<http://www.cnblogs.com/gwx238044/articles/6633252.html>



web中添加spring的配置文件

## 因为我们编译之后读取的是classpath的路径，所以即使配置文件在dao层也是可以的

**一、开发环境**

**Eclipse:eclipse-jee-luna-SR1a-win32;**

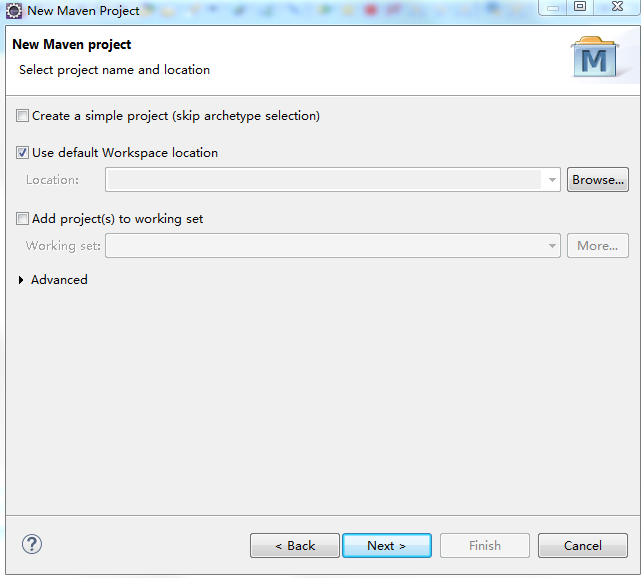
**JDK:jdk-8u121-windows-i586.exe;**

**MySql:MySQL Server 5.5;**

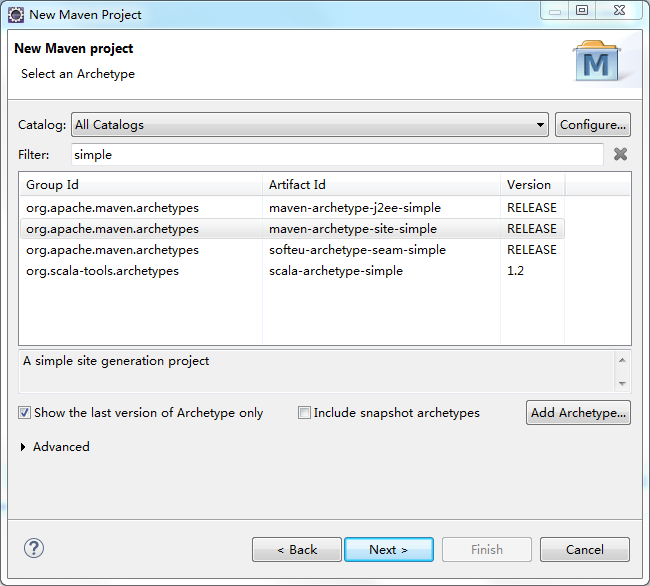
**二、创建Maven项目**

**1.创建父项目**

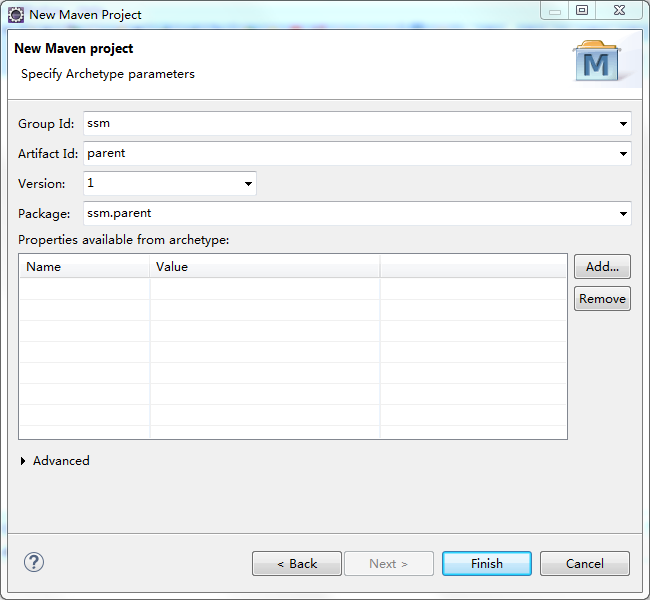
**NEW->PROJECT->MAVEN->Maven Project,点击next>进入下一步，可以看到New Maven Project界面,**

****

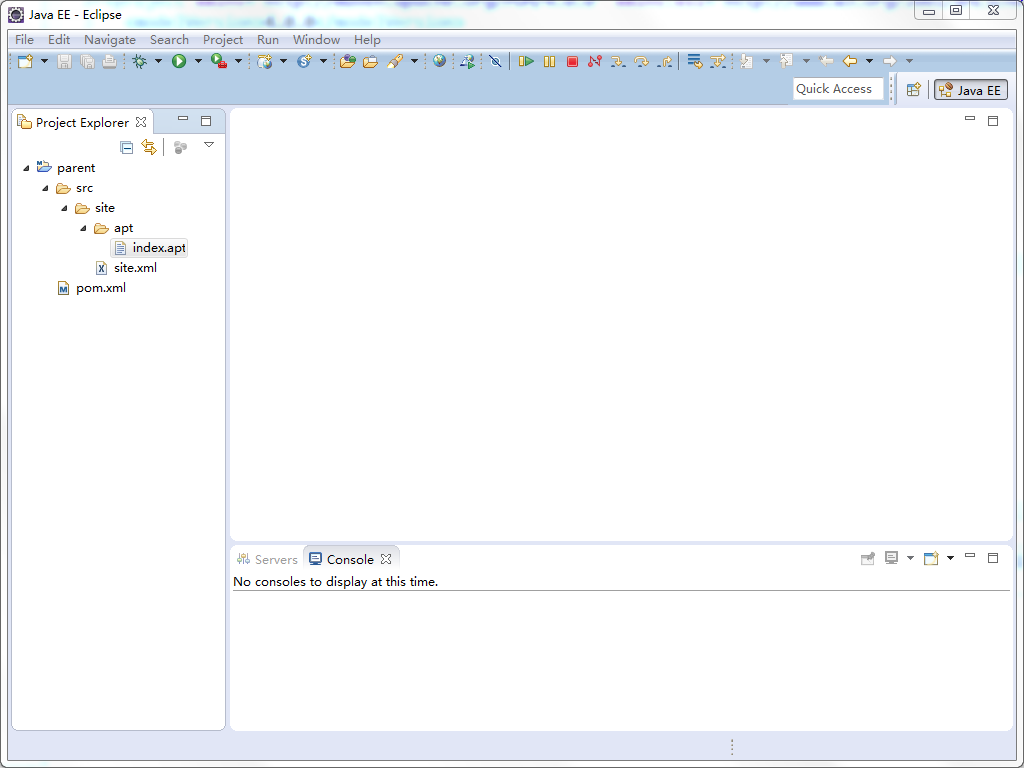
**点击Next >进入下一步,进入Select an Archetype界面，在Filter:输入simple,选择org.apache.maven.archetypes,**



