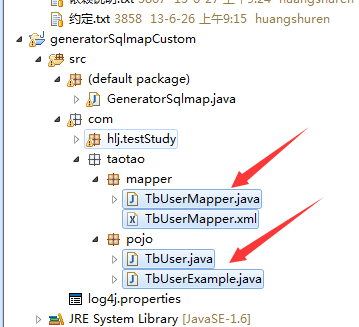
# 1、了解逆向生成代码结构以及作用



## 1、TbUser.java为数据库表tb\_user对应的映射java实体类

|  |
| --- |
| **public** **class** TbUser {  **private** Long id;  **private** String username;  **private** String password;  **private** String phone;  **private** String email;  **private** Date created;  **private** Date updated;  **public** Long getId() {  **return** id;  }  **public** **void** setId(Long id) {  **this**.id = id; |

## 2、TbUserMapper.java 为数据持久层接口类，类似于hibernate中的接口类。里面有自己的方法，可以直接用来使用

|  |
| --- |
| **public** **interface** TbUserMapper {  **int** countByExample(TbUserExample example);  **int** deleteByExample(TbUserExample example);  **int** deleteByPrimaryKey(Long id);  **int** insert(TbUser record);  **int** insertSelective(TbUser record);  List<TbUser> selectByExample(TbUserExample example);  TbUser selectByPrimaryKey(Long id);  **int** updateByExampleSelective(@Param("record") TbUser record, @Param("example") TbUserExample example);  **int** updateByExample(@Param("record") TbUser record, @Param("example") TbUserExample example);  **int** updateByPrimaryKeySelective(TbUser record);  **int** updateByPrimaryKey(TbUser record);  } |

