

大家好，很高兴我们可以继续交流学习Java相关框架知识点。在上一小节中，我们对当前热门的Spring框架进行了简单的介绍，重点阐述了其AOP和IOC两种关键技术。

9

本小节中，我们主要对当前热门的持久层框架MyBatis来进行介绍。MyBatis是一个优秀的持久层框架，基本上已经取代了Hibernate成为了当前市场上的主流持久层框架。在日常的工作中，我们经常在使用Spring和MyBatis来搭建我们的服务。所以，熟练的使用并且掌握框架的相关技术原理对于我们开发效率也是很有帮助的。接下来，我们先来看一个MyBatis框架的使用案例吧。

Demo展示：使用MyBatis+Druid连接MySQL数据库

数据库中表的建立，建立一张测试表mybatis_test，语句如下：

```
1 create table mybatis_test(  
2   id int primary key,  
3   k int not null,  
4   name varchar(16),  
5   index (k))engine=InnoDB;
```

插入一条记录，语句如下：

```
1 insert into mybatis_test values(1,1,'zhangsan');
```

```
mysql> create table mybatis_test(  
    -> id int primary key,  
    -> k int not null, 当前数据库的名字叫test  
    -> name varchar(16),  
    -> index (k))engine=InnoDB;  
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> desc mybatis_test;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
k	int(11)	NO	MUL	NULL	
name	varchar(16)	YES		NULL	

```
3 rows in set (0.02 sec)
```

```
mysql> insert into mybatis_test values(1,1,'zhangsan');  
Query OK, 1 row affected (0.01 sec)
```

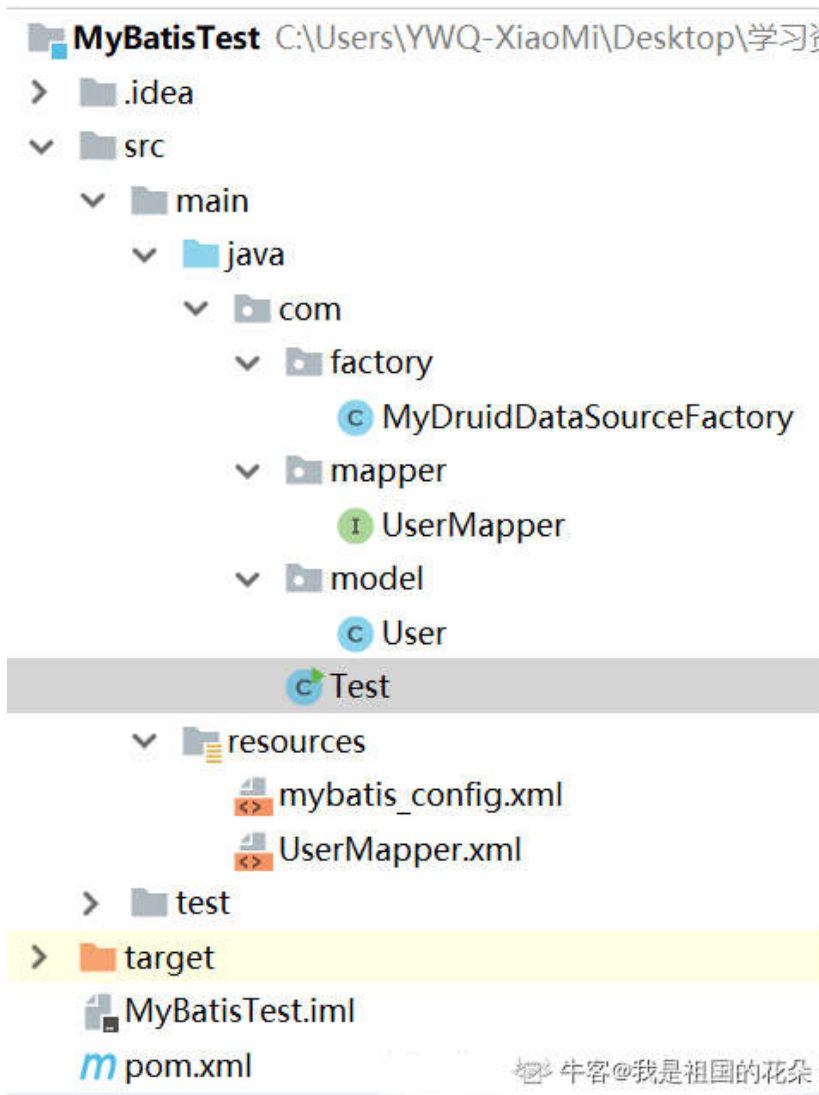
```
mysql> select * from mybatis_test;
```

id	k	name
1	1	zhangsan

