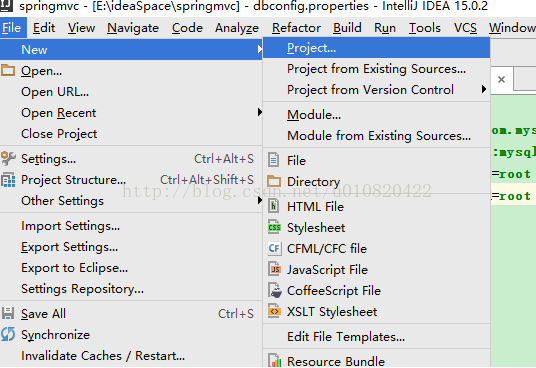
**idea创建springmvc+mavem+mybatis的web工程**

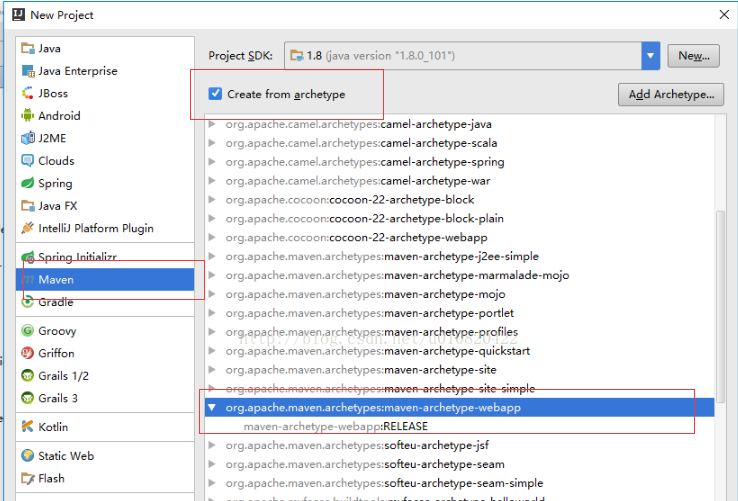
**一.使用idea创建maven工程**

（习惯把所有工程都放到同一路径，新建文件夹叫ideaSpace，之后所有新建project之前都在该文件夹下新建一个对应项目名称的文件夹，这样感觉比较整洁。）

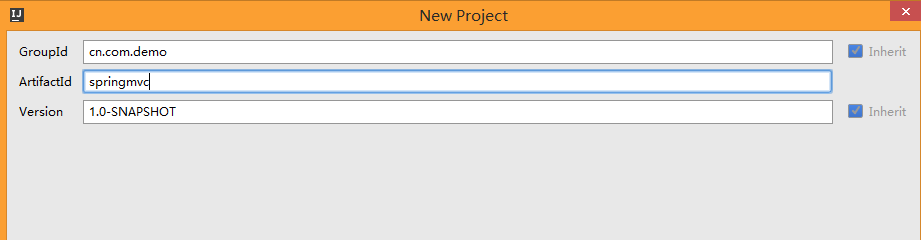
1.file-new-project



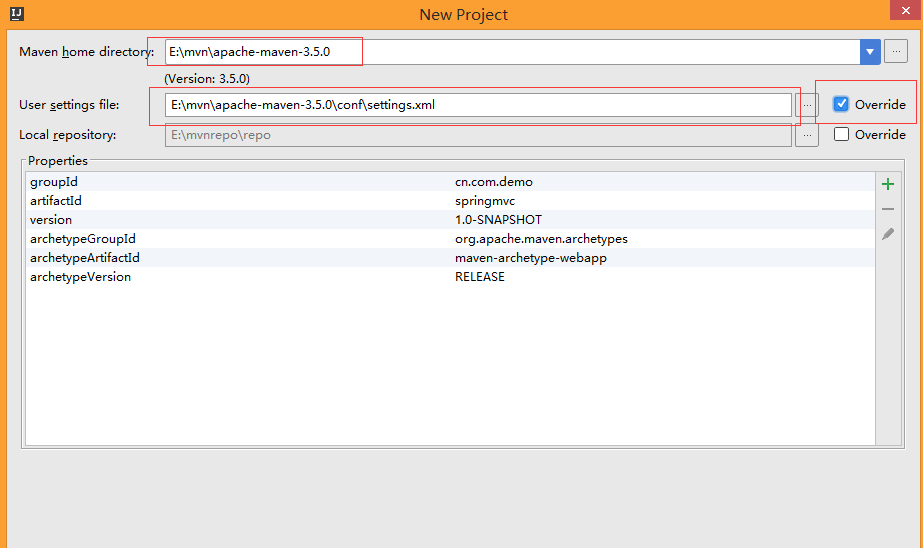
2.选择maven



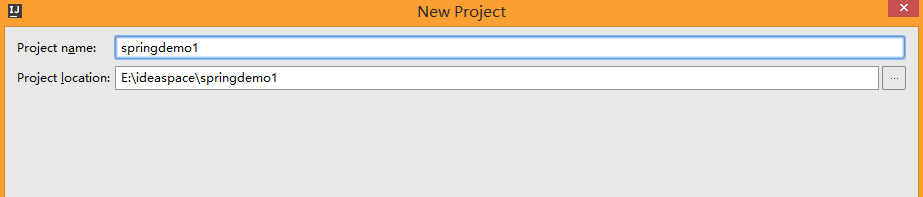
3.设置groupid



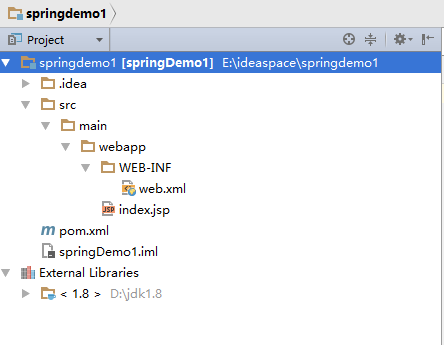
4.选择maven路径



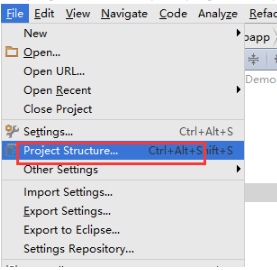
5.项目名称



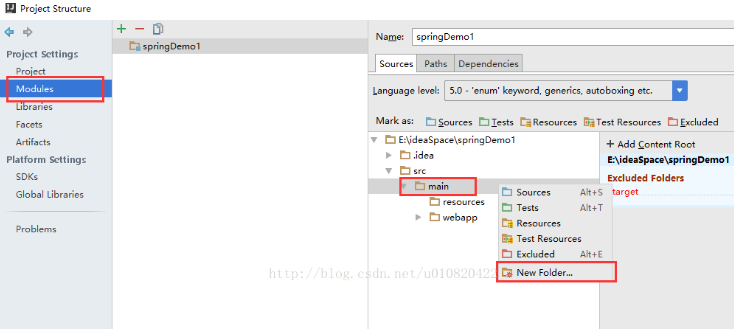
顺利生成项目：



不过很明显，项目结构与我们想象的不太相同，缺少的java目录，下面加上java目录。

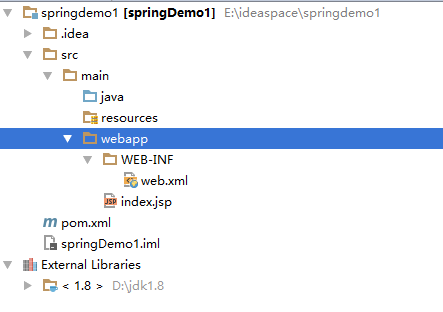


选择modules,在main文件夹上右击选择new folders,输入名称java



选择java文件夹，点击sources，将Java设置为源码文件夹；选择resources文件夹，点击resources。

至此，工程创建完毕。



**二.引入mybatis工具包，生成文件**

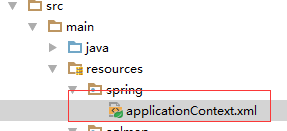
1.pom.xml中引入插件

<?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.springmvc</groupId>  
 <artifactId>helloworld</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>war</packaging>  
  
 <name>helloworld Maven Webapp</name>  
 <!-- FIXME change it to the project's website -->  
 <url>http://www.example.com</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <maven.compiler.source>1.7</maven.compiler.source>  
 <maven.compiler.target>1.7</maven.compiler.target>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.11</version>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-web</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-tx</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jdbc</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context-support</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-test</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>  
 <!-- springframe end -->  
  
