内网进行spring-boot开发

# 简介

使用Spring-Boot可以轻松的创建独立运行的程序，非常容易构建独立的服务组件，是实现分布式架构、微服务架构的利器。Spring-Boot简化了第三方包的引用，通过提供的starter，简化了依赖包的配置。

## Spring-Boot的优点

* 轻松创建独立于Spring的应用程序
* 内嵌Tomcat、jetty等web容器，不需要部署WAR文件
* 提供一系列的starter来简化maven配置。
* 提供生产就绪性功能，如指标，健康检查和外部配置

# 环境部署

因为要在内外网协作时完成jar包的解决，因此务必保持内网和外网的配置保持一致，这些配置包括如下：

## JDK环境部署

在内网和外网上部署的JDK版本为，因为安装在默认目录，因此配置的JAVA\_HOME系统变量值如下：

C:\Program Files\Java\jdk1.8.0\_131

编辑系统变量Path：

%JAVA\_HOME%\bin

放置在末尾即可。

验证JDK成功安装：

C:\Users\admin>java -version

java version "1.8.0\_131"

Java(TM) SE Runtime Environment (build 1.8.0\_131-b11)

Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)

## Maven环境部署

安装其他版本的Maven，可以访问url：

https://archive.apache.org/dist/maven-3/

保持内外网一致即可。配置Maven的环境需要确保JAVA\_HOME系统变量的存在；自己安装的maven版本为

apache-maven-3.3.9-bin.tar.gz

若解压至C盘，在环境变量中新建系统变量MAVEN\_HOME，值为Maven安装目录

C:\apache-maven-3.3.9

以同样的方式编辑系统变量Path，添加如下内容：

%MAVEN\_HOME%\bin

验证Maven环境配置成功

C:\Users\admin>mvn -v

Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-11T00:41:4

7+08:00)

Maven home: C:\apache-maven-3.3.9\bin\..

Java version: 1.8.0\_131, vendor: Oracle Corporation

Java home: C:\Program Files\Java\jdk1.8.0\_131\jre

Default locale: zh\_CN, platform encoding: GBK

OS name: "windows 7", version: "6.1", arch: "amd64", family: "dos"

## IDEA安装

下载IDEA安装破解，同样保持内外网的版本一致即可。

# 依赖解决步骤

由于在使用Spring-Boot进行开发时，一般情况下使用Maven管理jar包依赖，而在内网开发时，受限于局域网，一般也没有搭建Maven私服的情况下，则需要在外网下载好依赖，然后在内网进行本地库配置。具体的步骤如下。

