刘昱江 中心名称 + 班级号 + 名字 527161772

# 框架总结

## Mybatis

### 概念

Mybatis是持久层框架.Mybatis封装了JDBC,采用ORM(对象关系映射)方式封装了数据.简化代码的编辑和使用.

### 发展过程

JDBC方式操作数据库.

缺点:

1. 每次操作数据时,都需要频繁的开关链接. 后期引入连接池
2. 获取传输器对象和封装返回值结果繁琐.

### ORM对象关系映射

早期JDBC: sql----🡪 ResultSet--🡪GetString/GetInt----🡪对象

ORM:sql语句-----🡪对象

调用说明:

利用Mybatis中sqlSessionFactory对象,创建sqlSession主要的功能就是封装了JDBC.完成ORM对象关系映射.

## Spring

### 框架说明

概念:Spring框架可以整合所有的第三方框架.利用IOC和AOP极大的简化了程序的配置和调用的方式.让程序员编码时更加专注于业务的实现.

### IOC

说明:IOC控制反转.

知识回顾: 早期对象都是通过new关键字自己创建.

IOC说明:对象的创建的权利发生了变化.之前由程序员自己实例化对象.现在变成由Spring容器实例化对象.由Spring容器管理对象的生命周期.

#### IOC如何实例化对象

<bean id=”user” class=”com.jt.pojo.User”/>

Spring容器执行过程:

当tomcat服务器启动时会加载Spring容器的配置文件.当程序解析到Bean标签时.通过反射机制实例化对象.对象最终保存到了Spring容器自身维护的Map<Id,Object对象>

知识点:反射机制调用对象的无参构造实例化对象.

#### DI (依赖注入)

说明:为对象的**属性赋值**!!

将Spring容器中管理的对象最终通过set/构造器方式为对象的属性赋值.

注入方式:

1. Set注入
2. 构造器注入
3. 注解形式

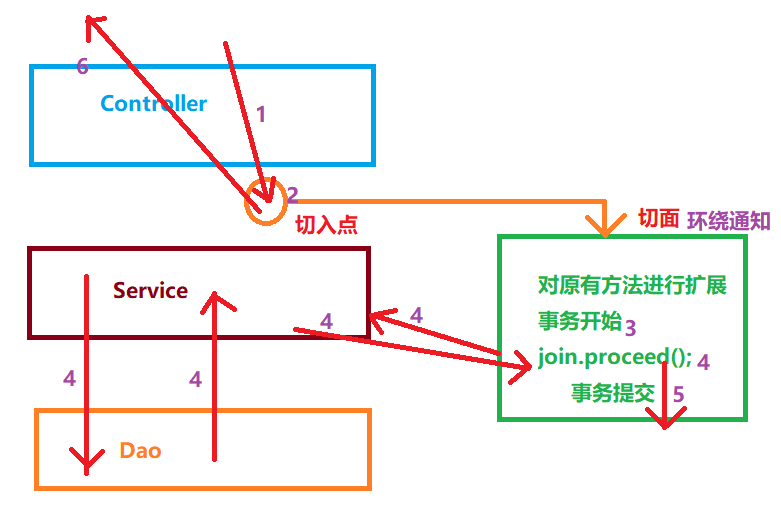
### AOP(面向切面编程)

核心思想:使用代理实现业务解耦.对原有的方式实现了扩展.