点击next>,进入Enter a group id for the artifact界面，

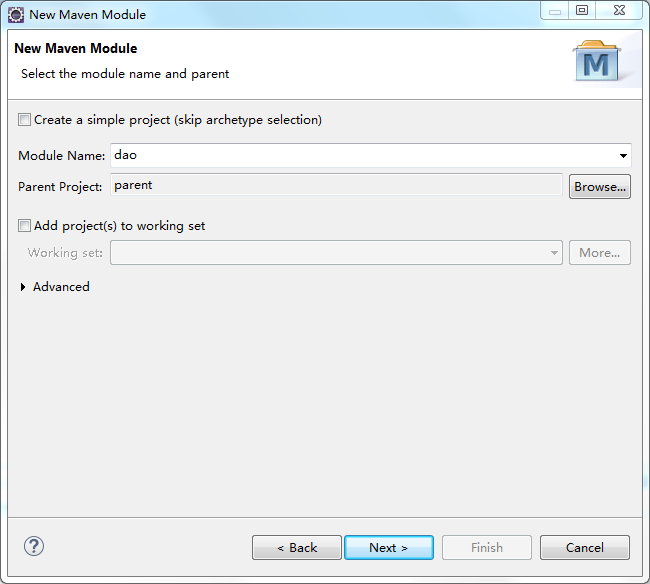


填写好Group Id和Artifact Id，我这里在ssm分组下创建了一个parent项目。点击Finish。

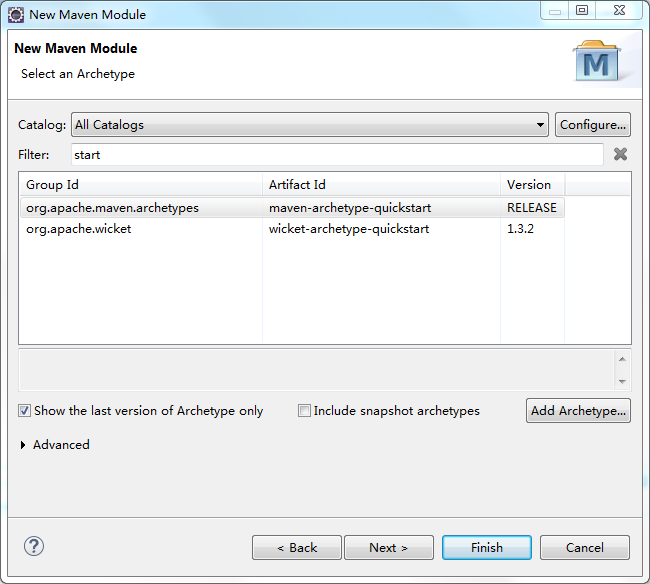


**2.创建子项目**

选中parent项目，右键，选择new->project->maven->maven Module,点击Next,进入Enter a module name界面，

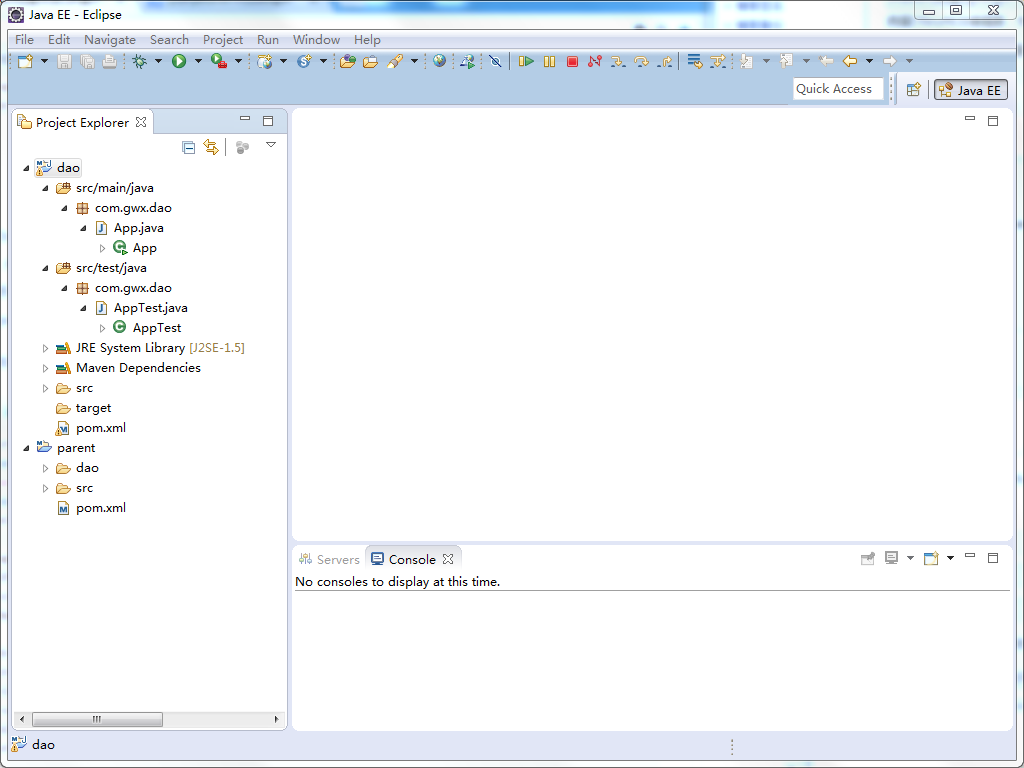


填写Module Name，点击next>,进入Select an Archetype界面，

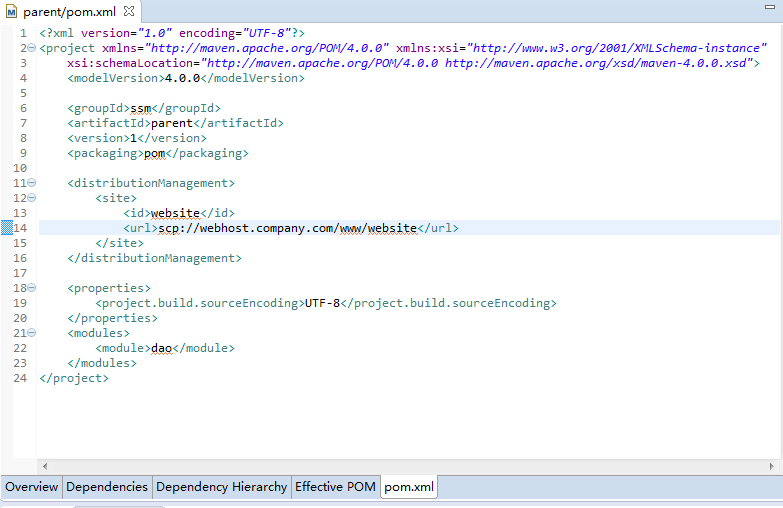


 在Filter输入start，选择maven-archetype-quickstart，点击next>,进入Specify Archetype parameters界面，

