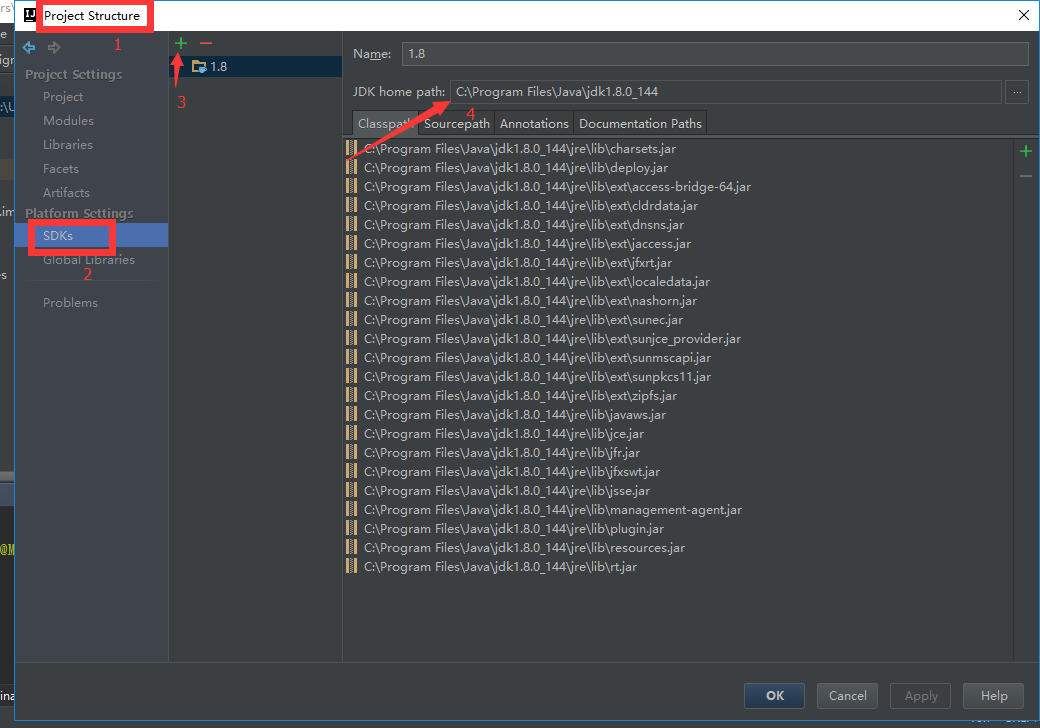
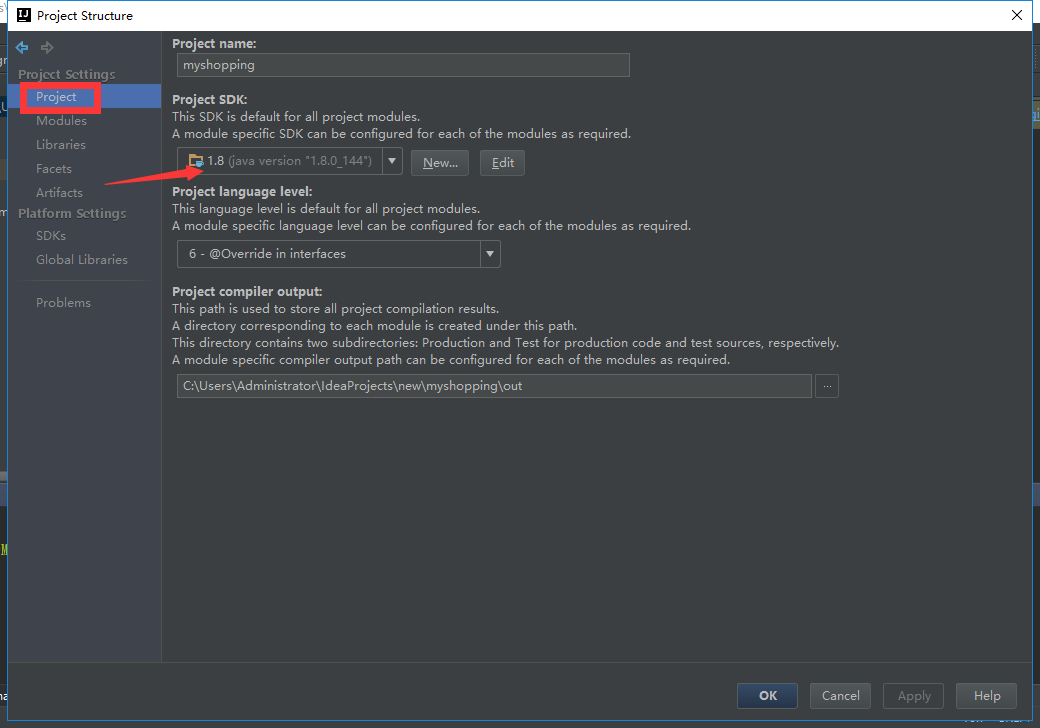
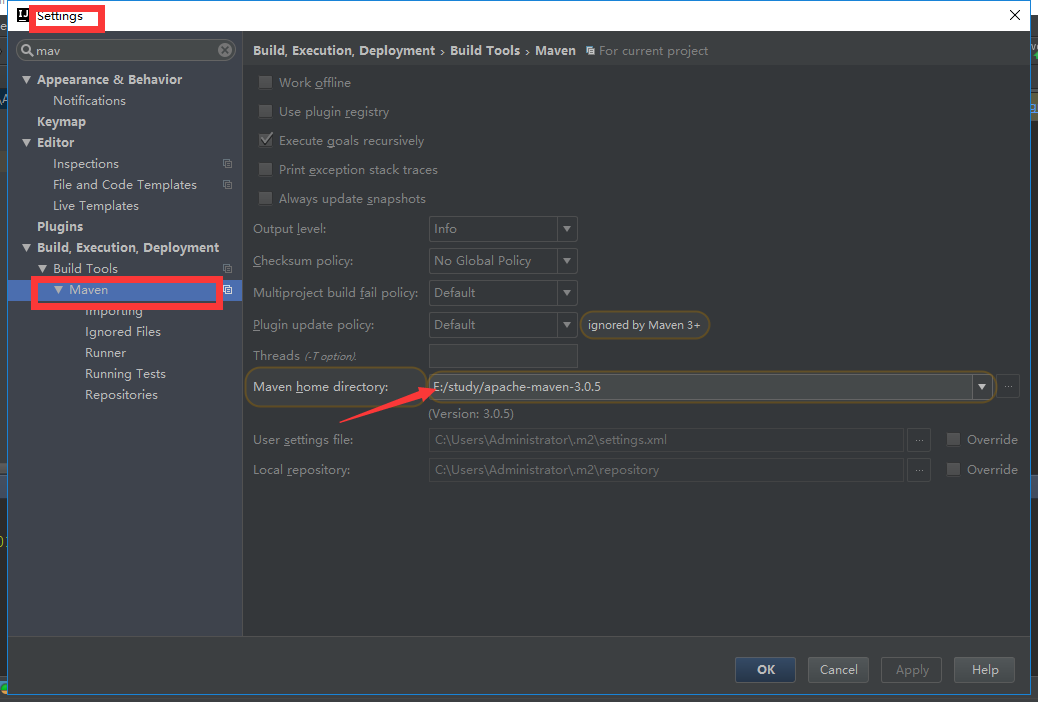
# Idea配置maven+git+ssm项目

