

# 阿进的写字台

博客园 首页 新随笔 联系 订阅 管理

随笔 - 49 文章 - 0 评论 - 54

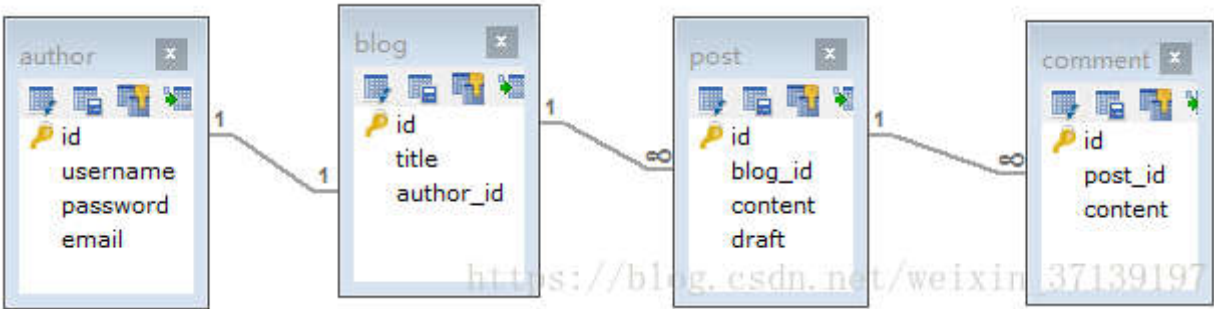
## mybatis-高级结果映射之一对一

mybatis的高级结果映射可以很轻松的帮助我们处理一对一， 一对多的数据关系。

### 1 数据准备

#### 1.1 数据库

创建以下的名为 **mybatis** 的数据库， 并在其下创建4个表。



在此就不贴出来建表的 SQL 语句了， 感兴趣的可以去我的 [Github:mybatis-mapping](https://github.com/137139197/mybatis-mapping) 中获取。

#### 1.2 实体类， 接口和XML

使用 [mybatis-代码生成器](#) 生成相应的实体类， 接口和XML。

昵称: 阿进的写字台  
园龄: 1年6个月  
粉丝: 47  
关注: 4  
[+加关注](#)

2018年12月						
<	日	一	二	三	四	五 六
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

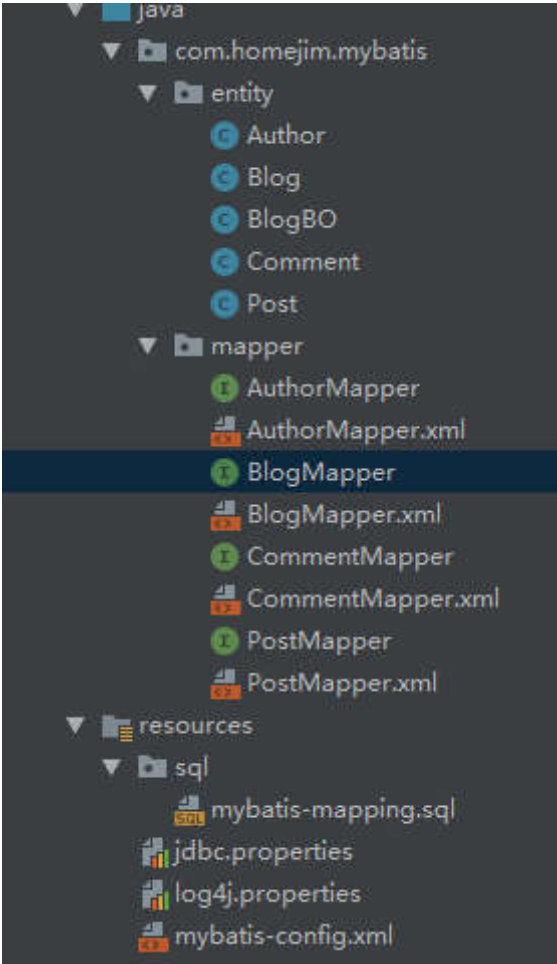
搜索

找找看

谷歌搜索

常用链接

- 我的随笔
- 我的评论
- 我的参与
- 最新评论



以上为生成的项目结构。

## 2 一对一映射

创建的表中，author 和 blog 就是一对一的关系。  
我们希望找到一个blog，然后就会把它的作者信息也关联出来。

### 2.1 resultType 方式一

注意：resultType方式要求获取到的列和成员变量名一致。

#### 我的标签

##### 我的标签

- Java(26)
- mybatis(17)
- mybatis使用(7)
- mybatis源码(6)
- iot(6)
- mqtt(6)
- paho(6)
- Thread(5)
- 多线程(4)
- 反射(3)
- 更多

##### 随笔分类

- C/C++(1)
- Clean Code
- Java(8)
- Java 进阶(15)
- kafka(1)
- linux(1)
- mqtt(6)
- mybatis-工具(1)