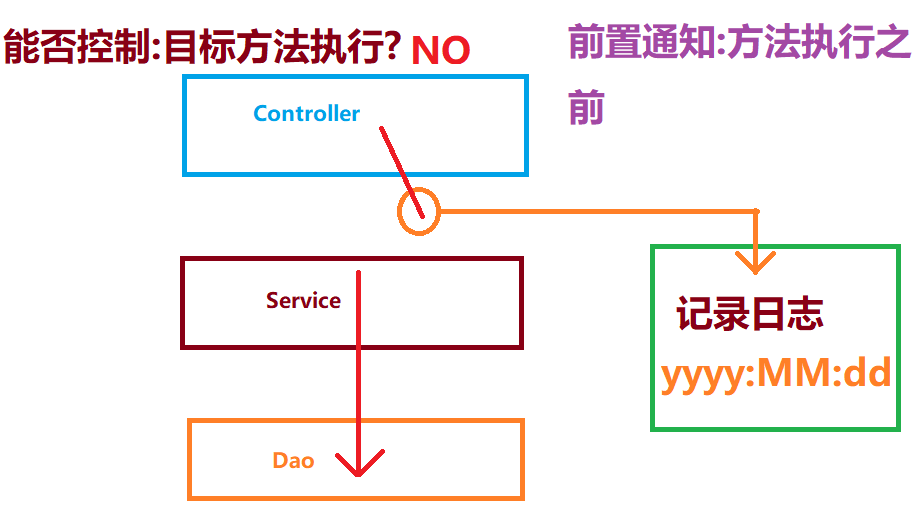
AOP组成部分:

1. 切入点(判断作用)
2. 通知(5大通知类型, 环绕通知功能最为强大.)

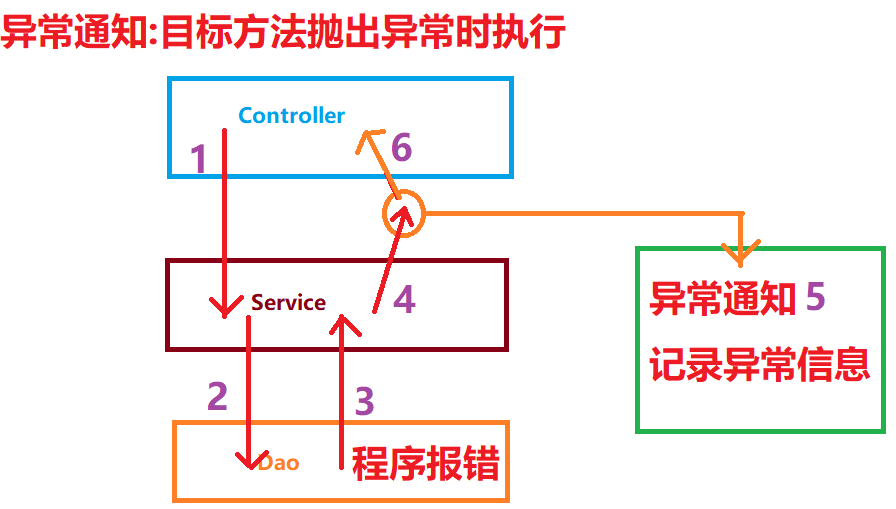
#### 环绕通知执行过程



#### 前置通知



#### 异常通知



#### 关于AOP总结

1. AOP中环绕通知使用是最多.因为可以控制目标方法是否执行.
2. 另外四大通知类型 只做记录.不做修改.

## SpringMVC

### SpringMVC作用

核心:**实现了用户的交互**.接收用户的参数,最终将”响应”的结果返回给用

户.

说明:SpringMVC底层封装了Servlet.极大的简化程序取值赋值的过程.

