链滴

java项目集成mybatis入门教程

作者：crick77

原文链接：https://ld246.com/article/1464055086458

来源网站：链滴

起因

公司项目中一直用了myBatis，但是将dao层封装了，并不知道是如何使用的。学习的第一步是查看方文档，发现虽然有中文文档，但是并没有完整的示例，只介绍特性和一些用法，但是并不知道怎么环境跑起来。于是按照官方blog的介绍，搭建了一个示例环境，留给有需要的人。

环境搭建

项目采用maven构造方式，IDE为eclipse，依赖mybatis包及mysql驱动，具体版本及jdk版本见下文om。

pom文件

采用了jdk1.8版本，以及最新的mysql驱动，如果jdk版本较低，需要选用相应的低版本驱动。

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
MLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/m
ven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>wang.crick</groupId>
  <artifactId>study.mybatis</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>study.mybatis</name>

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

  <dependencies>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>6.0.2</version>
    </dependency>
    <dependency>
      <groupId>org.mybatis</groupId>
      <artifactId>mybatis</artifactId>
      <version>3.4.0</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.1</version>
```

```
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
                <encoding>UTF-8</encoding>
            </configuration>
        </plugin>
    </plugins>
  </build>
</project>
```

数据脚本

数据库安装及环境搭建，可参照其他博客，这里只提供创建表的数据脚本。

```
DROP DATABASE IF EXISTS `zero`;
CREATE DATABASE IF NOT EXISTS `zero`;
USE `zero`;

DROP TABLE IF EXISTS `z_user`;
CREATE TABLE `z_user` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(50) NULL DEFAULT NULL,
  `password` VARCHAR(50) NULL DEFAULT NULL,
  `role_type` INT(11) NULL DEFAULT NULL,
  `address` VARCHAR(200) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `username` (`username`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB
AUTO_INCREMENT=13;

INSERT INTO `z_user` (`id`, `username`, `password`, `role_type`, `address`) VALUES
  (1, 'zhangsan', '123131', 1, 'shanghai,pudong');
```

XML配置文件

根据使用习惯，推荐使用XML配置文件。因为是maven项目，所以在src/main/resources目录下新一个配置文件mybatis-config.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>
  <!-- 数据库配置信息 -->
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC" />
      <dataSource type="POOLED">
```

```
            <property name="driver" value="com.mysql.cj.jdbc.Driver" />
            <property name="url" value="jdbc:mysql://172.16.99.121:3306/zero?useUnicode=t
ue&characterEncoding=utf-8" />
            <property name="username" value="root" />
            <property name="password" value="mysqlPwd123456!" />
        </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="wang/crick/study/mybatis/UserMapper.xml" />
  </mappers>
</configuration>
```

配置文件配置了一个数据源，和jdbc的配置方法类似。这里需要注意，如果在url增加额外属性配置，字符需要用转义后的&代替&。

在mappers标签内，指定了mybatis映射的 SQL 语句。

映射SQL语句

配置文件指定的UserMapper.xml文件内容如下

```
<?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="wang.crick.study.mybatis.UserMapper">

  <select id="selectByName" resultType="wang.crick.study.mybatis.User">
    select username , password , role_type AS role_type , address
    from z_user
    where username = #{username}
  </select>

  <insert id="add" parameterType="wang.crick.study.mybatis.User">
    insert into z_user
    (username , password , role_type , address)
    values(#{username} , #{password} , #{roleType} , #{address} )
  </insert>

</mapper>
```

namespace指定了唯一标识，为了保证其唯一性，通常以包名+文件名来命名，也可以自定义名称。

从标签名可以判断出两个方法的作用是select和insert。

wang.crick.study.mybatis.User指向了一个javeBean文件，通过#{}方式，可以映射javaBean的属性量。

```
package wang.crick.study.mybatis;

public class User {

  private int id;
```

```
    private String password;
    private String username;
    private int roleType;
    private String address;

    public int getId() {
        return id;
    }

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

    public String getPassword() {
        return password;
    }

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

    public String getUsername() {
        return username;
    }

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

    public int getRoleType() {
        return roleType;
    }

    public void setRoleType(int roleType) {
        this.roleType = roleType;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

}
```