1. 配置idea环境
2. 配置JDK

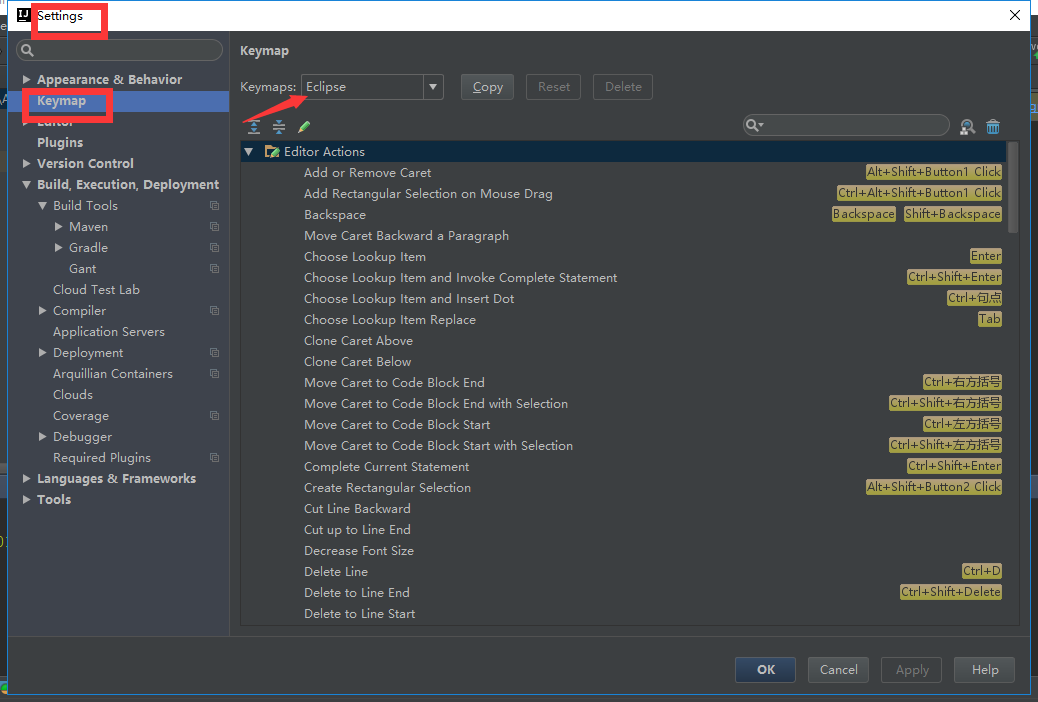




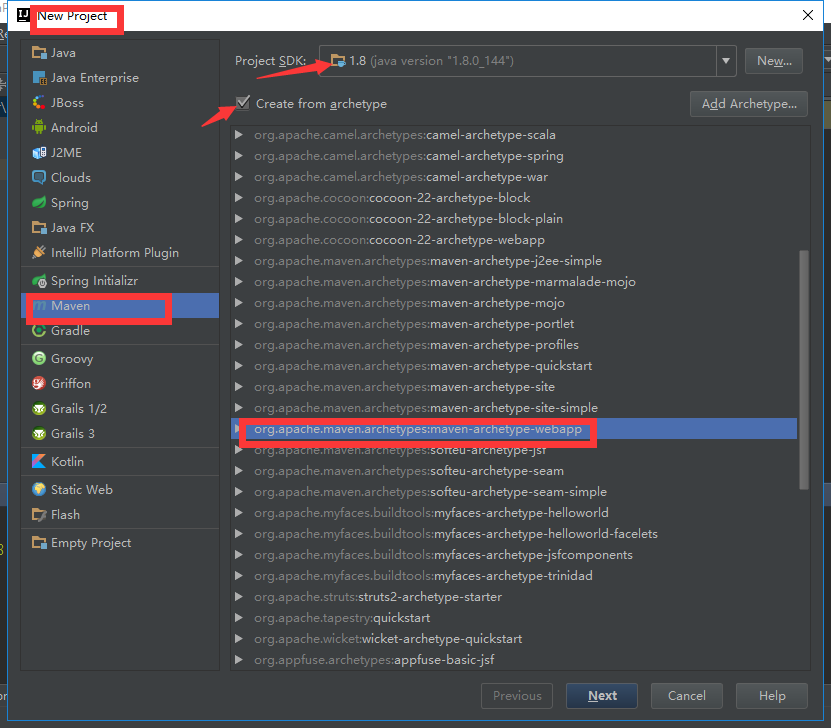
1. 配置maven

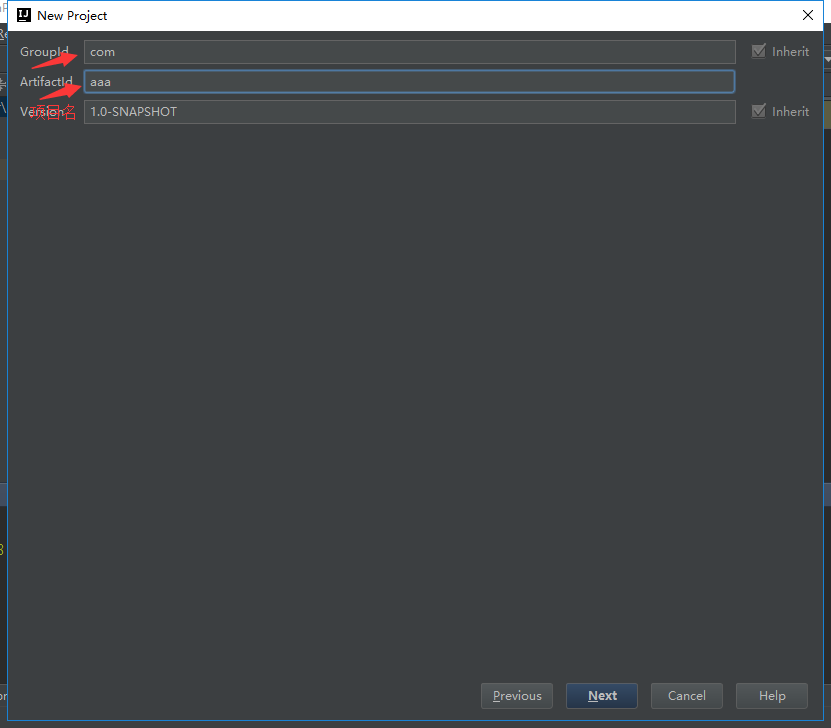


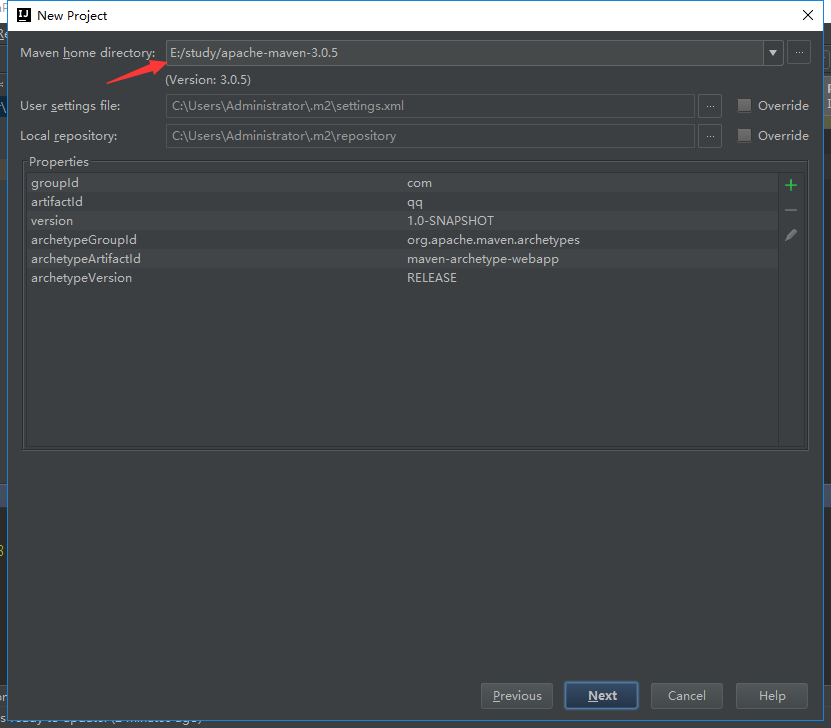
1. 配置快捷键（可选）

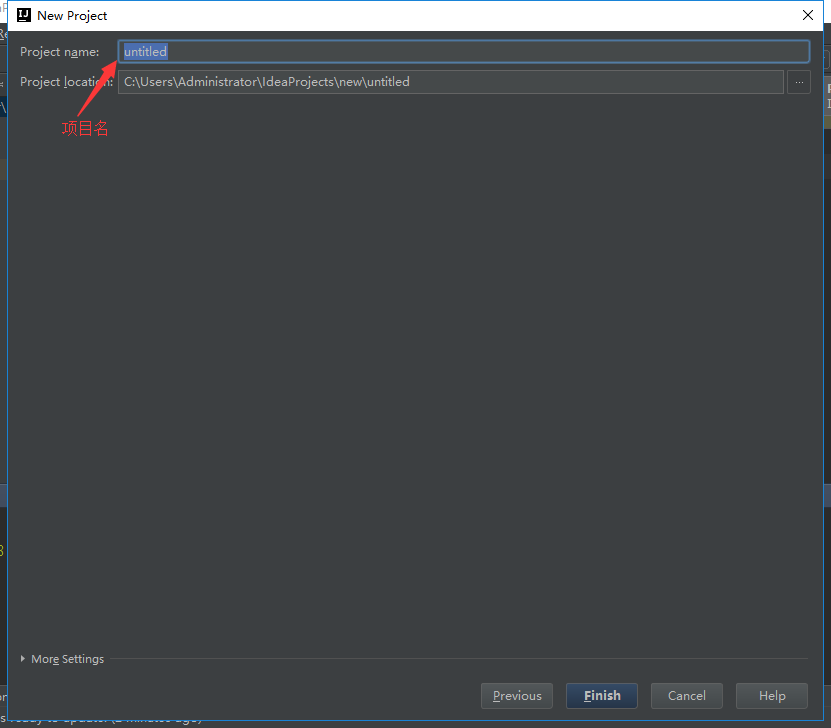


1. 创建、初始化项目
2. 创建

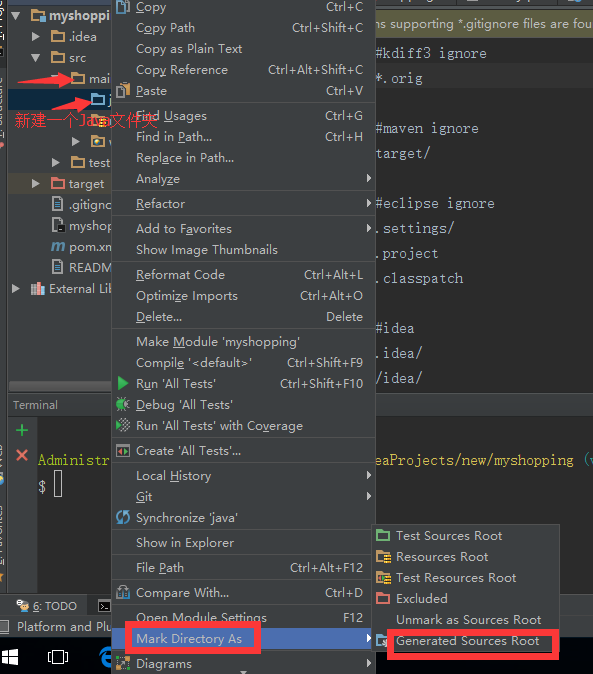


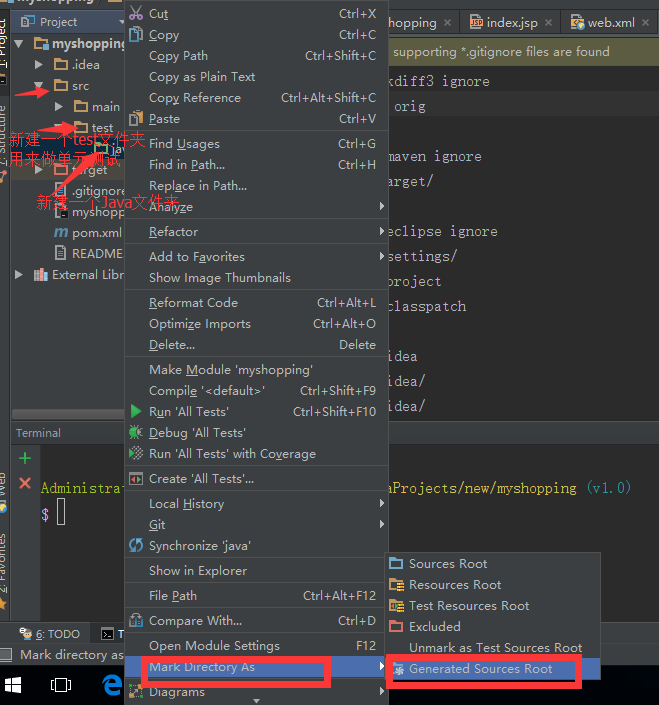




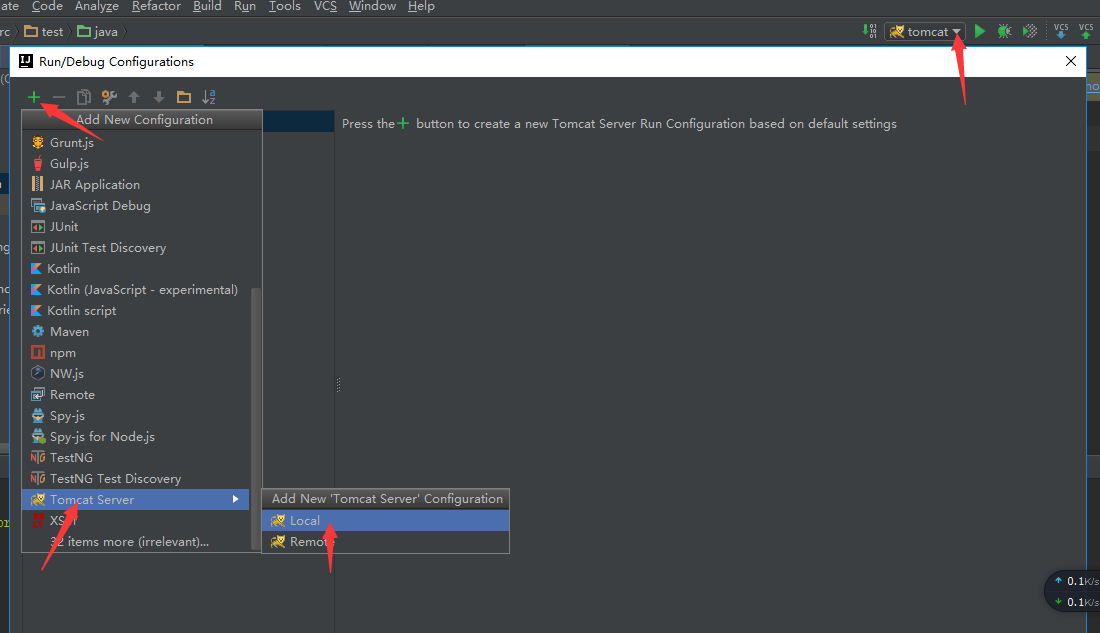


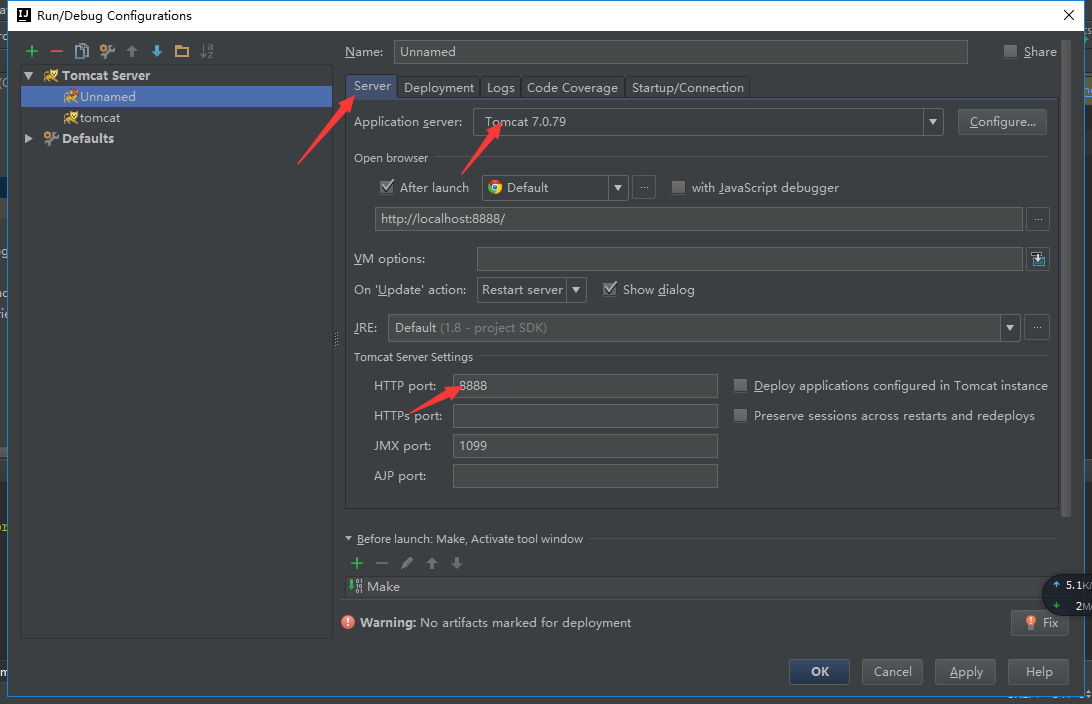
1. 配置项目

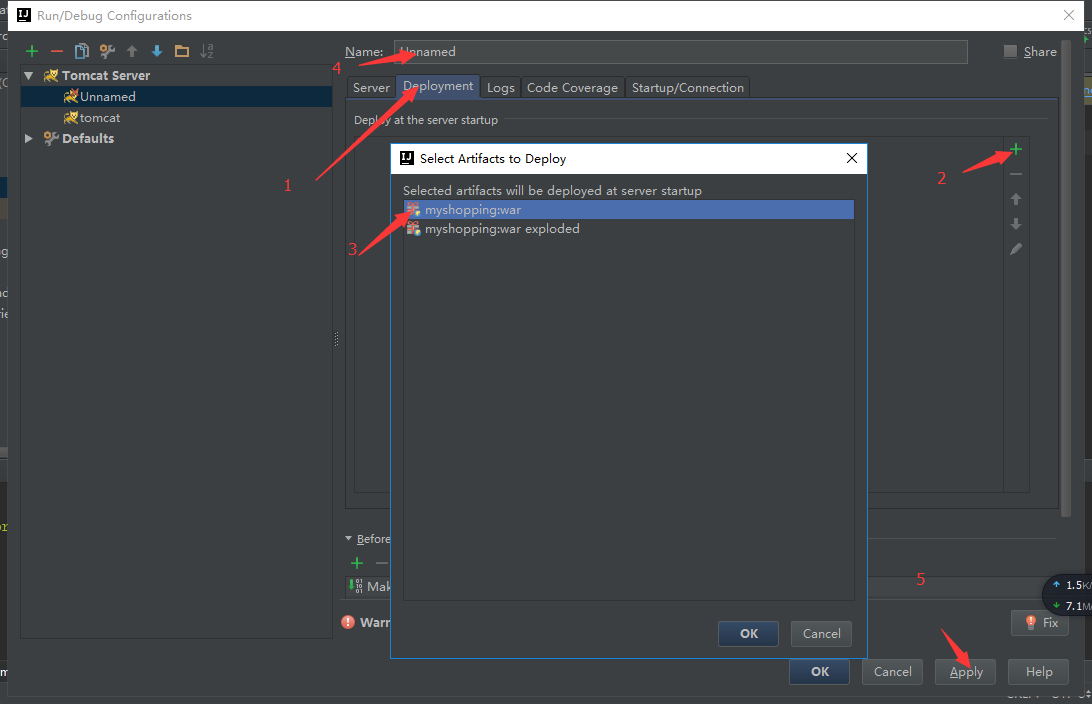




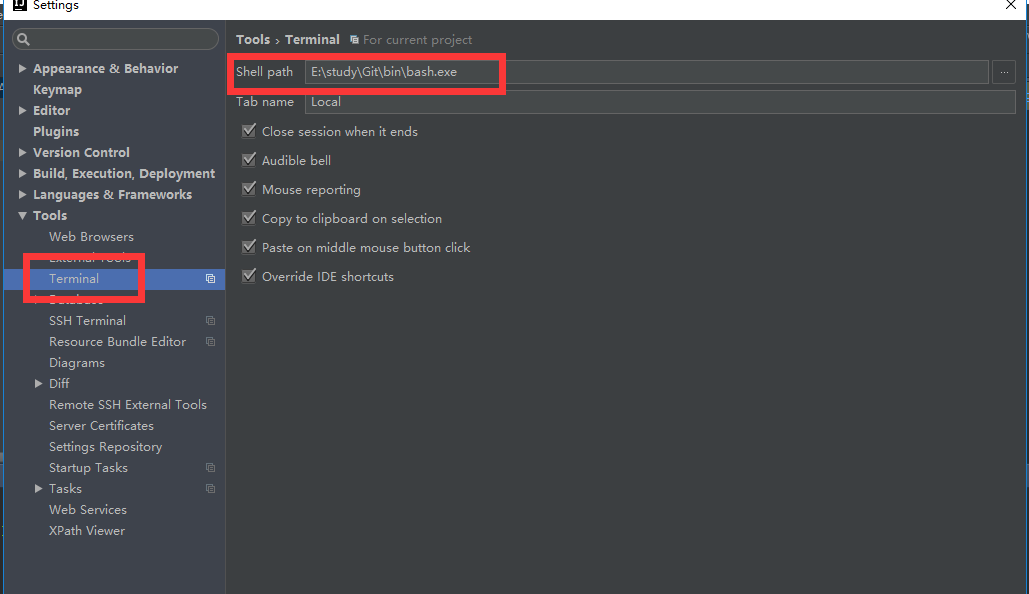
1. 配置tomcat





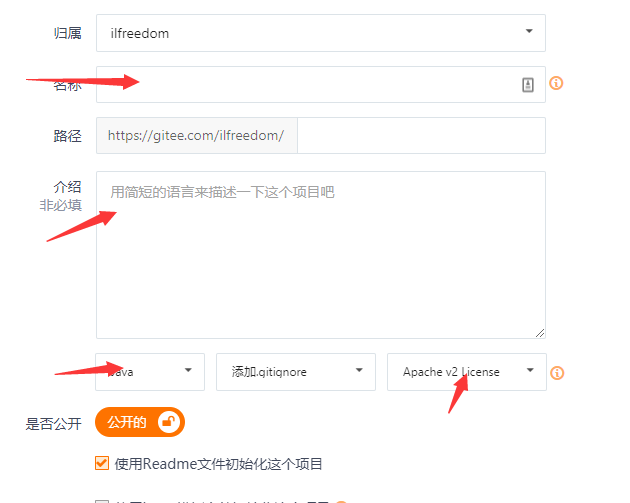


1. 启动tomcat，浏览器输入localhost:8888进行测试。
2. 初始化git
3. 将git配置到idea的Terminal



1. 在码云新建项目





1. 在idea中配置

在idea的terminal中输入 touch README.md 创建README.md文件。

touch .gitignore 创建.gitignore文件 配置不往仓库中推送的文件。

.gitignore文件的内容：（不是固定的）

#忽略什么就配置什么文件（表示不需要推送到git仓库）  
  
\*.class  
  
#package file  
\*.war  
\*.ear  
  
#kdiff3 ignore  
\*.orig  
  
#maven ignore  
target/  
  
#eclipse ignore  
.settings/  
.project  
.classpatch  
  
#idea  
.idea/  
/idea/  
\*.ipr  
\*.iml  
\*.iws  
  
# temp file  
\*.log  
\*.cache  
\*.diff  
\*.patch  
\*.tmp  
  
#system ignore  
.DS\_Store  
Thumbs.db

git init 进行初始化。

git status 可查看发生变化的文件。

git add **.** 添加。

git commit –am ‘注释’ 提交到本地仓库。

git remote add origin 远程git地址 关联远程仓库。

git push –u origin master 推送。

git checkout –b v1.0 origin/master 创建新分支。

git push origin HEAD –u 将新的分支推送到远程仓库。

注：如果github可以关联成功，但提交超时，解决如下：

