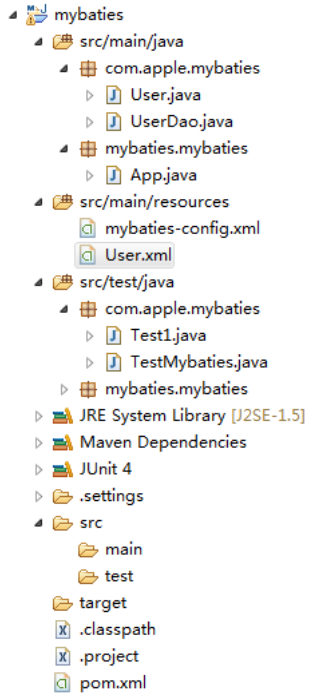
代码如下：



pom.xml

```xml
<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>mybaties</groupId>
    <artifactId>mybaties</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>mybaties</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.2.5</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.8</version>
        </dependency>


        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

```java
public class User {

    private int id;
    private String username;
    private String password;

    public User() {
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

}
```

```java
package com.apple.mybaties;

import java.io.IOException;
import java.io.InputStream;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

public class UserDao {

    public User selectByIdUser(int id) throws IOException {
        String resource = "mybaties-config.xml";

        InputStream inputStream = Resources.getResourceAsStream(resource);
        SqlSessionFactory sessionFactory = new SqlSessionFactoryBuilder().build(inputStream);
        SqlSession session = sessionFactory.openSession();
        User u = session.selectOne("com.apple.mybaties.User.selectByName",id);
        return u;
    }
}
```

```
<mappers>

        <mapper resources="com.ssm.example.dao.StudentDAO.xml"/>

    </mappers>
```

mybaties-config.xml

```xml
<?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>

      <environments default="mybaties">
          <environment id="mybaties">
              <transactionManager type="JDBC"/>
              <dataSource type="POOLED">
                  <property name="driver" value="com.mysql.jdbc.Driver"/>
                  <property name="url" value="jdbc:mysql://127.0.0.1:3306/test?characterEncoding=UTF-8"/>
                  <property name="username" value="root"/>
                  <property name="password" value="shenhong951015"/>
              </dataSource>
          </environment>
      </environments>

      <mappers>
          <mapper resource="User.xml"/>
      </mappers>
  </configuration>
```

User.xml(mapper.xml)

```xml
<?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.apple.mybaties.User">
     <resultMap type="com.apple.mybaties.User" id="userMap">
        <id column="id" property="id"/>
        <result column="username" property="username"/>
        <result column="password" property="password"/>
    </resultMap>

    <select id="selectByName" parameterType="Integer"
        resultMap="userMap">
        select id, username, password from user where id = #{id}
    </select>
</mapper>
```

```java
package com.apple.mybaties;

import java.io.IOException;

import org.junit.Before;
import org.junit.Test;

public class TestMybaties {

    private UserDao userDao = null;

    @Before
    public void signUp(){
        userDao = new UserDao();
    }

    @Test
    public void test2334() {
        User u;
        System.out.println("ok");
        try {
            u = userDao.selectByIdUser(22);
            System.out.println(u.getId());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Mybaties主要解决了JDBC编程的如下几个问题：

1.JDBC：数据库连接创建、释放频繁造成系统资源浪费从而影响系统性能，如果使用数据库连接池可以解决此问题。

Mybaties：在mybaties的配置文件中配置数据库连接池，使用连接池管理数据库连接。

2.JDBC：sql语句写在代码中不易维护，实际应用中sql变化的可能性较大，sql变动需要改变java代码。
mybaties:将sql语句写在maper.xml中，使得sql语句与java代码分离。
3.JDBC：向sql语句传参数比较麻烦，因为sql语句的where条件不一定，可能有多有少，占位符需要和参数一一对应。
Mybaties：mybaties自动将java对象映射至sql语句。
4.JDBC:对结果集解析比较麻烦，sql变化导致解析代码变化，且解析前需要遍历，如果能将数据库记录封装成pojo对象解析比较方便。
Mybaties:Mybaties自动将sql执行结果映射至java对象。

sqlMapConfig.xml（上例中为mybaties-config.xml）：
这是Mybaties的全局配置文件，配置mybaties运行环境，数据库配置，连接池配置等.
Mapper.xml（上例为User.xml）
是sql映射文件，每个mapper.xml对应一个mapper接口，改文件中包括sql语句，此文件需要在SqlMapConfig.xml中加载。
SqlSessionFactory:会话工厂，通过sqlMapConfig.xml文件构造。
sqlSession:由会话工厂创建会话，会话可以直接进行sql操作。
Mapper对象:由sqlSession通过动态代理的方式生成Mapper示例，每个Mapper实例对应着一个Mapper.xml，由Mapper对象进行数据库操作。
MapperStatement：是mybaties底层的一个封装对象，每个MapperStatement对应Mapper.xml中的一个sql语句，该对象包装了mybaties的映射信息，实现输入/输出到sql的映射。
#{id}表示传入的参数，起到占位符的作用。
#{}，${}的区别：
#{}在拼接sql时会为其增加一个单引号，这样起到占位符的作用
#{}可以有效防止sql注入，但是只能传入静态数据，例如order by #{name}，则会被解析为order by ‘name’ ,无法达到动态sql的目的。
${}在拼接时直接将内容拼接到sql语句中，上述若id=22，则sql会为其拼接为id=22