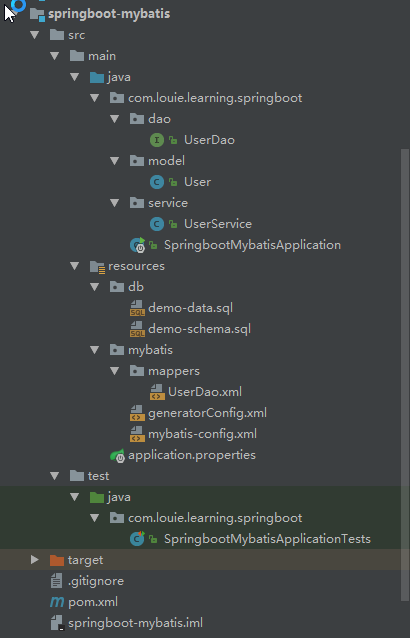
# 完整项目结构



# maven pom.xml

<?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.louie.learning.springboot</groupId>  
 <artifactId>springboot-mybatis</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>springboot-mybatis</name>  
 <description>Demo project for Spring Boot</description>  
  
 <!-- Spring Boot 启动父依赖 -->  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>1.5.6.RELEASE</version>  
 </parent>  
  
 <properties>  
 <!-- 文件拷贝时的编码 -->  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <java.version>1.8</java.version>  
 </properties>  
  
 <dependencies>  
 <!-- Spring Boot Mybatis 依赖 -->  
 <dependency>  
 <groupId>org.mybatis.spring.boot</groupId>  
 <artifactId>mybatis-spring-boot-starter</artifactId>  
 <version>1.3.1</version>  
 </dependency>  
  
 <!-- Spring Boot Test 依赖 -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- Spring Boot 热部署 -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <optional>true</optional>  
 </dependency>  
  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 <plugin>  
 <groupId>org.mybatis.generator</groupId>  
 <artifactId>mybatis-generator-maven-plugin</artifactId>  
 <version>1.3.5</version>  
 <configuration>  
 <configurationFile>src/main/resources/mybatis/generatorConfig.xml</configurationFile>  
 <verbose>true</verbose>  
 <overwrite>true</overwrite>  
 </configuration>  
 <executions>  
 <execution>  
 <id>Generate MyBatis Artifacts</id>  
 <goals>  
 <goal>generate</goal>  
 </goals>  
 </execution>  
 </executions>  
 <dependencies>  
 <dependency>  
 <groupId>org.mybatis.generator</groupId>  
 <artifactId>mybatis-generator-core</artifactId>  
 <version>1.3.5</version>  
 </dependency>  
 </dependencies>  
 </plugin>  
 </plugins>  
 </build>  
</project>

# application.properties

# 数据源配置  
spring.datasource.driver-class-name=org.h2.Driver  
spring.datasource.url=jdbc:h2:file:D:/WorkSpaceIDEA/springboot-learning-demo/db/demo  
spring.datasource.username=admin  
spring.datasource.password=admin  
  
# Mybatis 配置  
mybatis.typeAliasesPackage=com.louie.learning.springboot.model  
mybatis.mapperLocations=classpath:mybatis/mappers/\*.xml  
mybatis.configLocation=classpath:mybatis/mybatis-config.xml  
  
#数据初始化配置  
#进行该配置后，每次启动程序，程序都会运行resources/db/schema.sql文件，对数据库的结构进行操作。  
spring.datasource.schema=classpath:db/demo-schema.sql  
#进行该配置后，每次启动程序，程序都会运行resources/db/data.sql文件，对数据库的数据操作。  
spring.datasource.data=classpath:db/demo-data.sql

