## 一、简介

        Baits是第一个支持自定义sql语句、存储过程和高级映射的持久层框架。MyBatis去除了几乎所有的JDBC编码和手动设置参数以及获取结果集。MyBatis可以使用简单的XML或者Annotation来配置和原始映射。将接口和Java的Pojo映射成数据库中的记录。

## 二、下载安装

1.下载最新的mybatis框架，地址：[http://code.google.com/p/mybatis/downloads/list?can=3&q=Product%3DMyBatis](http://code.google.com/p/mybatis/downloads/list?can=3&q=Product=MyBatis)。我下载的版本是3.1.1

2.下载mybatis生成工具，地址：[http://code.google.com/p/mybatis/downloads/list?can=3&q=Product%3DGenerator](http://code.google.com/p/mybatis/downloads/list?can=3&q=Product=Generator)。我下载的版本是1.3.2

## 三、简单使用

1.每一个MyBatis应用程序都是围绕着SqlSessionFactory的实例为中心的。一个SqlSessionFactory实例可以通过一个XML配置文件的实例-SqlssionFactoryBuilder来创建，或者从自定义的配置类的实例来创建。

2.从XML文件来构建SqlSessionFactory

       从指定xml文件路径来构建SqlSessionFactory：

**[java]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. String resource = "org/mybatis/example/mybatis-config.xml";
2. InputStream inputStream = Resources.getResourceAsStream(resource);
3. sqlSessionFactory = **new** SqlSessionFactoryBuilder().build(inputStream);

mybatis-config.xml包含Mybatis系统的核心配置，主要包含数据库连接的配置，以及事务管理器的配置，这里是个简单例子：

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<?xml** version="1.0" encoding="UTF-8" **?>**
2. <!DOCTYPE configuration
3. PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4. "http://mybatis.org/dtd/mybatis-3-config.dtd"**>**
5. **<configuration>**
6. **<environments** default="development"**>**
7. **<environment** id="development"**>**
8. **<transactionManager** type="JDBC"**/>**
9. **<dataSource** type="POOLED"**>**
10. **<property** name="driver" value="${driver}"**/>**
11. **<property** name="url" value="${url}"**/>**
12. **<property** name="username" value="${username}"**/>**
13. **<property** name="password" value="${password}"**/>**
14. **</dataSource>**
15. **</environment>**
16. **</environments>**
17. **<mappers>**
18. **<mapper** resource="org/mybatis/example/BlogMapper.xml"**/>**
19. **</mappers>**
20. **</configuration>**

其中environment元素包含事务管理器的配置和数据连接池的设置。mapper元素包含一系列映射文件的列表，映射文件包括SQL编码和映射定义。

3.除了通过使用xml配置来生成SqlSessionFactory，还可以通过硬编码的方式来生成，不过这种方式意义不大，这里就不写了。

4.从SqlSessionFactory中获取SqlSession：

**[java]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. SqlSession session = sqlSessionFactory.openSession();
2. **try** {
3. Blog blog = session.selectOne("org.mybatis.example.BlogMapper.selectBlog", 101);
4. } **finally** {
5. session.close();
6. }

selectOne方法表示查找一条记录，selectList方法表示查找多条记录。第一个参数表示某个映射文件定义的namespace+某个Sql语句定义的id。例如：

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<?xml** version="1.0" encoding="UTF-8" **?>**
2. <!DOCTYPE mapper
3. PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4. "http://mybatis.org/dtd/mybatis-3-mapper.dtd"**>**
5. **<mapper** namespace="org.mybatis.example.BlogMapper"**>**
6. **<select** id="selectBlog" parameterType="int" resultType="Blog"**>**
7. select \* from Blog where id = #{id}
8. **</select>**
9. **</mapper>**

其中mapper为根元素,select标签，表示查询语句,parameterType表示请求参数类型，resultType表示返回参数类型，这个参数可以指定到相同namespace下的某个java类。

还可以用这种写法：

**[java]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. BlogMapper mapper = session.getMapper(BlogMapper.**class**);
2. Blog blog = mapper.selectBlog(101);

这种写法有两个好处：首先，它不依赖于一个字符串，安全性更高。其次，如果你的IDE有代码自动补充功能，很方便你编写这些代码。

另外，关于namespace不仅可以指某个具体的java类，也可以指定某个接口哦。

5.不想使用xml配置，还可以使用Annotation来配置：

**[java]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **package** org.mybatis.example;
2. **public** **interface** BlogMapper {
3. @Select("SELECT \* FROM blog WHERE id = #{id}")
4. Blog selectBlog(**int** id);
5. }