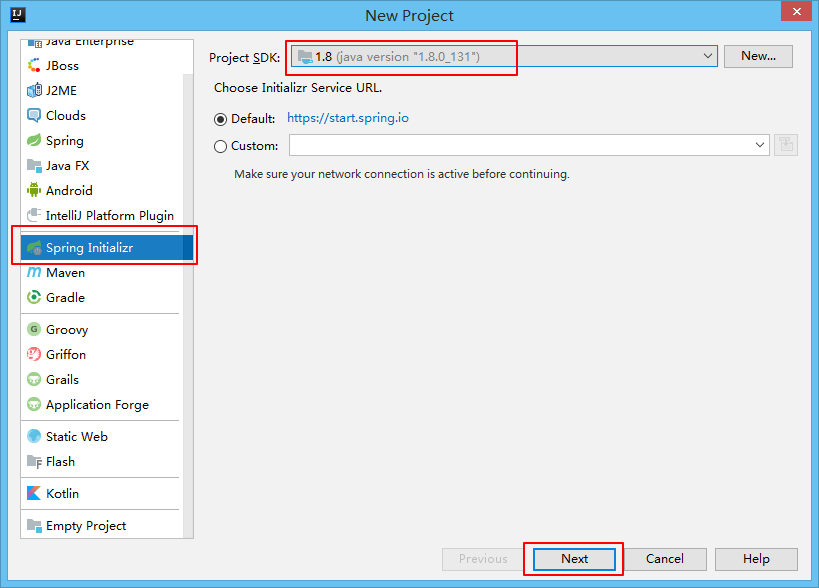
## 外网下载jar包依赖

### 外网创建项目

在外网创建项目，首先打开IDEA，界面如下：



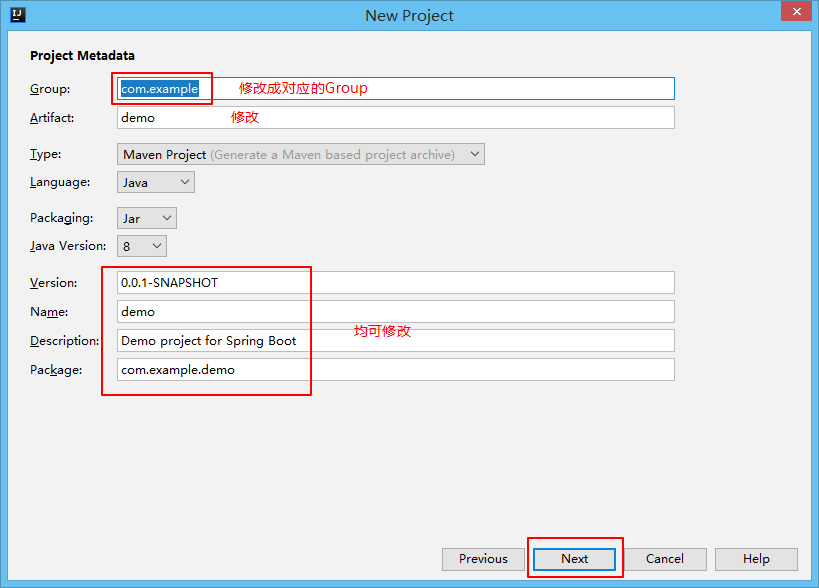
选择Create New Project，弹出如下界面：



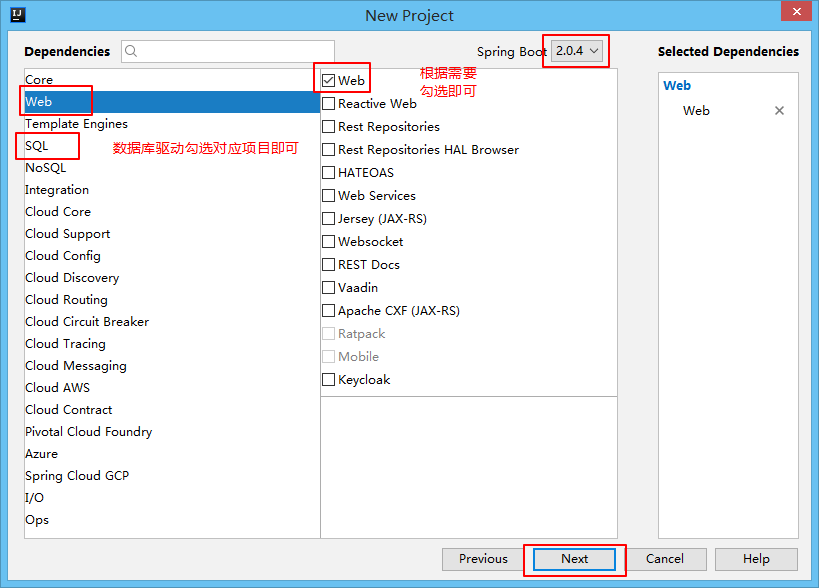
勾选Spring Initializr， Project SDK选择JDK安装目录，Initializr Service URL选择默认即可，因为外网一般可以连接网络，因此需要通过网站

https://start.spring.io

交互创建模板项目。选择Next



在上述界面中，根据开发的需要更改Project Metadata，在这里仅做演示，默认即可。Type勾选Maven Project，选择Next.



在演示的Demo中，取官网所提供的Restful程序，只需要勾选Web即可，如果需要与数据库进行交互，则可以勾选SQL，内含常用的数据库支持，如MySQL，JDBC，MyBatis等。因文档只演示内网开发时jar包依赖的解决，并不需要，因此仅勾选Web即可。若想要更改Spring-Boot的版本，选择左上的Spring Boot，选择合适的版本即可。

### 编写pom文件

IDEA会自动以我们在之前选择的程序结构生成对应的pom文件，如下：

<?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>com.example</groupId>

<artifactId>demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>demo</name>

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

<parent>

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

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

<version>2.0.4.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>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

