Mybatis

Mybatis：简单的持久层的框架，简化了JDBC操作，通过XML的配置或注解 实现 数据库中的数据和实体对象中的映射关系。

1.**********ORM（对象关系映射）的基本思路：

将数据表中的数据和实体类的属性进行关联，一个实体类对应一张表。

属性对应表中的字段，一个实例对象对应表中的一行记录。

2.**********Mybatis开发步骤**********

一个例子：http://www.cnblogs.com/jeffen/p/6211186.html

（1）. 导入Mbatis的核心jar包mybatis - xxx.jar

（2）. 导入数据库的驱动包

（3）. Mybatis的核心配置文件

配置文件名称：

自定义，通常使用 mybatis-config.xml

配置文件的位置

自定义，通常保存在src目录下面

功能

1.基本的设置（环境的配置，别名）

2.连接数据库的基本信息

3.注册映射文件

涉及框架有配置文件情况下要考虑两个问题

|-核心配置文件的名称是否固定

|-核心配置文件的位置是否固定

********** **mybatis-config.xml示例**********

```
<?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>
 <!-- 配置环境变量 -->
    <!-- environments:为不同环境（开发环境，测试环境）配置连接数据库的信息
        development是
    -->
 <environments default="development">
    <!-- environment为具体的环境配置数据库连接 -->
  <environment id="development">
   <transactionManage r type="JDBC" />
   <!-- 配置数据连接信息 -->
   <dataSource type="POOLED">
    <property name="driver" value="com.mysql.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://127.0.0.1:3066/test" />
        <!-- test是数据库连接名 -->
    <property name="username" value="root" />
    <property name="password" value="" />
   </dataSource>
  </environment>
 </environments>
 <mappers>
  <!-- 注册userMapper.xml文件，
```

```
 userMapper.xml位于com.luo.mapping这个包下，所以resource写成com/luo/mapping/userMapper.xml 格式要注意-->
  <mapper resource="com/luo/mapping/userMapper.xml"/>
 </mappers>
</configuration>
```

****

（4）.创建javabean (先创建包名 习惯为com.xxx.domain)

在包下创建一个实体类。

eg: com.luo.domain.User.java

一个实体类对应数据库中一张表，属性对应表中的字段，一个实例对象对应表中的一行记录。

****

（5）.创建一个mapping包用来放Mapper.xml

eg: com.luo.mapping userMapper.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">
<!--
 创建一个me.gacl.mapping包，专门用于存放sql映射文件，
 在包中创建一个userMapper.xml文件
-->

<!--
 为这个mapper指定一个唯一的namespace，namespace的值习惯上设置成包名+sql映射文件名，这样就能够保证namespace的值是唯一的
 例如namespace="com.luo.mapping.userMapper"就是com.luo.mapping(包名)+userMapper(userMapper.xml文件去除后缀)
-->
<mapper namespace="com.luo.mapping.userMapper">
 <!--
 在select标签中编写查询的SQL语句， 设置select标签的id属性为getUser，id属性值必须是唯一的，不能够重复使用
 parameterType属性指明查询时使用的参数类型，resultType属性指明查询返回的结果集类型
 resultType="com.luo.mapping.User"就表示将查询结果封装成一个User类的对象返回
 User类就是users表所对应的实体类
    -->
 <!--
        根据id查询得到一个user对象
    -->
    <select id="getUser" parameterType="int"
     resultType="com.luo.domain.User">
  select * from users where id=#{id}
    </select>
</mapper>
```

****

(6).在java中应用

1.引入mybatis的配置文件

2.使用类加载器加载mybatis的配置文件（同时也加载相关的sql映射文件）

3.构建sqlSession工厂

4.创建能执行映射文件中sql的sqlSession

5.引入映射sql的标志字符串statement

6.执行sql

```
        //mybatis的配置文件
 String resource = "mybatis-config.xml";
 //使用类加载器加载mybatis的配置文件（它也加载相关的映射文件）
 InputStream is = test2.class.getClassLoader().getResourceAsStream(resource);
 //构建sqlSession的工厂
 SqlSessionFactory sessionFactory = new SqlSessionFactoryBuilder().build(is);
  //使用MyBatis提供的Reasource类加载mybatis的配置文件（它也加载关联的映射文件）
  //Reader reader = Resources.getResourceAsReader(resource);
  //构建sqlSession工厂
  //SqlSessionFactory sessionFactory = new SqlSessionFactoryBuilder().build(reader);
 //创建能执行映射文件中sql的sqlSession
 SqlSession session = sessionFactory.openSession();
 /*
  * 映射sql的标志字符串
  * com.luo.mapping.userMapper是userMapper.xml文件中mapper标签的namespace属性的值
  * getUser是select标签的id属性值，通过select标签的id属性值就可以找到要执行的SQL
  */
 String statement = "com.luo.mapping.userMapper.getUser";//映射sql的标识字符串
 //执行查询返回一个唯一user对象的sql
 User user = (User)session.selectOne(statement,2);
 System.out.println(user);
}
```

3.**********Mybatis的核心类库**********

|-SqlSessionFactoryBuilder

    |- SqlSessionFactory，之后该对象可以销毁

|-SqkSessionFactory

    |-获得SqlSession，该对象要作用于整个应用中

|-SqlSession

    |-内部封装了Connection，和数据库进行交互，线程不安全

4.*********基本的ORM的开发

（1）.提供实体类

（2）.提供对应的表

（3）.提供操作表的规范（提供接口）

（4）.提供映射Mapper文件描述实体和表之间的关系