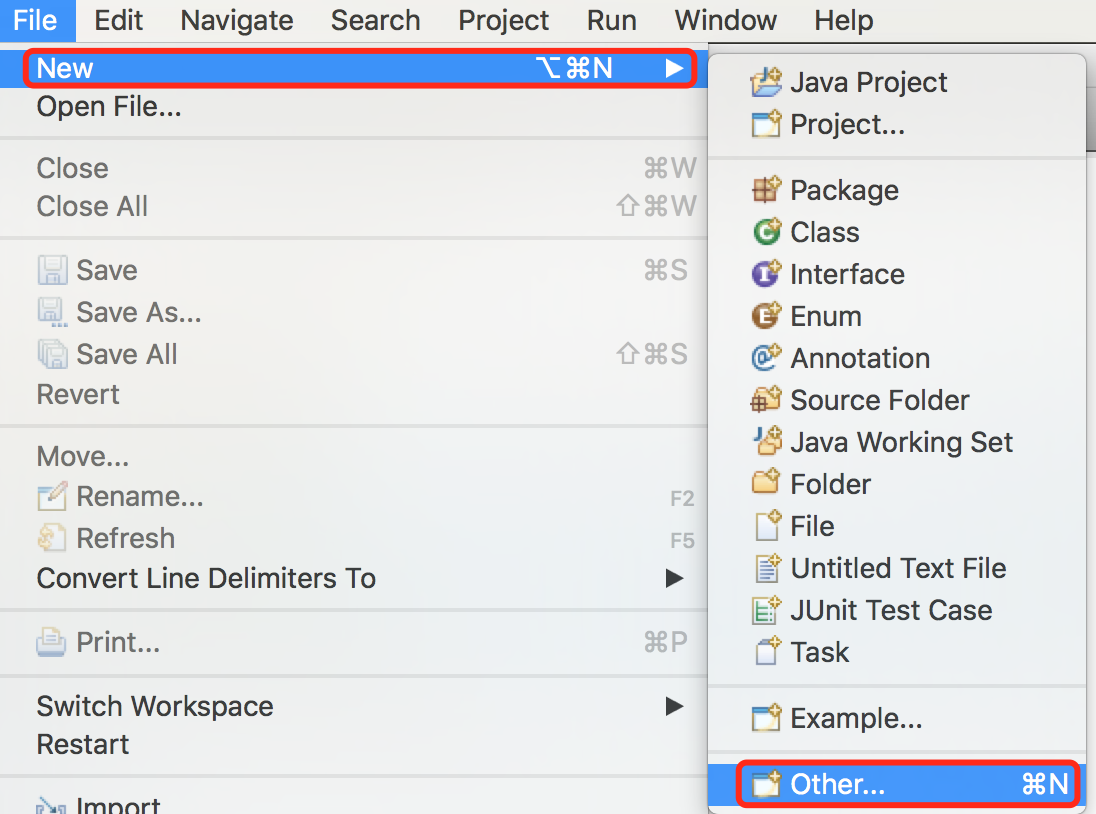
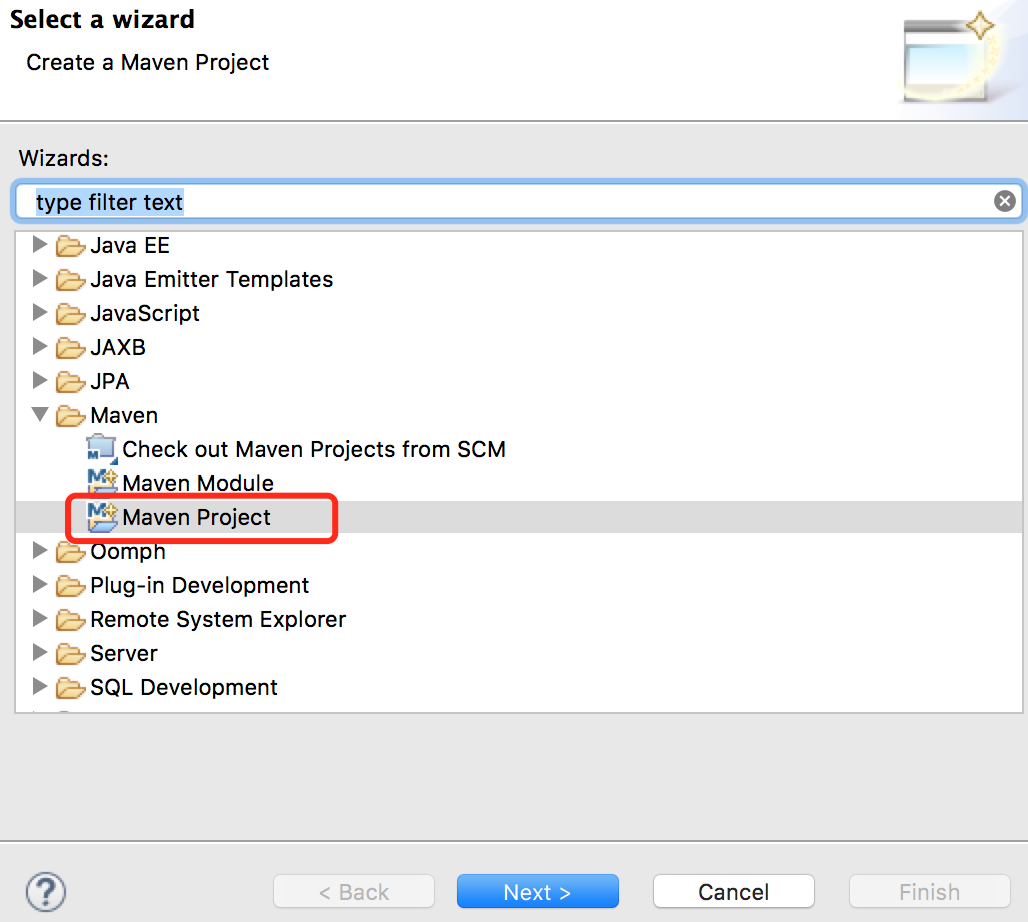
SSMM(Spring+Springmvc+Mybatis+Maven)整合