### IDEA配置settings.xml

因为外网需要下载jar包，而国外的仓库下载缓慢，因此一般在开发时会使用阿里云的maven仓库。因此在如下目录修改settings.xml如下：

<?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">

<!-- localRepository

| The path to the local repository maven will use to store artifacts.

|

| Default: ${user.home}/.m2/repository

<localRepository>/path/to/local/repo</localRepository>

-->

<localRepository>C:\apache-maven-3.3.9\conf\repository</localRepository>

<mirrors>

<mirror>

<id>alimaven</id>

<name>aliyun maven</name>

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

<mirrorOf>central</mirrorOf>

</mirror>

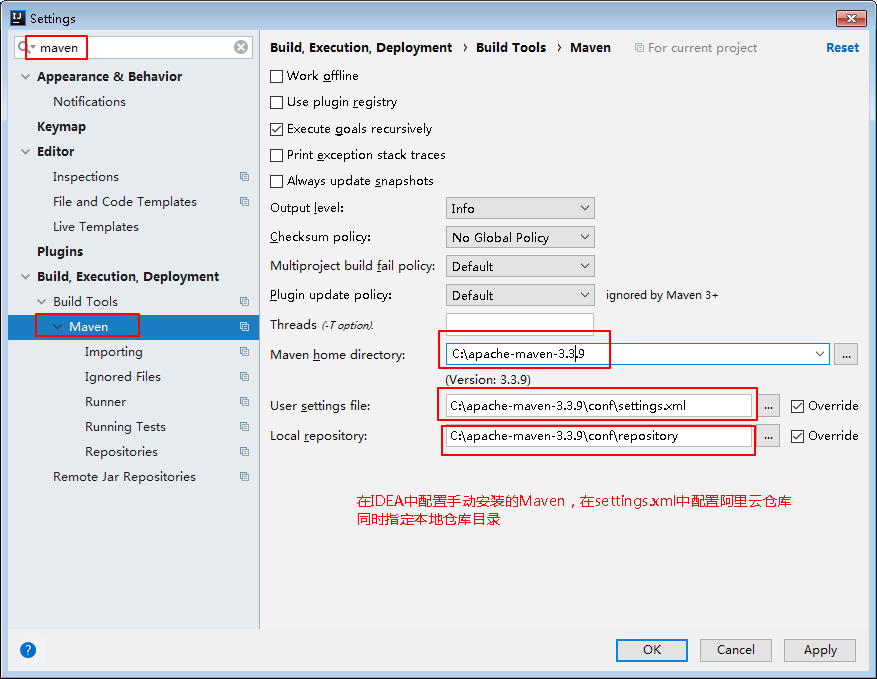
</mirrors>

</settings>

在上述的配置文件中，配置了本地仓库，位置如目录所示。

### 依赖下载Import Change

在IDEA中设置下搜索Maven

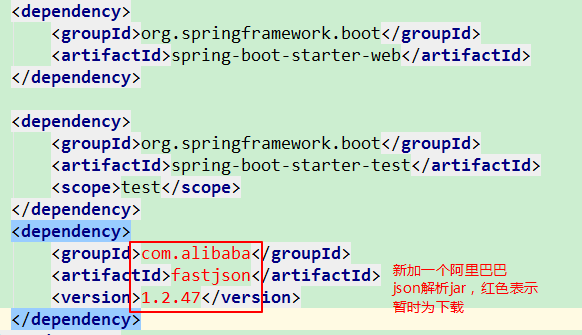


打开文件pom.xml，在空白处右击选择Maven->Reimport。可以看到IDEA正在为我们下载jar包依赖，并且会把下载好的依赖保存进本地仓库目录：

C:\apache-maven-3.3.9\conf\repository

### 导入fastjson包

在演示Demo中，使用了阿里巴巴提供的fastjson库进行json的解析和组装，因此，需要在pom.xml文件中添加fastjson的jar包依赖，如下：



依赖显示红色，表示该依赖在本地仓库中未找到，修改过后的pom.xml文件如下：

<?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>com.example</groupId>

<artifactId>demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>demo</name>

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

<parent>

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

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

<version>2.0.4.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>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.2.47</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