### 2.1.1 创建对象

创建一个类 `BlogCustom`，该类继承于 `Blog` 类，在该类上添加 `Author` 对象作为成员变量

```
public class BlogCustom extends Blog {  
    private Author author;  
  
    public Author getAuthor() {  
        return author;  
    }  
  
    public void setAuthor(Author author) {  
        this.author = author;  
    }  
}
```

### 2.1.2 创建接口方法和XML 语句

```
BlogBO selectBoById(int id);  
/**  
 * 根据博客的 id 获取博客及作者的信息  
 * @param id  
 * @return  
 */  
BlogCustom selectCutomById(int id);
```

对应的 XML 语句：

```
<select id="selectCutomById" parameterType="java.lang.Integer"  
resultType="com.homejim.mybatis.entity.BlogCustom">  
    SELECT  
    b.id,  
    b.title,  
    b.author_id AS authorId,  
    a.id AS "author.id",  
    a.username AS "author.username",  
    a.password AS "author.password",  
    a.email AS "author.email"
```

[mybatis-使用\(8\)](#)[mybatis-源码\(10\)](#)[paho\(5\)](#)[spring-boot使用](#)[tomcat\(1\)](#)[多线程那些事\(1\)](#)[集合框架那些事\(1\)](#)

#### 随笔档案

[2018年12月 \(5\)](#)[2018年11月 \(5\)](#)[2018年10月 \(11\)](#)[2018年9月 \(8\)](#)[2018年8月 \(5\)](#)[2018年3月 \(3\)](#)[2018年1月 \(3\)](#)[2017年12月 \(9\)](#)

#### linux

#### 积分与排名

积分 - 29332

排名 - 17422

#### 最新评论

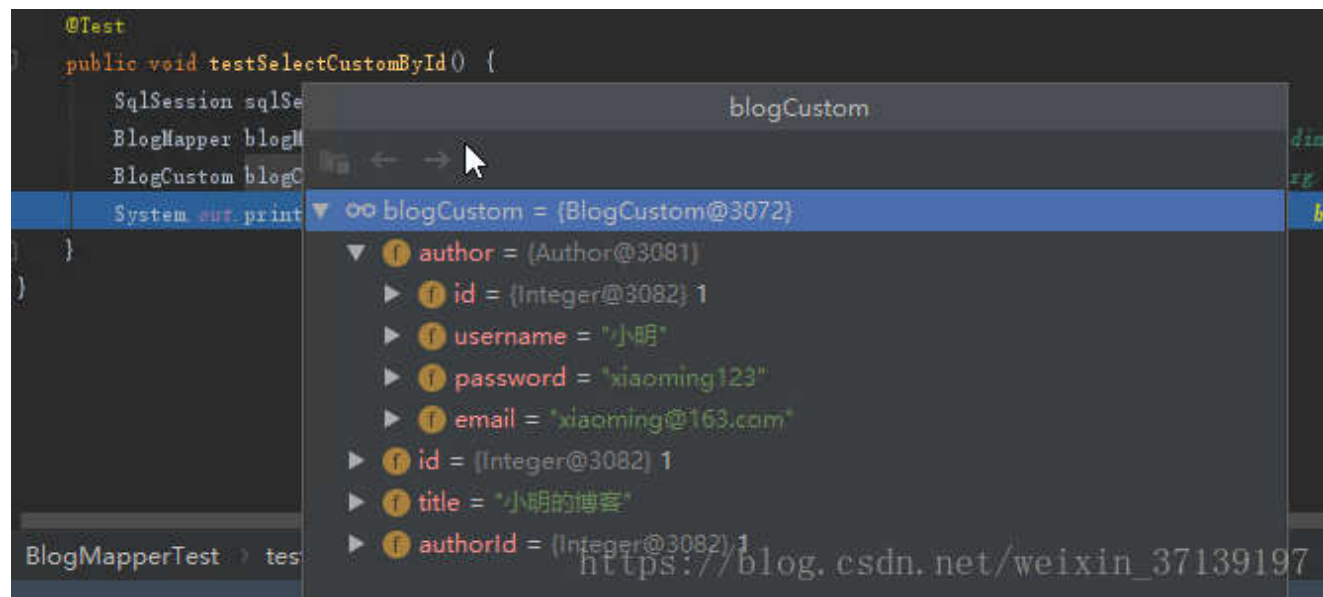
```
FROM blog b LEFT JOIN author a ON b.author_id=a.id
where b.id = #{id,jdbcType=INTEGER}
</select>
```

通过 **author.username** 这种方式可以设定 `author` 对象中的 **username** 属性。

### 2.1.3 测试

```
@Test
public void testSelectCustomById() {
    SqlSession sqlSession = sqlSessionFactory.openSession();
    BlogMapper blogMapper = sqlSession.getMapper(BlogMapper.class);
    BlogCustom blogCustom = blogMapper.selectCutomById(1);
    System.out.println(ToStringBuilder.reflectionToString(blogCustom,
ToStringStyle.MULTI_LINE_STYLE));
}
```

运行后的结果



有时候，我们获取的另外的属性不多，则我们也可以选择下面的方式二。

1. Re:MyBatis动态SQL (认真看看，以后写SQL就爽多了)

@灯塔下的守望者得到肯定，很开心，谢谢...