修改Package,点击Finish。就会创建parent项目下的一个子项目dao。



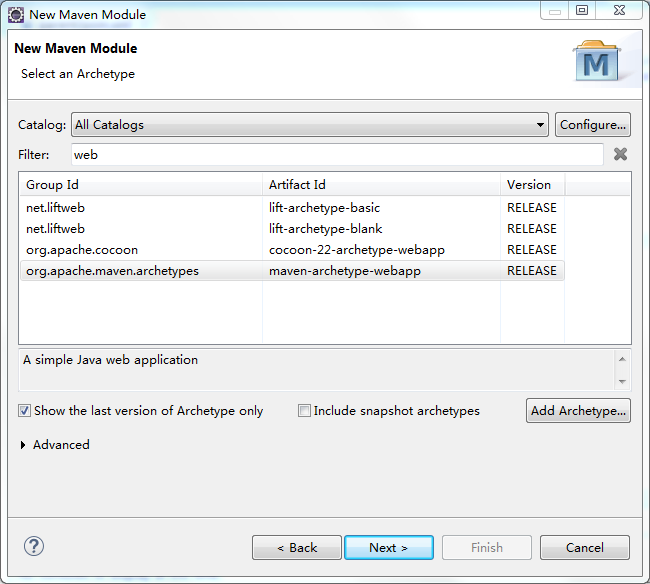
打开parent项目的pom.xml



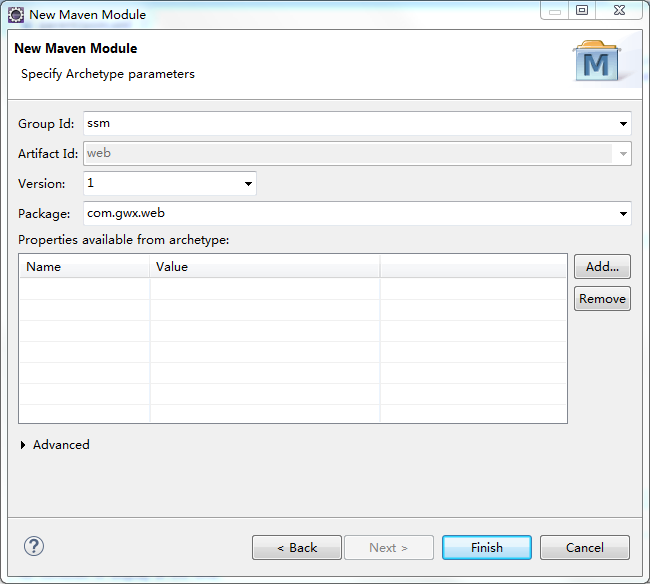
可以看到有个<moduls><module>dao</module></modules>,这说明dao已经是parent项目的子项目了。

接着我们按照上述方法继续创建子项目service和model。

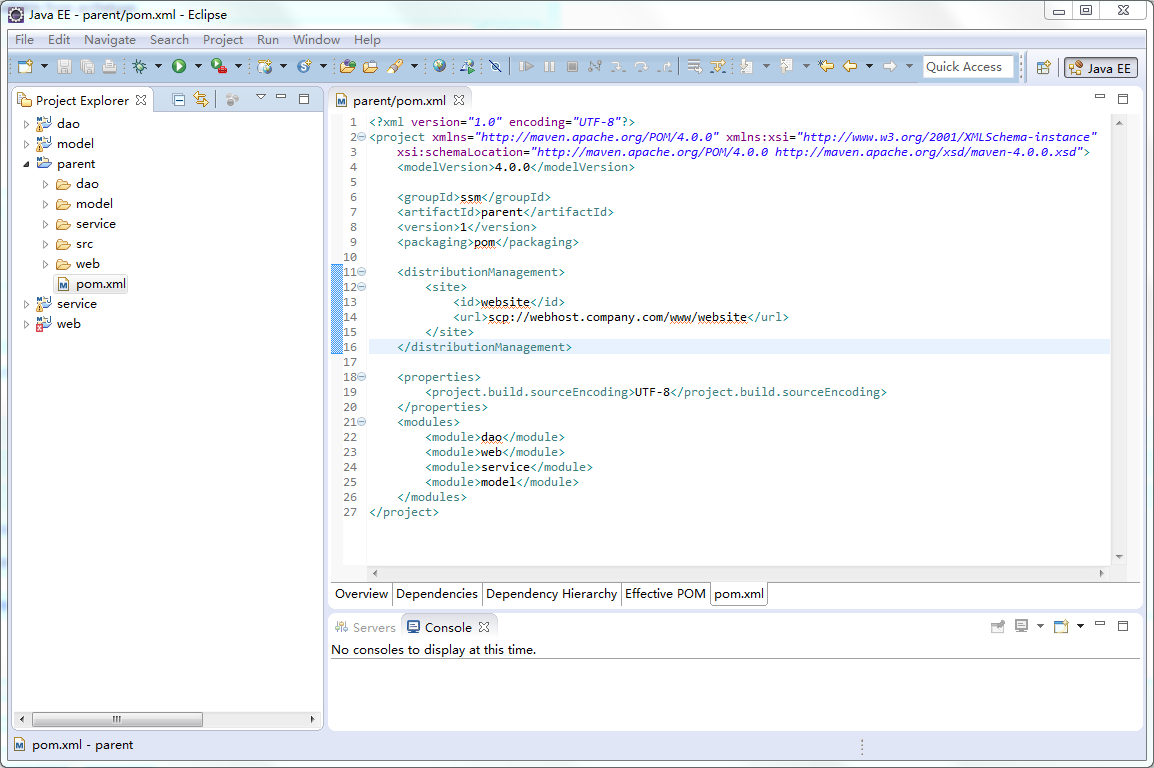
创建web子项目时，在进入Select an Archetype界面，在Filter填写web，选择org.apache.maven.archetypes.