## 3、TbUserMapper.xml ，是实现接口中方法的sql语句，与Mapper接口一一对应

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >  <mapper namespace=*"com.taotao.mapper.TbUserMapper"* >  <resultMap id=*"BaseResultMap"* type=*"com.taotao.pojo.TbUser"* >  <id column=*"id"* property=*"id"* jdbcType=*"BIGINT"* />  <result column=*"username"* property=*"username"* jdbcType=*"VARCHAR"* />  <result column=*"password"* property=*"password"* jdbcType=*"VARCHAR"* />  <result column=*"phone"* property=*"phone"* jdbcType=*"VARCHAR"* />  <result column=*"email"* property=*"email"* jdbcType=*"VARCHAR"* />  <result column=*"created"* property=*"created"* jdbcType=*"TIMESTAMP"* />  <result column=*"updated"* property=*"updated"* jdbcType=*"TIMESTAMP"* />  </resultMap>  <sql id=*"Example\_Where\_Clause"* >  <where >  <foreach collection=*"oredCriteria"* item=*"criteria"* separator=*"or"* >  <if test=*"criteria.valid"* >  <trim prefix=*"("* suffix=*")"* prefixOverrides=*"and"* >  <foreach collection=*"criteria.criteria"* item=*"criterion"* >  <choose >  <when test=*"criterion.noValue"* >  and ${criterion.condition}  </when>  <when test=*"criterion.singleValue"* >  and ${criterion.condition} #{criterion.value}  </when>  <when test=*"criterion.betweenValue"* >  and ${criterion.condition} #{criterion.value} and #{criterion.secondValue}  </when>  <when test=*"criterion.listValue"* >  and ${criterion.condition}  <foreach collection=*"criterion.value"* item=*"listItem"* open=*"("* close=*")"* separator=*","* >  #{listItem}  </foreach>  </when>  </choose>  </foreach>  </trim>  </if>  </foreach>  </where>  </sql>  <sql id=*"Update\_By\_Example\_Where\_Clause"* >  <where >  <foreach collection=*"example.oredCriteria"* item=*"criteria"* separator=*"or"* >  <if test=*"criteria.valid"* >  <trim prefix=*"("* suffix=*")"* prefixOverrides=*"and"* >  <foreach collection=*"criteria.criteria"* item=*"criterion"* >  <choose >  <when test=*"criterion.noValue"* >  and ${criterion.condition}  </when>  <when test=*"criterion.singleValue"* >  and ${criterion.condition} #{criterion.value}  </when>  <when test=*"criterion.betweenValue"* >  and ${criterion.condition} #{criterion.value} and #{criterion.secondValue}  </when>  <when test=*"criterion.listValue"* >  and ${criterion.condition}  <foreach collection=*"criterion.value"* item=*"listItem"* open=*"("* close=*")"* separator=*","* >  #{listItem}  </foreach>  </when>  </choose>  </foreach>  </trim>  </if>  </foreach>  </where>  </sql>  <sql id=*"Base\_Column\_List"* >  id, username, password, phone, email, created, updated  </sql>  <select id=*"selectByExample"* resultMap=*"BaseResultMap"* parameterType=*"com.taotao.pojo.TbUserExample"* >  select  <if test=*"distinct"* >  distinct  </if>  <include refid=*"Base\_Column\_List"* />  from tb\_user  <if test=*"\_parameter != null"* >  <include refid=*"Example\_Where\_Clause"* />  </if>  <if test=*"orderByClause != null"* >  order by ${orderByClause}  </if>  </select>  <select id=*"selectByPrimaryKey"* resultMap=*"BaseResultMap"* parameterType=*"java.lang.Long"* >  select  <include refid=*"Base\_Column\_List"* />  from tb\_user  where id = #{id,jdbcType=BIGINT}  </select>  <delete id=*"deleteByPrimaryKey"* parameterType=*"java.lang.Long"* >  delete from tb\_user  where id = #{id,jdbcType=BIGINT}  </delete>  <delete id=*"deleteByExample"* parameterType=*"com.taotao.pojo.TbUserExample"* >  delete from tb\_user  <if test=*"\_parameter != null"* >  <include refid=*"Example\_Where\_Clause"* />  </if>  </delete>  <insert id=*"insert"* parameterType=*"com.taotao.pojo.TbUser"* >  insert into tb\_user (id, username, password,  phone, email, created,  updated)  values (#{id,jdbcType=BIGINT}, #{username,jdbcType=VARCHAR}, #{password,jdbcType=VARCHAR},  #{phone,jdbcType=VARCHAR}, #{email,jdbcType=VARCHAR}, #{created,jdbcType=TIMESTAMP},  #{updated,jdbcType=TIMESTAMP})  </insert>  <insert id=*"insertSelective"* parameterType=*"com.taotao.pojo.TbUser"* >  insert into tb\_user  <trim prefix=*"("* suffix=*")"* suffixOverrides=*","* >  <if test=*"id != null"* >  id,  </if>  <if test=*"username != null"* >  username,  </if>  <if test=*"password != null"* >  password,  </if>  <if test=*"phone != null"* >  phone,  </if>  <if test=*"email != null"* >  email,  </if>  <if test=*"created != null"* >  created,  </if>  <if test=*"updated != null"* >  updated,  </if>  </trim>  <trim prefix=*"values ("* suffix=*")"* suffixOverrides=*","* >  <if test=*"id != null"* >  #{id,jdbcType=BIGINT},  </if>  <if test=*"username != null"* >  #{username,jdbcType=VARCHAR},  </if>  <if test=*"password != null"* >  #{password,jdbcType=VARCHAR},  </if>  <if test=*"phone != null"* >  #{phone,jdbcType=VARCHAR},  </if>  <if test=*"email != null"* >  #{email,jdbcType=VARCHAR},  </if>  <if test=*"created != null"* >  #{created,jdbcType=TIMESTAMP},  </if>  <if test=*"updated != null"* >  #{updated,jdbcType=TIMESTAMP},  </if>  </trim>  </insert>  <select id=*"countByExample"* parameterType=*"com.taotao.pojo.TbUserExample"* resultType=*"java.lang.Integer"* >  select count(\*) from tb\_user  <if test=*"\_parameter != null"* >  <include refid=*"Example\_Where\_Clause"* />  </if>  </select>  <update id=*"updateByExampleSelective"* parameterType=*"map"* >  update tb\_user  <set >  <if test=*"record.id != null"* >  id = #{record.id,jdbcType=BIGINT},  </if>  <if test=*"record.username != null"* >  username = #{record.username,jdbcType=VARCHAR},  </if>  <if test=*"record.password != null"* >  password = #{record.password,jdbcType=VARCHAR},  </if>  <if test=*"record.phone != null"* >  phone = #{record.phone,jdbcType=VARCHAR},  </if>  <if test=*"record.email != null"* >  email = #{record.email,jdbcType=VARCHAR},  </if>  <if test=*"record.created != null"* >  created = #{record.created,jdbcType=TIMESTAMP},  </if>  <if test=*"record.updated != null"* >  updated = #{record.updated,jdbcType=TIMESTAMP},  </if>  </set>  <if test=*"\_parameter != null"* >  <include refid=*"Update\_By\_Example\_Where\_Clause"* />  </if>  </update>  <update id=*"updateByExample"* parameterType=*"map"* >  update tb\_user  set id = #{record.id,jdbcType=BIGINT},  username = #{record.username,jdbcType=VARCHAR},  password = #{record.password,jdbcType=VARCHAR},  phone = #{record.phone,jdbcType=VARCHAR},  email = #{record.email,jdbcType=VARCHAR},  created = #{record.created,jdbcType=TIMESTAMP},  updated = #{record.updated,jdbcType=TIMESTAMP}  <if test=*"\_parameter != null"* >  <include refid=*"Update\_By\_Example\_Where\_Clause"* />  </if>  </update>  <update id=*"updateByPrimaryKeySelective"* parameterType=*"com.taotao.pojo.TbUser"* >  update tb\_user  <set >  <if test=*"username != null"* >  username = #{username,jdbcType=VARCHAR},  </if>  <if test=*"password != null"* >  password = #{password,jdbcType=VARCHAR},  </if>  <if test=*"phone != null"* >  phone = #{phone,jdbcType=VARCHAR},  </if>  <if test=*"email != null"* >  email = #{email,jdbcType=VARCHAR},  </if>  <if test=*"created != null"* >  created = #{created,jdbcType=TIMESTAMP},  </if>  <if test=*"updated != null"* >  updated = #{updated,jdbcType=TIMESTAMP},  </if>  </set>  where id = #{id,jdbcType=BIGINT}  </update>  <update id=*"updateByPrimaryKey"* parameterType=*"com.taotao.pojo.TbUser"* >  update tb\_user  set username = #{username,jdbcType=VARCHAR},  password = #{password,jdbcType=VARCHAR},  phone = #{phone,jdbcType=VARCHAR},  email = #{email,jdbcType=VARCHAR},  created = #{created,jdbcType=TIMESTAMP},  updated = #{updated,jdbcType=TIMESTAMP}  where id = #{id,jdbcType=BIGINT}  </update>  </mapper> |