 <!-- mybatis核心包 -->  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>3.3.1</version>  
 </dependency>  
 <!-- mybatis/spring包 -->  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis-spring</artifactId>  
 <version>1.2.2</version>  
 </dependency>  
 <!-- mysql驱动包 -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.29</version>  
 </dependency>  
 <!-- junit测试包 -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.11</version>  
 <scope>test</scope>  
 </dependency>  
 <!-- jstl -->  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>jstl</artifactId>  
 <version>1.2</version>  
 </dependency>  
 <dependency>  
 <groupId>javax.servlet.jsp</groupId>  
 <artifactId>jsp-api</artifactId>  
 <version>2.2.1-b03</version>  
 <scope>provided</scope>  
 </dependency>  
  
 <!-- servlet -->  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>servlet-api</artifactId>  
 <version>2.5</version>  
 <scope>provided</scope>  
 </dependency>  
 <!-- json数据 -->  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-core</artifactId>  
 <version>2.8.5</version>  
 </dependency>  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-databind</artifactId>  
 <version>2.8.5</version>  
 </dependency>  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-annotations</artifactId>  
 <version>2.8.5</version>  
 </dependency>  
 <dependency>  
 <groupId>org.codehaus.jackson</groupId>  
 <artifactId>jackson-mapper-asl</artifactId>  
 <version>1.9.4</version>  
 </dependency>  
 <dependency>  
 <groupId>net.sf.json-lib</groupId>  
 <artifactId>json-lib</artifactId>  
 <version>2.4</version>  
 <classifier>jdk15</classifier>  
 </dependency>  
 <!-- commons -->  
 <dependency>  
 <groupId>commons-lang</groupId>  
 <artifactId>commons-lang</artifactId>  
 <version>2.4</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-logging</groupId>  
 <artifactId>commons-logging</artifactId>  
 <version>1.1</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-pool</groupId>  
 <artifactId>commons-pool</artifactId>  
 <version>1.5.6</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-dbcp</groupId>  
 <artifactId>commons-dbcp</artifactId>  
 <version>1.4</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-beanutils</groupId>  
 <artifactId>commons-beanutils</artifactId>  
 <version>1.8.3</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-httpclient</groupId>  
 <artifactId>commons-httpclient</artifactId>  
 <version>3.1</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-collections</groupId>  
 <artifactId>commons-collections</artifactId>  
 <version>3.1</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-codec</groupId>  
 <artifactId>commons-codec</artifactId>  
 <version>1.9</version>  
 </dependency>  
 <dependency>  
 <groupId>javax.annotation</groupId>  
 <artifactId>jsr250-api</artifactId>  
 <version>1.0</version>  
 </dependency>  
  
 <dependency>  
 <groupId>net.sf.ezmorph</groupId>  
 <artifactId>ezmorph</artifactId>  
 <version>1.0.6</version>  
 </dependency>  
 <dependency>  
 <groupId>javax.activation</groupId>  
 <artifactId>activation</artifactId>  
 <version>1.1</version>  
 </dependency>  
 <dependency>  
 <groupId>taglibs</groupId>  
 <artifactId>standard</artifactId>  
 <version>1.1.2</version>  
 </dependency>  
 <!-- 日志文件管理包 -->  
 <!-- log start -->  
 <dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>1.2.17</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.7.5</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-log4j12</artifactId>  
 <version>1.7.21</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
  
 <finalName>helloworld</finalName>  
 <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->  
 <plugins>  
 <plugin>  
 <artifactId>maven-clean-plugin</artifactId>  
 <version>3.0.0</version>  
 </plugin>  
 <!-- see http://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin\_bindings\_for\_war\_packaging -->  
 <plugin>  
 <artifactId>maven-resources-plugin</artifactId>  
 <version>3.0.2</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.7.0</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>2.20.1</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-war-plugin</artifactId>  
 <version>3.2.0</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-install-plugin</artifactId>  
 <version>2.5.2</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-deploy-plugin</artifactId>  
 <version>2.8.2</version>  
 </plugin>  
 </plugins>  
 </pluginManagement>  
  
 <plugins>  
 <plugin>  
 <groupId>org.mybatis.generator</groupId>  
 <artifactId>mybatis-generator-maven-plugin</artifactId>  
 <version>1.3.2</version>  
 <configuration>  
 <!--允许移动生成的文件-->  
 <verbose>true</verbose>  
 <!--允许覆盖生成的文件-->  
 <overwrite>true</overwrite>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