但是这种配置方法适合简单的sql语句配置，如果要加一些复杂的判断什么的，还是xml配置给力一些。

6.关于使用范围和生命周期，很重要哦

       SqlSessionFactoryBuilder：

       这个类可以即刻实例化、使用和丢弃。一旦你生成了你的SqlSessionFactory，就没必要保留它了。

       SqlSessionFactory:

       一旦你创建了SqlSessionFactory，它将在你的程序运行过程中一直存在，没有必要再次创建。它的使用范围和生命周期可以看做是应用程序的使用范围和生命周期。(单例模式实现的,神马？你不知道单例模式，快去补补设计模式吧)

       SqlSession:

       每一个线程应当拥有自己的SqlSession实例。SqlSession是不能被线程共享，同时也是非线程安全的。所以它的范围就是方法的范围，你不应该用一个静态变量或者全局变量来引用sqlSession。这是标准的使用模板：

**[java]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. SqlSession session = sqlSessionFactory.openSession();
2. **try** {
3. // do work
4. } **finally** {
5. session.close();
6. }

获取到SqlSeesion和使用之后，立即close()。

      Mapper：

      和sqlSeesion一样，使用模版：

**[java]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. SqlSession session = sqlSessionFactory.openSession();
2. **try** {
3. BlogMapper mapper = session.getMapper(BlogMapper.**class**);
4. // do work
5. } **finally** {
6. session.close();
7. }

## 四、配置详解

1.Mapper xml文件配置

• cache – 指定一个namespace的cache配置.  
• cache-ref – 通过另一个namespace来引用某个cache.  
• resultMap – 描述如何从数据库结果集来生成你要的对象.  
• parameterMap –要弃用了，不写了.  
• sql – 一块可以被其他语句块重复使用的SQL语句.  
• insert – 插入语句.  
• update – 更新语句.  
• delete – 删除语句.  
• select – 查询语句.

例子：

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<select** id="selectPerson" parameterType="int" resultType="hashmap"**>**
2. SELECT \* FROM PERSON WHERE ID = #{id}
3. **</select>**

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<insert** id="insertAuthor" parameterType="domain.blog.Author"**>**
2. insert into Author (id,username,password,email,bio)
3. values (#{id},#{username},#{password},#{email},#{bio})
4. **</insert>**
5. **<update** id="updateAuthor" parameterType="domain.blog.Author"**>**
6. update Author set
7. username = #{username},
8. password = #{password},
9. email = #{email},
10. bio = #{bio}
11. where id = #{id}
12. **</update>**
13. **<delete** id="deleteAuthor" parameterType="int"**>**
14. delete from Author where id = #{id}
15. **</delete>**

如果某个表的id字段为自增的话，可以这么写：

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<insert** id="insertAuthor" parameterType="domain.blog.Author" useGeneratedKeys="true"
2. keyProperty="id"**>**
3. insert into Author (username,password,email,bio)
4. values (#{username},#{password},#{email},#{bio})
5. **</insert>**

sql标签的例子：

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<sql** id="userColumns"**>** id,username,password **</sql>**

被引用：

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<select** id="selectUsers" parameterType="int" resultType="hashmap"**>**
2. select **<include** refid="userColumns"**/>**
3. from some\_table
4. where id = #{id}
5. **</select>**

result maps简单配置：

首先，java类：

**[java]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **package** com.someapp.model;
2. **public** **class** User {
3. **private** **int** id;
4. **private** String username;
5. **private** String hashedPassword;
7. **public** **int** getId() {
8. **return** id;
9. }
10. **public** **void** setId(**int** id) {
11. **this**.id = id;
12. }
13. **public** String getUsername() {
14. **return** username;
15. }
16. **public** **void** setUsername(String username) {
17. **this**.username = username;
18. }
19. **public** String getHashedPassword() {
20. **return** hashedPassword;
21. }
22. **public** **void** setHashedPassword(String hashedPassword) {
23. **this**.hashedPassword = hashedPassword;
24. }
25. }

然后，把这个类设置一个别名：

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. <!-- In Config XML file -->
2. **<typeAlias** type="com.someapp.model.User" alias="User"**/>**
3. <!-- In SQL Mapping XML file -->
4. **<select** id="selectUsers" parameterType="int" resultType="User"**>**
5. select id, username, hashedPassword
6. from some\_table
7. where id = #{id}
8. **</select>**

