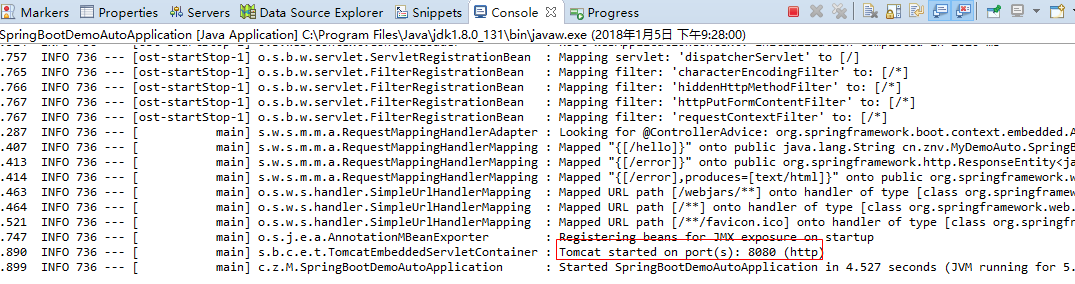
***public******static******void*** *main(String[] args) {*

*SpringApplication.run(SpringBootDemoAutoApplication.****class****, args);*

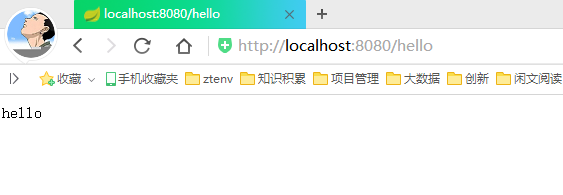
*}*

*}*

由于选择了web dependency默认启动一个Tomcat，8080端口监听



在IE中输入localhost:8080/hello



**7、在application.properties文件中，修改Tomcat的启动端口：**

server:

port: 8081

### 四、通过maven创建spring boot

通过上面的安装，我们已经简单实现了一个spring boot的web应用程序，springboot内置了tomcat，如果需要做发布版本，我们还需要一些步骤，后面在介绍，下面我们不通过插件的方式来进行springboot的环境搭建，采用maven的方式，有关maven的知识参考：