# mybatis-config.xml

这个文件基本上可以不需要

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE configuration  
 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-config.dtd">  
<configuration>  
 <!--<settings>-->  
 <!--&lt;!&ndash; 使全局的映射器启用或禁用缓存。 &ndash;&gt;-->  
 <!--<setting name="cacheEnabled" value="true" />-->  
 <!--&lt;!&ndash; 全局启用或禁用延迟加载。当禁用时，所有关联对象都会即时加载。 &ndash;&gt;-->  
 <!--<setting name="lazyLoadingEnabled" value="true" />-->  
 <!--&lt;!&ndash; 当启用时，有延迟加载属性的对象在被调用时将会完全加载任意属性。否则，每种属性将会按需要加载。 &ndash;&gt;-->  
 <!--<setting name="aggressiveLazyLoading" value="true"/>-->  
 <!--&lt;!&ndash; 是否允许单条sql 返回多个数据集 (取决于驱动的兼容性) default:true &ndash;&gt;-->  
 <!--<setting name="multipleResultSetsEnabled" value="true" />-->  
 <!--&lt;!&ndash; 是否可以使用列的别名 (取决于驱动的兼容性) default:true &ndash;&gt;-->  
 <!--<setting name="useColumnLabel" value="true" />-->  
 <!--&lt;!&ndash; 允许JDBC 生成主键。需要驱动器支持。如果设为了true，这个设置将强制使用被生成的主键，有一些驱动器不兼容不过仍然可以执行。 default:false &ndash;&gt;-->  
 <!--<setting name="useGeneratedKeys" value="false" />-->  
 <!--&lt;!&ndash; 指定 MyBatis 如何自动映射 数据基表的列 NONE：不隐射　PARTIAL:部分 FULL:全部 &ndash;&gt;-->  
 <!--<setting name="autoMappingBehavior" value="PARTIAL" />-->  
 <!--&lt;!&ndash; 这是默认的执行类型 （SIMPLE: 简单； REUSE: 执行器可能重复使用prepared statements语句；BATCH: 执行器可以重复执行语句和批量更新） &ndash;&gt;-->  
 <!--<setting name="defaultExecutorType" value="SIMPLE" />-->  
  
 <!--<setting name="defaultStatementTimeout" value="25" />-->  
  
 <!--<setting name="defaultFetchSize" value="100" />-->  
  
 <!--<setting name="safeRowBoundsEnabled" value="false" />-->  
 <!--&lt;!&ndash; 使用驼峰命名法转换字段。 &ndash;&gt;-->  
 <!--<setting name="mapUnderscoreToCamelCase" value="true" />-->  
 <!--&lt;!&ndash; 设置本地缓存范围 session:就会有数据的共享 statement:语句范围 (这样就不会有数据的共享 ) defalut:session &ndash;&gt;-->  
 <!--<setting name="localCacheScope" value="SESSION" />-->  
 <!--&lt;!&ndash; 默认为OTHER,为了解决oracle插入null报错的问题要设置为NULL &ndash;&gt;-->  
 <!--<setting name="jdbcTypeForNull" value="NULL" />-->  
 <!--<setting name="lazyLoadTriggerMethods" value="equals,clone,hashCode,toString" />-->  
  
 <!--指定MyBatis的日志实现使用。如果此设置是不存在的记录的实施将自动查找。-->  
 <!--<setting name="logImpl" value="LOG4J2"/>-->  
  
 <!--</settings>-->  
 <typeAliases>  
 <typeAlias alias="Integer" type="java.lang.Integer" />  
 <typeAlias alias="Long" type="java.lang.Long" />  
 <typeAlias alias="HashMap" type="java.util.HashMap" />  
 <typeAlias alias="LinkedHashMap" type="java.util.LinkedHashMap" />  
 <typeAlias alias="ArrayList" type="java.util.ArrayList" />  
 <typeAlias alias="LinkedList" type="java.util.LinkedList" />  
 <!-- 根据包取别名，把包下面的所有类都按类名来取别名 -->  
 <!-- 这用可以简化代码量 -->  
 <package name="com.louie.learning.springboot.model"/>  
 </typeAliases>  
</configuration>

# demo-schema.sql

-- ----------------------------  
-- Table structure for t\_user  
-- ----------------------------  
**DROP TABLE** IF **EXISTS** t\_user;  
**CREATE TABLE** t\_user (  
 id **varchar**(36) **NOT NULL PRIMARY KEY**,  
 user\_cd **varchar**(50) **default NULL** COMMENT '用户编号',  
 user\_name **varchar**(255) **default NULL** COMMENT '用户名',  
 user\_phone **varchar**(20) **default NULL** COMMENT '手机号',  
 user\_email **varchar**(255) **default NULL** COMMENT '邮箱地址',  
 user\_pwd **varchar**(32) **default NULL** COMMENT '用户密码',  
 create\_time datetime **default NULL** COMMENT '创建时间',  
 modify\_time datetime **default NULL** COMMENT '最后修改时间',  
 is\_delete tinyint(4) **default NULL** COMMENT '是否删除，0-未删除；1-已删除'  
) ;