或者不配置别名，也可以这么使用：

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<select** id="selectUsers" parameterType="int" resultType="com.someapp.model.User"**>**
2. select id, username, hashedPassword
3. from some\_table
4. where id = #{id}
5. **</select>**

如果你不想指定具体的某一个类，还可以这么使用：

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<select** id="selectUsers" parameterType="int" resultType="hashmap"**>**
2. select id, username, hashedPassword
3. from some\_table
4. where id = #{id}
5. **</select>**

再然后，如果你不想指定某个具体的java类，也就是不想指定ResultType，你还可以通过指定ResultMap来实现：

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<resultMap** id="userResultMap" type="User"**>**
2. **<id** property="id" column="user\_id" **/>**
3. **<result** property="username" column="user\_name"**/>**
4. **<result** property="password" column="hashed\_password"**/>**
5. **</resultMap>**

请注意，没有了resultType属性：

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<select** id="selectUsers" parameterType="int" resultMap="userResultMap"**>**
2. select user\_id, user\_name, hashed\_password
3. from some\_table
4. where id = #{id}
5. **</select>**

2.动态sql语句

有四种：

if

chose (when,otherwise)

trim(where,set)

foreach

if:

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<select** id="findActiveBlogWithTitleLike"
2. parameterType="Blog" resultType="Blog"**>**
3. SELECT \* FROM BLOG
4. WHERE state = ‘ACTIVE’
5. **<if** test="title != null"**>**
6. AND title like #{title}
7. **</if>**
8. **</select>**

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<select** id="findActiveBlogLike"
2. parameterType="Blog" resultType="Blog"**>**
3. SELECT \* FROM BLOG WHERE state = ‘ACTIVE’
4. **<if** test="title != null"**>**
5. AND title like #{title}
6. **</if>**
7. **<if** test="author != null and author.name != null"**>**
8. AND author\_name like #{author.name}
9. **</if>**
10. **</select>**

choose,when,otherwise

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<select** id="findActiveBlogLike"
2. parameterType="Blog" resultType="Blog"**>**
3. SELECT \* FROM BLOG WHERE state = ‘ACTIVE’
4. **<choose>**
5. **<when** test="title != null"**>**
6. AND title like #{title}
7. **</when>**
8. **<when** test="author != null and author.name != null"**>**
9. AND author\_name like #{author.name}
10. **</when>**
11. **<otherwise>**
12. AND featured = 1
13. **</otherwise>**
14. **</choose>**
15. **</select>**

trim,where,set

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<select** id="findActiveBlogLike"
2. parameterType="Blog" resultType="Blog"**>**
3. SELECT \* FROM BLOG
4. **<where>**
5. **<if** test="state != null"**>**
6. state = #{state}
7. **</if>**
8. **<if** test="title != null"**>**
9. AND title like #{title}
10. **</if>**
11. **<if** test="author != null and author.name != null"**>**
12. AND author\_name like #{author.name}
13. **</if>**
14. **</where>**
15. **</select>**

这里请注意使用了 where标签，而不是直接写字符串where，这个where标签知道你的表达式，跟在where后面的单词，是否是and或者or，如果是的话，它会去除。

set也很实用哦：

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<update** id="updateAuthorIfNecessary"
2. parameterType="domain.blog.Author"**>**
3. update Author
4. **<set>**
5. **<if** test="username != null"**>**username=#{username},**</if>**
6. **<if** test="password != null"**>**password=#{password},**</if>**
7. **<if** test="email != null"**>**email=#{email},**</if>**
8. **<if** test="bio != null"**>**bio=#{bio}**</if>**
9. **</set>**
10. where id=#{id}
11. **</update>**

这里注意，set标签会自动去除最后一个逗号。

foreach

**[html]** [view plain](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "view plain) [copy](http://blog.csdn.net/a107494639/article/details/7791907?locationNum=8" \o "copy)

1. **<select** id="selectPostIn" resultType="domain.blog.Post"**>**
2. SELECT \*
3. FROM POST P
4. WHERE ID in
5. **<foreach** item="item" index="index" collection="list"
6. open="(" separator="," close=")"**>**
7. #{item}
8. **</foreach>**
9. **</select>**

foreach一般用在sql语句包含in的语句。