找到git的安装目录，找到/etc/ssh/ssh\_config文件，

把如下内容复制到ssh\_config文件的末尾处：并记得保存

Host github.com

User git

Hostname ssh.github.com

PreferredAuthentications publickey

IdentityFile ~/.ssh/id\_rsa

Port 443

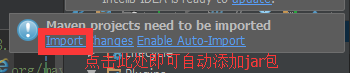
本地提交时，如果github中的README.md文件不在本地代码目录中，

需要先执行git pull --rebase origin master，然后才能git push。

1. 配置maven的pom.xml文件（配置jar包）

文件内容：

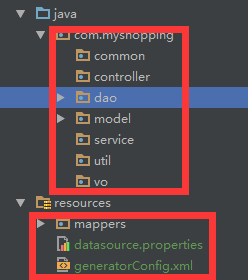
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>com</groupId>  
 <artifactId>myshopping</artifactId>  
 <packaging>war</packaging>  
 <version>1.0-SNAPSHOT</version>  
 <name>myshopping Maven Webapp</name>  
 <url>http://maven.apache.org</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <maven.compiler.encoding>UTF-8</maven.compiler.encoding>  
  
 <!--下面只需引用这里的版本-->  
 <org.springframework.version>4.3.3.RELEASE</org.springframework.version>  
 <org.mybatis.version>3.4.1</org.mybatis.version>  
 <org.mybatis.spring.version>1.3.0</org.mybatis.spring.version>  
 </properties>  
  
 <dependencies>  
  
 <dependency>  
 <groupId>org.apache.tomcat</groupId>  
 <artifactId>tomcat-servlet-api</artifactId>  
 <version>7.0.79</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>${org.springframework.version}</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-oxm</artifactId>  
 <version>${org.springframework.version}</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jdbc</artifactId>  
 <version>${org.springframework.version}</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-tx</artifactId>  