### Servlet复习

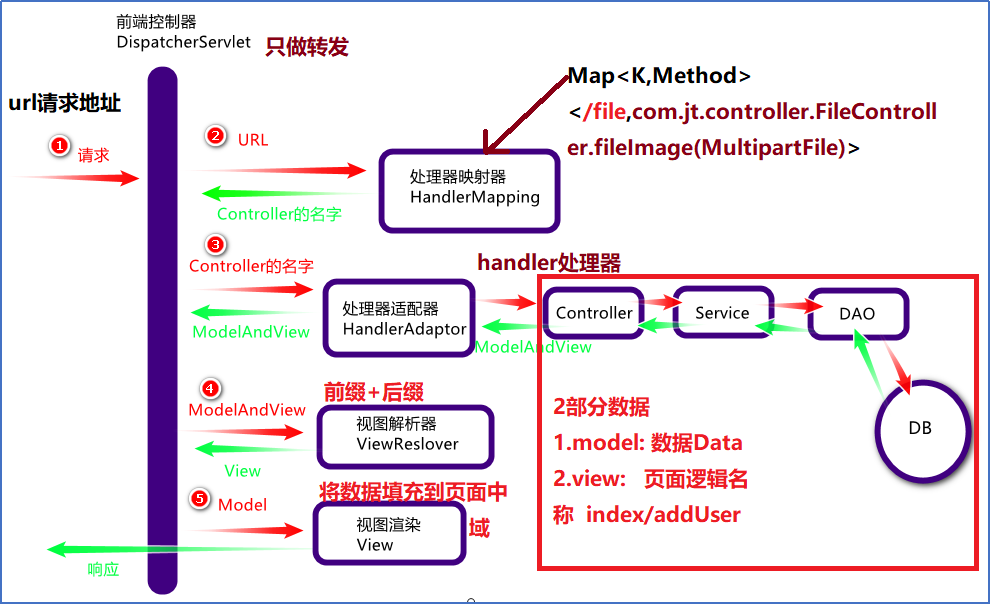
1. 配置繁琐:

在web.xml中必须配置8行标签.

1. 取值赋值繁琐
2. 利用Request/Response对象完成取值赋值操作.
3. 获取的数据都是String类型.如果需要获取其他数据类型需要进行强转.

### SpringMVC调用原理流程

重点知识:

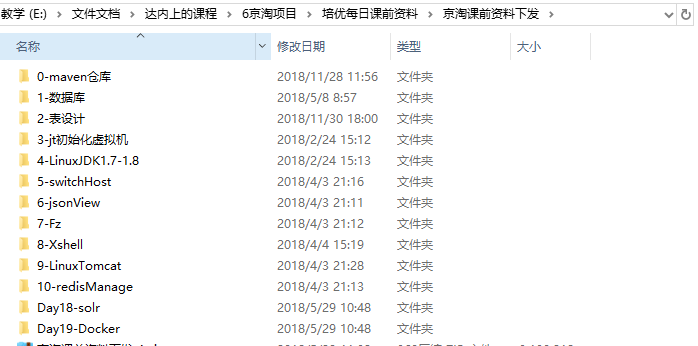


# SpringBoot

## 关于课前资料说明

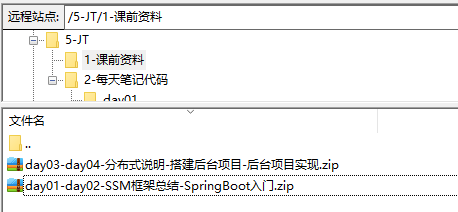
### 超大课前资料

说明:有些文件通过FTP进行输出速度很慢.包含软件安装包/JDK/Mysql/Vmware/PD.



### 课上资料

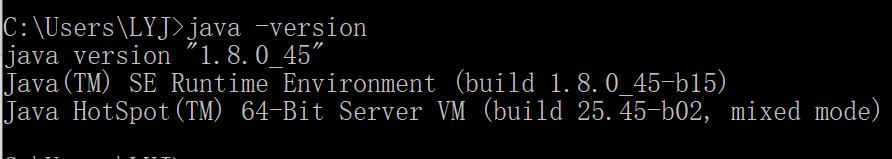
每天用到的一些文件,包括课上的配置文件等.