--阿进的写字台

2. Re:mybatis源码- 反射模块一（跟着MyBatis学反射）：类级别信息的封装

@Java深入浅出这个判断很简单的。if ((name.startsWith("get") && name.length() > 3) || (name.startsW.....

--阿进的写字台

3. Re:MyBatis动态SQL (认真看看，以后写SQL就爽多了)

支持，非常好

--灯塔下的守望者

4. Re:mybatis源码- 反射模块一（跟着MyBatis学反射）：类级别信息的封装

怎么没写怎么判断 setter和getter?

--Java深入浅出

5. Re:HashMap是如何工作的

mark

--Java深入浅出

### 阅读排行榜

1. Paho -物联网 MQTT C Client的实现和详解(9890)

## 2.2 resultType 方式二

### 2.2.1 创建对象

创建一个类 `BlogBO`，该类继承于 `Blog` 类，在该类上添加 `Author` 中的成员变量作为该类自己的成员变量。

```
public class BlogBO extends Blog{
    private String username;

    private String email;

    private String userId;
```

### 2.2.2 创建接口方法和XML 语句

接口方法

```
/**
 * 根据博客的 id 获取博客及作者的信息
 * @param id
 * @return
 */
BlogBO selectBoById(int id);
```

XML 对应的 SQL，其中 resultType 是上面定义的实体对象。

```
<select id="selectBoById" parameterType="java.lang.Integer"
resultType="com.homejim.mybatis.entity.BlogBO">
    select
        b.id, b.title, b.author_id as authorId, a.id as userId, a.username, a.email
    from blog b left join author a on b.author_id=a.id
    where b.id = #{id,jdbcType=INTEGER}
</select>
```

### 2.2.3 测试

创建一个测试例子。

2. 运行时动态库: not found 及介绍-linux 的-Wl,-rpath命令(3508)

3. JDBC详解系列（一）之流程(3348)

4. Paho - MQTT C Client的实现(3036)

5. MyBatis动态SQL（认真看看，以后写SQL就爽多了）(2198)

#### 评论排行榜

1. MyBatis动态SQL（认真看看，以后写SQL就爽多了）(14)

2. mybatis 多个接口参数的注解使用方式 (@Param)(7)

3. mybatis源码-解析配置文件（二）之解析的流程(3)

4. mybatis源码-原来resultMap解析完是这样(3)

5. Java设计模式-建造者(Builder)模式(2)

#### 推荐排行榜

1. MyBatis动态SQL（认真看看，以后写SQL就爽多了）(15)

2. HashMap是如何工作的(5)

3. 扒一扒：Java 中的枚举(4)

4. mybatis源码-原来resultMap解析完是这样(4)

5. mybatis源码-Mapper解析之SQL 语句节点解析（一条语句对应一个MappedStatement)(2)

```
@Test
public void testSelectBoById() {
    SqlSession sqlSession = sqlSessionFactory.openSession();
    BlogMapper blogMapper = sqlSession.getMapper(BlogMapper.class);
    BlogBO blogBO = blogMapper.selectBoById(1);
    System.out.println(ToStringBuilder.reflectionToString(blogBO,
ToStringStyle.MULTI_LINE_STYLE));
}
```

### 测试结果

```
2018-10-13 21:07:48,127 [main] TRACE [com.homejim.mybatis.mapper.BlogMapper.selectBoById] - <= Column: id, title, author_id, user_id, username, email
2018-10-13 21:07:48,127 [main] TRACE [com.homejim.mybatis.mapper.BlogMapper.selectBoById] - <= Row: 1, 小明的博客, 1, 1, 小明, xiaoming@163.com
2018-10-13 21:07:48,135 [main] DEBUG [com.homejim.mybatis.mapper.BlogMapper.selectBoById] - <= Total: 1
com.homejim.mybatis.entity.BlogBO@35f26e72[
  username=小明
  email=xiaoming@163.com
  userId=1
  id=1
  title=小明的博客
  authorId=1
]
```

[https://blog.csdn.net/weixin\\_37139197](https://blog.csdn.net/weixin_37139197)

## 2.3 resultMap 方式

### 2.3.1 创建对象

创建一个类 `BlogCustom`，该类继承于 `Blog` 类，在该类上添加 `Author` 对象作为成员变量



### 2.3.2 创建对应 resultMap

对应创建一个 resultMap

```
<resultMap id="BOResultMap" type="com.homejim.mybatis.entity.BlogCustom"
extends="BaseResultMap">
  <result column="username" jdbcType="VARCHAR" property="author.username" />
  <result column="user_id" jdbcType="VARCHAR" property="author.id" />
  <result column="email" jdbcType="VARCHAR" property="author.email" />
</resultMap>
```