<https://www.w3cschool.cn/maven/j3x41ht2.html>

步骤如下：

1. 配置maven，由于maven默认下载源太慢，在pom中配置依赖jar包时，下载需要很久，这里介绍下eclipse如何加载maven自定义配置文件。

首先我们新建一个settings.xml文件，内容如下：

<?xml version="1.0"?>

<settings>

<localRepository>C:\Users\wangdenuan\.m2\repository</localRepository>

<!--需要改成自己的maven的本地仓库地址-->

<mirrors>

<mirror>

<id>alimaven</id>

<name>aliyun maven</name>

<url>http://maven.aliyun.com/nexus/content/groups/public/</url>

<mirrorOf>central</mirrorOf>

</mirror>

</mirrors>

<profiles>

<profile>

<id>nexus</id>

<repositories>

<repository>

<id>nexus</id>

<name>local private nexus</name>

<url>http://maven.oschina.net/content/groups/public/</url>

<releases>

<enabled>true</enabled>

</releases>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

</repositories>

<pluginRepositories>

<pluginRepository>

<id>nexus</id>

<name>local private nexus</name>

<url>http://maven.oschina.net/content/groups/public/</url>

<releases>

<enabled>true</enabled>

</releases>

<snapshots>

<enabled>false</enabled>

</snapshots>

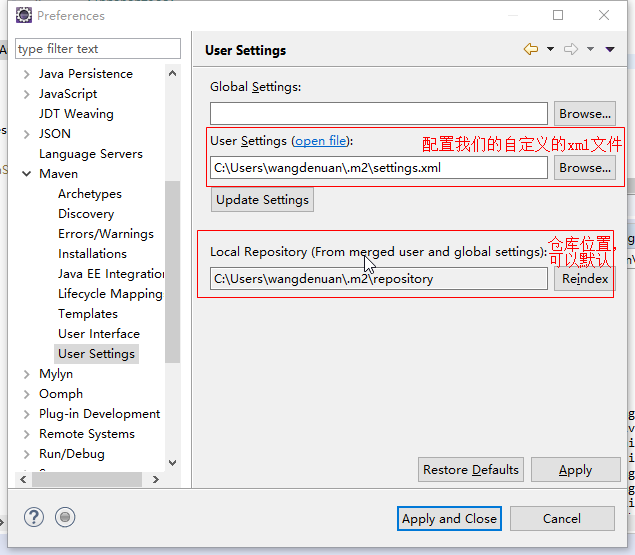
</pluginRepository>

</pluginRepositories>

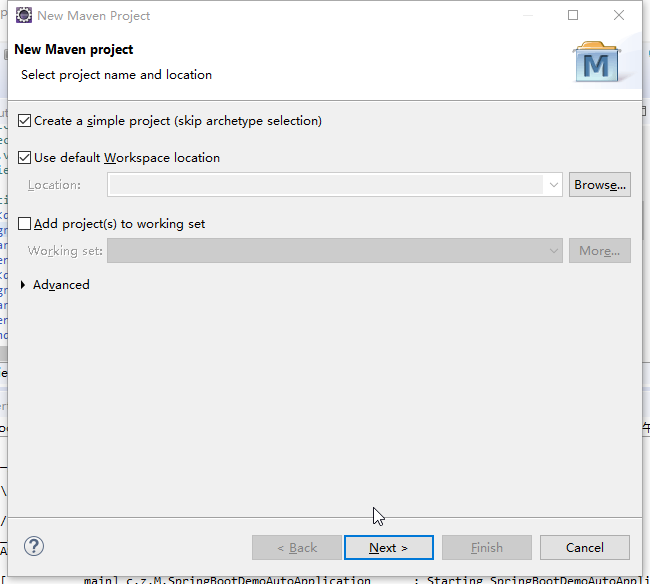
</profile></profiles>

</settings>

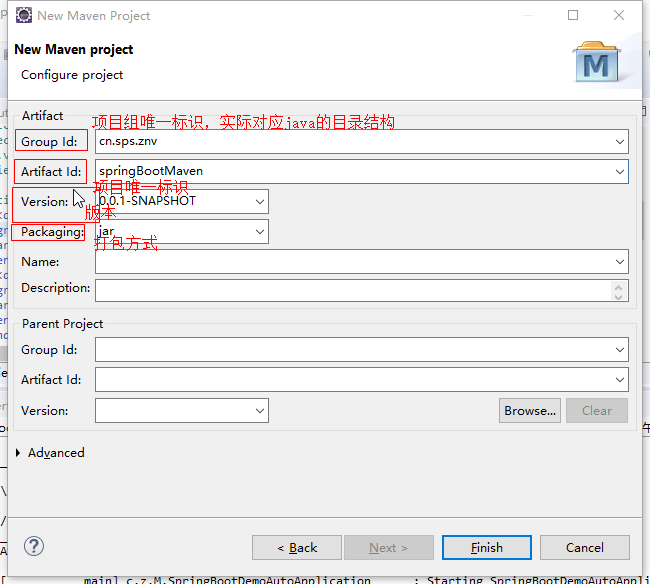
然后，打开eclipse，选择windows->preference,找到Maven->User Settings,配置如下：