## 4、TbUserExample.java ，主要是提供给mapper接口使用的类似于实体类这样的东西，但是它里面有各种的sql连接关系。

### 解释：文件太多，非常重要，建议自己点开观察

# 2、详解TbUserExample，很多以username这个字段举例说明

## 解释：

### 1、Example内有一个成员叫oredCriteria,是Criteria的集合,就想其名字所预示的一样，这个集合中的Criteria是由OR连接的，是逻辑或关系。oredCriteria就是ORed Criteria。

#### 个人举例

|  |
| --- |
| /\*\*  \* or用法  \*/  TbUserExample.Criteria criteria3 = example.createCriteria();  criteria3.andUsernameEqualTo("joe");  criteria3.andPasswordIsNull();    TbUserExample.Criteria criteria4 = example.createCriteria();  criteria4.andEmailEqualTo("mxzdhealer@gmail.com");  criteria4.andPhoneIsNull();  example.or(criteria4);    //出现 List<TbUser>list = TbUserMapper.selectByExample(example);  //相当于 select \* from TbUser where (username = 'joe' and password is null) or (email = 'mxzdhealer@gmail.com' and phone is null) |

### 2、Criteria Criteria包含一个Cretiron的集合,每一个Criteria对象内包含的Cretiron之间是由AND连接的,是逻辑与的关系。oredCriteria

|  |
| --- |
| /\*\*  \* and用法  \*/  TbUserExample.Criteria criteria2 = example.createCriteria();  criteria2.andUsernameEqualTo("joe");  criteria2.andPasswordIsNull();  example.setOrderByClause("username asc,email desc");  //出现 List<TbUser>list = TbUserMapper.selectByExample(example);  //相当于：select \* from TbUser where username = 'joe' and password is null order |

## 1、TbUserExample 中的用法

### 1、排序 orderByClause

|  |
| --- |
| /\*\*  example.setOrderByClause("字段名 ASC");  解释： 升序排列，desc为降序排列。  \*/  **protected** String orderByClause; |

### 2、重复、唯一 distinct

|  |
| --- |
| /\*\*  example.setDistinct(false)  解释：去除重复，boolean型，true为选择不重复的记录。  \*/  **protected** **boolean** distinct; |

### 3、IsNull：username是否为null

|  |
| --- |
| **public** Criteria andUsernameIsNull() {  // addCriterion("username is null");  **return** **null**;  }    **public** Criteria andUsernameIsNotNull() {  // addCriterion("username is not null");  **return** **null**;  } |

### 4、EqualTo：是否等于 value 参数

|  |
| --- |
| /\*\*  \* 2.  \* 是否等于 value 参数  \*/  **public** Criteria andUsernameEqualTo(String value) {  // addCriterion("username =", value, "username");  **return** **null**;  }    **public** Criteria andUsernameNotEqualTo(String value) {  // addCriterion("username <>", value, "username");  **return** **null**;  } |

### 5、GreaterThan：LessThan：是否大于、大于等于、小于、小于等于 value 参数

|  |
| --- |
| **public** Criteria andUsernameGreaterThan(String value) {  // addCriterion("username >", value, "username");  **return** **null**;  }  **public** Criteria andUsernameGreaterThanOrEqualTo(String value) {  // addCriterion("username >=", value, "username");  **return** **null**;  }  **public** Criteria andUsernameLessThan(String value) {  // addCriterion("username <", value, "username");  **return** **null**;  }    **public** Criteria andUsernameLessThanOrEqualTo(String value) {  // addCriterion("username <=", value, "username");  **return** **null**;  } |

### 6、like 匹配

|  |
| --- |
| /\*\*  4. like 匹配  \*/  **public** Criteria andUsernameLike(String value) {  // addCriterion("username like", value, "username");  **return** **null**;  }    **public** Criteria andUsernameNotLike(String value) {  // addCriterion("username not like", value, "username");  **return** **null**;  } |

### 7、in 匹配

|  |
| --- |
| /\*\*  5. in 匹配  \*/  **public** Criteria andUsernameIn(List<String> values) {  // addCriterion("username in", values, "username");  **return** **null**;  }  **public** Criteria andUsernameNotIn(List<String> values) {  // addCriterion("username not in", values, "username");  **return** **null**;  } |

### 8、between 匹配

|  |
| --- |
| /\*\*  5. between 匹配  \*/  **public** Criteria andUsernameBetween(String value1, String value2) {  // addCriterion("username between", value1, value2, "username");  **return** **null**;  }    **public** Criteria andUsernameNotBetween(String value1, String value2) {  // addCriterion("username not between", value1, value2, "username");  **return** **null**;  } |

# 3、详解Mapper接口方法的使用

## 1、int countByExample(TbUserExample example);

|  |
| --- |
| /\*\*  //下面是一个完整的案列  TbUserExample example = new TbUserExample();  TbUserExample.Criteria criteria = example.createCriteria();  criteria.andUsernameEqualTo("joe");  int count = tbUserMapper.countByExample(example);  相当于：select count(\*) from user where username='joe'  \*/ |