```
1 row in set (0.00 sec)
```

牛客@我是祖国的花朵

建立完整的项目，结构如下所示：



接下来，我们分别给出各个文件的内容：

User实体类：

```
1 package com.model;
2
3 public class User {
4     private int id;
5     private int k;
6     private String name;
7
8     public int getId() {
9         return id;
10    }
11
12    public void setId(int id) {
13        this.id = id;
14    }
15
16    public int getK() {
17        return k;
18    }
19
20    public void setK(int k) {
21        this.k = k;
22    }
23
24    public String getName() {
25        return name;
26    }
27 }
```

```

27
28     public void setName(String name) {
29         this.name = name;
30     }
31
32     @Override
33     public String toString() {
34         return "User{" +
35             "id=" + id +
36             ", k=" + k +
37             ", name='" + name + '\'' +
38             '}';
39     }
40 }

```

9

自定义的Druid数据库连接池MyDruidDataSourceFactory：

```

1  package com.factory;
2
3  import com.alibaba.druid.pool.DruidDataSourceFactory;
4  import org.apache.ibatis.datasource.DataSourceFactory;
5  import javax.sql.DataSource;
6  import java.util.Properties;
7
8  public class MyDruidDataSourceFactory implements DataSourceFactory {
9      private DataSource dataSource;
10
11     public DataSource getDataSource() {
12         return this.dataSource;
13     }
14
15     public void setProperties(final Properties props) {
16         try {
17             this.dataSource = DruidDataSourceFactory.createDataSource(props);
18         } catch (final RuntimeException e) {
19             throw e;
20         } catch (final Exception e) {
21             throw new RuntimeException("init datasource error", e);
22         }
23     }
24 }

```

映射器的定义：

```

1  package com.mapper;
2  import com.model.User;
3
4  public interface UserMapper {
5      public User getUserInfo(int id);
6  }

```

UserMapper.xml文件的定义

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE mapper PUBLIC "-//mybatis.org/DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis
3
4  <!-- namespace 与对应接口的全路径严格对应-->
5  <mapper namespace="com.mapper.UserMapper">
6
7      <select id="getUserInfo" parameterType="int" resultType="User">
8          select * from mybatis_test where id=#{id}
9      </select>
10
11  </mapper>

```

MyBatis配置文件mybatis_config.xml文件:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/
3  <configuration>
4
5      <!--定义别名-->
6      <typeAliases>
7          <typeAlias type="com.model.User" alias="User" />
8      </typeAliases>
9
10     <environments default="development">
11         <environment id="development">
12             <transactionManager type="JDBC"/>
13             <dataSource type="com.factory.MyDruidDataSourceFactory">
14                 <property name="driver" value="com.mysql.cj.jdbc.Driver" />
15                 <property name="url" value="jdbc:mysql://localhost:3306/test" />
16                 <property name="username" value="root" />
17                 <property name="password" value="123" />
18                 <property name="maxActive" value="30" /> <!--接下来是Druid连接池的属性配置-->
19                 <property name="initialSize" value="30" />
20                 <property name="minIdle" value="30" />
21                 <property name="testWhileIdle" value="false" />
22             </dataSource>
23         </environment>
24     </environments>
25
26     <!--引入定义好的UserMapper文件-->
27     <mappers>
28         <mapper resource="UserMapper.xml" />
29     </mappers>
30
31 </configuration>

```

9

依赖配置pom.xml:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/m
5      <modelVersion>4.0.0</modelVersion>
6
7      <groupId>com.ywq</groupId>
8      <artifactId>mybatis.test</artifactId>
9      <version>1.0-SNAPSHOT</version>
10
11     <dependencies>
12         <!--MyBatis依赖-->
13         <dependency>
14             <groupId>org.mybatis</groupId>
15             <artifactId>mybatis</artifactId>
16             <version>3.5.0</version>
17         </dependency>
18         <!--阿里巴巴Druid依赖-->
19         <dependency>
20             <groupId>com.alibaba</groupId>
21             <artifactId>druid</artifactId>
22             <version>1.1.10</version>
23         </dependency>
24         <!--MySQL相关依赖-->