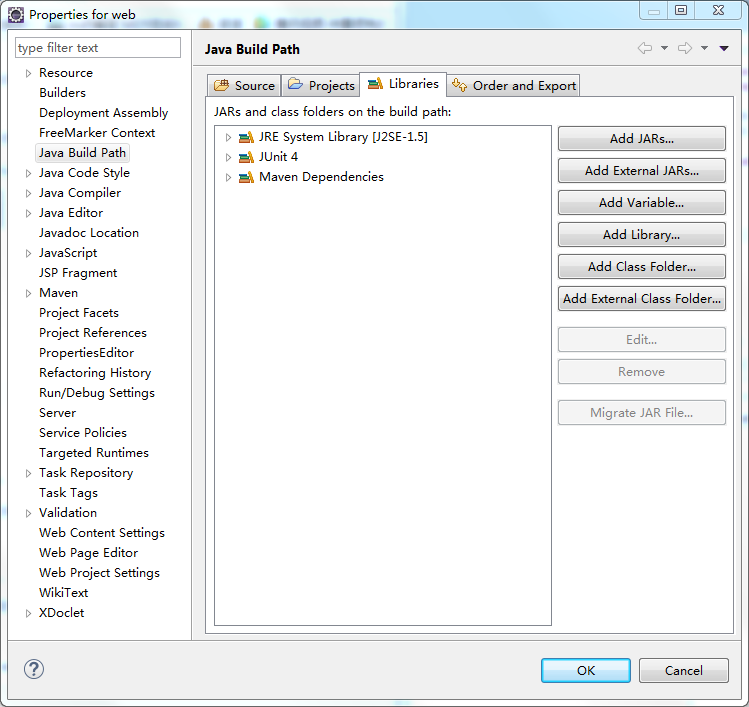
点击next>,修改package，最后点击Finish。



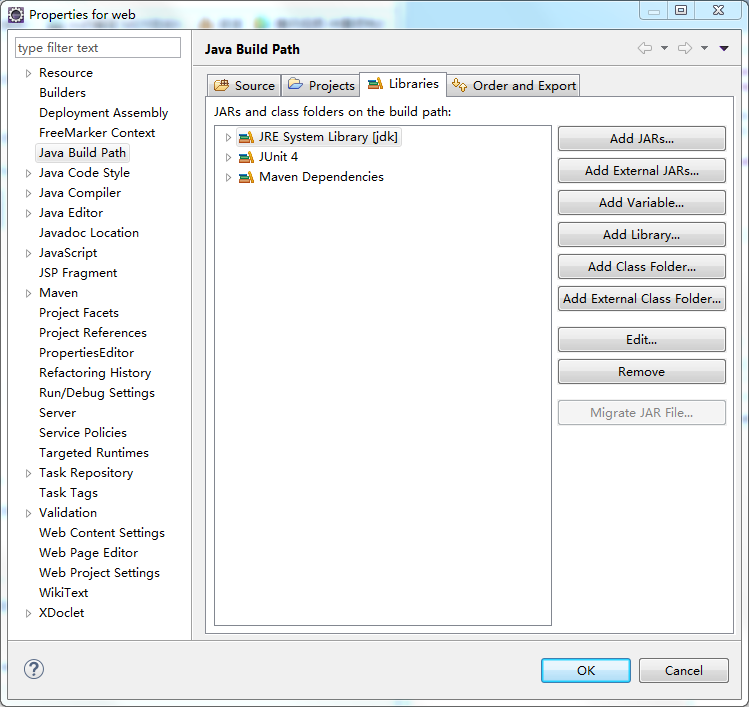
这样，我们的父项目和所有子项目就已经创建完了。如下图：



右键web子项目(其他子项目操作相同)，build path->configure build path,打开Java Build Path，点击Libraies页，



如图，删除JRE System Lirary[J2SE-1.5]，重新选择你自己的jdk；



**三、添加Springmvc+MyBatis依赖的jar包**

**在父类的pom.xml中添加SpringMvc+MyBatis所有依赖包**

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif pom.xml



**四，自动生成代码**

**1.创建数据库和表**

[复制代码](javascript:void(0);)

/\*

Navicat MySQL Data Transfer

Source Server : localhost

Source Server Version : 50521

Source Host : localhost:3306

Source Database : test

Target Server Type : MYSQL

Target Server Version : 50521

File Encoding : 65001

Date: 2017-03-28 11:11:09

\*/

SET FOREIGN\_KEY\_CHECKS=**0**;

-- ----------------------------

-- Table structure for `user\_t`

-- ----------------------------

DROP TABLE IF EXISTS `user\_t`;

CREATE TABLE `user\_t` (

`id` int(**10**) unsigned NOT NULL AUTO\_INCREMENT,

`user\_name` varchar(**255**) DEFAULT NULL,

`password` varchar(**255**) DEFAULT NULL,

`age` int(**11**) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=**2** DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of user\_t

-- ----------------------------

INSERT INTO `user\_t` VALUES ('1', '张三', '123', '22');

[复制代码](javascript:void(0);)

