就是根据数据库的表自动生成**Mybatis**所需要的配置文件及实体类，接口文件

需要：

1、配置文件

generatorConfig-base.xml

```
<generatorConfiguration>
    <!-- classPathEntry:数据库的JDBC驱动的jar包地址 -->
    <classPathEntry
        location="E:/mysql/mysql-connector-java-5.1.28-bin.jar" />

    <context id="caigouTables" 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/mybatis01" userId="root"
            password="admin">
        </jdbcConnection>

    <!-- targetProject:生成PO类的位置 -->
    <javaModelGenerator targetPackage="com.itheima.domain"
        targetProject=".\src">
        <!-- enableSubPackages:是否让schema作为包的后缀 -->
        <property name="enableSubPackages" value="true" />
        <!-- 从数据库返回的值被清理前后的空格 -->
        <property name="trimStrings" value="true" />
    </javaModelGenerator>
    <!-- targetProject:自动mapper的位置 -->
    <sqlMapGenerator targetPackage="com.itheima.dao"
        targetProject=".\src">
        <property name="enableSubPackages" value="false" />
    </sqlMapGenerator>

    <javaClientGenerator type="XMLMAPPER"
        targetPackage="com.itheima.dao" implementationPackage="com.itheima.dao"
        targetProject=".\src">
        <property name="enableSubPackages" value="false" />
    </javaClientGenerator>

    <table tableName="user"></table>
    <table tableName="orders"></table>
```

需要修改的

需要修改的

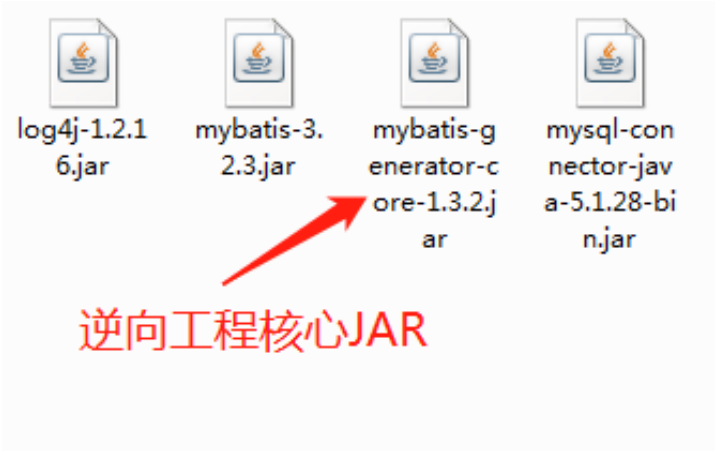
生成javaBean的位置

生成接口的位置

生成javaBean配置文件的路径

要读取的数据库表名

2、逆向工程的JAR



3、生成的代码，执行以下代码

```
List<String> warnings = new ArrayList<String>();
boolean overwrite = true;
File configFile = new File("src/generatorConfig-base.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);
```

使用：

1、创建批量扫描代理开发模式整合Spring项目

2、执行测试查询

```java
/**
 * 需求：测试Mybatis整合Spring
 */
@Test
public void mybatisAndSpring(){
    //加载Spring配置文件
    ApplicationContext app = new ClassPathXmlApplicationContext("applicationContext.xml");
    //获取dao对象
    UserMapper userMapper = app.getBean(UserMapper.class);
    //创建UserExamle对象  每个javaBean对象都会生成一个Examle对象用来处理封装javaBean属性，处理参数
    UserExample example = new UserExample();
    //获取封装criteria参数对象
    Criteria criteria = example.createCriteria();
    //设置封装参数
    criteria.andIdEqualTo(1);    //ID等于1
    criteria.andUsernameIsNotNull();  //username不为空
    //执行查询
    List<User> uList = userMapper.selectByExample(example);
    for (User user : uList) {
        System.out.println(user);
    }

}
```

/**

* 需求：测试Mybatis整合Spring

*/

@Test

public void mybatisAndSpring(){

//加载Spring配置文件

ApplicationContext app = new ClassPathXmlApplicationContext("applicationContext.xml");

//获取dao对象

UserMapper userMapper = app.getBean(UserMapper.class);

//创建UserExamle对象 每个javaBean对象都会生成一个Examle对象用来处理封装javaBean属性，处理参数

UserExample example = new UserExample();

//获取封装criteria参数对象

Criteria criteria = example.createCriteria();

//设置封装参数

criteria.andIdEqualTo(1); //ID等于1

criteria.andUsernameIsNotNull(); //username不为空

//执行查询

```
        List<User> uList = userMapper.selectByExample(example);
        for (User user : uList) {
         System.out.println(user);
        }
}
```