 <version>${org.springframework.version}</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-test</artifactId>  
 <version>${org.springframework.version}</version>  
 </dependency>  
  
  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 <version>1.7.3</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis-spring</artifactId>  
 <version>${org.mybatis.spring.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>${org.mybatis.version}</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjrt</artifactId>  
 <version>1.6.11</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.codehaus.jackson</groupId>  
 <artifactId>jackson-mapper-asl</artifactId>  
 <version>1.9.12</version>  
 </dependency>  
  
 <dependency>  
 <groupId>commons-dbcp</groupId>  
 <artifactId>commons-dbcp</artifactId>  
 <version>1.4</version>  
 <!--<scope>runtime</scope>-->  
 </dependency>  
  
  
 <dependency>  
 <groupId>ch.qos.logback</groupId>  
 <artifactId>logback-classic</artifactId>  
 <version>1.1.2</version>  
 <scope>compile</scope>  
 </dependency>  
 <dependency>  
 <groupId>ch.qos.logback</groupId>  
 <artifactId>logback-core</artifactId>  
 <version>1.1.2</version>  
 <scope>compile</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.6</version>  
 </dependency>  
  
 <dependency>  
 <groupId>com.google.guava</groupId>  
 <artifactId>guava</artifactId>  
 <version>20.0</version>  
 </dependency>  
  