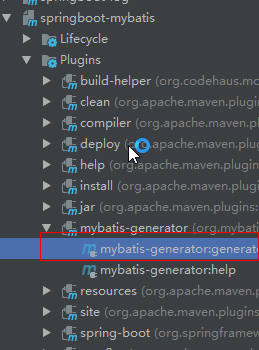
# demo-data.sql

-- ----------------------------  
-- Records of t\_user  
-- ----------------------------  
**INSERT INTO** t\_user **VALUES** ('1', '111111', '赵大宝', '13285250574', '1045221654@qq.com', '96e79218965eb72c92a549dd5a330112', '2016-07-15 23:38:56', '2016-07-15 23:39:09', '0');  
**INSERT INTO** t\_user **VALUES** ('2', '222222', '张三丰', '15985250574', '1198224554@qq.com', '96e79218965eb72c92a549dd5a330112', '2016-07-15 23:39:01', '2016-07-15 23:39:13', '0');  
**INSERT INTO** t\_user **VALUES** ('3', '333333', '王尼玛', '13685250574', '1256221654@qq.com', '96e79218965eb72c92a549dd5a330112', '2016-07-15 23:39:05', '2016-07-15 23:39:16', '0');

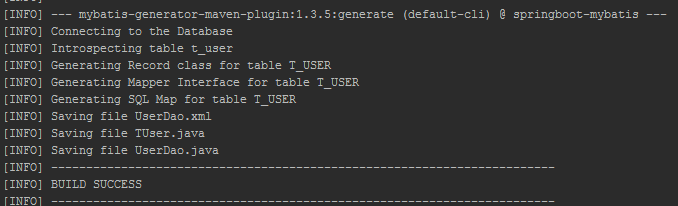
# generatorConfig.xml

通过配置这个文件，用来根据数据库表生成实体、dao接口和接口对应的sql操作的Mapper.xml文件。

<?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="application.properties"/>  
  
 <!--指定特定数据库的jdbc驱动jar包的位置 -->  
 <classPathEntry location="D:\.m2\repository\com\h2database\h2\1.4.194\h2-1.4.194.jar"/>  
  
 <context id="default" targetRuntime="MyBatis3">  
  
 <!-- optional，旨在创建class时，对注释进行控制 -->  
 <commentGenerator>  
 <!-- 是否去除自动生成的注释 true：是 ： false:否 -->  
 <property name="suppressDate" value="true"/>  
 <property name="suppressAllComments" value="true" />  
 </commentGenerator>  
  
 <!--jdbc的数据库连接 -->  
 <jdbcConnection driverClass="${spring.datasource.driver-class-name}"  
 connectionURL="${spring.datasource.url}"  
 userId="${spring.datasource.username}"  
 password="${spring.datasource.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>  
 <!-- 默认false，把JDBC DECIMAL 和 NUMERIC 类型解析为 Integer，  
 为 true时把JDBC DECIMAL 和NUMERIC 类型解析为java.math.BigDecimal -->  
 <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.louie.learning.springboot.model" targetProject="src/main/java">  
 <!--&lt;!&ndash; 是否对model添加 构造函数 &ndash;&gt;-->  
 <property name="constructorBased" value="false"/>  
  
 <!-- 是否允许子包，即targetPackage.schemaName.tableName -->  
 <property name="enableSubPackages" value="false"/>  
  
 <!-- 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"/>  
  
 <!--&lt;!&ndash; 给Model添加一个父类 &ndash;&gt;-->  
 <!--<property name="rootClass" value="com.foo.louis.Hello"/>-->  
  
 <!-- 是否对类CHAR类型的列的数据进行trim操作 -->  
 <property name="trimStrings" value="true"/>  
 </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="mybatis.mappers" targetProject="src/main/resources">  
 <!-- enableSubPackages:是否让schema作为包的后缀 -->  
 <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",基于注解的Mapper接口，不会有对应的XML映射文件  
 type="MIXEDMAPPER",XML和注解的混合形式，(上面这种情况中的)SqlProvider注解方法会被XML替代  
 type="XMLMAPPER",所有的方法都在XML中，接口调用依赖XML文件  
 -->  
 <javaClientGenerator targetPackage="com.louie.learning.springboot.dao" targetProject="src/main/java" type="XMLMAPPER">  
 <property name="enableSubPackages" value="false"/>  
 <!--&lt;!&ndash;-->  
 <!--定义Maper.java 源代码中的ByExample() 方法的可视性，可选的值有：-->  
 <!--public;-->  
 <!--private;-->  
 <!--protected;-->  
 <!--default-->  
 <!--注意：如果 targetRuntime="MyBatis3",此参数被忽略-->  
 <!--&ndash;&gt;-->  
 <!--<property name="exampleMethodVisibility" value=""/>-->  
 <!--&lt;!&ndash;-->  
 <!--方法名计数器-->  
 <!--Important note: this property is ignored if the target runtime is MyBatis3.-->  
 <!--&ndash;&gt;-->  
 <!--<property name="methodNameCalculator" value=""/>-->  
  