```

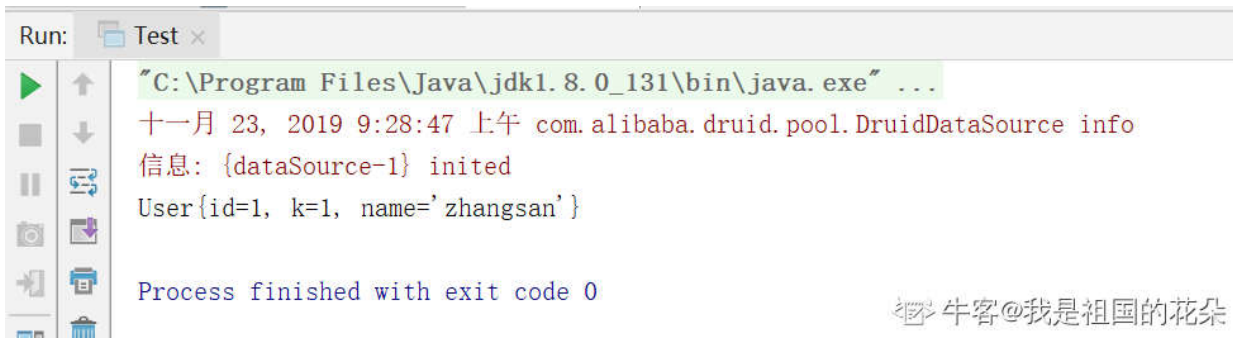
```
25     <dependency>
26         <groupId>mysql</groupId>
27         <artifactId>mysql-connector-java</artifactId>
28         <version>5.1.28</version>
29     </dependency>
30 </dependencies>
31
32 </project>
```

9

测试类Test.java:

```
1  package com;
2  import java.io.IOException;
3  import java.io.Reader;
4
5  import com.mapper.UserMapper;
6  import com.model.User;
7  import org.apache.ibatis.io.Resources;
8  import org.apache.ibatis.session.SqlSession;
9  import org.apache.ibatis.session.SqlSessionFactory;
10 import org.apache.ibatis.session.SqlSessionFactoryBuilder;
11
12 public class Test {
13     private static SqlSessionFactory factory;
14
15     public static void main(String[] args) {
16         initMyBatis();
17         SqlSession session = factory.openSession();
18         UserMapper userMapper = session.getMapper(UserMapper.class);
19         User userInfo = userMapper.getUserInfo(1);
20         session.commit();
21         System.out.println(userInfo);
22         session.close();
23     }
24
25     /**
26      * 初始化建立SqlSessionFactory
27      */
28     private static void initMyBatis() {
29         String resource = "mybatis_config.xml";
30         Reader reader = null;
31         try {
32             reader = Resources.getResourceAsReader(resource);
33         } catch (IOException e) {
34             System.out.println(e.getMessage());
35         }
36         factory = new SqlSessionFactoryBuilder().build(reader);
37     }
38
39 }
```

项目输出如下:



```
Run: Test x
"C:\Program Files\Java\jdk1.8.0_131\bin\java.exe" ...
十一月 23, 2019 9:28:47 上午 com.alibaba.druid.pool.DruidDataSource info
信息: {dataSource-1} inited
User{id=1, k=1, name='zhangsan'}
Process finished with exit code 0
```

9

牛客@我是祖国的花朵

在给出了一个MyBatis的使用Demo之后，我们接着来看MyBatis的相关知识点吧~

(1) MyBatis是什么?