### 2.3.3 创建接口方法和XML 语句

```
/**
 * 根据博客的 id 获取博客及作者的信息, resultMap 方式
 * @param id
 * @return
 */
BlogCustom selectCutomByIdMap(int id);
```

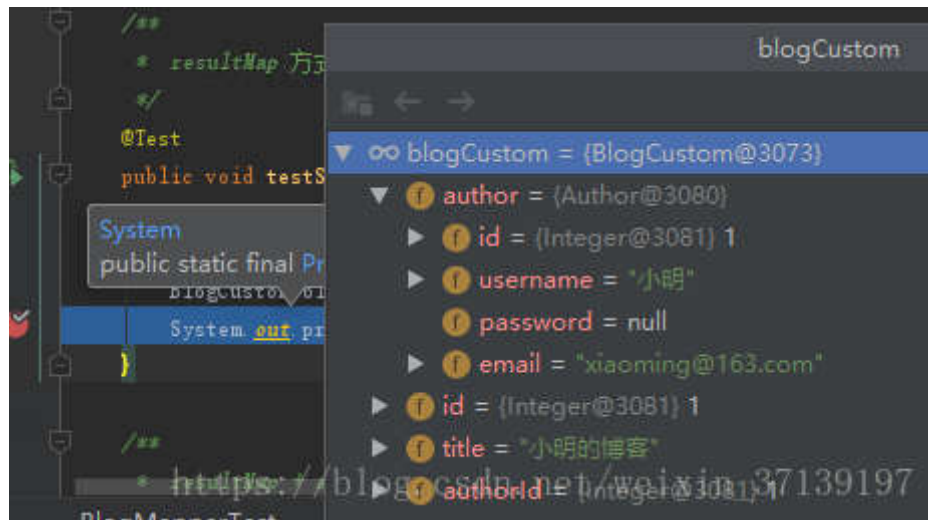
SQL 语句得出的列名与 **BOResultMap** 一致。

```
<select id="selectCutomByIdMap" parameterType="java.lang.Integer" resultMap="BOResultMap">
  SELECT
    b.id,
    b.title,
    b.author_id AS authorId,
    a.id AS "user_id",
    a.username,
    a.email
  FROM blog b LEFT JOIN author a ON b.author_id=a.id
  where b.id = #{id,jdbcType=INTEGER}
</select>
```

## 2.3.4 测试

```
/**
 * resultMap 方式一测试
 */
@Test
public void testSelectCustomByIdMap() {
    SqlSession sqlSession = sqlSessionFactory.openSession();
    BlogMapper blogMapper = sqlSession.getMapper(BlogMapper.class);
    BlogCustom blogCustom = blogMapper.selectCutomByIdMap(1);
    System.out.println(ToStringBuilder.reflectionToString(blogCustom,
ToStringStyle.MULTI_LINE_STYLE));
}
```

得出的结果



其实该类型也可以配置对应 `BlogBO` 类对应的方式，在此不做过多的讲解。

## 2.4 resultMap + association 方式

### 2.4.1 创建对象

创建一个类 `BlogCustom`，该类继承于 `Blog` 类，在该类上添加 `Author` 对象作为成员变量

```
public class BlogCustom extends Blog {  
    private Author author;  
  
    public Author getAuthor() {  
        return author;  
    }  
  
    public void setAuthor(Author author) {  
        this.author = author;  
    }  
}
```

### 2.4.2 创建 resultMap



```

<resultMap id="CustomResultMap" type="com.homejim.mybatis.entity.BlogCustom"
extends="BaseResultMap">
    <association property="author" javaType="com.homejim.mybatis.entity.Author">
        <id column="user_id" jdbcType="INTEGER" property="id" />
        <result column="username" jdbcType="VARCHAR" property="username" />
        <result column="email" jdbcType="VARCHAR" property="email" />
    </association>
</resultMap>

```

或者，可以引用别的 Mapper 中的结果集。

```

<resultMap id="CustomResultMap" type="com.homejim.mybatis.entity.BlogCustom"
extends="BaseResultMap">
    <association property="author" javaType="com.homejim.mybatis.entity.Author"
resultMap="com.homejim.mybatis.mapper.AuthorMapper.BaseResultMap">
    </association>
</resultMap>

```

### 2.4.3 创建接口方法和XML 语句

```

/**
 * 根据博客的 id 获取博客及作者的信息， resultMap + association方式
 * @param id
 * @return
 */
BlogCustom selectCutomByIdAMap(int id);