**2.利用mybatis-generator-core-1.3.5.jar自动创建代码**

1.在本地创建一个文件夹（我在D盘创建了一个generator文件夹）；

2.在generator文件夹下创建一个src文件夹，

3.添加2个jar包mybatis-generator-core-1.3.5.jar和mysql-connector-java-5.1.30.jar

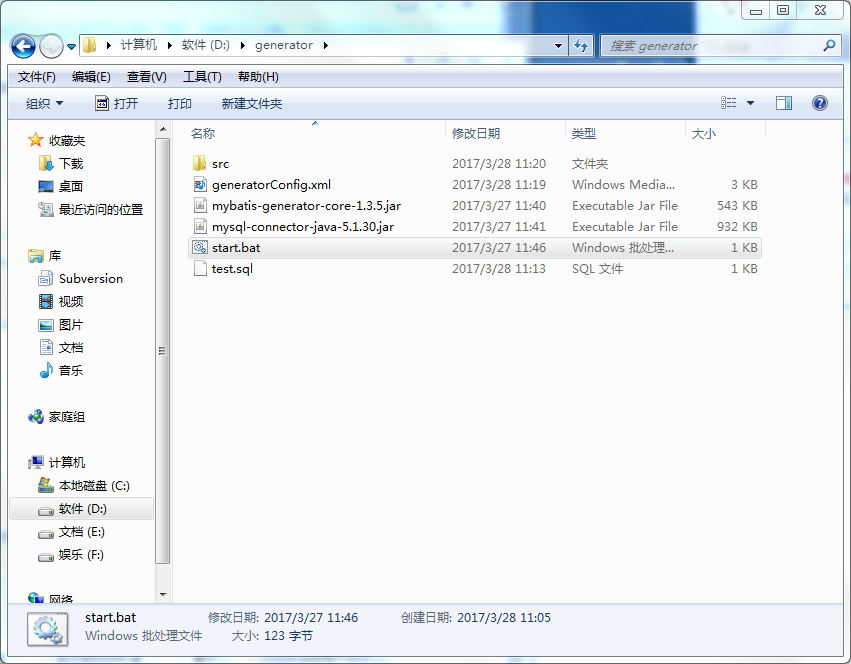
4.在创建一个generatorConfig.xml

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif generatorConfig.xml

5创建一个start.bat

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif start.bat

整体目录结构如下：



然后运行start.bat（双击），则自动生成了dao,mapping,model代码

**五、添加子项目之间的依赖**

1.给web子项目添加service子项目的依赖

打开web项目的pom.xml

给<dependencies>下添加

<dependency>  
<groupId>ssm</groupId>  
<artifactId>service</artifactId>  
<version>1</version>  
</dependency>

完整web/pom.xml如下：

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif web/pom.xml

2.给service子项目添加dao子项目的依赖

打开service项目的pom.xml

给<dependencies>下添加

<dependency>  
<groupId>ssm</groupId>  
<artifactId>dao</artifactId>  
<version>1</version>  
</dependency>

完整service/pom.xml如下：

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif service/pom.xml

3.给dao子项目添加model子项目的依赖

打开dao项目的pom.xml

给<dependencies>下添加

<dependency>  
<groupId>ssm</groupId>  
<artifactId>model</artifactId>  
<version>1</version>  
</dependency>

完整dao/pom.xml如下：

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

1 <?xml version="1.0"?>

2 <project

3 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"

4 xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

5 <modelVersion>4.0.0</modelVersion>

6 <parent>

7 <groupId>ssm</groupId>

8 <artifactId>parent</artifactId>

9 <version>1</version>

10 </parent>

11 <artifactId>dao</artifactId>

12 <name>dao</name>

13 <url>http://maven.apache.org</url>

14 <properties>

15 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

16 </properties>

17 <dependencies>

18 <dependency>

19 <groupId>junit</groupId>

20 <artifactId>junit</artifactId>

21 <version>3.8.1</version>

22 <scope>test</scope>

23 </dependency>

24 <dependency>

25 <groupId>ssm</groupId>

26 <artifactId>model</artifactId>

27 <version>1</version>

28 </dependency>

29 </dependencies>

30 </project>

[复制代码](javascript:void(0);)

整个项目依赖关系添加完毕：web依赖service，service依赖dao，dao依赖model，parent是父项目

**六、添加代码**

1.model子项目创建com.gwx.model包，将上面第四节自动生成model目录下的代码复制到com.gwx.model包下；

2.dao子项目src/main/java下创建com.gwx.dao和在src/main/resources下创建mapping包，将上面第四节自动生成dao目录下的的代码复制到

com.gwx.dao包下，将mapping目录下自动生成的代码复制到mapping包下；

3.service子项目创建com.gwx.service包和com.gwx.service.impl包，在com.gwx.service包下创建接口IUserService

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif IUserService.java

在com.gwx.service.impl包下创建UserService

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif UserService

**七、Spring与MyBatis的整合**

1.在子项目web项目中的resources目录下创建jdbc.properties

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

1 driver=com.mysql.jdbc.Driver

2 url=jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=UTF-8

3 username=root

4 password=root

5 #定义初始连接数

6 initialSize=0

7 #定义最大连接数

8 maxActive=20

9 #定义最大空闲

10 maxIdle=20

11 #定义最小空闲

12 minIdle=1

13 #定义最长等待时间

14 maxWait=60000

[复制代码](javascript:void(0);)

 2.Log4j的配置(可有可无)

在你需要查看日志的子项目的resources目录下创建log4j.properties

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif log4j.properties

3.在子项目dao层的resources目录下创建**spring-mybatis.xml**

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

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

2 <beans xmlns="http://www.springframework.org/schema/beans"

3 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p"

4 xmlns:context="http://www.springframework.org/schema/context"

5 xmlns:mvc="http://www.springframework.org/schema/mvc"

6 xsi:schemaLocation="http://www.springframework.org/schema/beans

7 http://www.springframework.org/schema/beans/spring-beans-4.0.xsd

8 http://www.springframework.org/schema/context

9 http://www.springframework.org/schema/context/spring-context-4.0.xsd

10 http://www.springframework.org/schema/mvc

11 http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">

12 <!-- 自动扫描 -->

13 <context:component-scan base-package="com.gwx.dao" />

14 <context:component-scan base-package="com.gwx.service" />

15 <context:component-scan base-package="com.gwx.service.impl" />

16 <context:component-scan base-package="com.gwx.controller" />

17 <!-- 引入配置文件 -->

18 <bean id="propertyConfigurer"

19 class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

20 <property name="location" value="classpath:jdbc.properties" />

21 </bean>

22

23 <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"

24 destroy-method="close">

25 <property name="driverClassName" value="${driver}" />

26 <property name="url" value="${url}" />

27 <property name="username" value="${username}" />

28 <property name="password" value="${password}" />

29 <!-- 初始化连接大小 -->

30 <property name="initialSize" value="${initialSize}"></property>

31 <!-- 连接池最大数量 -->

32 <property name="maxActive" value="${maxActive}"></property>

33 <!-- 连接池最大空闲 -->

34 <property name="maxIdle" value="${maxIdle}"></property>

35 <!-- 连接池最小空闲 -->

36 <property name="minIdle" value="${minIdle}"></property>

37 <!-- 获取连接最大等待时间 -->

38 <property name="maxWait" value="${maxWait}"></property>

39 </bean>

40 <!-- spring和MyBatis完美整合，不需要mybatis的配置映射文件 -->

41 <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

42 <property name="dataSource" ref="dataSource" />

43 <!-- 自动扫描mapping.xml文件 -->

44 <property name="mapperLocations" value="classpath:mapping/\*.xml"></property>

45 </bean>

46 <!-- DAO接口所在包名，Spring会自动查找其下的类 -->

47 <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

48 <property name="basePackage" value="com.gwx.dao" />

49 <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"></property>

50 </bean>

51 <!-- (事务管理)transaction manager, use JtaTransactionManager for global tx -->

52 <bean id="transactionManager"