  
 <dependency>  
 <groupId>org.apache.commons</groupId>  
 <artifactId>commons-lang3</artifactId>  
 <version>3.5</version>  
 </dependency>  
  
  
 <dependency>  
 <groupId>commons-collections</groupId>  
 <artifactId>commons-collections</artifactId>  
 <version>3.2.1</version>  
 </dependency>  
  
  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.12</version>  
 <!--<scope>test</scope>-->  
 </dependency>  
  
 <dependency>  
 <groupId>joda-time</groupId>  
 <artifactId>joda-time</artifactId>  
 <version>2.3</version>  
 </dependency>  
  
  
 <!-- id加密解密 -->  
 <dependency>  
 <groupId>org.hashids</groupId>  
 <artifactId>hashids</artifactId>  
 <version>1.0.1</version>  
 </dependency>  
  
  
 <!-- ftpclient -->  
 <dependency>  
 <groupId>commons-net</groupId>  
 <artifactId>commons-net</artifactId>  
 <version>3.1</version>  
 </dependency>  
  
 <!-- file upload -->  
  
 <!-- https://mvnrepository.com/artifact/commons-fileupload/commons-fileupload -->  
 <dependency>  
 <groupId>commons-fileupload</groupId>  
 <artifactId>commons-fileupload</artifactId>  
 <version>1.2.2</version>  
 </dependency>  
  
 <dependency>  
 <groupId>commons-io</groupId>  
 <artifactId>commons-io</artifactId>  
 <version>2.0.1</version>  
 </dependency>  
  
  
  
  
 <!-- mybatis pager -->  
  
 <dependency>  
 <groupId>com.github.pagehelper</groupId>  
 <artifactId>pagehelper</artifactId>  
 <version>4.1.0</version>  
 </dependency>  
  
 <dependency>  
 <groupId>com.github.miemiedev</groupId>  
 <artifactId>mybatis-paginator</artifactId>  
 <version>1.2.17</version>  
 </dependency>  
  
 <dependency>  
 <groupId>com.github.jsqlparser</groupId>  
 <artifactId>jsqlparser</artifactId>  
 <version>0.9.4</version>  
 </dependency>  
  
  
 <!-- alipay -->  
 <dependency>  
 <groupId>commons-codec</groupId>  
 <artifactId>commons-codec</artifactId>  
 <version>1.10</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-configuration</groupId>  
 <artifactId>commons-configuration</artifactId>  
 <version>1.10</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-lang</groupId>  
 <artifactId>commons-lang</artifactId>  
 <version>2.6</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-logging</groupId>  
 <artifactId>commons-logging</artifactId>  
 <version>1.1.1</version>  
 </dependency>  
 <dependency>  
 <groupId>com.google.zxing</groupId>  
 <artifactId>core</artifactId>  
 <version>2.1</version>  
 </dependency>  
 <dependency>  
 <groupId>com.google.code.gson</groupId>  
 <artifactId>gson</artifactId>  
 <version>2.3.1</version>  
 </dependency>  
 <dependency>  
 <groupId>org.hamcrest</groupId>  
 <artifactId>hamcrest-core</artifactId>  
 <version>1.3</version>  
 </dependency>  
  
 <dependency>  
 <groupId>redis.clients</groupId>  
 <artifactId>jedis</artifactId>  
 <version>2.9.0</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <finalName>myshopping</finalName>  
 <plugins>  
 <plugin>  
 <groupId>org.mybatis.generator</groupId>  
 <artifactId>mybatis-generator-maven-plugin</artifactId>  
 <version>1.3.2</version>  
 <configuration>  
 <verbose>true</verbose>  
 <overwrite>true</overwrite>  
 </configuration>  
 </plugin>  
 <!-- geelynote maven的核心插件之-complier插件默认只支持编译Java 1.4，因此需要加上支持高版本jre的配置，在pom.xml里面加上 增加编译插件 -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 <encoding>UTF-8</encoding>  
 <compilerArguments>  
 <extdirs>${project.basedir}/src/main/webapp/WEB-INF/lib</extdirs>  
 </compilerArguments>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

注：



若此处报错，可点击行前的小灯泡，选择第一个进行添加。

1. 配置mybaties-generator（自动生产model、dao和相关xml配置文件）
2. 创建包



1. 创建数据库表
2. 配置generatorConfig.xml

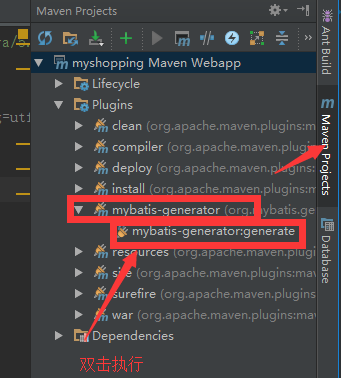
内容：

<?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="datasource.properties"></properties>  
  
 <!--指定特定数据库的jdbc驱动jar包的位置-->  
 <classPathEntry location="${db.driverLocation}"/>  
  
 <context id="default" targetRuntime="MyBatis3">  
  
 <!-- optional，旨在创建class时，对注释进行控制 -->  
 <commentGenerator>  
 <property name="suppressDate" value="true"/>  
 <property name="suppressAllComments" value="true"/>  
 </commentGenerator>  
  
 <!--jdbc的数据库连接 -->  
 <jdbcConnection  
 driverClass="${db.driverClassName}"  
 connectionURL="${db.url}"  
 userId="${db.username}"  
 password="${db.password}">  
 </jdbcConnection>  
  
  
 <!-- 非必需，类型处理器，在数据库类型和java类型之间的转换控制-->  
 <javaTypeResolver>  
 <property name="forceBigDecimals" value="false"/>  
 </javaTypeResolver>  
  
  
 <!-- Model模型生成器,用来生成含有主键key的类，记录类 以及查询Example类  
 targetPackage 指定生成的model生成所在的包名  
 targetProject 指定在该项目下所在的路径  
 -->  
 <!--<javaModelGenerator targetPackage="com.myshopping.model" targetProject=".\src\main\java">-->  
 <javaModelGenerator targetPackage="com.myshopping.model" targetProject="./src/main/java">  
 <!-- 是否允许子包，即targetPackage.schemaName.tableName -->  
 <property name="enableSubPackages" value="false"/>  
 <!-- 是否对model添加 构造函数 -->  
 <property name="constructorBased" value="true"/>  
 <!-- 是否对类CHAR类型的列的数据进行trim操作 -->  
 <property name="trimStrings" value="true"/>  
 <!-- 建立的Model对象是否 不可改变 即生成的Model对象不会有 setter方法，只有构造方法 -->  
 <property name="immutable" value="false"/>  
 </javaModelGenerator>  
  
 <!--mapper映射文件生成所在的目录 为每一个数据库的表生成对应的SqlMap文件 -->  
 <!--<sqlMapGenerator targetPackage="mappers" targetProject=".\src\main\resources">-->  
 <sqlMapGenerator targetPackage="mappers" targetProject="./src/main/resources">  
 <property name="enableSubPackages" value="false"/>  
 </sqlMapGenerator>  
  
 <!-- 客户端代码，生成易于使用的针对Model对象和XML配置文件 的代码  
 type="ANNOTATEDMAPPER",生成Java Model 和基于注解的Mapper对象  
 type="MIXEDMAPPER",生成基于注解的Java Model 和相应的Mapper对象  
 type="XMLMAPPER",生成SQLMap XML文件和独立的Mapper接口  
 -->  
  
 <!-- targetPackage：mapper接口dao生成的位置 -->  
 <!--<javaClientGenerator type="XMLMAPPER" targetPackage="com.myshopping.dao" targetProject=".\src\main\java">-->  
 <javaClientGenerator type="XMLMAPPER" targetPackage="com.myshopping.dao" targetProject="./src/main/java">  
 <!-- enableSubPackages:是否让schema作为包的后缀 -->  
 <property name="enableSubPackages" value="false" />  
 </javaClientGenerator>  
  