执行数据操作

mybatis提供了一个SqlSessionFactory，用来根据配置文件，创建资源。SqlSessionFactory创建出后，在程序运行期间，一直存在，并且不应该被修改或清除。在需要执行数据操作的时候，通过单例SqlSessionFactory打开一个SqlSession，执行CRUD操作，在执行完毕后，关闭SqlSession。

```
package wang.crick.study.mybatis;

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 Main {
    public static void main(String[] args) throws IOException {
        String resource = "mybatis-config.xml";
        InputStream inputStream = Resources.getResourceAsStream(resource);
        SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream)

        SqlSession session = sqlSessionFactory.openSession();

        User liUser = new User();
        liUser.setUsername("lisi");
        liUser.setPassword("123654");
        liUser.setRoleType(2);
        liUser.setAddress("辽宁省大连市");
        int result = session.insert("wang.crick.study.mybatis.UserMapper.add", liUser);
        System.out.println(result);

        User zhang = (User)session.selectOne("wang.crick.study.mybatis.UserMapper.selectByN
me", "lisi");
        System.out.println(zhang.getUsername() + ": from " + zhang.getAddress());
//        session.commit();
    }
}
```

程序通过SqlSessionFactoryBuilder读取xml配置文件的方式构建了一个SqlSessionFactory，并打开个SqlSession，执行了insert和select操作，对应Mapper文件中的两个标签，具体的映射规则为，根Mapper文件中的namespace+方法id。参数直接传入javaBean，或者用javaBean接收参数，通过属名映射规则，将属性与数据库字段对应。

只有执行commit()方法后，操作才会提交至数据库。

因为只有一个main方法，所以没有关闭session的操作。正常程序，应该将session.close()放在finall中，保证一定会执行。

至此，一个最基本的mybatis环境搭建完毕，但是这只是一个简单而丑陋的方式，还有很多方法，可将项目精简与美化，让mybatis发挥出自身的优势。

项目优化

properties配置文件

在src/main/resources/prop目录下新建一个配置文件config.properties

```
db.driver=com.mysql.cj.jdbc.Driver
```

```
db.url=jdbc:mysql://172.16.99.121:3306/zero?useUnicode=true&characterEncoding=utf-8
db.username=root
db.password=mysqlPwd123456!
```

注意，此时不需转义，不能用&替换&

将配置信息抽离到此文件中，在mybatis-config.xml中通过properties标签注入属性。

```
<properties resource="prop/config.properties">
    <!-- <property name="driver" value="com.mysql.cj.jdbc.Driver" /> -->
</properties>
```

此标签在<configuration>标签内部，property标签会替换properties中的内容。

别名

在Mapper文件中，指定了javaBean的位置，但是需要指定包名+类名。在这里，我们可以单独指定个类的别名，可以在Mapper中用别名替换全路径名，或者指定基础包名，然后就可以在基础包上加路径类名，简化操作。

```
<typeAliases>
    <typeAlias alias="User" type="wang.crick.study.mybatis.User" />
    <package name="wang.crick.study.mybatis"/>
</typeAliases>
```

此标签同样要在<configuration>标签内部。

此时UserMapper.xml就可以修改为

```
<?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="wang.crick.study.mybatis.UserMapper">

    <select id="selectByName" resultType="User">
        select username , password , role_type AS role_type , address
        from z_user
        where username = #{username}
    </select>

    <insert id="add" parameterType="User" >
        insert into z_user
        (username , password , role_type , address)
        values(#{username} , #{password} , #{roleType} , #{address} )
    </insert>

</mapper>
```

接口

在main方法中，对于sql的调用语句如下：

```
User zhang = (User)session.selectOne("wang.crick.study.mybatis.UserMapper.selectByName",
lisi");
```