53 class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

54 <property name="dataSource" ref="dataSource" />

55 </bean>

56 </beans>

[复制代码](javascript:void(0);)

4.Junit测试

 先将parent/pom.xml,model/pom.xml,dao/pom.xml,service/pom.xml中的

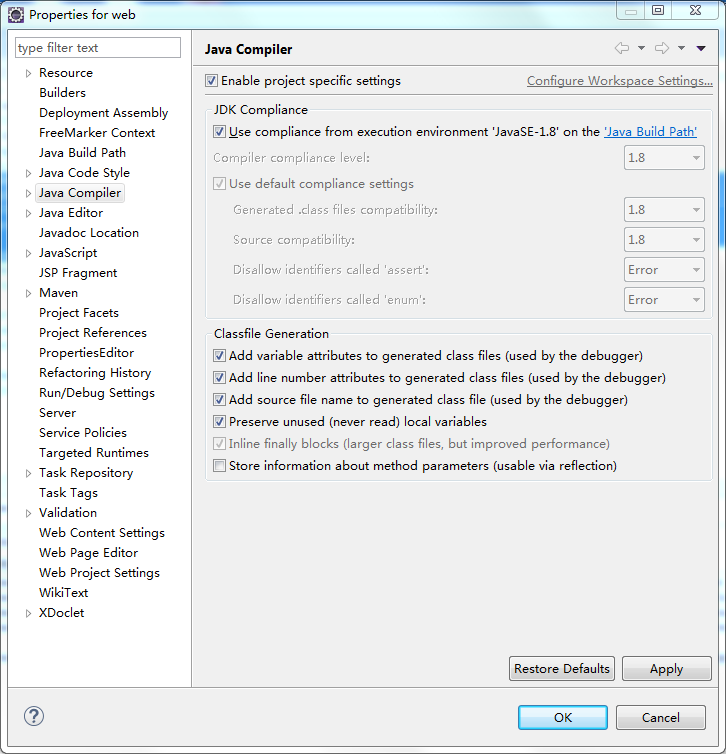
<dependency>  
<groupId>junit</groupId>  
<artifactId>junit</artifactId>  
<version>4.11</version>  
<!-- <scope>test</scope> -->  
</dependency>

删除；将web/pom.xml

中的修改为

<dependency>  
<groupId>junit</groupId>  
<artifactId>junit</artifactId>  
<version>4.11</version>  
<!-- <scope>test</scope> -->  
</dependency>

再右键web项目->properties->Java Compiler->修改为和你当前jdk版本一致；(我本地的jdk版本为1.8)如图



在web项目中的src/main/java包下创建com/gwx/test包，创建TestMyBatis.java

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

1 package com.gwx.test;

2 import javax.annotation.Resource;

3

4 import org.junit.Test;

5 import org.junit.runner.RunWith;

6 import org.springframework.test.context.ContextConfiguration;

7 import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

8

9 import com.gwx.model.User;

10 import com.gwx.service.IUserService;

11

12 @RunWith(SpringJUnit4ClassRunner.class) // 表示继承了SpringJUnit4ClassRunner类

13 @ContextConfiguration(locations = { "classpath:spring-mybatis.xml" })

14

15 public class TestMyBatis {

16 @Resource

17 private IUserService userService;

18

19 @Test

20 public void test() {

21 User user = userService.getUserById(1);

22 System.out.println(user.getUserName());

23 }

24 }

[复制代码](javascript:void(0);)

运行。控制台出现'张三'则表示整合成功！

**八、SpringMVC整合**

在web子项目的resources下创建spring-mvc.xml

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

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

2 <beans xmlns="http://www.springframework.org/schema/beans"

3 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p"

4 xmlns:context="http://www.springframework.org/schema/context"

5 xmlns:mvc="http://www.springframework.org/schema/mvc"

6 xsi:schemaLocation="http://www.springframework.org/schema/beans

7 http://www.springframework.org/schema/beans/spring-beans-3.1.xsd

8 http://www.springframework.org/schema/context

9 http://www.springframework.org/schema/context/spring-context-3.1.xsd

10 http://www.springframework.org/schema/mvc

11 http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">

12

13 <!--避免IE执行AJAX时，返回JSON出现下载文件 -->

14 <bean id="stringHttpMessageConverter"

15 class="org.springframework.http.converter.StringHttpMessageConverter">

16 <property name="supportedMediaTypes">

17 <list>

18 <value>text/plain;charset=UTF-8</value>

19 <value>application/json;charset=UTF-8</value>

20 </list>

21 </property>

22 </bean>

23 <!-- 避免IE执行AJAX时,返回JSON出现下载文件 -->

24 <!-- Jackson 是Spring 自带的功能 -->

25 <bean id="mappingJackson2HttpMessageConverter"

26 class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter">

27 <property name="supportedMediaTypes">

28 <list>

29 <value>application/json;charset=UTF-8</value>

30 <value>application/xml;charset=UTF-8</value>

31 <value>text/html;charset=UTF-8</value>

32 <value>text/plain;charset=UTF-8</value>

33 <value>text/xml;charset=UTF-8</value>

34 <value>text/javascript;charset=UTF-8</value>

35 </list>

36 </property>

37 </bean>

38

39 <!-- 启动SpringMVC的注解功能，完成请求和注解POJO的映射 -->

40 <bean

41 class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter">

42 <property name="messageConverters">

43 <list>

44 <!-- JSON转换器 -->

45 <ref bean="mappingJackson2HttpMessageConverter" />

46 </list>

47 </property>

48 </bean>

49

50 <!-- 对静态资源的访问 -->

51 <!-- <mvc:resources mapping="/js/\*\*" location="/js/" /><mvc:resources mapping="/css/\*\*"

52 location="/css/" /> -->

53

54 <!-- 自动扫描该包，使SpringMVC认为包下用了@controller注解的类是控制器 -->

55 <context:component-scan base-package="com.gwx.controller" />

56

57 <!-- 添加注解驱动 -->

58 <mvc:annotation-driven enable-matrix-variables="true" >

59 <mvc:message-converters>

60 <ref bean="stringHttpMessageConverter" />

61 <ref bean="mappingJackson2HttpMessageConverter" />

62 </mvc:message-converters>

63 </mvc:annotation-driven>

64 <!-- 允许对静态资源文件的访问 -->

65

66 <mvc:default-servlet-handler />

67

68 <!-- 定义跳转的文件的前后缀 ，视图模式配置 -->

69 <bean

70 class="org.springframework.web.servlet.view.InternalResourceViewResolver">

71 <!-- 这里的配置我的理解是自动给后面action的方法return的字符串加上前缀和后缀，变成一个 可用的url地址 -->

72 <property name="prefix" value="/WEB-INF/jsp/" />

73 <property name="suffix" value=".jsp" />

74 </bean>

75

76 <!-- 配置文件上传，如果没有使用文件上传可以不用配置，当然如果不配，那么配置文件中也不必引入上传组件包 -->

77 <bean id="multipartResolver"