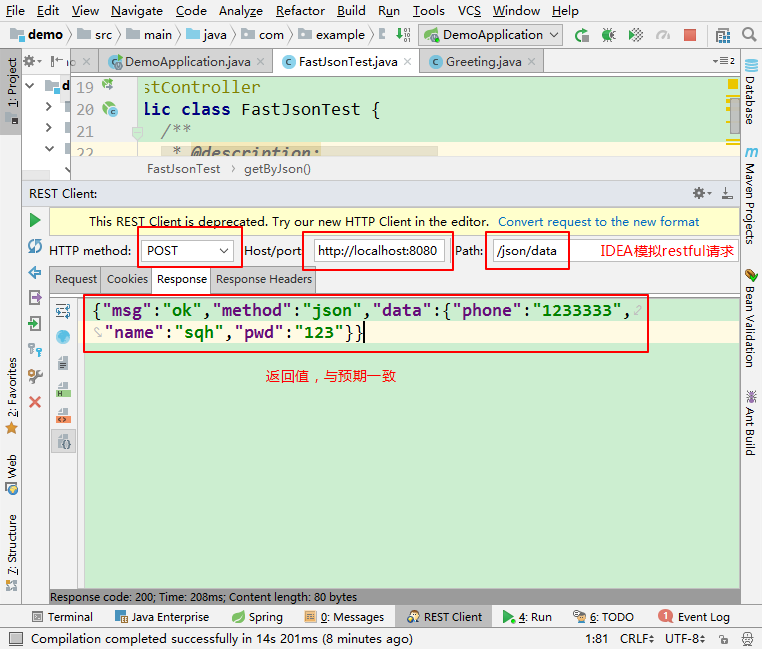
</build>

</project>

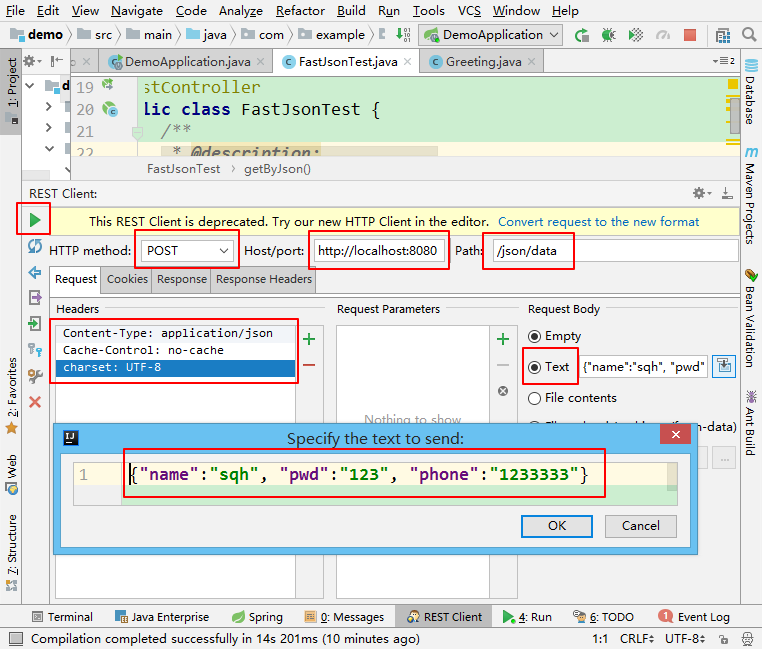
右击选择Maven->Reimport，待下载完成fastjson依赖，可以看到红色变为黑色，表示依赖已经解决。