# 1.新建maven项目

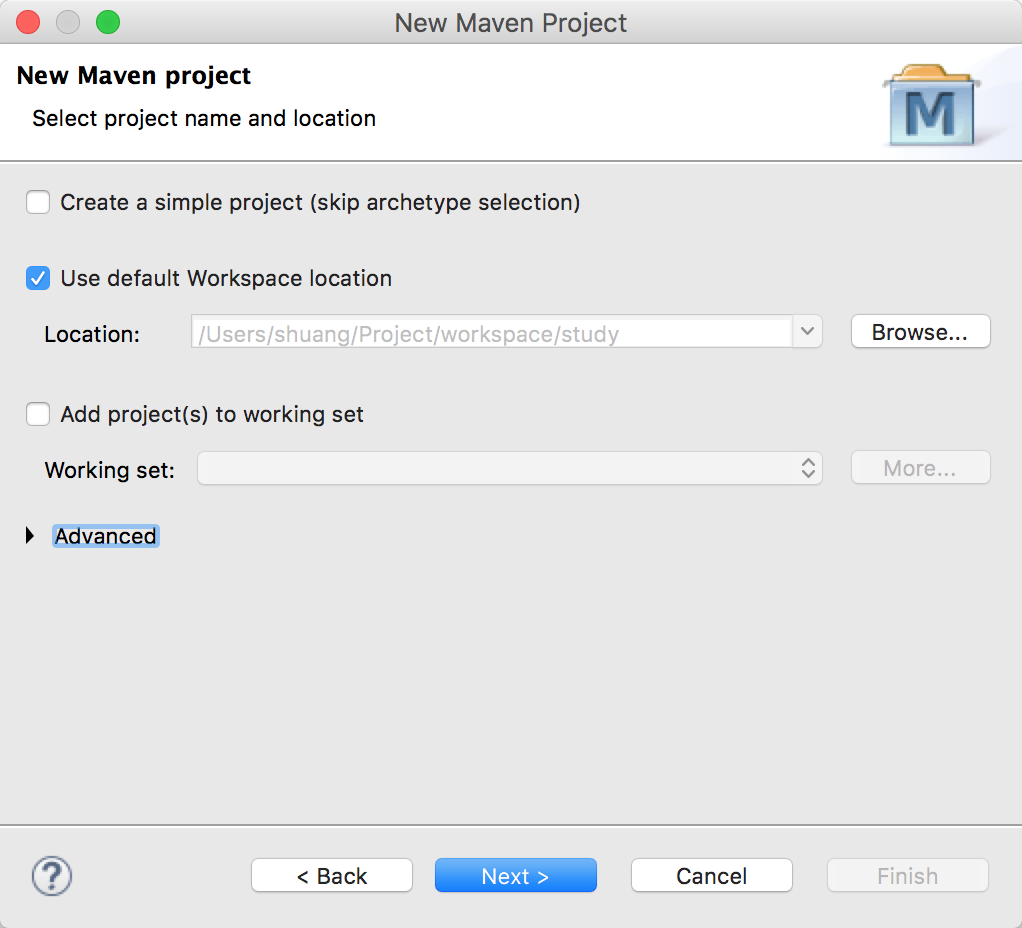
使用Eclipse工具File---New---Other,如下图:



