兮克

Java EE框架 ---MyBatis

Java EE framework --MyBatis

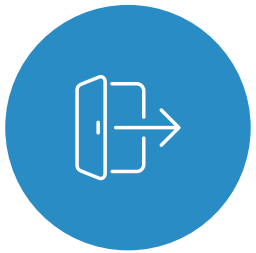
王磊

部门：研发部

CONTENTS

新建java项目

植入log4j文件



创建generatorConfig.xml

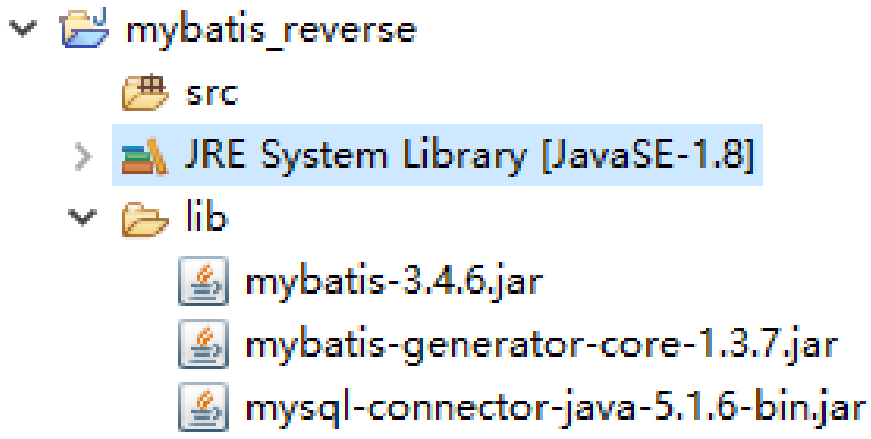
创建Generator类

Mybatis使用逆向工程

需求：使用Mybatis逆向工程自动生成model和mapper映射接口。

Mybatis属于半自动ORM，可以利用mybatis工具generatorConfig.xml自动生成DAO、实体、映射文件的方式来代替手动书写的方式，这样既提高了工作效率也可以在项目避免出现的一些细微难调试的BUG。

1、新建java项目mybatis_reverse
2、引入jar包
(1)mybatis-3.4.6.jar是基础jar包，使用mybatis操作数据库必须引用
(2)mybatis-generator-core-1.3.7.jar包，使用mybatis进行逆向工程必须使用
(3) mysql-connector-java-5.1.6-bin.jar，使用MySQL数据必须使用