### 启动验证

运行项目，通过IDEA自带的Restful模拟工具进行调用，如下图所示：



同时需要修改Request的参数，如下图所示：

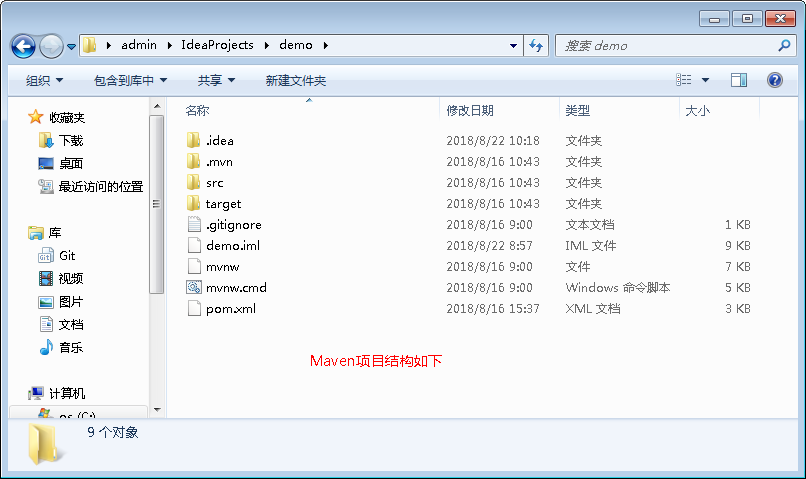


## 拷贝

### 拷贝项目和repository

拷贝完整的项目和IDEA拉取的repository。

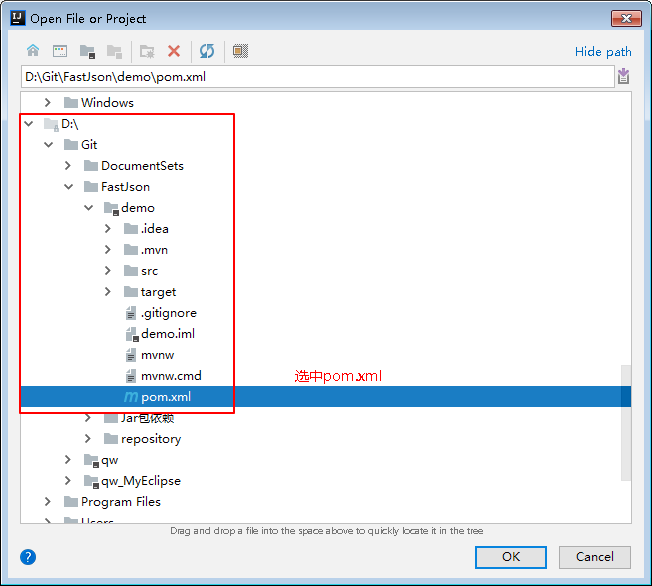
demo项目结构如下：



## 内网配置

### 项目导入IDEA

选择File->Open->导航到demo所在目录，选择pom.xml



点击OK，在弹出的对话框中选择Open as Project->Open Existing Project->New Window即打开了在外网创建的工程。

### 配置settings.xml和本地仓库

新建目录如下：

C:\Users\admin\test

在该目录下创建settings.xml文件，内容如下：

<?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">

<!-- localRepository

| The path to the local repository maven will use to store artifacts.

|

| Default: ${user.home}/.m2/repository

<localRepository>/path/to/local/repo</localRepository>

-->

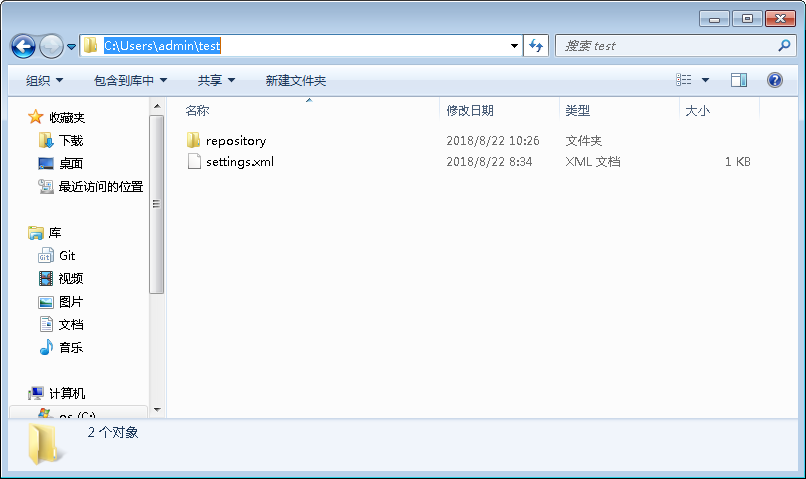
<localRepository>C:\Users\admin\test\repository</localRepository>