  
 <table tableName="mmall\_shipping" domainObjectName="Shipping" enableCountByExample="false" enableUpdateByExample="false" enableDeleteByExample="false" enableSelectByExample="false" selectByExampleQueryId="false"></table>  
 <table tableName="mmall\_cart" domainObjectName="Cart" enableCountByExample="false" enableUpdateByExample="false" enableDeleteByExample="false" enableSelectByExample="false" selectByExampleQueryId="false"></table>  
 <table tableName="mmall\_cart\_item" domainObjectName="CartItem" enableCountByExample="false" enableUpdateByExample="false" enableDeleteByExample="false" enableSelectByExample="false" selectByExampleQueryId="false"></table>  
 <table tableName="mmall\_category" domainObjectName="Category" enableCountByExample="false" enableUpdateByExample="false" enableDeleteByExample="false" enableSelectByExample="false" selectByExampleQueryId="false"></table>  
 <table tableName="mmall\_order" domainObjectName="Order" enableCountByExample="false" enableUpdateByExample="false" enableDeleteByExample="false" enableSelectByExample="false" selectByExampleQueryId="false"></table>  
 <table tableName="mmall\_order\_item" domainObjectName="OrderItem" enableCountByExample="false" enableUpdateByExample="false" enableDeleteByExample="false" enableSelectByExample="false" selectByExampleQueryId="false"></table>  
 <table tableName="mmall\_pay\_info" domainObjectName="PayInfo" enableCountByExample="false" enableUpdateByExample="false" enableDeleteByExample="false" enableSelectByExample="false" selectByExampleQueryId="false"></table>  
 <table tableName="mmall\_product" domainObjectName="Product" enableCountByExample="false" enableUpdateByExample="false" enableDeleteByExample="false" enableSelectByExample="false" selectByExampleQueryId="false">  
 <columnOverride column="detail" jdbcType="VARCHAR" />  
 <columnOverride column="sub\_images" jdbcType="VARCHAR" />  
 </table>  
 <table tableName="mmall\_user" domainObjectName="User" enableCountByExample="false" enableUpdateByExample="false" enableDeleteByExample="false" enableSelectByExample="false" selectByExampleQueryId="false"></table>  
  
  
 <!-- geelynote mybatis插件的搭建 -->  
 </context>  
</generatorConfiguration>

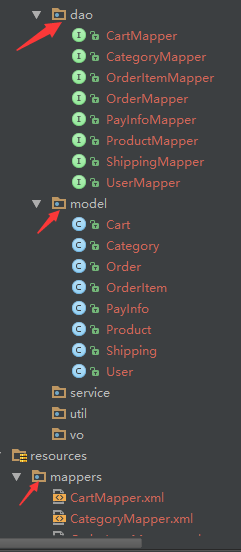
1. 配置datasource.properties（数据库的配置）

内容：

db.driverLocation=/Users/Administrator/.m2/repository/mysql/mysql-connector-java/5.1.6/mysql-connector-java-5.1.6.jar  
db.driverClassName=com.mysql.jdbc.Driver  
  
#db.url=jdbc:mysql://你的数据库IP:你的数据库Port/你的database?characterEncoding=utf-8  
db.url=jdbc:mysql://localhost:3306/myshopping?characterEncoding=utf-8  
db.username=root  
db.password=12345678900

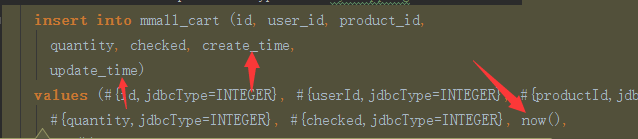
1. 

结束后：



注：时间戳问题：

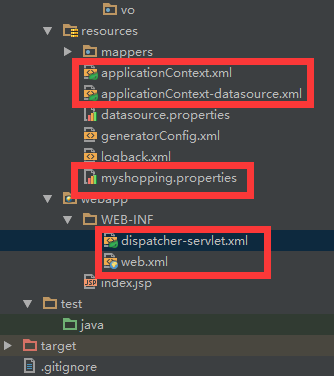
将每个model对应的xml配置文件中的时间戳类型的值都换为now()函数



旨在将对时间戳的处理交给DB，就不用自己在代码中处理了。

1. 配置mybaties-plugin
2. 配置spring和springmvc

需要添加或更改的文件：



1. web.xml:

<?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\_2\_5.xsd"

id="WebApp\_ID" version="2.5">

<display-name>Archetype Created Web Application</display-name>

<filter>

<!--配置转码的过滤器-->

<filter-name>characterEncodingFilter</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>characterEncodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!--监听web的启动和关闭-->

<listener>

<listener-class>org.springframework.web.context.request.RequestContextListener</listener-class>

</listener>

<!--web容器和spring容器进行整合-->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<context-param>

<!--通过ContextLoaderListener加载applicationContext.xml文件-->

<param-name>contextConfigLocation</param-name>

<param-value>

classpath:applicationContext.xml

</param-value>

</context-param>

<servlet>

<!--配置springMVC-->

<servlet-name>dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

<url-pattern>\*.do</url-pattern>

</servlet-mapping>

</web-app>

1. applicationContext.xml:

<?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:aop="http://www.springframework.org/schema/aop"

xmlns:tx="http://www.springframework.org/schema/tx" xmlns:jdbc="http://www.springframework.org/schema/jdbc"

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

xsi:schemaLocation="

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

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

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd

http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd">

<!--扫描com.myshopping包下的注解-->

<context:component-scan base-package="com.myshopping" annotation-config="true"/>

<!--<context:annotation-config/>-->

<aop:aspectj-autoproxy/>

<import resource="applicationContext-datasource.xml"/>

</beans>

1. applicationContext-datasource.xml:

<?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:aop="http://www.springframework.org/schema/aop"

xmlns:tx="http://www.springframework.org/schema/tx" xmlns:jdbc="http://www.springframework.org/schema/jdbc"

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

xsi:schemaLocation="

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

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

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd

http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd">

<context:component-scan base-package="com.myshopping" annotation-config="true"/>

<bean id="propertyConfigurer"

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

<property name="order" value="2"/>

<property name="ignoreUnresolvablePlaceholders" value="true"/>

<property name="locations">

<list>

<value>classpath:datasource.properties</value>

</list>

</property>

<property name="fileEncoding" value="utf-8"/>

</bean>

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">

<property name="driverClassName" value="${db.driverClassName}"/>

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

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

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

<!-- 连接池启动时的初始值 -->

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

<!-- 连接池的最大值 -->

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

<!-- 最大空闲值.当经过一个高峰时间后，连接池可以慢慢将已经用不到的连接慢慢释放一部分，一直减少到maxIdle为止 -->

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

<!-- 最小空闲值.当空闲的连接数少于阀值时，连接池就会预申请去一些连接，以免洪峰来时来不及申请 -->

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

<!-- 最大建立连接等待时间。如果超过此时间将接到异常。设为－1表示无限制 -->

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

<!--#给出一条简单的sql语句进行验证 -->

<!--<property name="validationQuery" value="select getdate()" />-->

<property name="defaultAutoCommit" value="${db.defaultAutoCommit}"/>

<!-- 回收被遗弃的（一般是忘了释放的）数据库连接到连接池中 -->

<!--<property name="removeAbandoned" value="true" />-->

<!-- 数据库连接过多长时间不用将被视为被遗弃而收回连接池中 -->

<!--<property name="removeAbandonedTimeout" value="120" />-->

<!-- #连接的超时时间，默认为半小时。 -->

<property name="minEvictableIdleTimeMillis" value="${db.minEvictableIdleTimeMillis}"/>

<!--# 失效检查线程运行时间间隔，要小于MySQL默认-->

<property name="timeBetweenEvictionRunsMillis" value="40000"/>

<!--# 检查连接是否有效-->

<property name="testWhileIdle" value="true"/>

<!--# 检查连接有效性的SQL语句-->

<property name="validationQuery" value="SELECT 1 FROM dual"/>

</bean>

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

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

<property name="mapperLocations" value="classpath\*:mappers/\*Mapper.xml"></property>

<!-- 分页插件 -->

<property name="plugins">

<array>

<bean class="com.github.pagehelper.PageHelper">

<property name="properties">

<value>

dialect=mysql

</value>

</property>

</bean>

</array>

</property>

</bean>

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

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

</bean>

<!-- 使用@Transactional进行声明式事务管理需要声明下面这行 -->

<tx:annotation-driven transaction-manager="transactionManager" proxy-target-class="true" />

<!-- 事务管理 -->

<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

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

<property name="rollbackOnCommitFailure" value="true"/><!--提交失败进行回滚-->

</bean>

</beans>

1. dispatcher-servlet.xml:

<?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" xmlns:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

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

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

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

<!--配置springMVC-->

<context:component-scan base-package="com.myshopping" annotation-config="true"/>

<mvc:annotation-driven>

<mvc:message-converters>

<bean class="org.springframework.http.converter.StringHttpMessageConverter">

<property name="supportedMediaTypes">

<list>

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

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

</list>

</property>

</bean>

<bean class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter">

<property name="supportedMediaTypes">

<list>

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

</list>

</property>

</bean>

</mvc:message-converters>

</mvc:annotation-driven>

<!-- 文件上传 -->

<bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

<property name="maxUploadSize" value="10485760"/> <!-- 10m -->

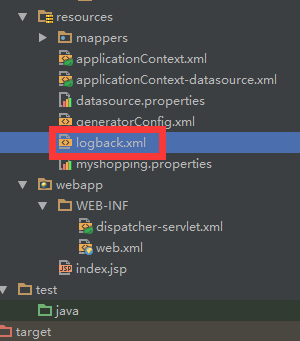
<property name="maxInMemorySize" value="4096" />