## 2、int insert(TbUser record);

|  |
| --- |
| /\*\*  TbUser record=new TbUser();  record.setName("sansan");  // 使用所有的属性作为字段使用  this.tbUserMapper.insert(record);  相当于 INSERT INTO tb\_user (AGE,USER\_NAME,ID,NAME,BIRTHDAY,SEX,PASSWORD,UPDATED,CREATED) VALUES ( ?,?,?,?,?,?,?,?,? )  \*/ |

## 3、int insertSelective(TbUser record);

|  |
| --- |
| /\*\*  TbUser record=new TbUser();  record.setName("sansan");  // 使用所有的属性作为字段使用  this.tbUserMapper.insert(record);  INSERT INTO tb\_user ( ID,NAME ) VALUES ( ?,? )  \*/ |

## 4、TbUser selectByPrimaryKey(Long id);

|  |
| --- |
| /\*\*  \* Mapper.selectByPrimaryKey(100);  \* 相当于select \* from user where id = 100  \* **@param** id  \* **@return**  \*/ |

## 5、List<TbUser> selectByExample(TbUserExample example);

|  |
| --- |
| /\*\*  \* List<TbUser> selectByExample(TbUserExample example);  \*  语句逻辑上存在问题，但是语句是正确的  serExample example = new UserExample();  TbUserExample.Criteria criteria = example.createCriteria();  criteria.andUsernameEqualTo("joe");  criteria.andUsernameIsNull();  example.setOrderByClause("username asc,email desc");  List<?>list = ##tbUserMapper.selectByExample(example);  相当于：select \* from user where username = 'joe' and username is null order    \*/ |

## 6、int updateByExampleSelective(@Param("record") TbUser record, @Param("example") TbUserExample example);

|  |
| --- |
| 第一个参数 是要修改的部分值组成的对象，其中有些属性为null则表示该项不修改。 第二个参数 是一个对应的查询条件的类， 通过这个类可以实现 order by 和一部分的where 条件。 |

### 解释：对应xml中传入的参数为map

<update id=*"updateByExample"* parameterType=*"map"* >

update CSIP\_BENEFICIARY

## 7、int updateByPrimaryKey(TbUser record);

|  |
| --- |
| /\*\*  \* updateByPrimaryKey对你注入的字段全部更新 ，根据主键  \*/  **int** updateByPrimaryKey(TbUser record); |

# 4、如果是自己想写sql语句，也可以先写好接口方法TbUserMapper.java ，再去xml接口配置文件TbUserMapper.xml中写相应的sql语句

# 5、利用注解执行sql语句，类似于hibernate

## 1、@insert

|  |
| --- |
| @Insert({  "insert into CSIP\_BENEFICIARY (PK\_BENEFICIARY, PK\_PERSON, ",  "NAME, IDTYPE, IDNO, ",  "BENESSID, BENEBANK, ",  "BENEBANKID, BENEACCNAME, ",  "BENEACC, BENERELATION, ",  "BENEORDER, BENERATE, ",  "MEMO, TS, BENEBANKPROV, ",  "BENEBANKCITY)",  "values (#{pkBeneficiary,jdbcType=CHAR}, #{pkPerson,jdbcType=CHAR}, ",  "#{name,jdbcType=VARCHAR}, #{idtype,jdbcType=VARCHAR}, #{idno,jdbcType=VARCHAR}, ",  "#{benessid,jdbcType=VARCHAR}, #{benebank,jdbcType=VARCHAR}, ",  "#{benebankid,jdbcType=VARCHAR}, #{beneaccname,jdbcType=VARCHAR}, ",  "#{beneacc,jdbcType=VARCHAR}, #{benerelation,jdbcType=VARCHAR}, ",  "#{beneorder,jdbcType=VARCHAR}, #{benerate,jdbcType=DECIMAL}, ",  "#{memo,jdbcType=VARCHAR}, #{ts,jdbcType=TIMESTAMP}, #{benebankprov,jdbcType=VARCHAR}, ",  "#{benebankcity,jdbcType=VARCHAR})"  })  **int** insert(BeneficiaryVO record); |