```

SQL 语句

```

<select id="selectCutomByIdAMap" parameterType="java.lang.Integer" resultMap="CustomResultMap">
    SELECT
        b.id,
        b.title,
        b.author_id AS authorId,
        a.id AS "user_id",
        a.username,
        a.email
    FROM blog b LEFT JOIN author a ON b.author_id=a.id

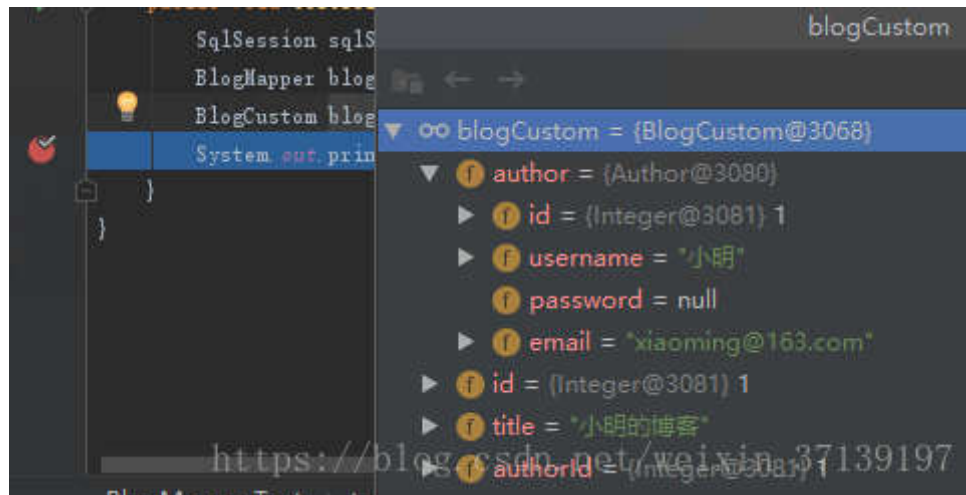
```

```
where b.id = #{id,jdbcType=INTEGER}
</select>
```

## 2.4.4 测试

```
/**
 * resultMap + association 方式测试
 */
@Test
public void testSelectCustomByIdAMap() {
    SqlSession sqlSession = sqlSessionFactory.openSession();
    BlogMapper blogMapper = sqlSession.getMapper(BlogMapper.class);
    BlogCustom blogCustom = blogMapper.selectCutomByIdAMap(1);
    System.out.println(ToStringBuilder.reflectionToString(blogCustom,
ToStringStyle.MULTI_LINE_STYLE));
}
```

结果



## 2.5 resultMap + association 嵌套查询

以上几种方法都是通过 **left join** 的 SQL 语句获取多个对象的结果。还可以经过简单的 SQL 语句，多次查询而转化为我们需要的结果。

### 2.5.1

## 2.5.1 创建对象

创建一个类 `BlogCustom`，该类继承于 `Blog` 类，在该类上添加 `Author` 对象作为成员变量

```
public class BlogCustom extends Blog {  
    private Author author;  
  
    public Author getAuthor() {  
        return author;  
    }  
  
    public void setAuthor(Author author) {  
        this.author = author;  
    }  
}
```

## 2.5.2 创建 resultMap

```
<resultMap id="blogAuthorMap" type="com.homejim.mybatis.entity.BlogCustom"  
extends="BaseResultMap">  
    <association property="author" column="author_id"  
select="com.homejim.mybatis.mapper.AuthorMapper.selectById" fetchType="eager" />  
</resultMap>
```

该结果集与之前的不同了，**association** 标签中使用了 **select** 属性。

**select:** 另一个查询的 id，mybatis 会额外执行这个查询来获取嵌套的结果

**column:** 列名（别名），将主查询中的列作为嵌套查询的参数，多个参数使用逗号（英文）分隔开。

**fetchType:** 数据加载的方式，可选择为 lazy(延迟加载) 或者 eager(积极加载)，该配置会覆盖全局的 lazyLoadingEnabled 配置。

## 2.5.3 创建接口方法和XML 语句

```
/**  
 * 根据博客的 id 获取博客及作者的信息，resultMap + association嵌套方式  
 * @param id  
 * @return
```

```
*/  
BlogCustom selectBlogAndAuthorByIdSelect (int id);
```

获取的数据是博客和用户的信息， 以下的 SQL 只是获取博客信息， 用户信息通过

`com.homejim.mybatis.mapper.AuthorMapper.selectById` 获取。

```
<select id="selectBlogAndAuthorByIdSelect" parameterType="java.lang.Integer"  
resultMap="blogAuthorMap">  
  SELECT  
    b.id,  
    b.title,  
    b.author_id  
  FROM blog b  
  where b.id = #{id,jdbcType=INTEGER}  
</select>
```

`com.homejim.mybatis.mapper.AuthorMapper.selectById` 是一个全限定名， 即 `AuthorMapper` 下的 **`selectById`**

方法

```
/**  
 * 嵌套查询使用的方法  
 * @param id  
 * @return  
 */  
Author selectById(Integer id);
```