<mirrors>

</mirrors>

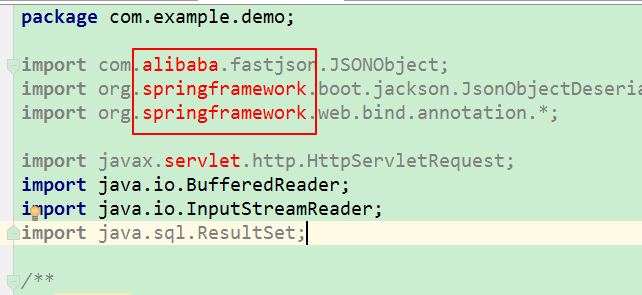
</settings>

这是因为在内网中，因为没有网络连接，只需要配置本地仓库即可，所有的Maven依赖均在本地仓库中，根据<localRepository>的含义，把拷贝进内网的仓库放置在指定位置。该配置文件指定的位置与settigns.xml处在同级目录下，因此把拷贝进入内网的仓库拷贝进该目录即可。

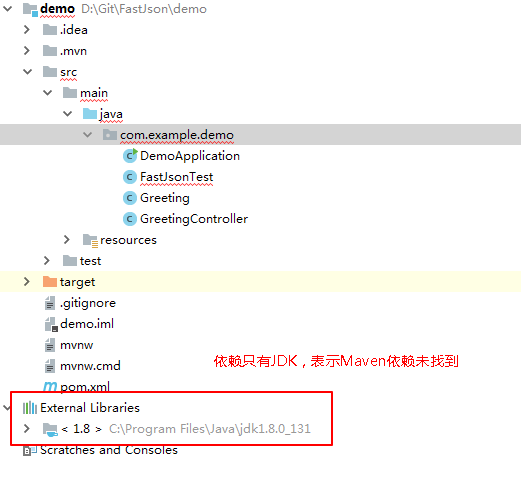


### IDEA内网配置Maven

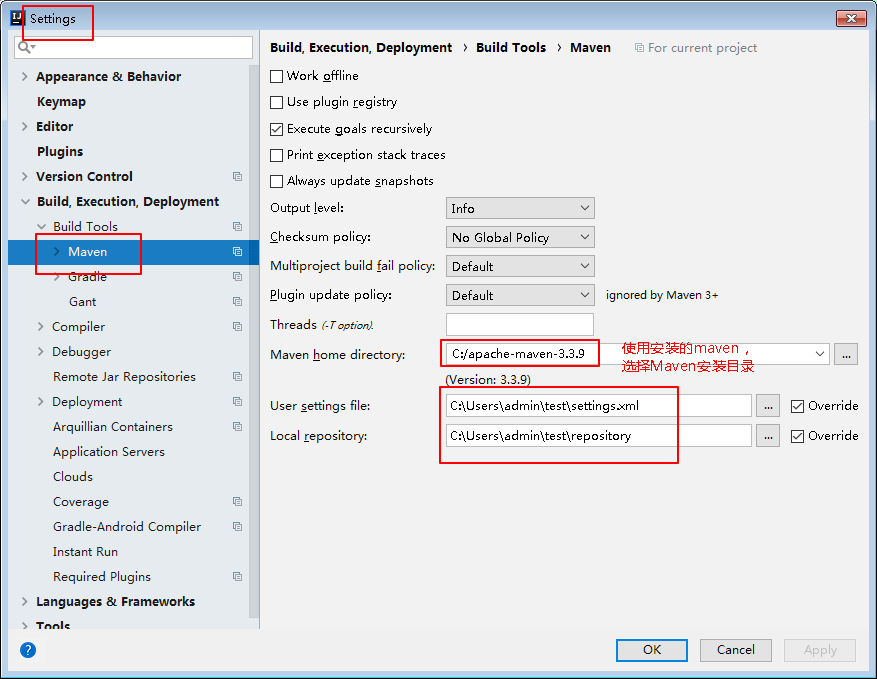
打开FastJsonTest.java，可以看到未配置Maven的IDEA在alibaba和springframework显示标红，表示未找到。



在项目结构中查看External Libraries中查看工程依赖：



配置maven如下：



确保Maven home directory勾选为安装的Maven目录，然后设置User settings file和Local repository。点击OK。