<property name="defaultEncoding" value="UTF-8"></property>

</bean>

</beans>

1. 配置日志



内容：

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

<configuration scan="true" scanPeriod="60 seconds" debug="false">

<!--配置log日志-->

<appender name="console" class="ch.qos.logback.core.ConsoleAppender">

<encoding>UTF-8</encoding>

<encoder>

<pattern>[%d{HH:mm:ss.SSS}][%p][%c{40}][%t] %m%n</pattern> <!--格式-->

</encoder>

<filter class="ch.qos.logback.classic.filter.ThresholdFilter">

<level>DEBUG</level> <!--级别大于DEBUG，便于查看mybaties中的sql语句-->

</filter>

</appender>

<!--项目的日志-->

<appender name="myshopping" class="ch.qos.logback.core.rolling.RollingFileAppender">

<!--<File>d:/myshoppinglog/myshopping.log</File>-->

<File>/developer/apache-tomcat-7.0.79/logs/myshopping.log</File>

<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<fileNamePattern>/developer/apache-tomcat-7.0.79/logs/myshopping.log.%d{yyyy-MM-dd}.gz</fileNamePattern>

<append>true</append>

<maxHistory>10</maxHistory>

</rollingPolicy>

<encoder>

<pattern>[%d{HH:mm:ss.SSS}][%p][%c{40}][%t] %m%n</pattern>

</encoder>

</appender>

<appender name="error" class="ch.qos.logback.core.rolling.RollingFileAppender">

<!--<File>d:/myshoppinglog/error.log</File>-->

<File>/developer/apache-tomcat-7.0.79/logs/error.log</File>

<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<fileNamePattern>/devsoft/apache-tomcat-7.0.79/logs/error.log.%d{yyyy-MM-dd}.gz</fileNamePattern>

<!--<fileNamePattern>d:/myshoppinglog/error.log.%d{yyyy-MM-dd}.gz</fileNamePattern>-->

<append>true</append>

<maxHistory>10</maxHistory>

</rollingPolicy>

<encoder>

<pattern>[%d{HH:mm:ss.SSS}][%p][%c{40}][%t] %m%n</pattern>

</encoder>

<filter class="ch.qos.logback.classic.filter.LevelFilter">

<level>ERROR</level>

<onMatch>ACCEPT</onMatch>

<onMismatch>DENY</onMismatch>

</filter>

</appender>

<logger name="com.myshopping" additivity="false" level="INFO" >

<appender-ref ref="myshopping" />

<appender-ref ref="console"/>

</logger>

<!-- geelynote mybatis log 日志 -->

<logger name="com.myshopping.dao" level="DEBUG"/>

<!--<logger name="com.ibatis.sqlmap.engine.impl.SqlMapClientDelegate" level="DEBUG" >-->

<!--<appender-ref ref="console"/>-->

<!--</logger>-->

<!--<logger name="java.sql.Connection" level="DEBUG">-->

<!--<appender-ref ref="console"/>-->

<!--</logger>-->

<!--<logger name="java.sql.Statement" level="DEBUG">-->

<!--<appender-ref ref="console"/>-->

<!--</logger>-->

<!--<logger name="java.sql.PreparedStatement" level="DEBUG">-->

<!--<appender-ref ref="console"/>-->

<!--</logger>-->

<root level="DEBUG">

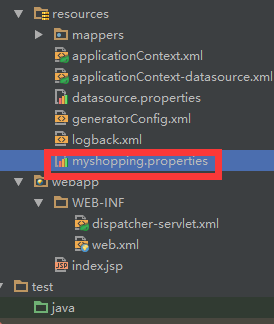
<appender-ref ref="console"/>

<appender-ref ref="error"/>

</root>

</configuration>

1. 配置ftp服务器



内容：

#配置ftp服务器

ftp.server.ip=你的FTP服务器ip地址

ftp.user=ypy

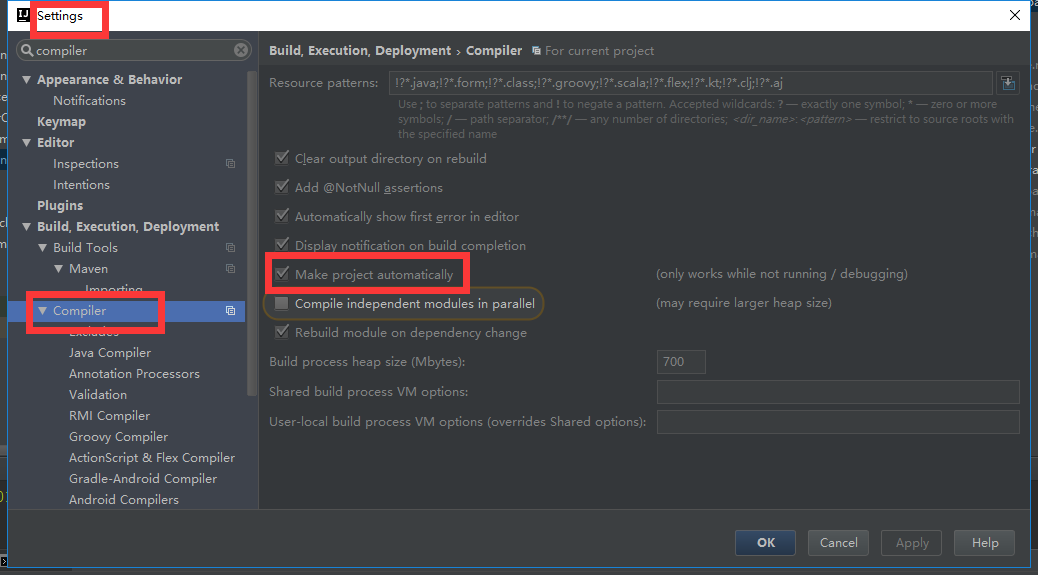
ftp.pass=ypy

ftp.server.http.prefix=http://img.ypy.com/

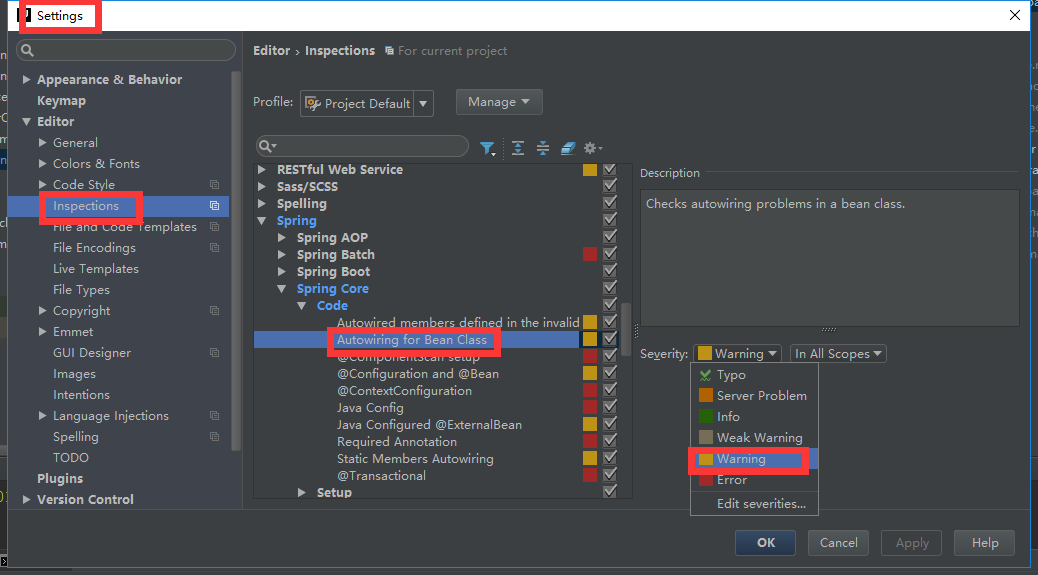
alipay.callback.url=http://www.happymmall.com/order/alipay\_callback.do