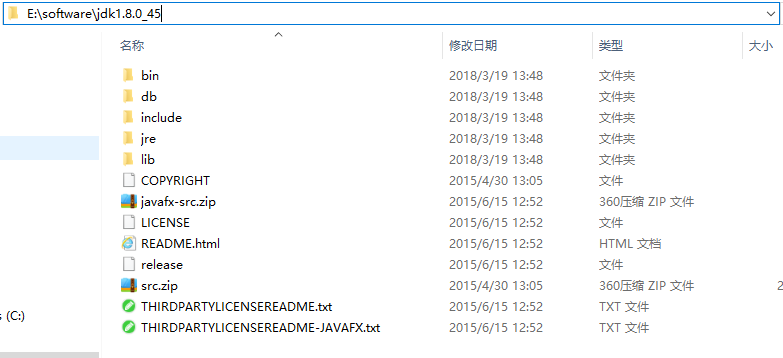
## 环境配置

### 安装JDK

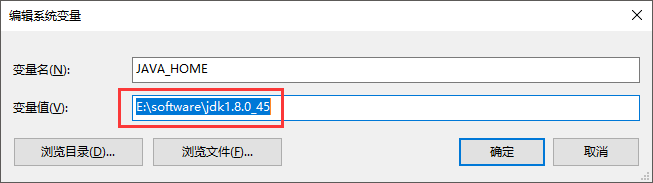
版本说明:必须使用JDK1.8版本.



1. JDK文件



1. 配置环境变量

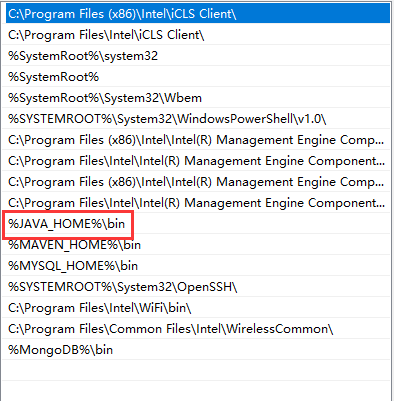


配置path:

Java中必须配置JDK.否则程序运行出错.注意写法

;%JAVA\_HOME%\bin

;D:/JDK1.8\bin



### 开发工具(IDE)

1. Eclipse
2. STS <https://spring.io/tools>
3. Idea

### Maven

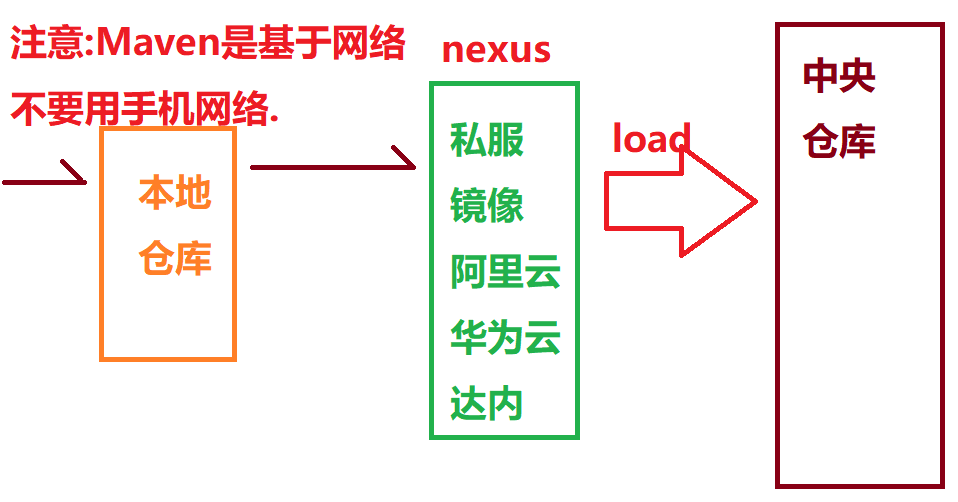
#### 版本要求

说明:Maven要求至少3.3及以上版本.

#### Maven工作原理

介绍:Maven是一站式的项目管理工具. 项目创建/项目打包/聚合/测试/jar包依赖.

原理:



#### Maven配置文件说明

说明:Maven中有settings配置文件.  
本地仓库设定(不是必须)