2、jdbc.properties

driver=com.mysql.jdbc.Driver  
url=jdbc:mysql://localhost:3306/springmvc\_test?useUnicode=true&characterEncoding=utf-8  
username=root  
#下面输入自己数据库的密码  
password=6666  
#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

3.applicationContext.xml（配置spring）



<?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: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-4.3.xsd  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context-4.3.xsd  
 http://www.springframework.org/schema/mvc  
 http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd" >  
  
 <!-- ①：对com.springmvc包中的所有类进行扫描，以完成Bean创建和自动依赖注入的功能 -->  
 <context:component-scan base-package="com.springmvc"/>  
 <mvc:annotation-driven />  
 <!-- 静态资源访问 -->  
 <!--如果webapp下你新建了文件夹，想访问里面的静态资源，那么就要在这配置一下-->  
 <mvc:resources location="/images/" mapping="/images/\*\*"/>  
 <mvc:resources location="/css/" mapping="/css/\*\*"/>  
 <mvc:resources location="/styles/" mapping="/styles/\*\*"/>  
 <mvc:resources location="/js/" mapping="/js/\*\*"/>  
  
 <!-- Configures the @Controller programming model  
 <mvc:annotation-driven />-->  
 <!-- ②：启动Spring MVC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"/>  
 </list>  
 </property>  
 </bean>  
  
 <bean id="mappingJacksonHttpMessageConverter" class="org.springframework.http.converter.json.MappingJacksonHttpMessageConverter">  
 <property name="supportedMediaTypes">  
 <list>  
 <value>application/json;charset=UTF-8</value>  
 </list>  
 </property>  
 </bean>  
  
 <!-- 配置视图解析器，把控制器的逻辑视频映射为真正的视图 -->  
 <!-- /WEB-INF/jsp/start.jsp -->  
 <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
 <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>  
 <property name="prefix" value="/WEB-INF/views/" />  
 <property name="suffix" value=".jsp" />  
 </bean>  
  
 <!-- 配置dbcp数据库连接池 -->  
  
 <!-- <context:property-placeholder location="classpath:db.properties"/> -->  
 <!--数据库配置 -->  
 <bean id = "propertyConfigurer" class = "org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">  
 <property name="locations">  
 <list>  
 <value>classpath:jdbc.properties</value>  
 </list>  
 </property>  
  
 </bean>  
  
 <!-- 数据库连接池 -->  
 <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}"/>  
 <property name="initialSize" value="1"/>  
 <property name="maxActive" value="100"/>  
 <property name="maxIdle" value="5"/>  
 <property name="maxWait" value="80000"/>  
 </bean>  
  
  
 <!-- 配置事务管理器 -->  
 <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
 <property name="dataSource" ref="dataSource" />  
 </bean>  
  
  
 <!-- 拦截器 -->  
 <mvc:interceptors>  
 <!-- 国际化操作拦截器 如果采用基于（请求/Session/Cookie）则必需配置 -->  
 <bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor" />  
 </mvc:interceptors>  
 <!-- 定义无Controller的path<->view直接映射 -->  
 <mvc:view-controller path="/" view-name="redirect:/" />  
 <!-- 配置会话工厂SqlSessionFactory -->  
  
 <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  
 <!-- 数据源 -->  
 <property name="dataSource" ref="dataSource"/>  
 <property name="mapperLocations" value="classpath\*:sqlmap/\*.xml"/>  
 <property name="typeAliasesPackage" value="com.springmvc.entity" />  
 </bean>  
  
 <!-- 在spring容器中配置mapper的扫描器产生的动态代理对象在spring的容器中自动注册，bean的id就是mapper类名（首字母小写）-->  
 <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  
 <!-- 指定扫描包的路径，就是mapper接口的路径，多个包中间以 半角逗号隔开 -->  
 <property name="basePackage" value="com.springmvc.dao"/>  
 <!-- 配置sqlSessionFactoryBeanName -->  
 <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>  
 </bean>  
  
</beans>

4．log4j.properties

### set log levels ###  
log4j.rootLogger = INFO,D  
  
###输出到控制台###  
log4j.logger.toConsole=debug,stdout  
log4j.appender.stdout = org.apache.log4j.ConsoleAppender  
log4j.appender.stdout.Target = System.out  
log4j.appender.stdout.layout = org.apache.log4j.PatternLayout  
  
  
log4j.appender.stdout.layout.ConversionPattern =%d{yyyy-MM-dd HH\:mm\:ss} [%5p] - %c -%F(%L) -%m%n  
  