 <!--&lt;!&ndash;-->  
 <!--为生成的接口添加父接口-->  
 <!--&ndash;&gt;-->  
 <!--<property name="rootInterface" value=""/>-->  
  
 </javaClientGenerator>  
  
  
 <!--<table tableName="sys\_user" mapperName="UserDao" schema="" enableCountByExample="false"-->  
 <!--enableUpdateByExample="false" enableDeleteByExample="false"-->  
 <!--enableSelectByExample="false" selectByExampleQueryId="false">-->  
  
 <!-- optional , only for mybatis3 runtime  
 自动生成的键值（identity,或者序列值）  
 如果指定此元素，MBG将会生成<selectKey>元素，然后将此元素插入到SQL Map的<insert> 元素之中  
 sqlStatement 的语句将会返回新的值  
 如果是一个自增主键的话，你可以使用预定义的语句,或者添加自定义的SQL语句. 预定义的值如下:  
 Cloudscape This will translate to: VALUES IDENTITY\_VAL\_LOCAL()  
 DB2: VALUES IDENTITY\_VAL\_LOCAL()  
 DB2\_MF: SELECT IDENTITY\_VAL\_LOCAL() FROM SYSIBM.SYSDUMMY1  
 Derby: VALUES IDENTITY\_VAL\_LOCAL()  
 HSQLDB: CALL IDENTITY()  
 Informix: select dbinfo('sqlca.sqlerrd1') from systables where tabid=1  
 MySql: SELECT LAST\_INSERT\_ID()  
 SqlServer: SELECT SCOPE\_IDENTITY()  
 SYBASE: SELECT @@IDENTITY  
 JDBC: This will configure MBG to generate code for MyBatis3 suport of JDBC standard generated keys. This is a database independent method of obtaining the value from identity columns.  
 identity: 自增主键 If true, then the column is flagged as an identity column and the generated <selectKey> element will be placed after the insert (for an identity column). If false, then the generated <selectKey> will be placed before the insert (typically for a sequence).  
 -->  
 <!--<generatedKey column="id" sqlStatement="" identity="true" type=""/>-->  
  
  
 <!--&lt;!&ndash; optional.-->  
 <!--列的命名规则：-->  
 <!--MBG使用 <columnRenamingRule> 元素在计算列名的对应 名称之前，先对列名进行重命名，-->  
 <!--作用：一般需要对BUSI\_CLIENT\_NO 前的BUSI\_进行过滤-->  
 <!--支持正在表达式-->  
 <!--searchString 表示要被换掉的字符串-->  
 <!--replaceString 则是要换成的字符串，默认情况下为空字符串，可选-->  
 <!--&ndash;&gt;-->  
 <!--<columnRenamingRule searchString="" replaceString=""/>-->  
  
  
 <!--&lt;!&ndash; optional.告诉 MBG 忽略某一列-->  
 <!--column，需要忽略的列-->  
 <!--delimitedColumnName:true ,匹配column的值和数据库列的名称 大小写完全匹配，false 忽略大小写匹配-->  
 <!--是否限定表的列名，即固定表列在Model中的名称-->  
 <!--&ndash;&gt;-->  
 <!--&lt;!&ndash;<ignoreColumn column="PLAN\_ID" delimitedColumnName="true"/>&ndash;&gt;-->  
  