答：MyBatis是一个**半ORM（对象关系映射）框架**，内部封装了JDBC，开发时只需要关注SQL语句本身，不需要花费精力去处理加载驱动、创建连接、创建statement等繁杂的过程。通过直接编写原生态SQL，可以严格控制SQL语句的执行性能，灵活度高(**支持动态SQL语句**)。

MyBatis **使用XML或注解来配置和映射原生信息**，将 POJO映射成数据库中的记录，避免了JDBC代码手动设置参数以及获取结果集的繁琐步骤。

MyBatis**通过xml文件或注解的方式将要执行的各种 statement 配置起来**，并通过Java对象和 statement中SQL的动态参数进行映射生成最终执行的SQL语句，最后由MyBatis框架执行SQL语句，并将结果映射为Java对象并返回。

解析：

MyBatis是一个优秀的持久层框架，我们来看看其与Hibernate和JDBC的比较吧。

MyBatis和Hibernate的区别：

- **MyBatis**的优点是代码开发量少、简单易上手、SQL语句和代码分离（便于修改）、数据库可移植；但是，其缺点是SQL语句需要自己写，并且参数只能有一个。
- **Hibernate**的优点是进行了对象关系数据库映射、完全面向对象、提供缓存机制和HQL编程；但是，其缺点是不能灵活使用原生SQL、无法对SQL优化、全表映射效率低下并且存在N+1的问题。

JDBC、MyBatis和Hibernate的主要区别：

JDBC更为灵活，更加有效率，系统运行速度快。但是代码繁琐复杂，如果使用了存储过程就不方便数据库移植了。

Hibernate和MyBatis是关系数据库框架，开发速度快，更加面向对象，可以移植更换数据库，但影响系统性能。

(2) MyBatis的核心组件有哪些?

答：MyBatis的核心组件包括**SqlSessionFactoryBuilder**，**SqlSessionFactory**，**SqlSession**和**Mapper**。在开头给出的Demo中，我们也可以看到这四大核心组件。下边我们依次介绍。

SqlSessionFactoryBuilder是一个构建器，通过XML配置文件或者Java编码获得资源来构建SqlSessionFactory，通过Builder可以构建多个SessionFactory。其生命周期一般只存在于方法的局部，用完即可回收。一般通过如下语句建

立：

```
1 | SqlSessionFactory factory = SqlSessionFactoryBuilder.build(inputStream);
```

SqlSessionFactory的作用就是创建SqlSession，也就是创建一个会话。每次程序需要访问数据库，就需要使用到SqlSession。所以SqlSessionFactory应该在MyBatis应用的整个生命周期中。为了减少每次都创建一个会话带来的资源消耗，一般情况下都会使用单例模式来创建SqlSession。创建方式如下：

```
1 | SqlSession sqlSession = SqlSessionFactory.openSession();
```

SqlSession就是一个会话，相当于JDBC中的Connection对象，既可以发送SQL去执行并返回结果，也可以获取Mapper接口。SqlSession是一个线程不安全的对象，其生命周期应该是请求数据库处理事务的过程中。每次创建的SqlSession对象必须及时关闭，否则会使数据库连接池的活动资源减少，影响系统性能。

Mapper也叫做映射器，由Java接口和XML文件（或者是注解）共同组成，给出了对应的SQL和映射规则，主要负责发送SQL去执行，并且返回结果。通过下边的语句来获取Mapper，接下来就可以调用Mapper中的各个方法去获取数据库结果了。

```
1 | XXMapper xxMapper = sqlSession.getMapper(XXMapper.class);
```

解析：

四大核心组件是MyBatis框架的重要组成部分，必须熟练掌握使用并且了解组件作用。

根据Demo展示，我们看到Test测试类中创建SqlSessionFactory的过程首先是通过读取xml配置，然后使用SqlSessionFactorybuilder来build一个SqlSessionFactory。具体创建过程的步骤如下：

- 通过XMLConfigBuilder解析配置的XML文件，读出配置参数，封装到Configuration类中
- 使用Configuration对象去创建SqlSessionFactory，SqlSessionFactory在MyBatis中是一个接口而不是一个类，所以提供了一个其默认实现类DefaultSqlSessionFactory