### 输出到日志文件 ###  
## 其中File 可以采用绝对路径 亦可以采用相对路径 catalina.home 是tomcat目录 如果没有logs文件夹 将会报错 ##  
## 更加倾向于将日志文件放在 tomcat的日志目录下${catalina.home}/logs/salesmanb2bError.log ##  
## 绝对路径如：e:/salesmanb2bError.log ##  
log4j.logger.daily=INFO,D  
log4j.appender.D = org.apache.log4j.DailyRollingFileAppender  
log4j.appender.D.File = ${catalina.home}/logs/helloworld/helloworld.log  
log4j.appender.D.Append = true  
log4j.appender.D.Threshold = INFO  
log4j.appender.D.layout = org.apache.log4j.PatternLayout  
log4j.appender.D.layout.ConversionPattern =%d{yyyy-MM-dd HH\:mm\:ss} [%5p] - %c -%F(%L) -%m%n  
  
  
##log4j.logger.org.apache.ibatis=debug,stdout  
##log4j.logger.java.sql=debug,stdout

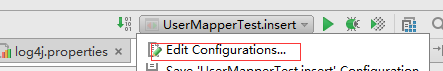
5. web.xml（在WEB-INF下）

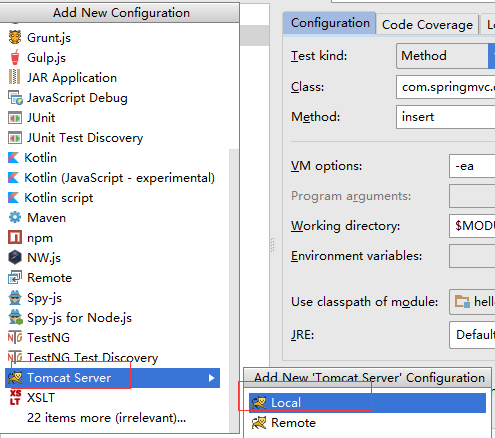
<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns="http://java.sun.com/xml/ns/javaee"  
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  
 version="3.0">  
 <display-name>helloworld2</display-name>  
 <!-- 配置编码方式-->  
 <filter>  
 <filter-name>encodingFilter</filter-name>  
 <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  
 <init-param>  
 <param-name>encoding</param-name>  
 <param-value>UTF-8</param-value>  
 </init-param>  
 <init-param>  
 <param-name>forceEncoding</param-name>  
 <param-value>true</param-value>  
 </init-param>  
 </filter>  
 <filter-mapping>  
 <filter-name>encodingFilter</filter-name>  
 <url-pattern>/\*</url-pattern>  
 </filter-mapping>  
  
  
 <!-- 配置springmvc的前端控制器 指向spring-mvc.xml 程序在启动的时候就加载springmvc 可以接受所有请求 load-on-startup：表示启动容器时初始化该Servlet； -->  
 <servlet>  
 <servlet-name>springServlet</servlet-name>  
 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
 <!-- 可以自定义servlet.xml配置文件的位置和名称， 默认为WEB-INF目录下，名称为[<servlet-name>]-servlet.xml，如spring-servlet.xml -->  
 <init-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value> classpath:spring/applicationContext.xml</param-value>  
 </init-param>  
 <load-on-startup>1</load-on-startup>  
 </servlet>  
 <!-- 将前端URL请求和后台处理方法controller建立对应关系-->  
 <servlet-mapping>  
 <servlet-name>springServlet</servlet-name>  
 <url-pattern>/</url-pattern>  
 </servlet-mapping>  
  
 <!-- Spring配置 -->  
 <listener>  
 <listener-class>  
 org.springframework.web.context.ContextLoaderListener  
 </listener-class>  
 </listener>  
  
 <context-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value> classpath\*:spring/applicationContext.xml</param-value>  
 </context-param>  
 <!-- 欢迎页面-->  
 <welcome-file-list>  
 <welcome-file>/WEB-INF/views/index.jsp</welcome-file>  
 </welcome-file-list>  
 <session-config>  
 <session-timeout>600</session-timeout>  
 </session-config>  
</web-app>