## 2、@delete

|  |
| --- |
| @Delete({  "delete from CSIP\_BENEFICIARY",  "where PK\_BENEFICIARY = #{pkBeneficiary,jdbcType=CHAR}"  })  **int** deleteByPrimaryKey(String pkBeneficiary); |

## 3、@update

|  |
| --- |
| @Update({  "update CSIP\_BENEFICIARY",  "set PK\_PERSON = #{pkPerson,jdbcType=CHAR},",  "NAME = #{name,jdbcType=VARCHAR},",  "IDTYPE = #{idtype,jdbcType=VARCHAR},",  "IDNO = #{idno,jdbcType=VARCHAR},",  "BENESSID = #{benessid,jdbcType=VARCHAR},",  "BENEBANK = #{benebank,jdbcType=VARCHAR},",  "BENEBANKID = #{benebankid,jdbcType=VARCHAR},",  "BENEACCNAME = #{beneaccname,jdbcType=VARCHAR},",  "BENEACC = #{beneacc,jdbcType=VARCHAR},",  "BENERELATION = #{benerelation,jdbcType=VARCHAR},",  "BENEORDER = #{beneorder,jdbcType=VARCHAR},",  "BENERATE = #{benerate,jdbcType=DECIMAL},",  "MEMO = #{memo,jdbcType=VARCHAR},",  "TS = #{ts,jdbcType=TIMESTAMP},",  "BENEBANKPROV = #{benebankprov,jdbcType=VARCHAR},",  "BENEBANKCITY = #{benebankcity,jdbcType=VARCHAR}",  "where PK\_BENEFICIARY = #{pkBeneficiary,jdbcType=CHAR}"  })  **int** updateByPrimaryKey(BeneficiaryVO record); |

## 4、@selete @ResultMap，BaseResultMap 在mapper接口对应的xml文件中有说明

|  |
| --- |
| @Select({  "select",  "PK\_BENEFICIARY, PK\_PERSON, NAME, IDTYPE, IDNO, BENESSID, BENEBANK, BENEBANKID, ",  "BENEACCNAME, BENEACC, BENERELATION, BENEORDER, BENERATE, MEMO, TS, BENEBANKPROV, ",  "BENEBANKCITY",  "from CSIP\_BENEFICIARY",  "where PK\_BENEFICIARY = #{pkBeneficiary,jdbcType=CHAR}"  })  @ResultMap("BaseResultMap")  BeneficiaryVO selectByPrimaryKey(String pkBeneficiary);  @Select({  " select count(cb.pk\_bene) from csip\_bene cb ",  " join csip\_taskfile ct on ct.pk\_taskfile = cb.pk\_taskfile ",  " join csip\_payapplytask cpt on ct.pk\_task=cpt.pk\_payapplytask ",  " join csip\_payapplyinfo cpi on cpt.pk\_payapplyinfo=cpi.pk\_payapplyinfo ",  " where cb.pk\_paysetlist=#{pk\_person,jdbcType=VARCHAR} " ,  " and cpi.applyno=#{applyNo,jdbcType=VARCHAR} ",  " and cpt.bill\_status = #{billStatus,jdbcType=VARCHAR} ",  })  **int** queryByPKSetlistAndApplyNo(@Param("applyNo")String applyNo,  @Param("pk\_person")String pk\_person,@Param("billStatus")String billStatus); |

# 6、其他注解

## 1、@SelectKey

SelectKey在Mybatis中是为了解决Insert数据时不支持主键自动生成的问题，他可以很随意的设置生成主键的方式。

不管SelectKey有多好，尽量不要遇到这种情况吧，毕竟很麻烦。

[**http://blog.csdn.net/isea533/article/details/21153791**](http://blog.csdn.net/isea533/article/details/21153791)