  
 <!--&lt;!&ndash;optional.覆盖MBG对Model 的生成规则-->  
 <!--column: 数据库的列名-->  
 <!--javaType: 对应的Java数据类型的完全限定名-->  
 <!--在必要的时候可以覆盖由JavaTypeResolver计算得到的java数据类型. For some databases, this is necessary to handle "odd" database types (e.g. MySql's unsigned bigint type should be mapped to java.lang.Object).-->  
 <!--jdbcType:该列的JDBC数据类型(INTEGER, DECIMAL, NUMERIC, VARCHAR, etc.)，该列可以覆盖由JavaTypeResolver计算得到的Jdbc类型，对某些数据库而言，对于处理特定的JDBC 驱动癖好 很有必要(e.g. DB2's LONGVARCHAR type should be mapped to VARCHAR for iBATIS).-->  
 <!--typeHandler:-->  
  
 <!--&ndash;&gt;-->  
 <!--<columnOverride column="" javaType="" jdbcType="" typeHandler="" delimitedColumnName=""/>-->  
  
 <!--</table>-->  
  
 <table schema="" tableName="t\_user" mapperName="UserDao" enableCountByExample="false"  
 enableUpdateByExample="false" enableDeleteByExample="false"  
 enableSelectByExample="false" selectByExampleQueryId="false"></table>  
 </context>  
</generatorConfiguration>



运行mybatis-generator:generate



可以看到生成了三个文件。之后改一下TUser.java的名字User.java

# UserService.java

package com.louie.learning.springboot.service;  
  
import com.louie.learning.springboot.dao.UserDao;  
import com.louie.learning.springboot.model.User;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
@Service  
@Transactional(rollbackFor = Exception.class)  
public class UserService {  
 @Autowired  
 private UserDao userDao;  
  
 public int deleteByPrimaryKey(String id) {  
 return userDao.deleteByPrimaryKey(id);  
 }  
  
 public int insert(User record) {  
 return userDao.insert(record);  
 }  
  
 public int insertSelective(User record) {  
 return userDao.insertSelective(record);  
 }  
  
 public User selectByPrimaryKey(String id) {  
 return userDao.selectByPrimaryKey(id);  
 }  
  
 public int updateByPrimaryKeySelective(User record) {  
 return userDao.updateByPrimaryKeySelective(record);  
 }  
  
 public int updateByPrimaryKey(User record) {  
 return userDao.updateByPrimaryKey(record);  
 }  
}

# SpringbootMybatisApplication.java

package com.louie.learning.springboot;  
  
import org.mybatis.spring.annotation.MapperScan;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.boot.web.servlet.ServletComponentScan;  
import org.springframework.transaction.annotation.EnableTransactionManagement;  
  
*/\*\*  
 \* Spring Boot 应用启动类  
 \*/*@SpringBootApplication// Spring Boot 应用的标识  
@EnableTransactionManagement //如果mybatis中service实现类中加入事务注解，需要此处添加该注解  
@MapperScan("com.louie.learning.springboot.dao")// mapper 接口类扫描包配置  
public class SpringbootMybatisApplication {  
  
 public static void main(String[] args) {  
 // 程序启动入口  
 // 启动嵌入式的 Tomcat 并初始化 Spring 环境及其各 Spring 组件  
 SpringApplication.*run*(SpringbootMybatisApplication.class, args);  
 }  
}

# SpringbootMybatisApplicationTests.java

package com.louie.learning.springboot;  
  
import com.louie.learning.springboot.model.User;  
import com.louie.learning.springboot.service.UserService;  
import org.junit.Assert;  
import org.junit.Test;  
import org.junit.runner.RunWith;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.test.context.SpringBootTest;  
import org.springframework.test.context.junit4.SpringRunner;  
  
import java.util.Date;  
import java.util.UUID;  
  
@RunWith(SpringRunner.class)  
@SpringBootTest  
public class SpringbootMybatisApplicationTests {  
  
 @Autowired  
 private UserService userService;  
  
 @Test  
 public void contextLoads() {  
 //create  
 User user = new User();  
 user.setId(UUID.*randomUUID*().toString().replace("-", ""));  
 user.setCreateTime(new Date());  
 user.setUserCd("admin");  
 user.setUserName("管理员");  
 int result = userService.insertSelective(user);  
 Assert.*assertEquals*(1,result);  
  
 //read  
 user=userService.selectByPrimaryKey("1");  
 Assert.*assertNotNull*(user);  
  
 //update  
 user.setModifyTime(new Date());  
 result=userService.updateByPrimaryKeySelective(user);  
 Assert.*assertEquals*(1,result);  
  
 //delete  
 result=userService.deleteByPrimaryKey(user.getId());  
 Assert.*assertEquals*(1,result);  
  
 }  
}