6、在WEB-INF/views/下创建index.jsp

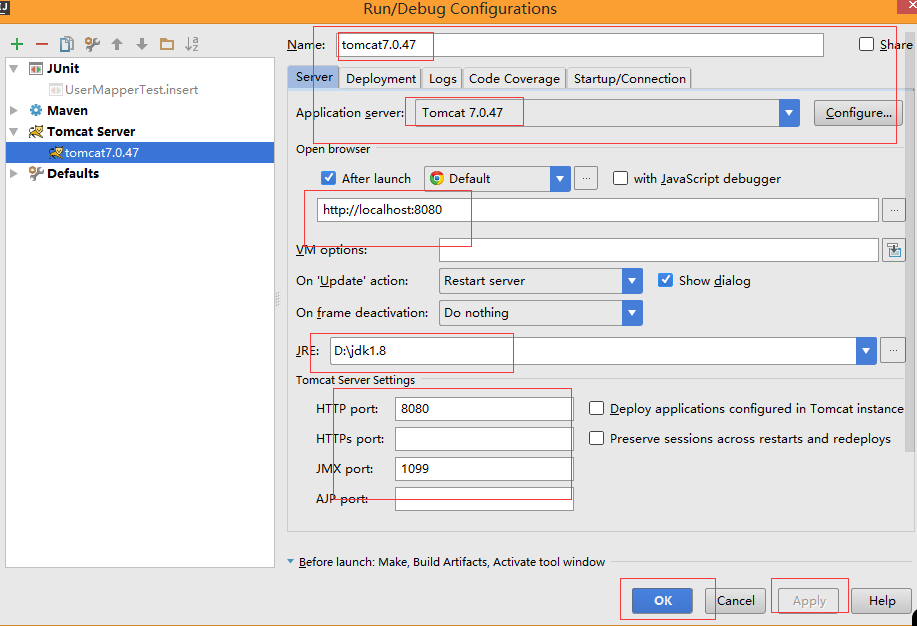
**3）配置Tomcat**

点击如图所示区域，进入“Edit Configurations”

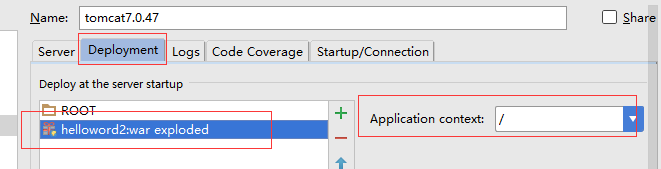




进入设置Tomcat：



然后将项目部署到Tomcat中



运行：

## 4）mybatis generator配置

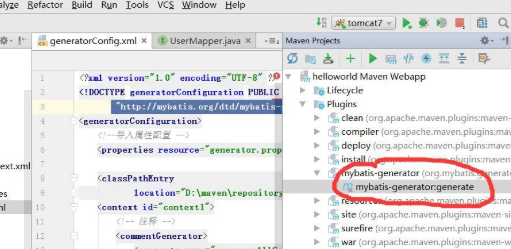
generatorConfig.xml：

*<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE generatorConfiguration PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"  
 "http://mybatis.org/dtd/mybatis-generator-config\_1\_0.dtd">  
<generatorConfiguration>  
 <!--导入属性配置 -->  
 <properties resource="generator.properties"/>  
  
 <classPathEntry  
 location="E:\ideaspace\mysql\mysql-connector-java-5.0.8-bin.jar" />  
 <context id="context1">  
 <!-- 注释 -->  
 <commentGenerator>  
 <property name="suppressAllComments" value="true" /><!-- 是否取消注释 -->  
 <property name="suppressDate" value="true" /> <!-- 是否生成注释代时间戳 -->  
 </commentGenerator>  
  
 <jdbcConnection driverClass="${driver}"  
 connectionURL="${url}"  
 userId="${username}"  
 password="${password}" />  
  
 <!-- 类型转换 -->  
 <javaTypeResolver>  
 <!-- 是否使用bigDecimal， false可自动转化以下类型（Long, Integer, Short, etc.） -->  
 <property name="forceBigDecimals" value="false" />  
 </javaTypeResolver>  
  
 <javaModelGenerator targetPackage="${modelPackage}"  
 targetProject="${modelProject}" />  
 <sqlMapGenerator targetPackage="${sqlPackage}" targetProject="${sqlProject}" />  
 <javaClientGenerator targetPackage="${mapperPackage}"  
 targetProject="${mapperProject}" type="XMLMAPPER" />  
  