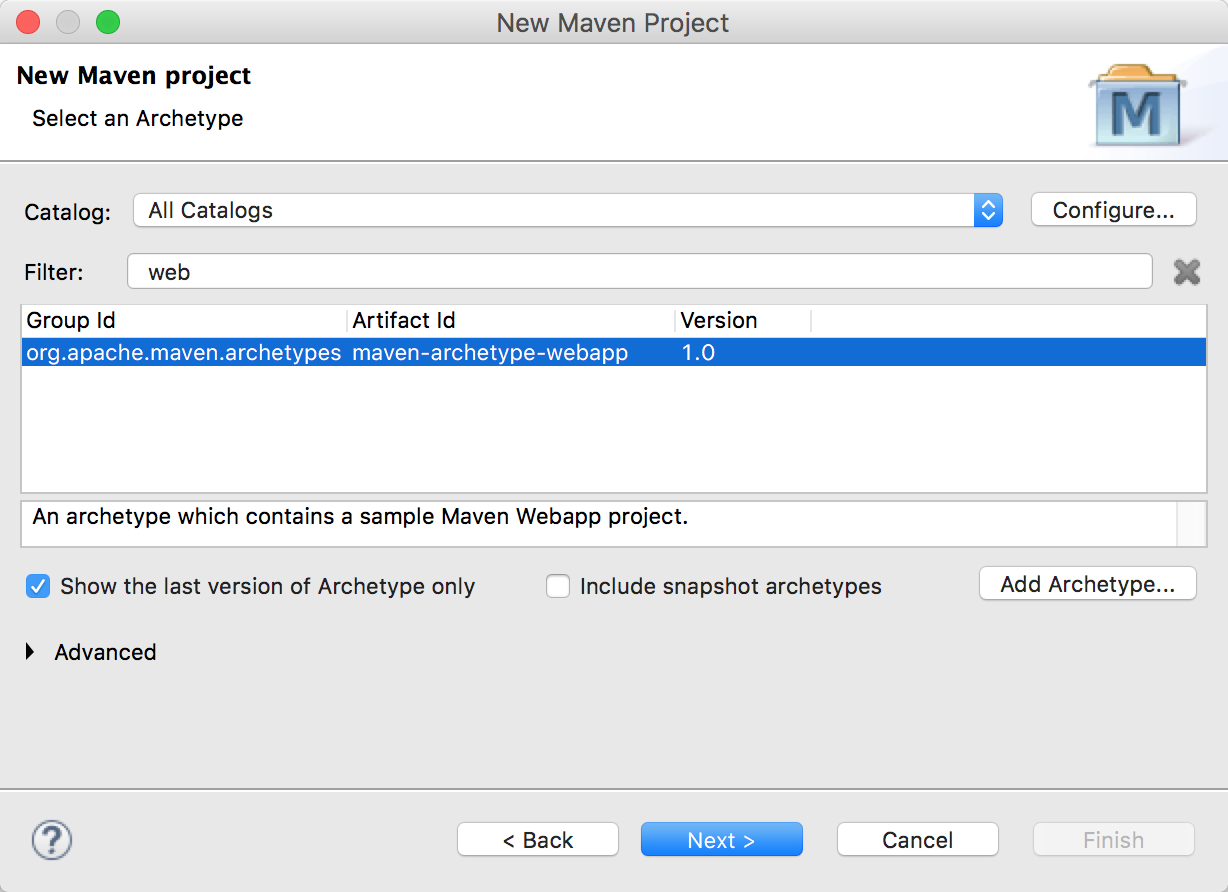
点击Other后如下图选择后点击下一步,



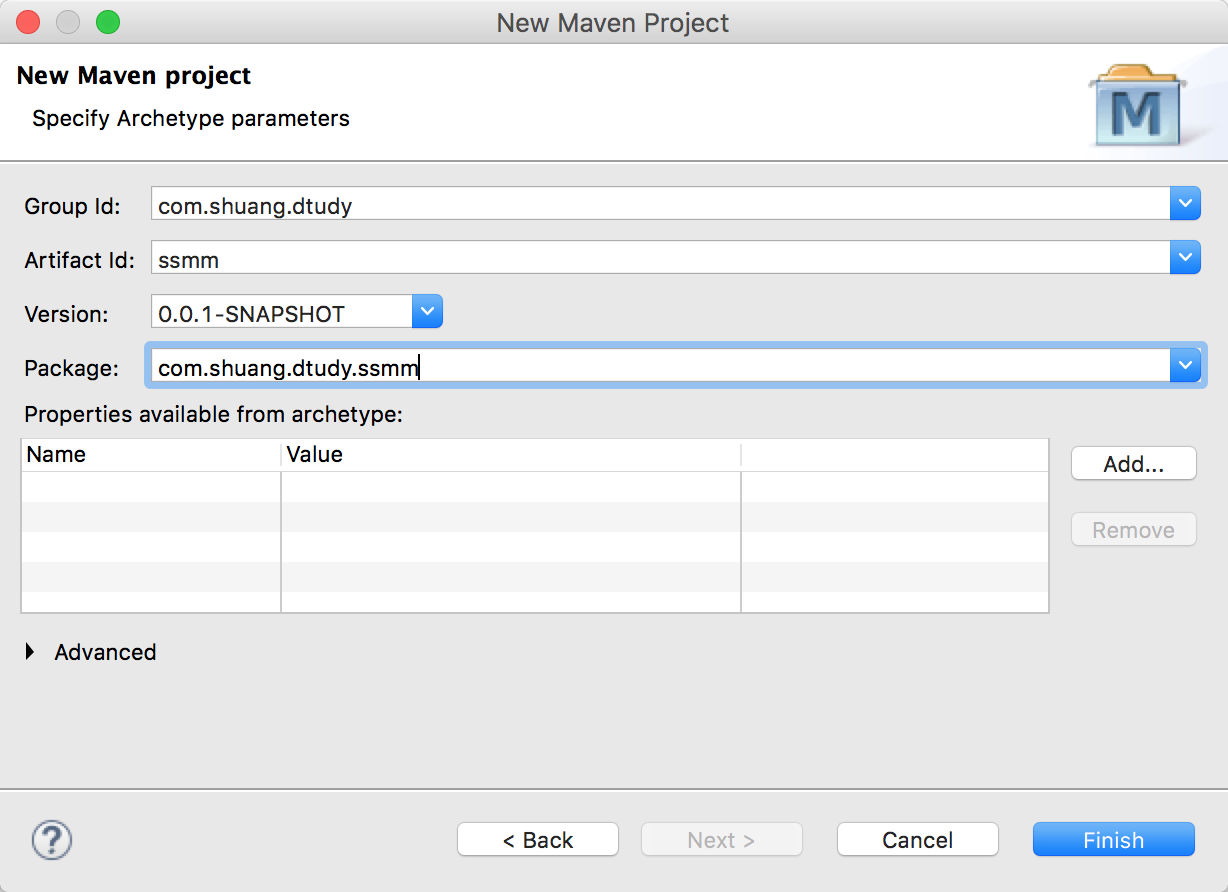
下面选择自己默认的工作空间就是了



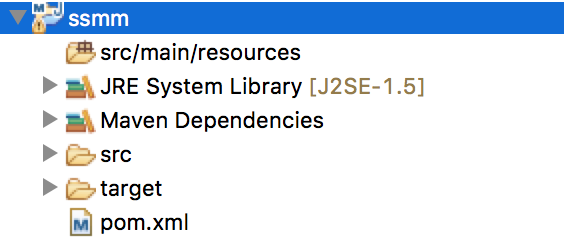
地filter中输入web后出显发下界面,选中maven-archetype-webapp如下图:



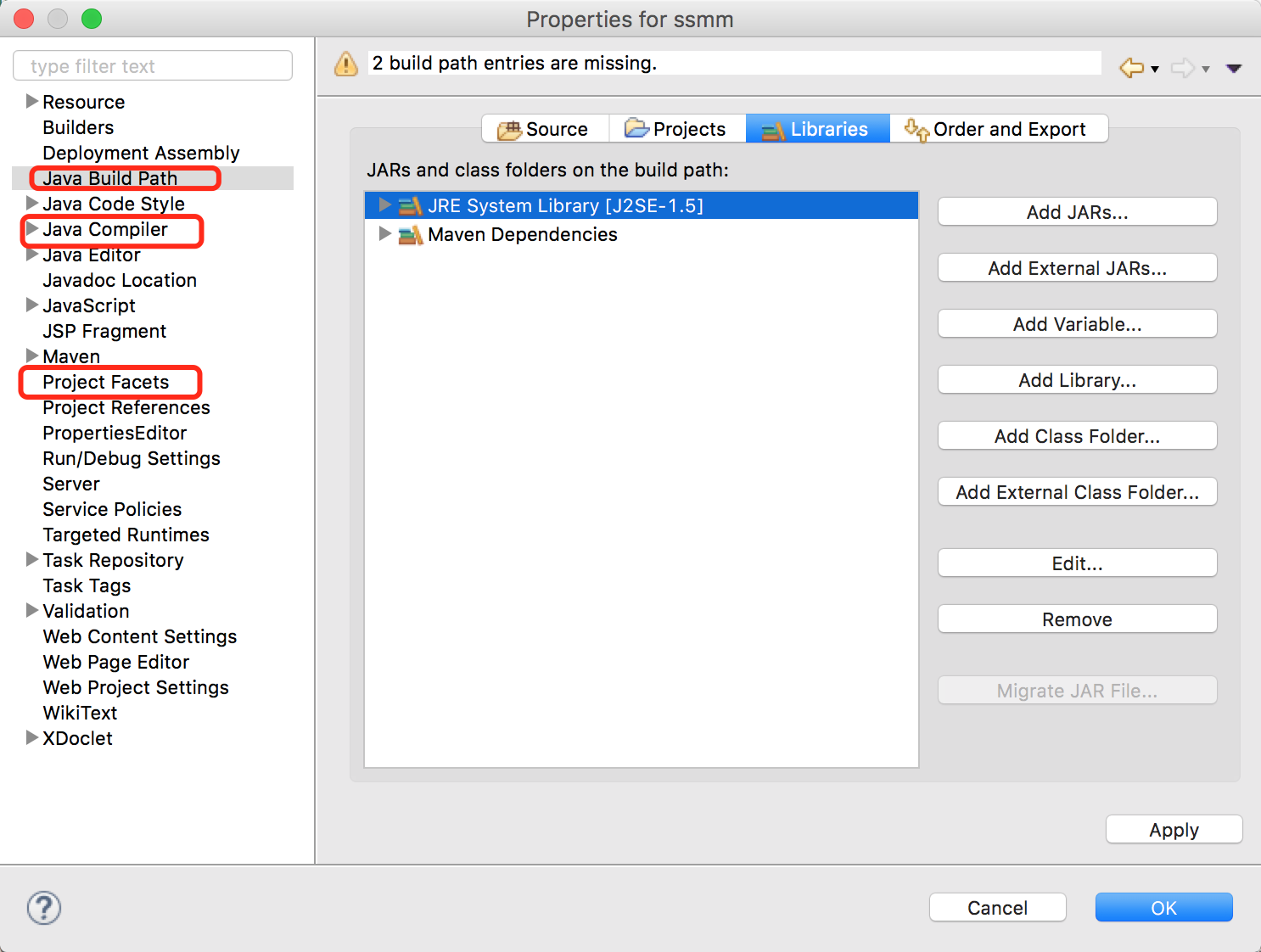
在Artifact Id与Package上输入内容,如下内容可以前不理解,后续学习Maven就明白了



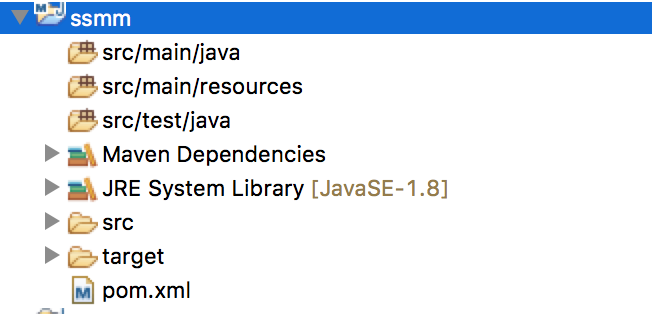
最后创建出来的项目如下图:



修改下图中框选的地方(不明白各个选项可以度娘一下),如下图:



修改后如下图:



# 2.导入相应的JAR包

在项目的pom.xml文件中添加如下配置,如下:

|  |
| --- |
| <properties>  <!-- spring版本号 -->  <spring.version>4.0.2.RELEASE</spring.version>  <!-- mybatis版本号 -->  <mybatis.version>3.2.6</mybatis.version>  <!-- log4j日志文件管理包版本 -->  <slf4j.version>1.7.7</slf4j.version>  <log4j.version>1.2.17</log4j.version>  <!-- 文件拷贝时的编码 -->  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <!-- 编译时的编码 -->  <maven.compiler.encoding>UTF-8</maven.compiler.encoding>  </properties>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.11</version>  <!-- 表示开发的时候引入，发布的时候不会加载此包 -->  <scope>test</scope>  </dependency>  <!-- spring核心包 -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-core</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-web</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-oxm</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-tx</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aop</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-test</artifactId>  <version>${spring.version}</version>  </dependency>  <!-- mybatis核心包 -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>${mybatis.version}</version>  </dependency>  <!-- mybatis/spring包 -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  <version>1.2.2</version>  </dependency>  <!-- 导入java ee jar 包 -->  <dependency>  <groupId>javax</groupId>  <artifactId>javaee-api</artifactId>  <version>7.0</version>  </dependency>  <!-- 导入Mysql数据库链接jar包 -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.30</version>  </dependency>  <!-- 导入dbcp的jar包，用来在applicationContext.xml中配置数据库 -->  <dependency>  <groupId>commons-dbcp</groupId>  <artifactId>commons-dbcp</artifactId>  <version>1.2.2</version>  </dependency>  <!-- JSTL标签类 -->  <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <!-- 日志文件管理包 -->  <!-- log start -->  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>${log4j.version}</version>  </dependency>  <!-- 格式化对象，方便输出日志 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  <version>1.1.41</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>${slf4j.version}</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>${slf4j.version}</version>  </dependency>  <!-- log end -->  <!-- 映入JSON -->  <dependency>  <groupId>org.codehaus.jackson</groupId>  <artifactId>jackson-mapper-asl</artifactId>  <version>1.9.13</version>  </dependency>  <!-- 上传组件包 -->  <dependency>  <groupId>commons-fileupload</groupId>  <artifactId>commons-fileupload</artifactId>  <version>1.3.1</version>  </dependency>  <dependency>  <groupId>commons-io</groupId>  <artifactId>commons-io</artifactId>  <version>2.4</version>  </dependency>  <dependency>  <groupId>commons-codec</groupId>  <artifactId>commons-codec</artifactId>  <version>1.9</version>  </dependency>  </dependencies> |

# 3.简单jdbc.properties配置

在项目的src/main/resources目录下创建jdbc.properties文件,并在当中添加如下内容:

|  |
| --- |
| #数据库驱动  driver=com.mysql.jdbc.Driver  #连接地址  url=jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=UTF-8  #账号  username=dev  #密码  password=dev  #定义初始连接数  initialSize=0  #定义最大连接数  maxActive=20  #定义最大空闲  maxIdle=20  #定义最小空闲  minIdle=1  #定义最长等待时间  maxWait=60000 |

# 4.简单log4j.properties配置

在项目的src/main/resources目录下创建log4j.properties文件,并在当中添加如下内容:

|  |
| --- |
| log4j.rootLogger=INFO,Console,File  #定义日志输出目的地为控制台  log4j.appender.Console=org.apache.log4j.ConsoleAppender  log4j.appender.Console.Target=System.out  #可以灵活地指定日志输出格式，下面一行是指定具体的格式  log4j.appender.Console.layout = org.apache.log4j.PatternLayout  log4j.appender.Console.layout.ConversionPattern=[%c] - %m%n    #文件大小到达指定尺寸的时候产生一个新的文件  log4j.appender.File = org.apache.log4j.RollingFileAppender  #指定输出目录  log4j.appender.File.File = ${user.dir}/logs/ssm.log  #定义文件最大大小  log4j.appender.File.MaxFileSize = 10MB  #输出所以日志，如果换成DEBUG表示输出DEBUG以上级别日志  log4j.appender.File.Threshold = ALL  log4j.appender.File.layout = org.apache.log4j.PatternLayout  log4j.appender.File.layout.ConversionPattern =[%p] [%d{yyyy-MM-dd HH\:mm\:ss}][%c]%m%n |

# 5.mybatis的整合

在项目的src/main/resources目录下创建spring-mybatis文件,并添加如下内容:

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:p=*"http://www.springframework.org/schema/p"* xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.1.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.1.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd"*>  <!-- 自动扫描 -->  <context:component-scan base-package=*"com.ssmm"* />  <!-- 引入配置文件 -->  <bean id=*"propertyConfigurer"* class=*"org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"*>  <property name=*"location"* value=*"classpath:jdbc.properties"* />  </bean>  <bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"* destroy-method=*"close"*>  <property name=*"driverClassName"* value=*"${driver}"* />  <property name=*"url"* value=*"${url}"* />  <property name=*"username"* value=*"${username}"* />  <property name=*"password"* value=*"${password}"* />  <!-- 初始化连接大小 -->  <property name=*"initialSize"* value=*"${initialSize}"*></property>  <!-- 连接池最大数量 -->  <property name=*"maxActive"* value=*"${maxActive}"*></property>  <!-- 连接池最大空闲 -->  <property name=*"maxIdle"* value=*"${maxIdle}"*></property>  <!-- 连接池最小空闲 -->  <property name=*"minIdle"* value=*"${minIdle}"*></property>  <!-- 获取连接最大等待时间 -->  <property name=*"maxWait"* value=*"${maxWait}"*></property>  </bean>  <!-- spring和MyBatis完美整合，不需要mybatis的配置映射文件 -->  <bean id=*"sqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>  <property name=*"dataSource"* ref=*"dataSource"* />  <!-- 自动扫描mapping.xml文件 -->  <property name=*"mapperLocations"* value=*"classpath:com/ssmm/dao/\*.xml"*></property>  </bean>  <!-- DAO接口所在包名，Spring会自动查找其下的类 -->  <bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>  <property name=*"basePackage"* value=*"com.ssmm.dao"* />  <property name=*"sqlSessionFactoryBeanName"* value=*"sqlSessionFactory"*></property>  </bean>  <!-- (事务管理)transaction manager, use JtaTransactionManager for global tx -->  <bean id=*"transactionManager"* class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>  <property name=*"dataSource"* ref=*"dataSource"* />  </bean>  </beans> |

注: 自动扫描注解, 自动扫描mapping.xml文件与DAO接口路径配置

# 6.数据库创建表

数据库创建表,根据表生成实体对象,如下脚本:

|  |
| --- |
| CREATE TABLE `t\_user` (  `id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '用户ID',  `user\_name` varchar(40) NOT NULL COMMENT '用户姓名',  `password` varchar(255) NOT NULL COMMENT '用户密码',  `age` int(4) NOT NULL COMMENT '用户年龄',  PRIMARY KEY (`id`)  ) ENGINE=InnoDB AUTO\_INCREMENT=13 DEFAULT CHARSET=utf8 COMMENT='用户表' |

注:因使用mybatis工具生成实体对象.