### 导入依赖

打开pom.xml文件，右击选择Maven->Reimport

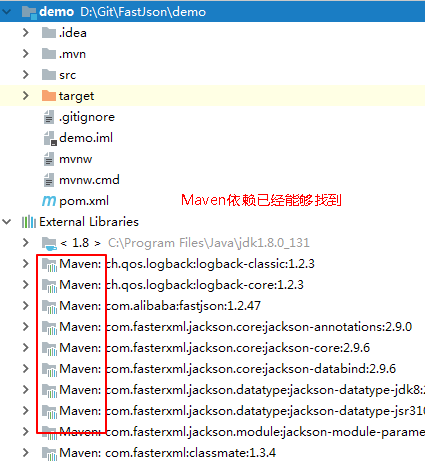
待Maven在本地仓库中解决所有的依赖，可以看到如下的结果

### 验证

#### FastJsonTest.java



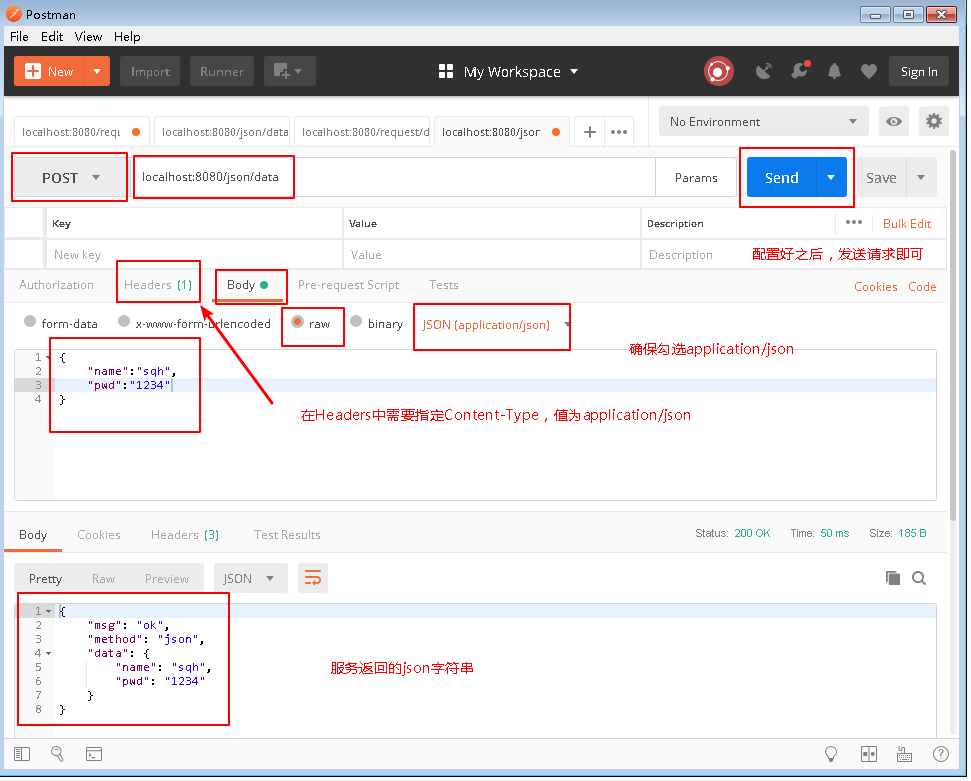
#### External Libraries



#### 运行程序

点击IDEA的运行程序。

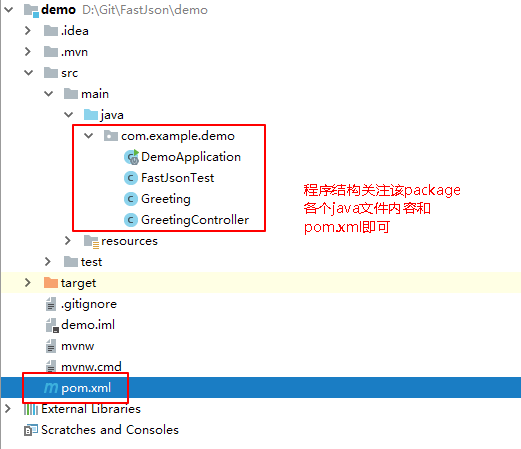
启动Postman程序模拟Restful请求



可以看到，程序正常能够工作，并且返回中也已经得到了预期的json字符串。

# 实例FastsonTest

## 程序结构



## DemoApplication.java

main函数所在文件，代码如下：

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

System.out.println("End");