 <!-- 如果需要通配所有表 直接用sql的通配符%即可 -->  
 <table schema="" tableName="${table}" enableCountByExample="true"  
 enableUpdateByExample="true" enableDeleteByExample="true"  
 enableSelectByExample="true"* selectByExampleQueryId="true"/>  
 </context>  
</generatorConfiguration>

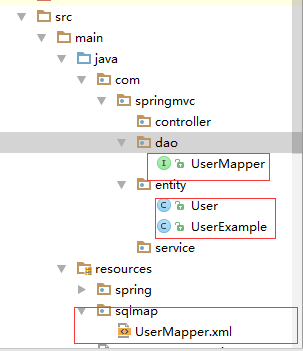
generator.properties：

driver=com.mysql.jdbc.Driver  
url=jdbc:mysql://localhost:3306/springmvc\_test?useUnicode=true&characterEncoding=utf-8  
username=root  
password=6666  
  
#entity 包名和 java目录  
modelPackage=com.springmvc.entity  
modelProject=src/main/java  
#sqlmap包名 和resources目录  
sqlPackage=sqlmap  
sqlProject=src/main/resources  
#mapper包名和 java目录  
mapperPackage=com.springmvc.dao  
mapperProject=src/main/java  
  
table=user

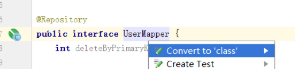
配置完成，双击如下：



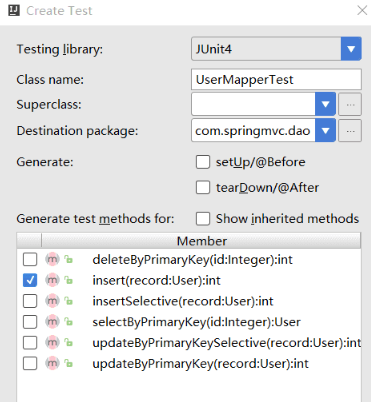
于是会自动创建图中四个文件：



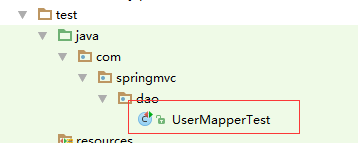
进入UserMapper.java。在接口上方添加@Repository注释，然后将鼠标放到public interface UserMapper处，点击alt+enter键，接着点击create test



勾选insert



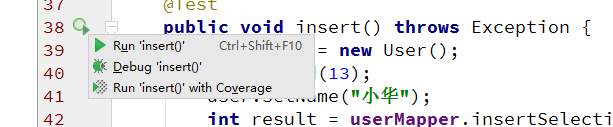
于是IDEA会自动帮我们创建个测试类：



代码如下：

**package** com.springmvc.dao;  
  
**import** com.springmvc.entity.User;  
**import** org.junit.After;  
**import** org.junit.Before;  
**import** org.junit.Test;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.context.ApplicationContext;  
**import** org.springframework.context.support.ClassPathXmlApplicationContext;  
  
**import static** org.junit.Assert.\*;  
  
/\*\*  
 \* Created by dell on 2018/7/27.  
 \*/  
**public class** UserMapperTest {  
 **private** ApplicationContext **applicationContext**;  
  
 //注入dao层接口  
 @Autowired  
 **private** UserMapper **userMapper**;  
  
 //insert（）方法前执行（体现了aop切面编程思想）  
 @Before  
 **public void** setup() **throws** Exception{  
 //加载spring容器  
 **applicationContext** = **new** ClassPathXmlApplicationContext(**"classpath:spring/applicationContext.xml"**);  
 **userMapper** = **applicationContext**.getBean(UserMapper.**class**);  
 }  
  
 //insert()方法后执行  
 @After  
 **public void** tearDown() **throws** Exception{  
  
 }  
  
 @Test  
 **public void** insert() **throws** Exception {  
 User user = **new** User();  
 user.setId(13);  
 user.setName(**"小华"**);  
 **int** result = **userMapper**.insertSelective(user);  
 System.**out**.println(result);  
 **assert**(result==1);  
 }  
}

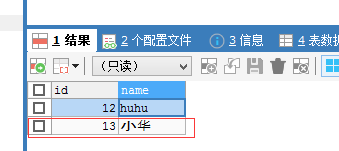
然后点击图中有个小感叹号的地方，选择”run insert()”



结果如下：



查看数据库数据：



结束