1. 创建maven工程，选择file->New->Maven Project,打开New Maven Project对话框，配置如下，点击下一步：



1. 填写Maven工程相关的信息，如下图所示，点击finish

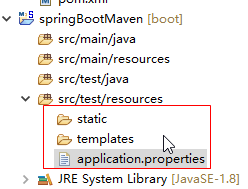


1. 配置pom.xml文件，导入SpringBoot的相关依赖，这里并没由自动产生需要在pom.xml中添加如下依赖：

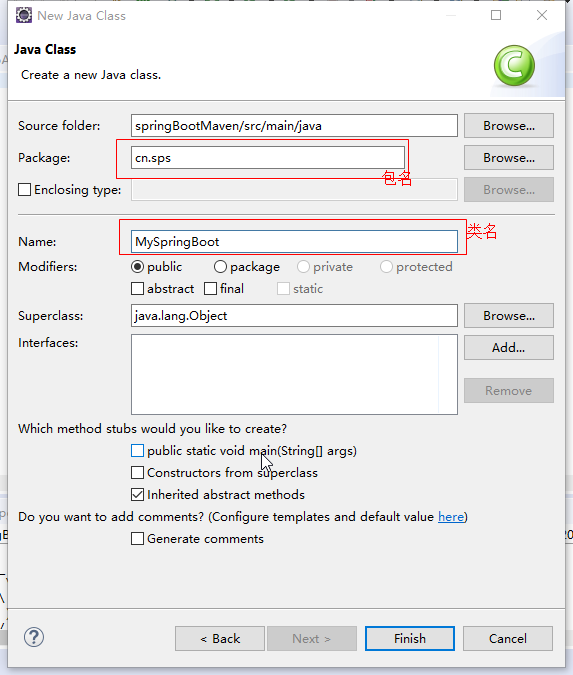
|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>cn.sps.znv</groupId>  <artifactId>springBootMaven</artifactId>  <version>0.0.1-SNAPSHOT</version>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.9.RELEASE</version>  <relativePath /> <!-- lookup parent from repository -->  </parent>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</java.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  </dependencies>  </project> |

这样就会自动加载springboot基础框架依赖的jar包了

1. 在资源文件夹下建立web应用所需要的资源文件夹，当然如果不需要页面，也可以不创建，这里创建三个目录，static、templates，application.propertys



6,创建应用程序入口类，在src/main/class上右击，new一个class，配置如下，点击finish：



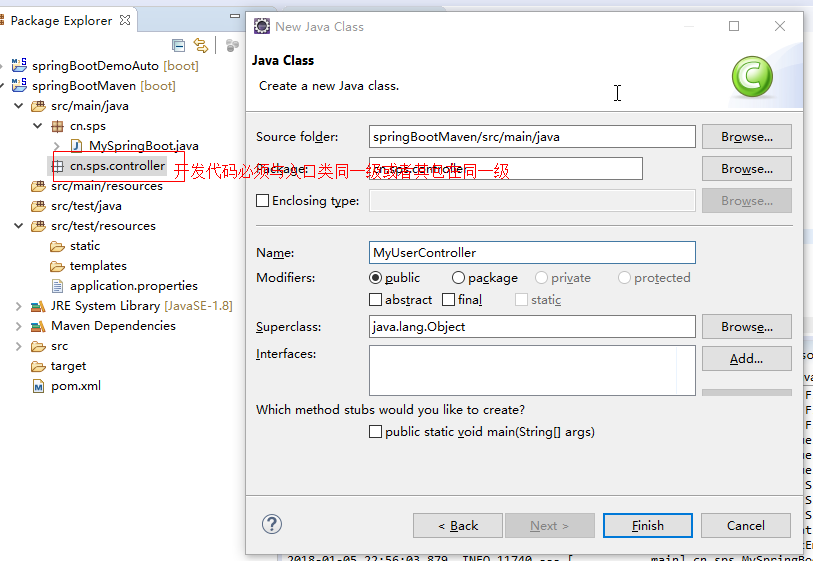
在MySpringBoot.java中编写如下代码：

|  |
| --- |
| **package** cn.sps;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  //加入springboot注解,springboot通过注解找到入口  @SpringBootApplication  **public** **class** MySpringBoot {  //项目的入口  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(MySpringBoot.**class**, args);    }  } |
|  |

此时就可以运行了，运行在终端窗口可以看到tomcat的启动信息，由于我们没有建立控制代码，所以还不能浏览器访问。

注意：开发的代码必须与入口代码在同一级，或者建立的包与其同一级，才能通过注解自动扫描到，自动加载，这里开发的代码，不需要配置。就能够自动加载。

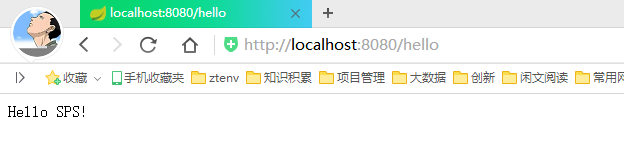
1. 创建用户控制代码，首先创建包cn.sps.controller，在此包里创建类，类名MyUserController