Mybatis如何使用逆向工程

2、创建包

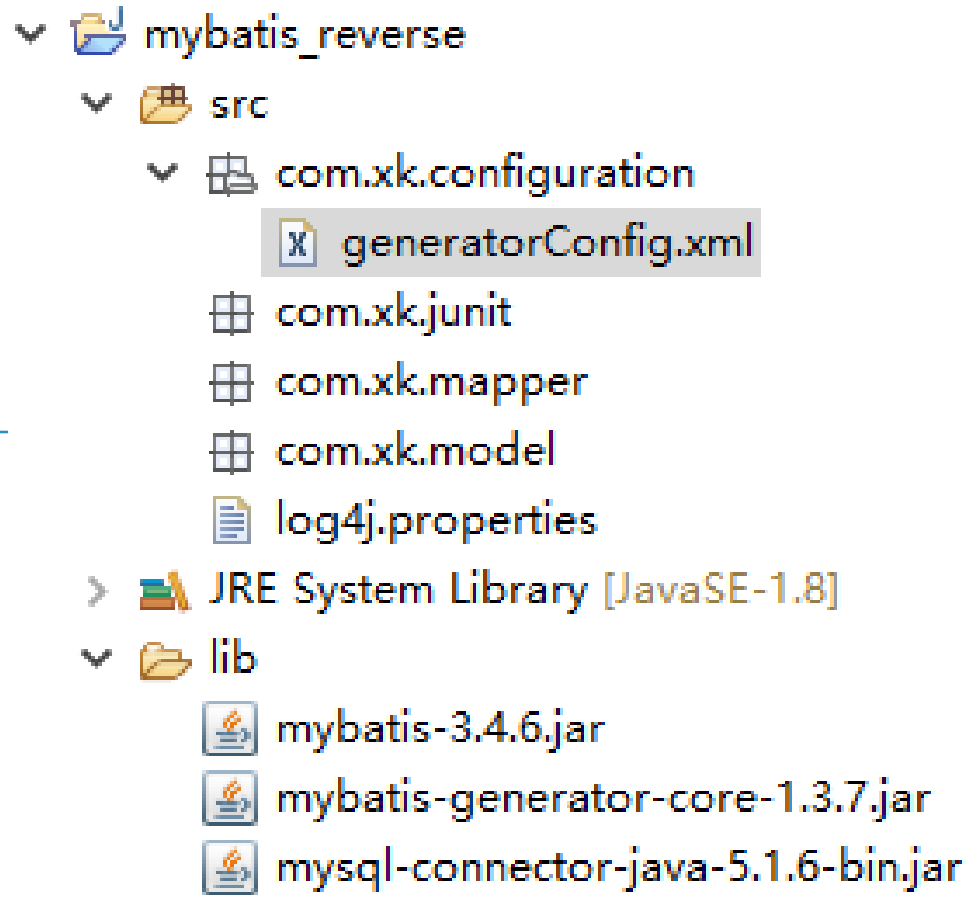
com.xk.configuration

com.xk.junit

com.xk.mapper

com.xk.model

3、植入log4j文件



Mybatis如何使用逆向工程

4、在com.xk.configuration包之下，创建generatorConfig.xml配置文件

```
<generatorConfiguration>
    <context id="testTables" targetRuntime="MyBatis3">
        <commentGenerator>
            <!-- 是否去除自动生成的注释 true：是 ： false:否 -->
            <property name="suppressAllComments" value="true" />
        </commentGenerator>
        <!--数据库连接的信息：驱动类、连接地址、用户名、密码 -->
        <jdbcConnection driverClass="com.mysql.jdbc.Driver"
            connectionURL="jdbc:mysql://localhost:3306/usermanager"
            userId="root"
            password="root">
        </jdbcConnection>
        <!-- <jdbcConnection driverClass="oracle.jdbc.OracleDriver"
              connectionURL="jdbc:oracle:thin:@127.0.0.1:1521:yycg"
              userId="xk" password="123456"> </jdbcConnection> -->
        <!-- 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 -->
        <javaTypeResolver>
            <property name="forceBigDecimals" value="false" />
        </javaTypeResolver>
```

Mybatis如何使用逆向工程

4、在com.xk.configuration包之下，创建generatorConfig.xml配置文件

```
        <!-- targetProject:生成PO类的位置 -->
        <javaModelGenerator targetPackage="com.xk.model"
            targetProject=".\src">
            <!-- enableSubPackages:是否让schema作为包的后缀 -->
            <property name="enableSubPackages" value="false" />
            <!-- 从数据库返回的值被清理前后的空格 -->
            <property name="trimStrings" value="true" />
        </javaModelGenerator>
        <!-- targetProject:mapper映射文件生成的位置 -->
        <sqlMapGenerator targetPackage="com.xk.mapper"
            targetProject=".\src">
            <!-- enableSubPackages:是否让schema作为包的后缀 -->
            <property name="enableSubPackages" value="false" />
        </sqlMapGenerator>
        <!-- targetPackage：mapper接口生成的位置 -->
        <javaClientGenerator type="XMLMAPPER"
            targetPackage="com.xk.mapper" targetProject=".\src">
            <!-- enableSubPackages:是否让schema作为包的后缀 -->
            <property name="enableSubPackages" value="false" />
        </javaClientGenerator>
        <!-- 指定数据库表 -->
        <table tableName="tuserlogin"></table>
    </context>
</generatorConfiguration>
```