78 class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

79 <!-- 默认编码 -->

80 <property name="defaultEncoding" value="utf-8" />

81 <!-- 文件大小最大值 -->

82 <property name="maxUploadSize" value="10485760000" />

83 <!-- 内存中的最大值 -->

84 <property name="maxInMemorySize" value="40960" />

85 </bean>

86 </beans>

[复制代码](javascript:void(0);)

再配置web.xml

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif web.xml

在web子项目的 Java Resources下创建package com.gwx.controller,在com.gwx.controller下面创建UserController.java

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

1 package com.gwx.controller;

2

3 import javax.annotation.Resource;

4 import javax.servlet.http.HttpServletRequest;

5

6 import org.springframework.stereotype.Controller;

7 import org.springframework.ui.Model;

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

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

10

11 import com.gwx.model.User;

12 import com.gwx.service.IUserService;

13 import com.gwx.util.JacksonUtil;

14

15 @Controller

16 @RequestMapping("/user")

17 public class UserController {

18

19 @Resource

20 private IUserService userService;

21

22 @RequestMapping("/showUser")

23 public String toIndex(HttpServletRequest request, Model model) {

24 User user = this.userService.getUserById(1);

25 model.addAttribute("user", user);

26 return "User";

27 }

28 @RequestMapping("/showJson")

29 @ResponseBody

30 public String showJson(int id) {

31 User user = this.userService.getUserById(id);

32 String jSon = JacksonUtil.toJSon(user);

33 return jSon;

34 }

35 }

[复制代码](javascript:void(0);)

在model子项目下创建package  com.gwx.util,在com.gwx.util下面创建JacksonUtil.java和httpModel.java

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

1 package com.gwx.util;

2

3 import java.io.Serializable;

4

5 public class HttpModel implements Serializable{

6

7 /\*\*

8 \*

9 \*/

10 private static final long serialVersionUID = -7484615771443870211L;

11

12 private int code = 0;

13 private Object object;

14 public int getCode() {

15 return code;

16 }

17 public void setCode(int code) {

18 this.code = code;

19 }

20 public Object getObject() {

21 return object;

22 }

23 public void setObject(Object object) {

24 this.object = object;

25 }

26

27 }

[复制代码](javascript:void(0);)

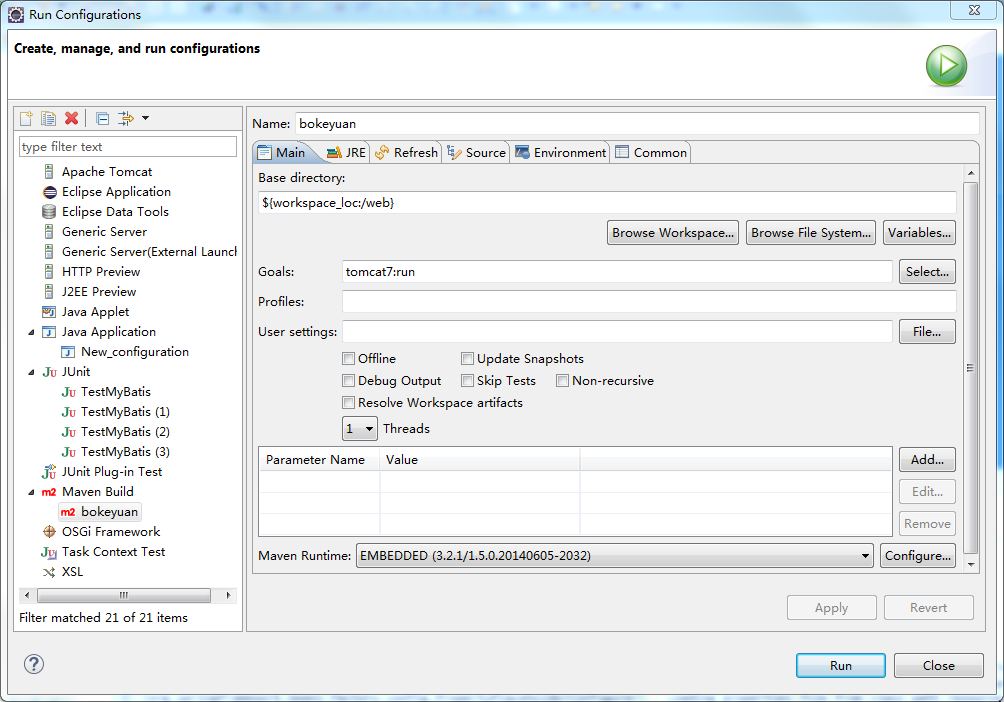
http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif JacksonUtil.java