填写如下代码：

|  |
| --- |
| **package** cn.sps.controller;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RestController;  //加入rest风格的controller,自动返回json格式  @RestController  **public** **class** MyUserController {  // 返回一个字符串  //首先需要加入一个注解，接收uri请求  @RequestMapping("/hello")  **public** String Hello(){  **return** "Hello SPS!";  }  } |

直接运行代码，在浏览器中输入localhost:8080/hello,会出现如下界面：



至此，springboot的运行环境已经搭建完成。

## 第二章 spring boot 入门案例

### 一、返回json格式的消息

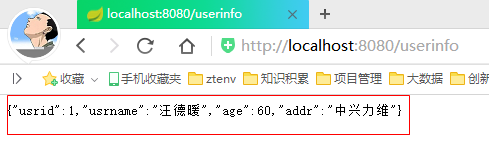
我们按照上面的实例，建立号spring boot环境，建立用户控制类MyUserController.java，在这里我们新建一个包，来开发模型类，包名叫cn.sps.domain,建立一个UserInfo类，类的代码如下：

|  |
| --- |
| **package** cn.sps.domain;  **import** java.io.Serializable;  **public** **class** UserInfo **implements** Serializable {  /\*\*  \*  \*/  **private** **static** **final** **long** ***serialVersionUID*** = -7176148921253559720L;  //userid  **private** Integer usrid;  //usrname  **private** String usrname;  //age  **private** Integer age;  //address  **private** String addr;  **public** Integer getUsrid() {  **return** usrid;  }  **public** **void** setUsrid(Integer usrid) {  **this**.usrid = usrid;  }  **public** String getUsrname() {  **return** usrname;  }  **public** **void** setUsrname(String usrname) {  **this**.usrname = usrname;  }  **public** Integer getAge() {  **return** age;  }  **public** **void** setAge(Integer age) {  **this**.age = age;  }  **public** String getAddr() {  **return** addr;  }  **public** **void** setAddr(String addr) {  **this**.addr = addr;  }    } |

修改MyUserController.java，加入一个请求映射，创建User返回请求，代码如下：

|  |
| --- |
| **package** cn.sps.controller;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RestController;  **import** cn.sps.domain.UserInfo;  //加入rest风格的controller,自动返回json格式  @RestController  **public** **class** MyUserController {  // 返回一个字符串  //首先需要加入一个注解，接收uri请求  @RequestMapping("/hello")  **public** String Hello(){  **return** "Hello SPS!";  }  // 返回POJO（普通的java对象，javabean）对象，  @RequestMapping("/userinfo")  **public** UserInfo GetUserInfo() {  UserInfo usrinfo = **new** UserInfo();  usrinfo.setUsrid(1);  usrinfo.setUsrname("汪德暖");  usrinfo.setAge(60);  usrinfo.setAddr("中兴力维");  **return** usrinfo;  }  } |

运行程序，在浏览器中输入<http://localhost:8080/userinfo>，界面显示如下：



自动返回了json格式

### 二、redis安装部署

1，linux下redis的安装与部署

操作环境：centos7

安装部署步骤如下：

下载Redis安装包

执行命令：wget <http://download.redis.io/releases/redis-4.0.6.tar.gz>

然后解压: tar -zxvf redis-4.0.6.tar.gz

进入目录执行：

cd redis-4.0.6

make

make install

建立配置文件目录：

Mkdir -p /usr/local/redis/etc/

复制配置文件到新路径：

cp redis.conf /usr/local/redis/etc/

安装完毕

2，运行redis服务

首先修改redis配置文件让其在后台运行

cd /usr/local/redis/etc/

vi redis.conf

把daemonize no改成daemonize yes

运行：redis-server /usr/local/redis/etc/redis.conf

查看进行：pgrep -l redis

查看端口：netstat -anop |grep redis

发现其默认端口为6379

3，运行redis客户端，验证服务

[root@localhost etc]# redis-cli

127.0.0.1:6379> set foo bar

OK

127.0.0.1:6379> get foo

"bar"

127.0.0.1:6379>

### 三、spring boot引入redis作为缓存