这有2个问题，一是通过字符串定位映射，容易出现误差，而是需要执行强制转型。

mybatis提供了更为清晰和类型安全的实现形式：利用接口。

```
UserMapper userMapper = session.getMapper(UserMapper.class);
```

然后可以用

```
User zhang = userMapper.selectByName("lisi");
```

替换之前的

```
User zhang = (User)session.selectOne("wang.crick.study.mybatis.UserMapper.selectByName",
lisi");
```

除此之外，接口还提供了另一种简便的方法，可以通过标签自定义sql语句，而不需要xml的映射文件。

```
package wang.crick.study.mybatis;

import org.apache.ibatis.annotations.Select;

public interface UserMapper {

    User selectByName(String username);

    int add(User user);

    @Select("select * from z_user where id = #{id}")
    User selectById(int id);
}
```

调用方法为

```
package wang.crick.study.mybatis;

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 Main {
    public static void main(String[] args) throws IOException {
        String resource = "mybatis-config.xml";
        InputStream inputStream = Resources.getResourceAsStream(resource);
        SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream)
```

```
        SqlSession session = sqlSessionFactory.openSession();
        UserMapper userMapper = session.getMapper(UserMapper.class);

        User liUser = new User();
        liUser.setUsername("lisi");
        liUser.setPassword("123654");
        liUser.setRoleType(2);
        liUser.setAddress("辽宁省大连市");
        int result = userMapper.add(liUser);
        System.out.println(result);

        User zhang = userMapper.selectByName("lisi");
        System.out.println(zhang.getUsername() + ": from " + zhang.getAddress());

        User firstUser = userMapper.selectById(1);
        System.out.println(firstUser.getUsername() + ": from " + firstUser.getAddress());
//        session.commit();
    }
}
```

注意，此时UserMapper.xml文件中的namespace必须指向接口文件UserMapper.java

mappers指向包

可以将Mapper的接口和xml单独抽离到某个包下，然后在mybatis-config.xml中修改<mappers>标的配置。

因为<mappers>是不支持通配符操作的，按之前的配置方式，没增加一个映射，就需要增加一个配。

抽离之后，可以扫描整个包目录，增加映射后，不需要修改配置文件。

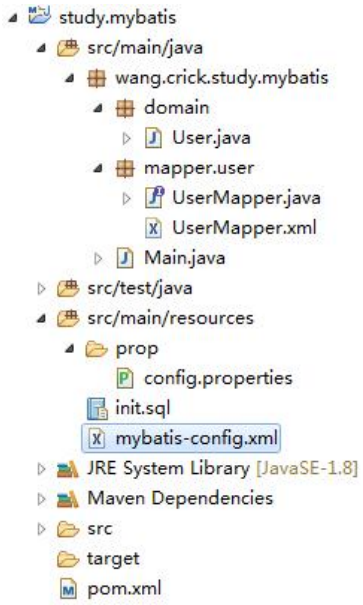
修改后的mybatis-config.xml文件为

```
<?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>
    <properties resource="prop/config.properties">
    </properties>
        <typeAliases>
        <package name="wang.crick.study.mybatis.domain"/>
    </typeAliases>
    <!-- 数据库配置信息 -->
    <environments default="development">
        <environment id="development">
            <transactionManager type="JDBC" />
            <dataSource type="POOLED">
                <property name="driver" value="${driver}" />
                <property name="url" value="${db.url}" />
                <property name="username" value="${db.username}" />
```

```
            <property name="password" value="${db.password}" />
        </dataSource>
    </environment>
</environments>
<mappers>
    <package name="wang.crick.study.mybatis.mapper"/>
</mappers>
</configuration>
```

目录结构为



Mapper XML 文件

xml中的映射sql文件，是mybatis最强大的功能，但不是本篇文章的重点。所以会在另一篇文章中详介绍一些技巧、已经容易遇到的问题。

至此，一个集成了mybatis的java项目已经搭建完成，可以在此基础上编写业务逻辑代码。

打赏区什么内容也没有 就是看在这么多字的份上 求打赏