对应的 SQL

```
<select id="selectById" parameterType="java.lang.Integer" resultMap="BaseResultMap">  
  select  
    <include refid="Base_Column_List" />  
  from author  
  where id = #{id}  
</select>
```

## 2.5.4 测试

使用时， 调用 `selectBlogAndAuthorByIdSelect` 方法即可。

```

/**
 * resultMap + association 嵌套查询方式测试
 */
@Test
public void testSelectBlogAndAuthorByIdSelect() {
    SqlSession sqlSession = sqlSessionFactory.openSession();
    BlogMapper blogMapper = sqlSession.getMapper(BlogMapper.class);
    BlogCustom blogCustom = blogMapper.selectBlogAndAuthorByIdSelect(1);
    System.out.println(ToStringBuilder.reflectionToString(blogCustom,
ToStringStyle.MULTI_LINE_STYLE));

    Assert.assertNotNull(blogCustom);
    Assert.assertNotNull(blogCustom.getAuthor());
}

```

输出，会发送两次 SQL 语句。

```

2018-10-13 23:54:24.656 [main] DEBUG [org.apache.ibatis.datasource.pooled.PooledDataSource] - Created connection 1804126860.
2018-10-13 23:54:24.664 [main] DEBUG [org.apache.ibatis.transaction.jdbc.JdbcTransaction] - Setting autocommit to false on JDBC Connection [com.mysql.jdbc.JDBC4Connection@6b88ca8c]
2018-10-13 23:54:24.664 [main] DEBUG [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - > Preparing: SELECT b.id, b.title, b.author_id FROM blog b where b.id = ?
2018-10-13 23:54:24.820 [main] DEBUG [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - > Parameters: 1(Integer)
2018-10-13 23:54:24.906 [main] TRACE [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - <= Columns: id, title, author_id
2018-10-13 23:54:24.906 [main] TRACE [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - <= Row: 1, 小明的小博客, 1
2018-10-13 23:54:24.914 [main] DEBUG [com.homejim.mybatis.mapper.AuthorMapper.selectById] - > Preparing: select id, username, 'password', email from author where id = ?
2018-10-13 23:54:24.914 [main] DEBUG [com.homejim.mybatis.mapper.AuthorMapper.selectById] - > Parameters: 1(Integer)
2018-10-13 23:54:24.914 [main] TRACE [com.homejim.mybatis.mapper.AuthorMapper.selectById] - <= Columns: id, username, password, email
2018-10-13 23:54:24.914 [main] TRACE [com.homejim.mybatis.mapper.AuthorMapper.selectById] - <= Row: 1, 小明, xiaoming123, xiaoming@163.com
2018-10-13 23:54:24.914 [main] DEBUG [com.homejim.mybatis.mapper.AuthorMapper.selectById] - <= Total: 1
2018-10-13 23:54:24.914 [main] DEBUG [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - <= Total: 1
com.homejim.mybatis.entity.BlogCustom@52bd8fca[
  author=com.homejim.mybatis.entity.Author@37c7595
  id=1
  title=小明的小博客
  authorId=1
]

```

[https://blog.csdn.net/weixin\\_37139197](https://blog.csdn.net/weixin_37139197)

可以看到，上面的结果示意图中，发送了两次 SQL。

## 2.5.5 延迟加载

如果是一个对象中只是包含一两个对象，使用上面的方式还好。但是如果包含有很多，那要一次性发送很多次 SQL，性能上就会很有影响。延迟加载可以解决此类的问题。

延迟加载就是说，只有在调用内部的对象时，才会把获取该对象的 SQL 发送出去。

更改结果集

```
<resultMap id="blogAuthorMapLazy" type="com.homejim.mybatis.entity.BlogCustom"
extends="BaseResultMap">
    <association fetchType="lazy" property="author" column="author_id"
select="com.homejim.mybatis.mapper.AuthorMapper.selectById" />
</resultMap>
```

将上面的查询中的结果集更改 resultMap="blogAuthorMapLazy"

```
<select id="selectBlogAndAuthorByIdSelect" parameterType="java.lang.Integer"
resultMap="blogAuthorMapLazy"> <!--resultMap="blogAuthorMap"-->
    SELECT
    b.id,
    b.title,
    b.author_id
FROM blog b
where b.id = #{id,jdbcType=INTEGER}
</select>
```

更改延迟加载总开关

```
<setting name="lazyLoadingEnabled" value="true"/>
<setting name="aggressiveLazyLoading" value="false"/>
```