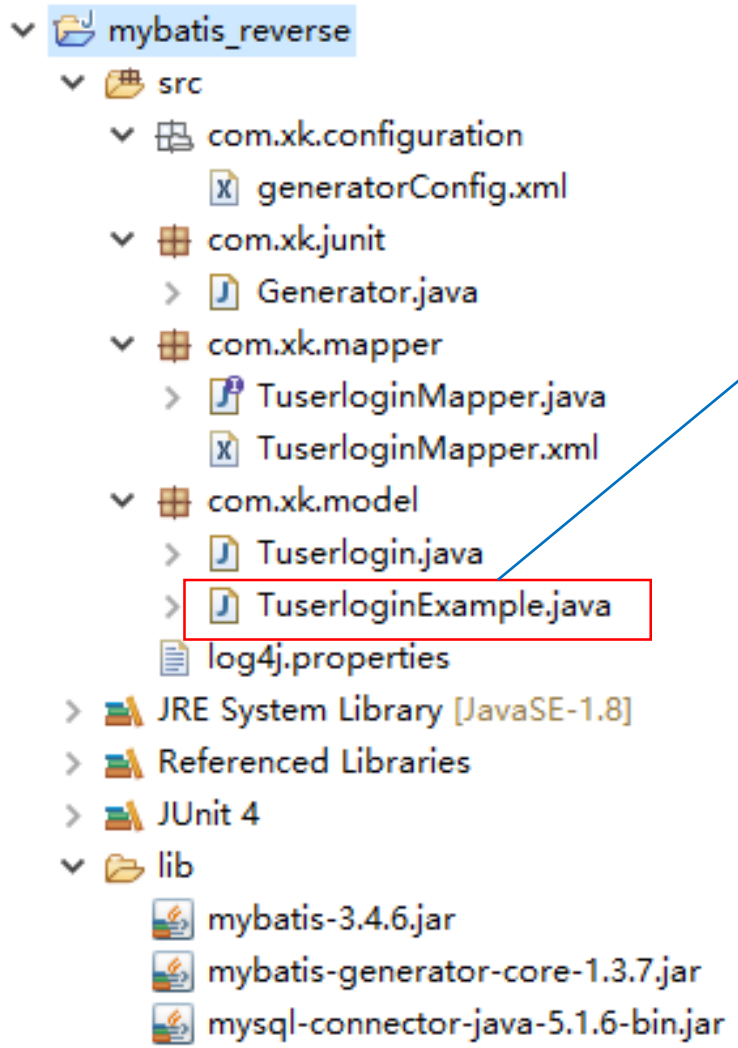
Mybatis如何使用逆向工程

4、在com.xk.junit包之下，创建Generator类，负责生成model类和mapper接口及配置文件。

```
package com.xk.junit;
import java.io.File;
import java.util.ArrayList;
import java.util.List;
import org.junit.Test;
import org.mybatis.generator.api.MyBatisGenerator;
import org.mybatis.generator.config.Configuration;
import org.mybatis.generator.config.xml.ConfigurationParser;
import org.mybatis.generator.internal.DefaultShellCallback;
public class Generator {
    @Test
    public void create() throws Exception{
        List<String> warnings = new ArrayList<String>();
        boolean overwrite = true;
        File configFile = new File("./src/com/xk/configuration/generatorConfig.xml");
        ConfigurationParser cp = new ConfigurationParser(warnings);
        Configuration config = cp.parseConfiguration(configFile);
        DefaultShellCallback callback = new DefaultShellCallback(overwrite);
        MyBatisGenerator myBatisGenerator = new MyBatisGenerator(config, callback, warnings);
        myBatisGenerator.generate(null);
    }
}
```

Mybatis如何使用逆向工程

5、运行Generat0r类中的create方法，进行逆向工程，刷新src文件夹，查看运行结果。



生成的持久化对象中，多了一个xxxExample.java类，这个类是用来[构造复杂的筛选条件]，通俗点讲就是专门用来封装自定义查询条件。

Mybatis如何使用逆向工程

6、测试，将逆向工程所生成的mapper相关的类和接口等内容加入到项目中。

(1)加入sqlMapConfig.xml

(2)增加com.xk.util工具包，加入MybatisUtil.java测试类

```
package com.xk.util;
import java.io.IOException;
public class MybatisUtil {
    public static SqlSessionFactory getSqlSessionFactory(){
        String resource = "sqlMapConfig.xml";
        SqlSessionFactory  sqlSessionFactory = null;
        InputStream in;
        try {
            in = Resources.getResourceAsStream(resource);
             sqlSessionFactory = new SqlSessionFactoryBuilder().build(in);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return sqlSessionFactory;
    }
    public static SqlSession getSqlSession(){
        return getSqlSessionFactory().openSession();
    }
    //true表示SqlSession对象在执行完SQL后自动提交事务，false不会自动提交事务，需要手动调用sqlSession.commit()
    public static SqlSession getSqlSession(boolean autoCommit) throws IOException {
        return getSqlSessionFactory().openSession(autoCommit);
    }
}
```

Mybatis如何使用逆向工程

6、测试，将逆向工程所生成的mapper相关的类和接口等内容加入到项目中。

(3)在com.xk.junit包下创建测试类。

```
package com.xk.junit;
import java.io.IOException;
public class MybatisMapperTest {
    @Test
    public void testGetUser() throws IOException {
        SqlSession sqlSession = MybatisUtil.getSqlSession(true);
        TuserloginMapper userMapper = sqlSession.getMapper(TuserloginMapper.class);
        Tuserlogin user = userMapper.selectByPrimaryKey(15);
        System.out.println(user.getUsername()+" "+user.getEmail());
    }
    @Test
    public  void testGetAllUser() throws IOException{
        SqlSession sqlSession = MybatisUtil.getSqlSession(true);
        TuserloginMapper userMapper = sqlSession.getMapper(TuserloginMapper.class);
        TuserloginExample example = new TuserloginExample();
        List<Tuserlogin> users = userMapper.selectByExample(example);
        System.out.println(users);
    }
    @Test
    public void testDeleteUser() throws IOException {
        SqlSession sqlSession = MybatisUtil.getSqlSession(true);
        TuserloginMapper userMapper = sqlSession.getMapper(TuserloginMapper.class);
        userMapper.deleteByPrimaryKey(16);
    }
}
```