password.salt = geelysdafaqj23ou89ZXcj@#$@#$#@KJdjklj;D../dSF.,

1. 两个注意的配置
2. 调出problems界面



1. 将关于mybatis的一个错误提示调成警告



1. 将初始化项目推送到git的仓库

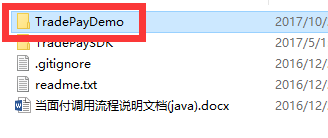
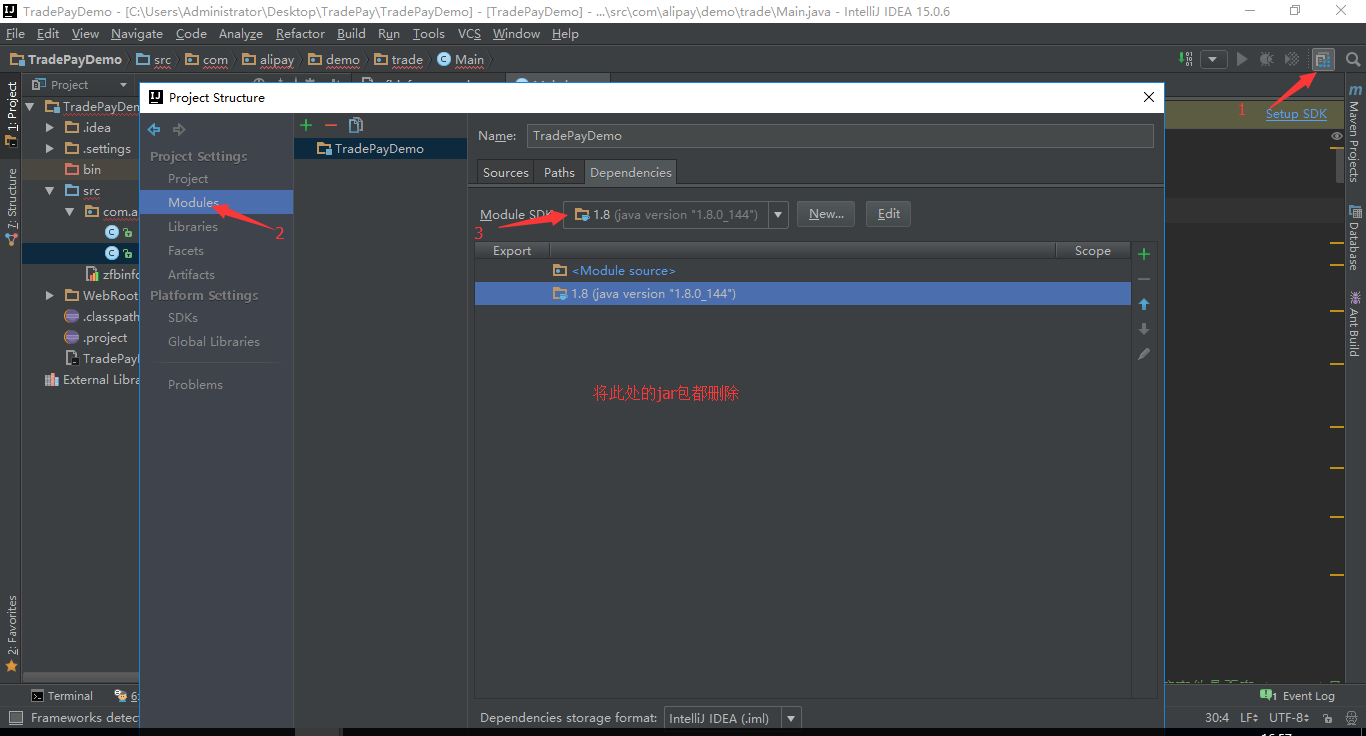
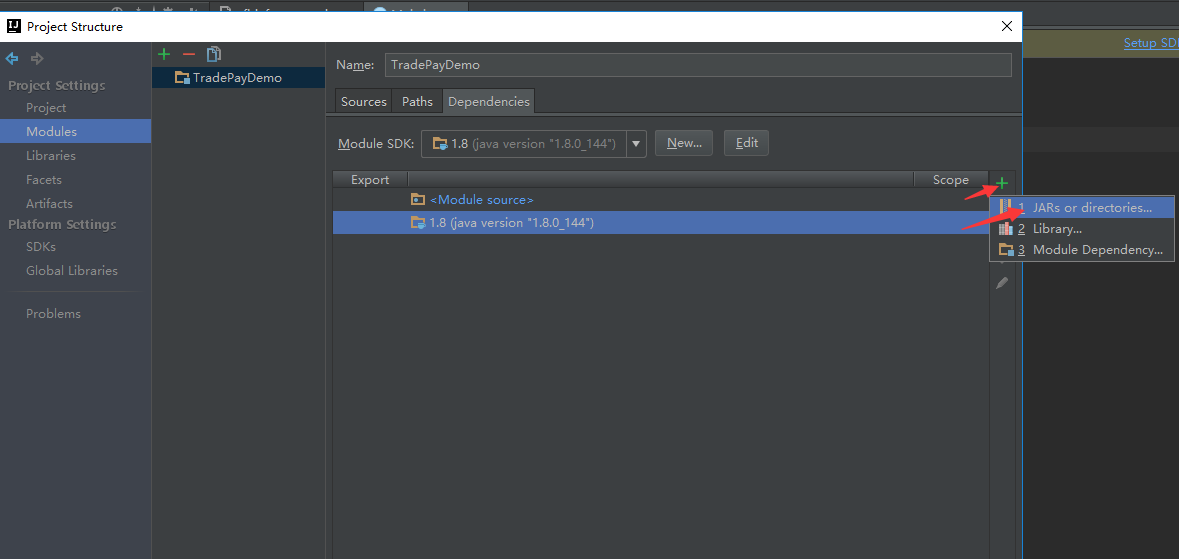
git add **.**

git commit -am ‘progect init commit’

git push

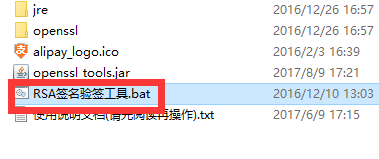
1. 配置沙箱支付宝
2. 访问网站<https://doc.open.alipay.com/>
3. 

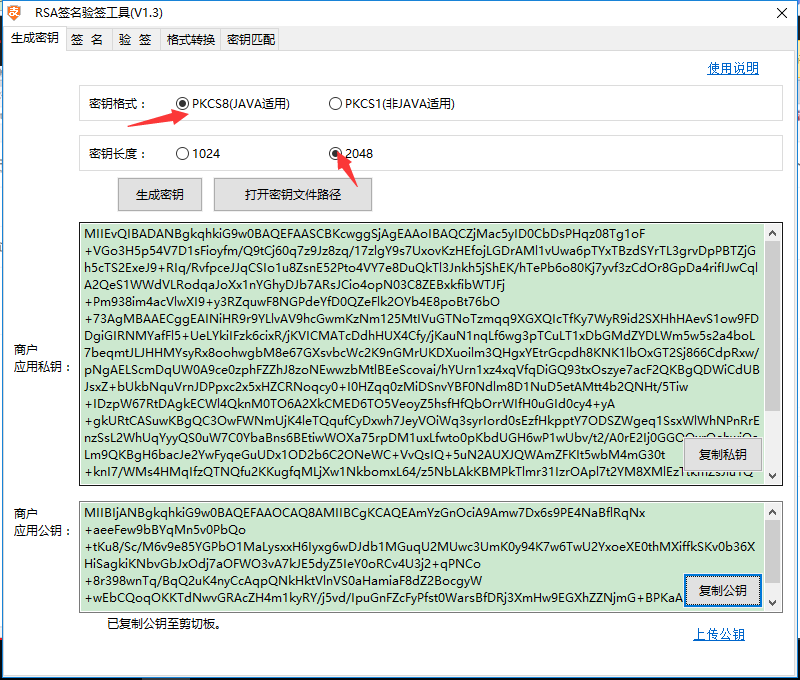
点击下载demo

1. 解压后用idea打开
2.  
3. 

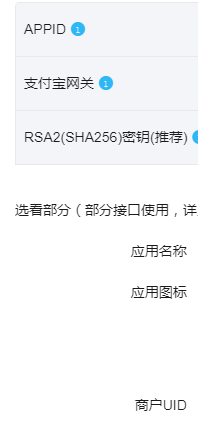
将此项目中WEB-INF下的lib下的包，除了两个source包，都导入

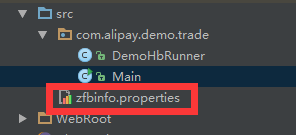
1. 下载RSA签名工具<https://doc.open.alipay.com/docs/doc.htm?treeId=291&articleId=105971&docType=1>
2. 下载完点击此处进行运行





点击生成密钥





配置此文件

open\_api\_domain对应支付宝网关

pid对应商户UID

appid对应APPID

私钥对应生成的私钥

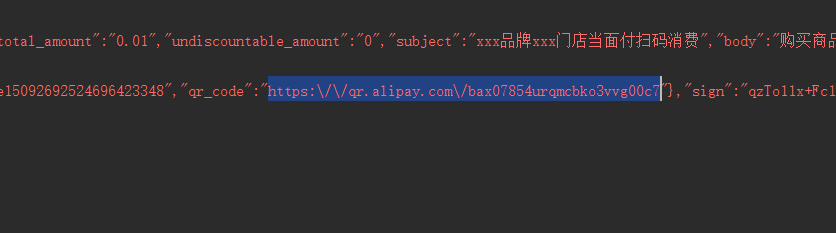
公钥对应生成的公钥

支付宝公钥配置：



点击修改，将前面复制的公钥复制过来，保存，然后点击查看支付宝公钥，复制。

运行main方法进行测试。



复制生成的二维码，用二维码生成器生成二维码。

用android手机下载测试支付宝app，使用沙箱账号中的测试账号进行测试。