测试

```
2018-10-14 10:55:13.124 [main] DEBUG [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - ==> Preparing: SELECT b.id, b.title, b.author_id FR
2018-10-14 10:55:13.210 [main] DEBUG [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - ==> Parameters: 1(Integer)
2018-10-14 10:55:13.241 [main] TRACE [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - <= Columns: id, title, author_id
2018-10-14 10:55:13.241 [main] TRACE [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - <= Row: 1, 小明的博客, 1
2018-10-14 10:55:13.335 [main] DEBUG [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - <= Total: 1
开始使用author对象
2018-10-14 10:55:13.335 [main] DEBUG [com.homejim.mybatis.mapper.AuthorMapper.selectById] - ==> Preparing: select id, username, 'password', email from author wh
2018-10-14 10:55:13.335 [main] DEBUG [com.homejim.mybatis.mapper.AuthorMapper.selectById] - ==> Parameters: 1(Integer)
2018-10-14 10:55:13.335 [main] TRACE [com.homejim.mybatis.mapper.AuthorMapper.selectById] - <= Columns: id, username, password, email
2018-10-14 10:55:13.335 [main] TRACE [com.homejim.mybatis.mapper.AuthorMapper.selectById] - <= Row: 1, 小明, xiaoming123, xiaoming@163.com
2018-10-14 10:55:13.335 [main] DEBUG [com.homejim.mybatis.mapper.AuthorMapper.selectById] - <= Total: 1
https://blog.csdn.net/weixin_37139197
```

注意：延迟加载是在 `SqlSession` 的声明周期内的，如果超出该声明周期，如 spring 中，只能在 Service 层使用延迟加载的对象，如果返回Controller层在获取延迟加载属性，则会抛出异常。

有时候，我们配置了延迟加载，但是却想要一次性加载，怎么办？

有一个配置属性可以帮我们解决 **lazyLoadTriggerMethods**，它的默认配置如下：

```
<setting name="lazyLoadTriggerMethods" value="equals,clone,hashCode,toString"/>
```

就是说我们使用上面配置中的任何一个方法（上面的是默认的，我们可以不配置），就可以加载属性啦。

测试

```
/**
 * resultMap + association 嵌套查询方式测试 (延迟加载不延迟lazyLoadTriggerMethods)
 */
@Test
public void testSelectBlogAndAuthorByIdSelectTrigger() {
    SqlSession sqlSession = sqlSessionFactory.openSession();
    BlogMapper blogMapper = sqlSession.getMapper(BlogMapper.class);
    BlogCustom blogCustom = blogMapper.selectBlogAndAuthorByIdSelect(1);
    blogCustom.equals(null);
    Assert.assertNotNull(blogCustom);
    sqlSession.close();
    System.out.println("开始使用author对象");
    Assert.assertNotNull(blogCustom.getAuthor());
}
```

结果

```
2018-10-14 11:22:29,639 [main] DEBUG [org.apache.ibatis.transaction.jdbc.JdbcTransaction] - Setting autocommit to false on JDBC Connection [com.mysql.jdbc.JDBC4Connection
2018-10-14 11:22:29,647 [main] DEBUG [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - ==> Preparing: SELECT b.id, b.title, b.author_id FROM blog b
2018-10-14 11:22:29,725 [main] DEBUG [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - ==> Parameters: 1(Integer)
2018-10-14 11:22:29,764 [main] TRACE [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - <= Columns: id, title, author_id
2018-10-14 11:22:29,764 [main] TRACE [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - <= Row: 1, 小明的博客, 1
2018-10-14 11:22:29,850 [main] DEBUG [com.homejim.mybatis.mapper.BlogMapper.selectBlogAndAuthorByIdSelect] - <= Total: 1
2018-10-14 11:22:29,850 [main] DEBUG [com.homejim.mybatis.mapper.AuthorMapper.selectById] - ==> Preparing: select id, username, 'password', email from author where id =
2018-10-14 11:22:29,850 [main] DEBUG [com.homejim.mybatis.mapper.AuthorMapper.selectById] - ==> Parameters: 1(Integer)
2018-10-14 11:22:29,858 [main] TRACE [com.homejim.mybatis.mapper.AuthorMapper.selectById] - <= Columns: id, username, password, email
2018-10-14 11:22:29,858 [main] TRACE [com.homejim.mybatis.mapper.AuthorMapper.selectById] - <= Row: 1, 小明, xiaoming123, xiaoming@163.com
2018-10-14 11:22:29,858 [main] DEBUG [com.homejim.mybatis.mapper.AuthorMapper.selectById] - <= Total: 1
2018-10-14 11:22:29,858 [main] DEBUG [org.apache.ibatis.transaction.jdbc.JdbcTransaction] - Resetting autocommit to true on JDBC Connection [com.mysql.jdbc.JDBC4Connection
2018-10-14 11:22:29,858 [main] DEBUG [org.apache.ibatis.transaction.jdbc.JdbcTransaction] - Closing JDBC Connection [com.mysql.jdbc.JDBC4Connection@336f1079]
2018-10-14 11:22:29,858 [main] DEBUG [org.apache.ibatis.datasource.pooled.PooledDataSource] - Returned connection 862916729 to pool
开始使用author对象
https://blog.csdn.net/weixin_37139197
```