<!--定义本地仓库路径 如果pom.xml文件报错,则切换本地仓库-->

<localRepository>E:/software/mvn\_repo</localRepository>

<!-- 达内内网 Maven 镜像服务器配置 -->

<!--<mirror>

<id>nexus</id>

<name>Tedu Maven</name>

<mirrorOf>\*</mirrorOf>

<url>http://maven.tedu.cn/nexus/content/groups/public/</url>

</mirror>-->

<!--阿里云镜像-->

<mirror>

<id>nexus-aliyun</id>

<mirrorOf>\*</mirrorOf>

<name>Nexus aliyun</name>

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

</mirror>

JDK版本配置

<!-- 全局jdk配置，settings.xml -->

<profile>

<id>jdk18</id>

<activation>

<activeByDefault>true</activeByDefault>

<jdk>1.8</jdk>

</activation>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

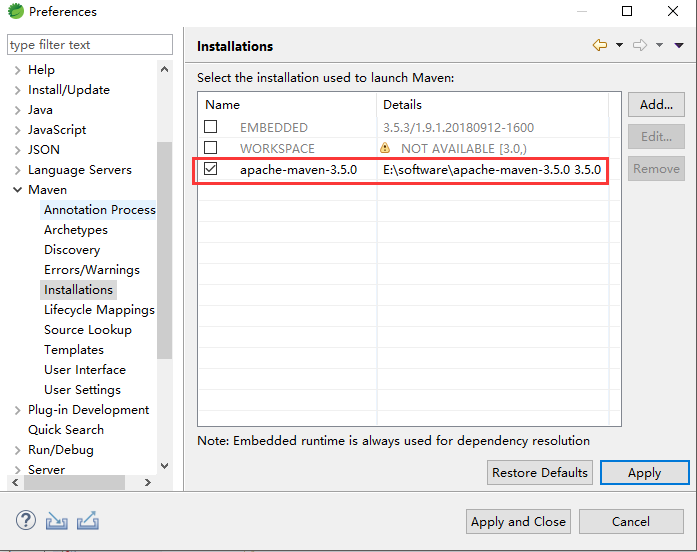
<maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>