}

}

## FastJsonTest

该文件演示了使用阿里巴巴的fastjson包进行json的解析和组装的具体过程，非常实用。

package com.example.demo;

import com.alibaba.fastjson.JSONObject;

import org.springframework.boot.jackson.JsonObjectDeserializer;

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

import javax.servlet.http.HttpServletRequest;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.sql.ResultSet;

/\*\*

\* @param: jsonParam

\* @return

\* 直接通过@RequestBody的方式，直接将json的数据注入到JSONObject里面

\*/

@RestController

public class FastJsonTest {

/\*\*

\* @description:

\* @url: localhost:8080/json/data

\* @author:

\* @date: 2018/8/16-17:16

\* @return:

\*/

@ResponseBody

@RequestMapping(value = "/json/data", method = RequestMethod.POST, produces = "application/json;charset=UTF-8")

public String getByJson(@RequestBody JSONObject jsonParam) {

System.out.println("jsonParam: "+jsonParam.toJSONString());

if(jsonParam.containsKey("name")) {

System.out.println("Json contains key name");

}

if (jsonParam.containsKey("pwd")) {

System.out.println("Json contains key password");

}

if (jsonParam.containsKey("phone")) {

System.out.println("Json contains key phone");

}

String name = jsonParam.getString("name");

String pwd = jsonParam.getString("pwd");

JSONObject result = new JSONObject();

result.put("msg", "ok");

result.put("method", "json");

result.put("data", jsonParam);

return result.toJSONString();

}

@ResponseBody

@RequestMapping(value = "/request/data", method = RequestMethod.POST)

public String getByRequest(HttpServletRequest request) {

System.out.println(request.getMethod());

System.out.println(request.getQueryString());

System.out.println(request.getParameterNames().toString());

System.out.println(request.getRequestURI());

System.out.println(request.getRequestURL());

JSONObject jsonParam = this.getJsonParam(request);

JSONObject result = new JSONObject();

result.put("msg", "ok");

result.put("method", "request");

result.put("data", jsonParam);

System.out.println(jsonParam.toJSONString());

return result.toJSONString();

}

public JSONObject getJsonParam(HttpServletRequest request) {

JSONObject jsonParam = null;

try {

BufferedReader streamReader = new BufferedReader(new InputStreamReader(request.getInputStream(), "UTF-8"));

StringBuilder sb = new StringBuilder();

String line = null;

while ((line = streamReader.readLine())!=null) {

System.out.println(line);

sb.append(line);

}

jsonParam = JSONObject.parseObject(sb.toString());

} catch (Exception e) {

e.printStackTrace();

}

return jsonParam;

}

}

## Greeting

Greeting是一个资源表示类(resource representation class)，该类型代码如下：

package com.example.demo;

public class Greeting {

private final long id;

private final String content;

public Greeting(long id, String content) {

this.id = id;

this.content = content;

}

public long getId() {

return id;

}

public String getContent() {

return content;

}

}

## GreetingController

GreetingController类是一个资源控制器类，用来进行url请求与类型方法的绑定。该类型代码如下：

package com.example.demo;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.support.spring.messaging.MappingFastJsonMessageConverter;

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

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

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

import java.util.concurrent.atomic.AtomicLong;

@RestController

public class GreetingController {

private static final String template = "Hello, %s";

private final AtomicLong counter = new AtomicLong();

@RequestMapping("/greeting")

public Greeting greeting(@RequestParam(value = "name", defaultValue = "world") String name) {

return new Greeting(counter.incrementAndGet(), String.format(template, name));

}

}

# 总结

在之前内网Jar包依赖的问题曾困扰了很久，请教了许多人也没能解决，最后在出差时在同事的帮助下，自己竟然把这个问题给解决了，之前使用的方式是在内网搭建maven私服，这种方式也可以自然的解决maven依赖的问题，但由于出差的工作环境内网是虚拟机，因此无法部署虚拟机和docker，因此暂时就用这种repository仓库拷贝的方式解决依赖。整个过程并不复杂，但确实能够解决问题，为进一步使用spring-boot开发微服务解决了基础问题。