### 3. 代码

本来还要写的一对多，鉴别器的，但由于篇幅的原因，后续继续吧。

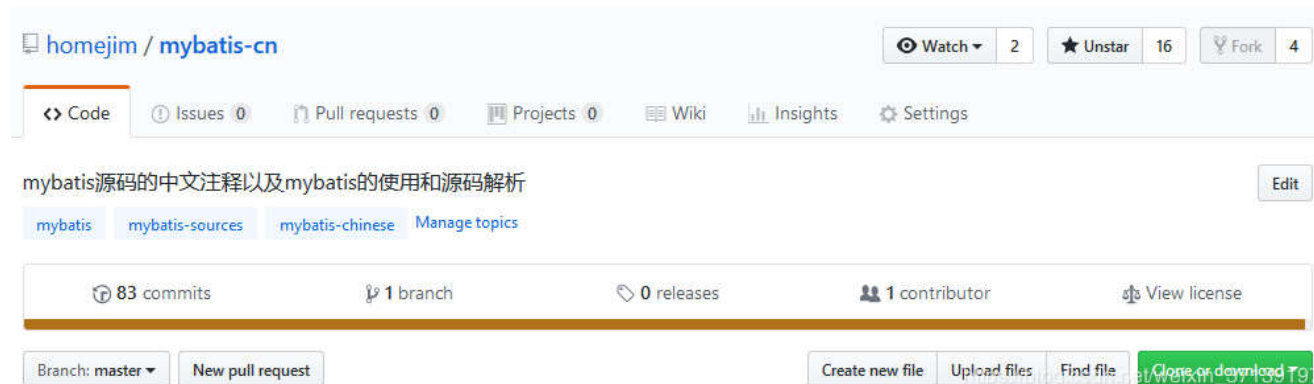
我的 [Github:mybatis-mapping](#)

# 一起学 mybatis

你不想来学习 mybatis? 学习其使用和源码呢? 那么, 在博客园关注我吧!!

我自己打算把这个源码系列更新完毕, 同时会更新相应的注释。快去 star 吧!!

[mybatis最新源码和注释](#)



The screenshot shows the GitHub repository page for 'homejim / mybatis-cn'. At the top, it displays repository statistics: 2 watchers, 16 unstars, and 4 forks. Below this is a navigation bar with links to Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. The main content area shows the repository name 'mybatis-cn' and a description: 'mybatis源码的中文注释以及mybatis的使用和源码解析'. There are tags for 'mybatis', 'mybatis-sources', 'mybatis-chinese', and 'Manage topics'. Below the description, it shows repository statistics: 83 commits, 1 branch, 0 releases, and 1 contributor. At the bottom, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'.

作者: [阿进的写字台](#)

出处: <https://www.cnblogs.com/homejim/>

本文版权归作者和博客园共有, 欢迎转载, 但未经作者同意必须保留此段声明, 且在文章页面明显位置给出原文连接, 否则保留追究法律责任的权利。

分类: [Java 进阶](#), [mybatis-使用](#)

标签: [Java](#), [mybatis](#), [mybatis使用](#)

好文要顶

关注我

收藏该文



阿进的写字台

关注 - 4

粉丝 - 47

[+加关注](#)

1

0



« 上一篇: [mybatis 代码生成器 \(IDEA, Maven\)及配置详解 \(部分配置你应该不知道\)](#)

» 下一篇: [mybatis-高级结果映射之一对多](#)

posted @ 2018-10-14 11:51 阿进的写字台 阅读(172) 评论(2) 编辑 收藏

#1楼 2018-10-14 23:11 瓦尔登湖的一滴

“csdn的图片坏了

支持(0) 反对(0)”

#2楼[楼主] 2018-10-14 23:44 阿进的写字台

“@ 瓦尔登湖的一滴  
纳尼? 我都可以看得到啊

支持(0) 反对(0)”

[刷新评论](#) [刷新页面](#) [返回顶部](#)

**注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。**

【推荐】超50万VC++源码: 大型组态工控、电力仿真CAD与GIS源码库!

【活动】华为云12.12会员节 云产品1折起 满额送Mate20 点击抢购

【推荐】服务器100%基准CPU性能，1核1G首年168元，限时特惠!



#### 相关博文：

- 辞职诗
- How to get a Chinese character's PinYin automaticly?
- determine if there is an active connection to the internet
- 來點不一樣的一烤田雞&腐皮臭豆腐卷
- 今天写了一篇<<胡思乱想>>，回头一看把自己吓了一跳，但是决定，再胡思乱想一下。

#### 最新新闻：

- 癌症疫苗再获突破！证实抗癌疫苗对“冷”肿瘤起效
  - 为什么我觉得Python烂的要死？原因有八
  - 王兴：20年到 C，20年到 B
  - 硅谷是个什么谷（第二十五章）：更好的世界
  - 戴威的董事会股东的签字权 谁搞垮了ofo的资金和前途？
- » 更多新闻...

Copyright ©2018 阿进的写字台