</properties>

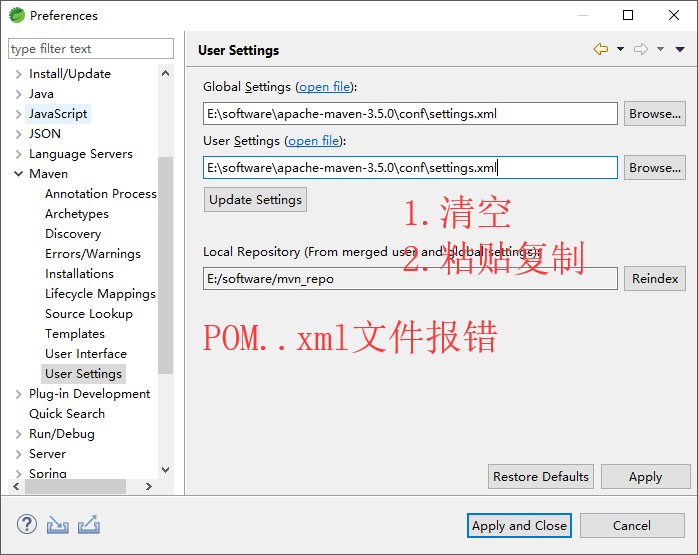
</profile>

**编辑路径时额外注意 不要有中文/不要放C盘**

#### STS中引入Maven

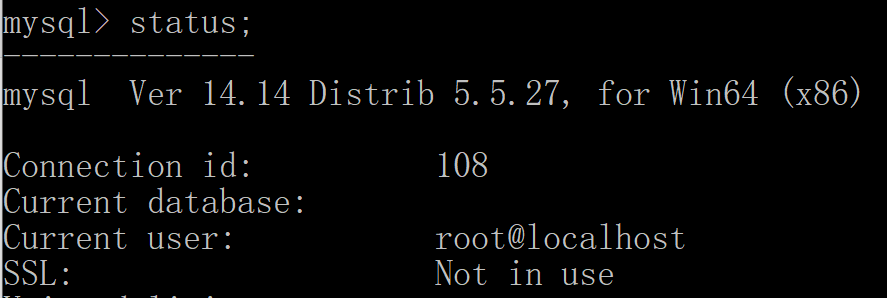


#### Maven Settings文件配置



### Mysql

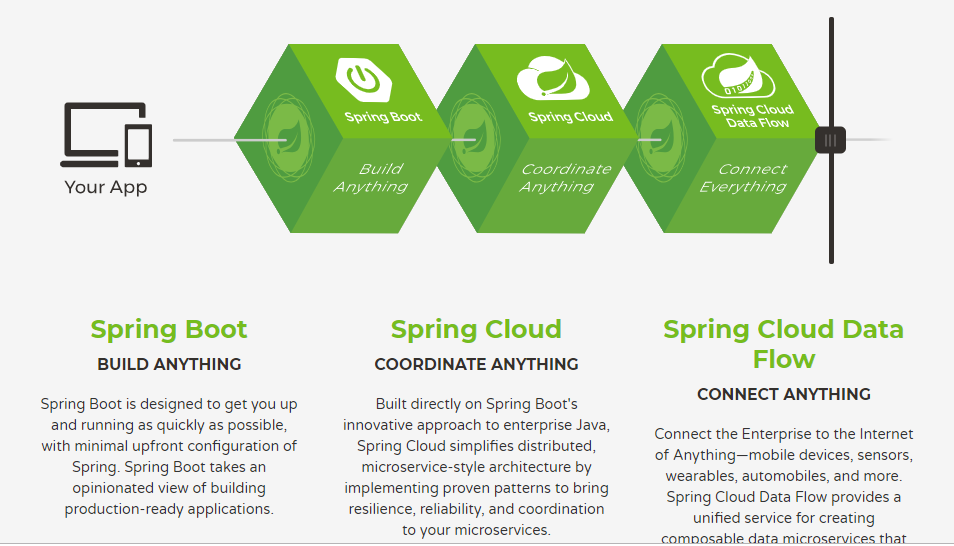
说明:Mysql要求至少5.5及5.7



## SpringBoot

### SpringBoot介绍

SpringBoot是学习SpringCloud的基石.也是当下最火的技术.



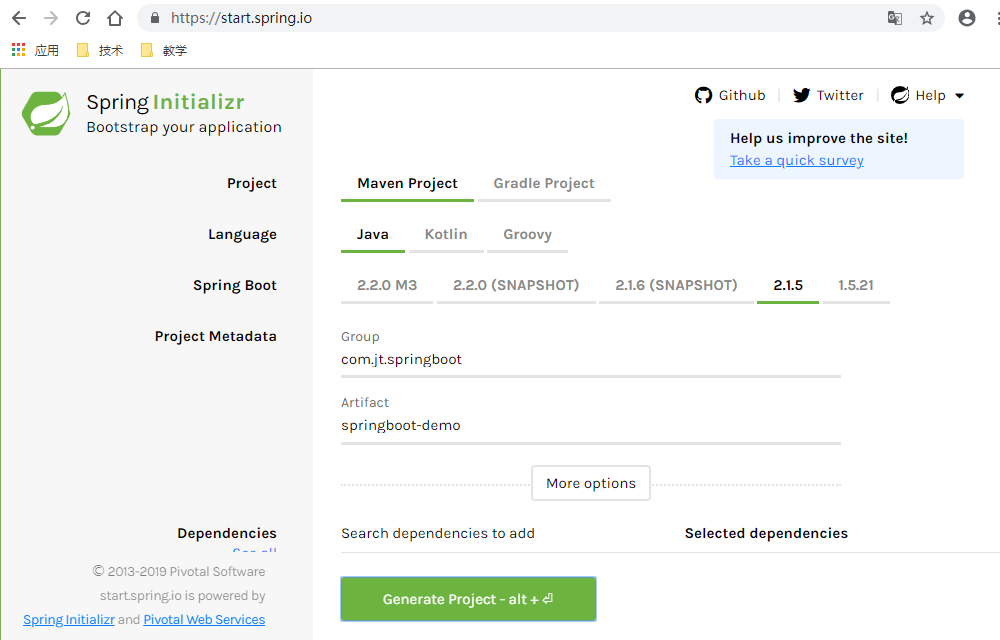
概念:

Spring Boot是构建所有基于Spring的应用程序的起点。Spring Boot旨在尽可能快地启动和运行，只需最少的Spring前端配置。自己内部添加了单独tomcat服务器.**要求项目尽可能独立运行**.