# 7.service接口

用户service接口,如下:

|  |
| --- |
| **package** com.ssmm.service;  **import** com.ssmm.model.User;  **public** **interface** IUserService {  **void** addUserInfo(User user) **throws** Exception;  } |

# 8. service接口实现

用户service实现,如下:

|  |
| --- |
| package com.ssmm.service.impl;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.ssmm.dao.UserDao;  import com.ssmm.model.User;  import com.ssmm.service.IUserService;  @Service("userService")  public class UserServiceImpl implements IUserService {  @Autowired  UserDao userDao;  @Override  public void addUserInfo(User user) throws Exception {  int rec = userDao.insert(user);  System.out.println("添加用户返回状态: " + rec);  }  } |

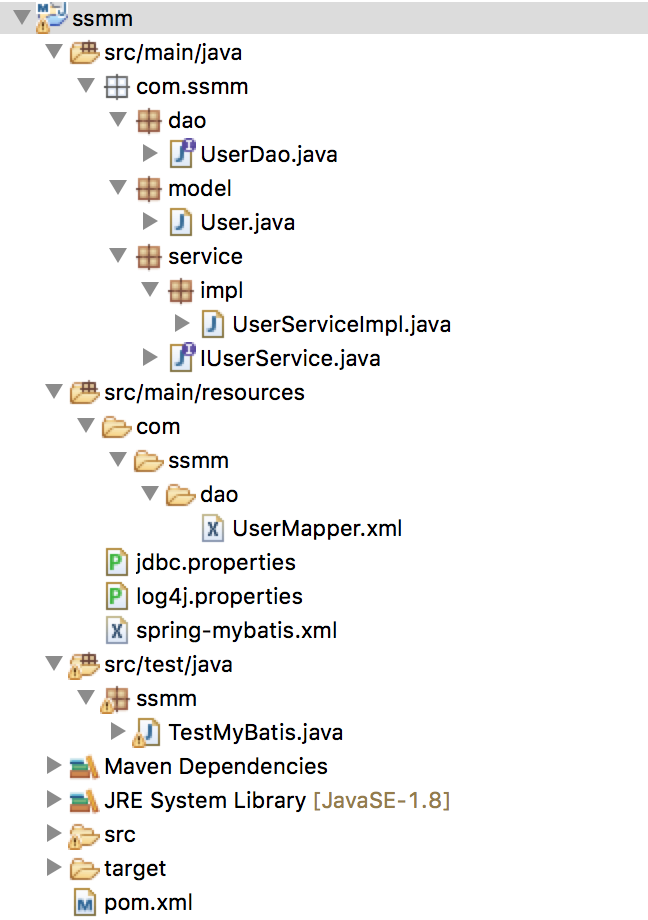
# 9.编写JUNIT单元测试

测试类实现,如下:

|  |
| --- |
| **package** ssmm;  **import** javax.annotation.Resource;  **import** org.apache.log4j.Logger;  **import** org.junit.Before;  **import** org.junit.Test;  **import** org.junit.runner.RunWith;  **import** org.springframework.context.ApplicationContext;  **import** org.springframework.context.support.ClassPathXmlApplicationContext;  **import** org.springframework.test.context.ContextConfiguration;  **import** org.springframework.test.context.junit4.SpringJUnit4ClassRunner;  **import** com.alibaba.fastjson.JSON;  **import** com.ssmm.model.User;  **import** com.ssmm.service.IUserService;  @RunWith(SpringJUnit4ClassRunner.**class**) // 表示继承了SpringJUnit4ClassRunner类  @ContextConfiguration(locations = { "classpath:spring-mybatis.xml" })  **public** **class** TestMyBatis {  **private** **static** Logger *logger* = Logger.*getLogger*(TestMyBatis.**class**);  /\*\*  \* 注释掉的部分是不使用Spring时，一般情况下的一种测试方法  \*/  // private ApplicationContext ac = null;  @Resource  **private** IUserService userService = **null**;  // @Before  // public void before() {  // ac = new ClassPathXmlApplicationContext("applicationContext.xml");  // userService = (IUserService) ac.getBean("userService");  // }  @Test  **public** **void** test1() {  **try** {  // userService.addUserInfo(new User(11, "admin", "123456", 20));  userService.addUserInfo(**new** User(12, "张三", "123456", 20));  } **catch** (Exception e) {  e.printStackTrace();  *logger*.error(e.getMessage(), e);  }  // logger.info(JSON.toJSONString(user));  }  } |

# 10. spring+mybatis的整合

spring+mybatis的整合后项目,如下图:



# 11.整合Spring MVC框架

上面已经完成了2大框架的整合，SpringMVC的配置文件单独放，然后在web.xml中配置整合。

11.1配置spring-mvc.xml文件,如下:

配置里面的注释也很详细，在此就不说了，主要是自动扫描控制器，视图模式，注解的启动这三个,在resources目录中创建spring-mvc.xml文件,并在文夹中添加发下内容:

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:p=*"http://www.springframework.org/schema/p"* xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.1.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.1.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd"*>  <!-- 自动扫描该包，使SpringMVC认为包下用了@controller注解的类是控制器 -->  <context:component-scan base-package=*"com.ssmm.controller"* />  <!--避免IE执行AJAX时，返回JSON出现下载文件 -->  <bean id=*"mappingJacksonHttpMessageConverter"* class=*"org.springframework.http.converter.json.MappingJacksonHttpMessageConverter"*>  <property name=*"supportedMediaTypes"*>  <list>  <value>text/html;charset=UTF-8</value>  </list>  </property>  </bean>  <!-- 启动SpringMVC的注解功能，完成请求和注解POJO的映射 -->  <bean class=*"org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter"*>  <property name=*"messageConverters"*>  <list>  <ref bean=*"mappingJacksonHttpMessageConverter"* /> <!-- JSON转换器 -->  </list>  </property>  </bean>  <!-- 定义跳转的文件的前后缀 ，视图模式配置 -->  <bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <!-- 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 -->  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>  <!-- 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 -->  <bean id=*"multipartResolver"* class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"*>  <!-- 默认编码 -->  <property name=*"defaultEncoding"* value=*"utf-8"* />  <!-- 文件大小最大值 -->  <property name=*"maxUploadSize"* value=*"10485760000"* />  <!-- 内存中的最大值 -->  <property name=*"maxInMemorySize"* value=*"40960"* />  </bean>  </beans> |

# 12配置web.xml文件

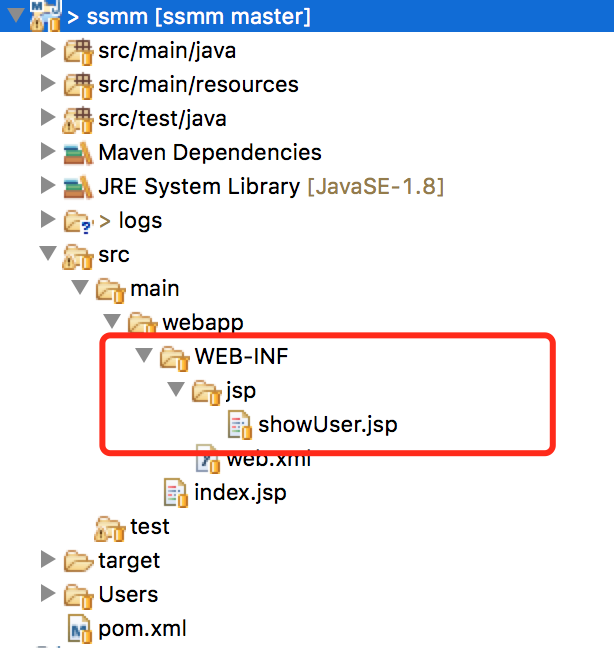
这里面对spring-mybatis.xml的引入以及配置的spring-mvc的Servlet就是为了完成SSM整合，之前2框架整合不需要在此处进行任何配置。配置一样有详细注释，不多解释了。

12.1web.xml文件文件内容,如下:

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://java.sun.com/xml/ns/javaee"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"* version=*"3.0"*>  <display-name>Archetype Created Web Application</display-name>  <!-- Spring和mybatis的配置文件 -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring-mybatis.xml</param-value>  </context-param>  <!-- 编码过滤器 -->  <filter>  <filter-name>encodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <async-supported>true</async-supported>  <init-param>  <param-name>encoding</param-name>  <param-value>UTF-8</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>encodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <!-- Spring监听器 -->  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <!-- 防止Spring内存溢出监听器 -->  <listener>  <listener-class>org.springframework.web.util.IntrospectorCleanupListener</listener-class>  </listener>  <!-- Spring MVC servlet -->  <servlet>  <servlet-name>SpringMVC</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring-mvc.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  <async-supported>true</async-supported>  </servlet>  <servlet-mapping>  <servlet-name>SpringMVC</servlet-name>  <!-- 此处可以可以配置成\*.do，对应struts的后缀习惯 -->  <url-pattern>/</url-pattern>  </servlet-mapping>  <welcome-file-list>  <welcome-file>/index.jsp</welcome-file>  </welcome-file-list>  </web-app> |

# 13.测试

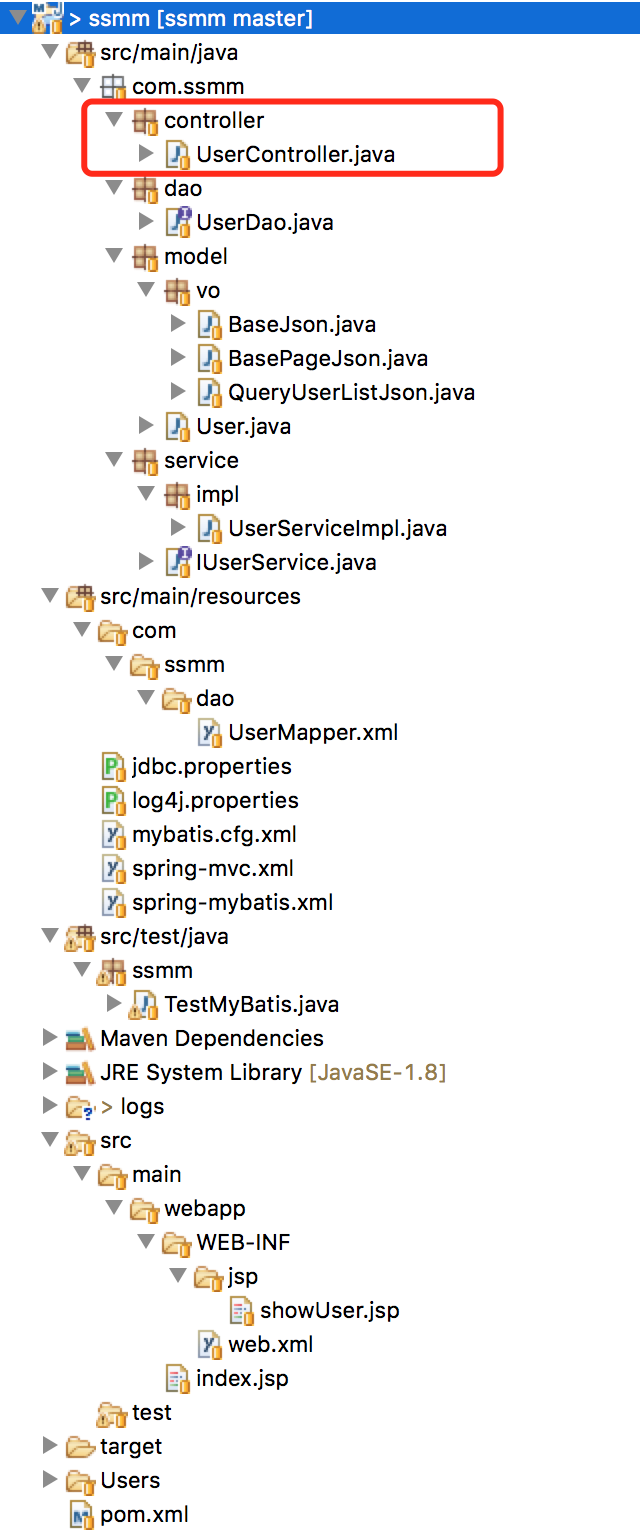
至此已经完成了SSM三大框架的整合了，接下来测试一下，如果成功了，那么恭喜你，如果失败了，继续调试吧，作为程序员就是不停的与BUG做斗争！现在我们创建一个JSP页面做一个测试,如下图为当前项目的目录与创建JSP文件路径:



13.1JSP文件中的内容,如下:

|  |
| --- |
| <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"utf-8"*%>  <%@ taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"*%>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <title>测试</title>  </head>  <body>  <c:forEach items=*"*${user}*"* varStatus=*"i"* var=*"item"*>  <h2>${item.userName}</h2>  </c:forEach>  </body>  </html> |

13.2创建Controll(控制器)类,如下图:

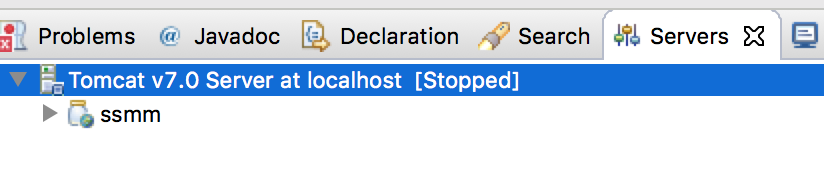


13.3UserCointroller.java控制器的内容如下:

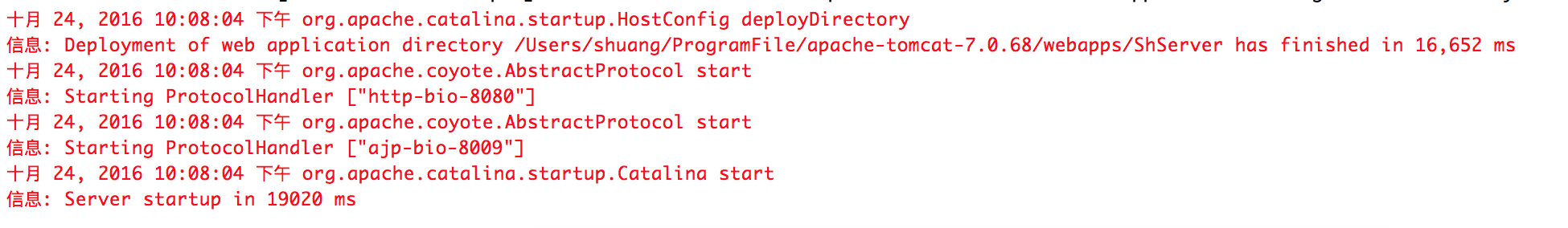
|  |
| --- |
| package com.ssmm.controller;  import java.util.HashMap;  import java.util.Map;  import javax.annotation.Resource;  import javax.servlet.http.HttpServletRequest;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  import com.ssmm.model.User;  import com.ssmm.model.vo.QueryUserListJson;  import com.ssmm.service.IUserService;  @Controller  @RequestMapping("/user")  public class UserController {  @Resource  private IUserService userService;  @RequestMapping("/showUser")  public String toIndex(HttpServletRequest request, Model model) {  try {  Map<String, Object> parameter = new HashMap<>();  parameter.put("pageNum", "1");  parameter.put("pageSize", 20);  QueryUserListJson quj = userService.queryUserList(parameter);  System.out.println("用户名:");  for (User user : quj.getList()) {  System.out.println(user.getUserName());  }  System.out.println("pageNum: " + quj.getPageNum() + ",pageSize: " + quj.getPageSize() + ",total: " + quj.getTotal());  model.addAttribute("user", quj.getList());  } catch (Exception e) {  e.printStackTrace();  }  return "showUser";  }  } |

# 14.部署项目

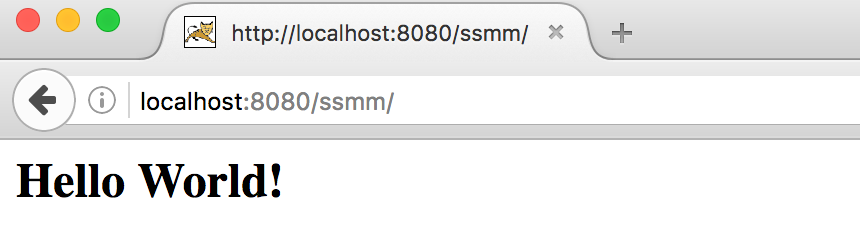
添加Tomcat服务并添加项目,如下图:



14.1启动服务,如下图:



14.2访问项目地址,如下:

15.结束

目前Spring+Springmvc+mybaits+maven的项目已经整合完成.在基础之前可以添加其它功能.也可以研究每个框架的使用.