在web项目的src/main/webapp/WEB-INF下创建jsp文件夹，然后创建User.jsp

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif User.jsp

此时，springmvc和json都已经整合完成！

开始运行web项目—>右键web->Run as->Run configurations->



左侧选择Maven Build

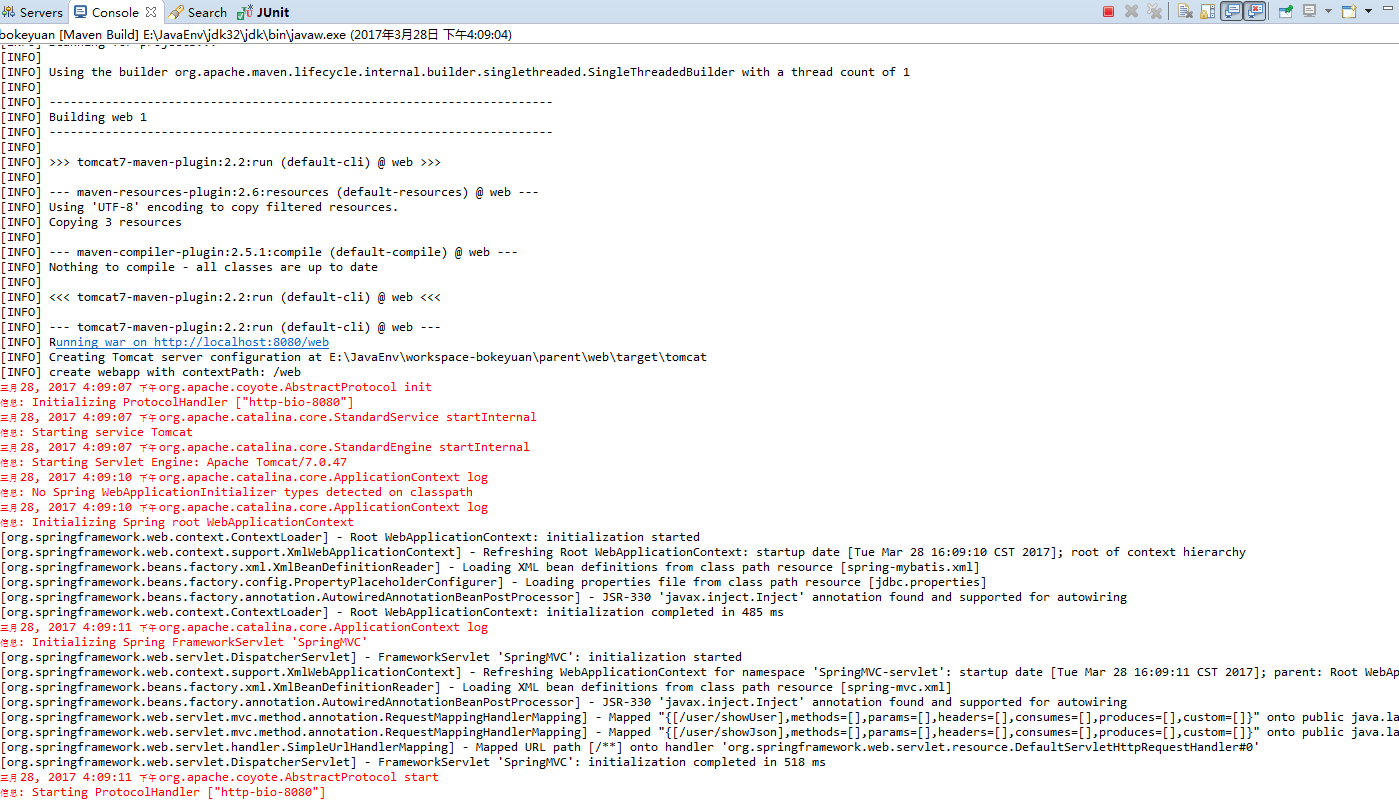
右侧在Name出填写你的name

在Main页的Base directory出选择web项目，在Goals出输入 tomcat7:run

在JRE页选择你自己的jdk

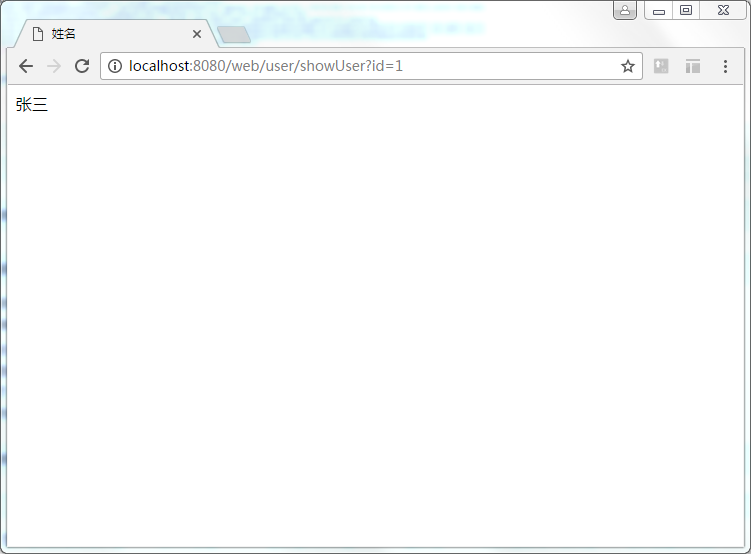
然后切换到Main页，点击Run。

如果没有报错或者异常，应该就没问题了！

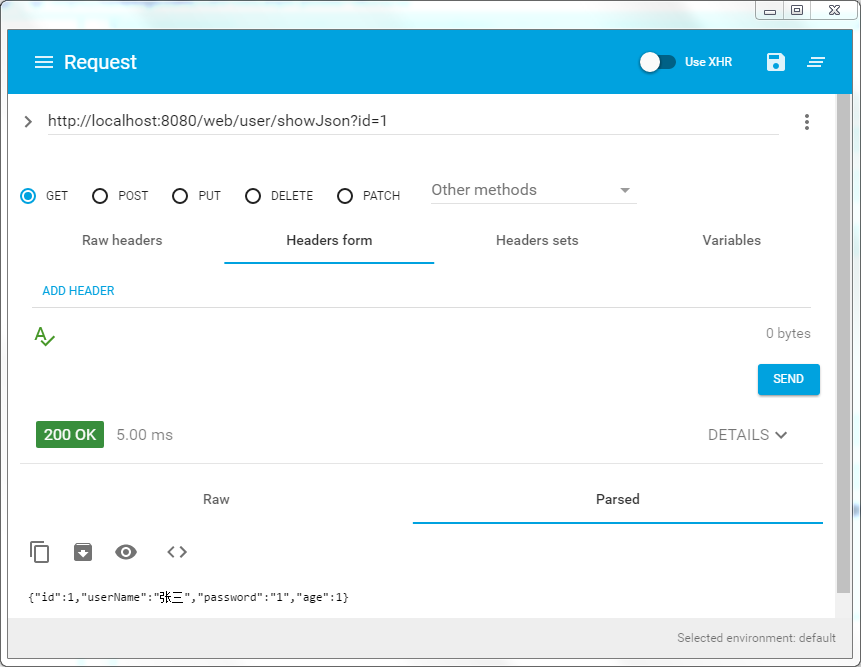


然后springmvc测试

在浏览器上输入网址http://localhost:8080/web/user/showUser?id=1就会返回到User.jsp页面。如图：



json测试：输入网址http://localhost:8080/web/user/showJson?id=1返回结果：



OK！

**九、总结**

以上就是最近整合的springmvc+mybatis+json的Maven管理一整套项目！是我亲自一步步搭建成功！

如果大家发现有什么问题，欢迎指正批评！

如果想要转载，请加载转载地址【http://www.cnblogs.com/gwx238044/articles/6633252.html】!