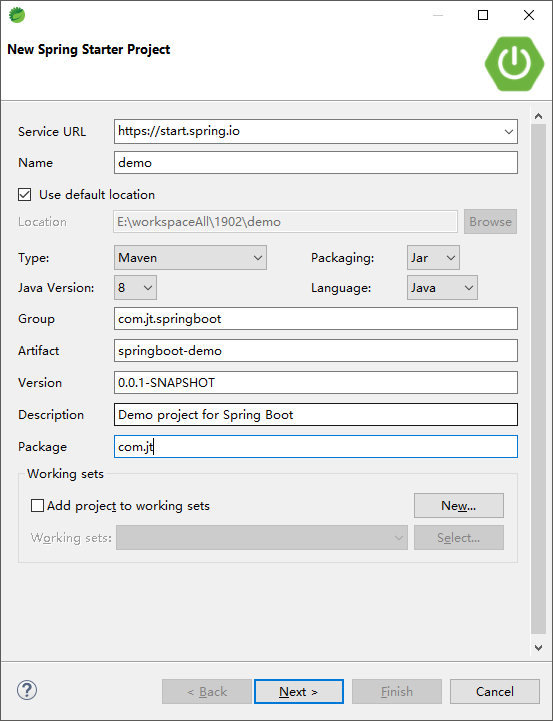
## SpringBoot入门案例

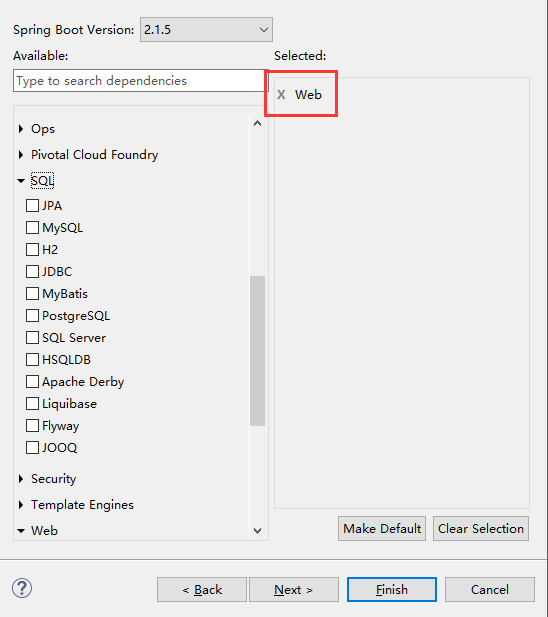
### 项目构建

#### 官网构建

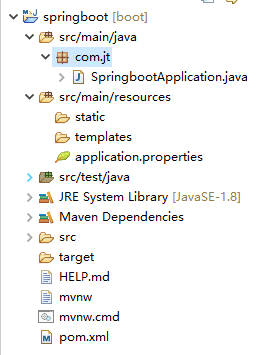


2.STS构建项目





项目创建结果,以下内容全部都是自动生成.



注意springBoot版本号编辑.将默认的SpringBoot版本改为2.1.3.RELEASE即可.

### 代码编辑

#### 编辑主启动类

注意:主启动类的包路径.就是包扫描的路径

@SpringBootApplication

**public** **class** SpringbootApplication {

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

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

}

}

#### 编辑测试类Controller

@SpringBootApplication

**public** **class** SpringbootApplication {

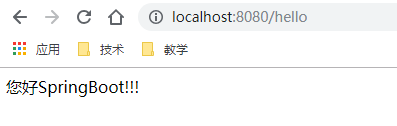
**public** **static** **void** main(String[] args) {

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

}

}

#### 页面效果



### 关于SpringBoot项目报错说明

#### 项目版本问题

将主版本由2.1.5改为2.1.3

<parent>

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

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

<version>2.1.3.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

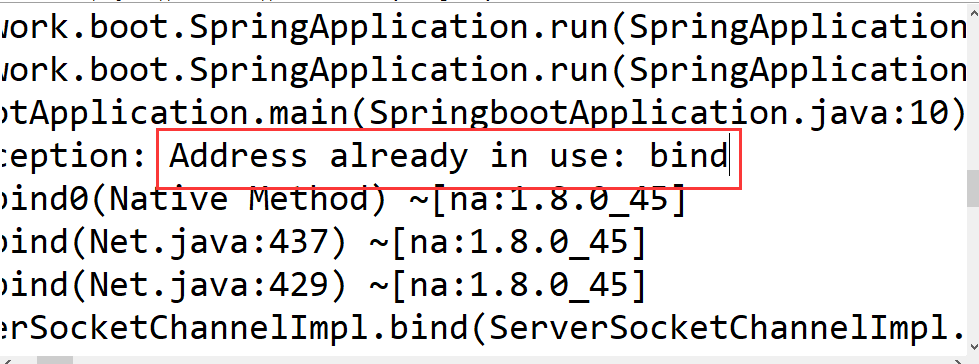
### POM.xml文件报错

通过检查发现jar包现在不完整.

1. 强制更新Maven文件
2. 修改maven链接镜像 将达内镜像改为阿里云镜像.

#### 端口被占用

说明:端口被占用.关闭进程项,重启启动.



### 关于SpringBoot中Pom.xml说明

<?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>

<!--继承springBoot程序中定义的公共jar包 -->

<parent>

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

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

<version>2.1.3.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<!--当前项目定义的坐标 -->

<groupId>com.jt.springboot</groupId>

<artifactId>springboot</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>springboot</name>

<description>Demo project for Spring Boot</description>

<!--定义JDK版本 -->

<properties>

<java.version>1.8</java.version>

</properties>

<!--jar包依赖

@EnableAutoConfiguration

主要依赖中添加了以spring-boot-starter

会自动的加载包中的配置信息.为了程序简化.

-->

<dependencies>

<dependency>

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

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

</dependency>

<!--自动添加SpringBoot测试类功能 -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<!--内部配置tomcat服务器.将来项目打包部署时必须依赖该配置 -->

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

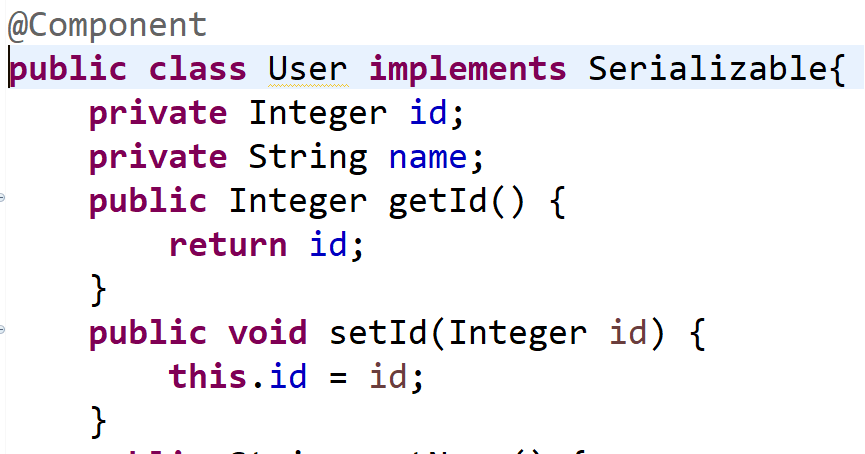
</build>

</project>

## SpringBoot测试类使用

### 定义实体对象

定义POJO对象.



### 测试类测试

@RunWith(SpringRunner.**class**)

@SpringBootTest//springboot启动加载对象

**public** **class** SpringbootApplicationTests {

@Autowired

**private** User user;

@Test

**public** **void** testUser() {

user.setId(100);

user.setName("1902班!!!!");

System.***out***.println(user);

}

}