创建SqlSessionFactory的本质是一种Builder模式。Configuration的主要作用如下：

- 读入配置文件
- 初始化基础配置。包括全局配置，别名，类型处理器，环境数据库标识以及Mapper映射器等
- 提供单例
- 执行一些重要的对象方法和初始化配置信息

(3) MyBatis的动态SQL有了解吗？

答：MyBatis动态SQL可以在XML映射文件内，以标签的形式编写动态SQL。动态SQL的执行原理是，**根据表达式的值完成逻辑判断并动态拼接SQL语句。**

解析：

对动态SQL的支持是MyBatis的一大优点。MyBatis提供了9种动态SQL标签：

- **if**：单条件分支的判断语句
- **choose, when, otherwise**：多条件的分支判断语句

- **foreach**: 列举条件, 遍历集合, 实现循环语句
- **trim,where,set**: 是一些辅助元素, 可以对拼接的SQL进行处理
- **bind**: 进行模糊匹配查询的时候使用, 提高数据库的可移植性

9

(4) MyBatis的Mapper中的常见标签有哪些?

答: Mapper中常见的标签如下:

- select | insert | update | delete
- resultMap | parameterMap | sql | include | selectKey
- trim | where | set | foreach | if | choose | when | otherwise | bind

解析:

MyBatis中的标签有很多很多, 单纯介绍意义并不大, 希望大家可以搭建一个实际的项目来进行学习。面试中挑选最常见的标签来给面试官阐述即可。

(5) MyBatis的Dao接口的工作原理有了解吗? (了解)

答: **Dao接口即Mapper接口**。接口的全限名, 就是映射文件中的namespace的值。接口的方法名, 就是映射文件中Mapper的Statement的id值。接口方法内的参数, 就是传递给SQL的参数。

Mapper接口是没有实现类的, 当调用接口方法时, 接口**全限名+方法名**拼接字符串作为key值, 可唯一定位一个**MapperStatement**。在MyBatis中, 每一个<select>、<insert>、<update>、<delete>标签, 都会被解析为一个**MapperStatement对象**。

例如: com.mapper.UserMapper.getUserInfo, 可以唯一找到namespace为com.mapper.UserMapper下面id 为 getUserInfo的 MapperStatement。

Mapper接口的工作原理是**JDK动态代理 (dai) 理**, MyBatis运行时会使用JDK动态代理 (dai) 理为Mapper接口生成代理对象proxy, 代理对象会拦截接口方法, 转而执行MapperStatement所代表的SQL, 然后将SQL执行结果返回。

解析:

在了解了Dao接口的工作原理之后, 我们再来看如下的问题:

面试官: Dao接口中的方法可以重载吗? (掌握)

当Dao接口中的方法参数不同时, 可以构成重载吗? 回答这个问题之前, 我们可以先来温习下何为重载?

- 重载是指一个类里面 (包括父类的方法) 存在方法名相同, 但是参数不一样的方法, 参数不一样可以是不同的参数个数、类型或顺序
- 如果仅仅是修饰符、返回值、throw的异常不同, 那么这是2个相同的方法

了解了映射器的工作原理之后, 我们应该可以得出下边的结论:

Mapper接口里的方法, 是不能重载的, 因为是使用 全限名+方法名 的保存和寻找策略。

我们继续来看下边的的问题。

面试官：不同的映射文件xml中的id值可以重复吗？

前面我们说了，MyBatis使用全限名+方法名来寻找对应的MapperStatement，那么看起来我们是不可以定义相同的id值的。但是注意需要分情况讨论该问题。

- 不同的xml映射文件，如果配置了namespace，那么id可以重复
- 如果没有配置namespace，那么id不能重复

(6) MyBatis中 # 和 \$ 的区别是什么？（高频考点）

答：MyBatis中我们**能用#号就尽量不要使用\$符号**，它们的区别主要体现在下边几点：

- #符号将传入的数据都当做一个字符串，会对自动传入的数据加一个双引号
- \$符号将传入的数据**直接显示**在生成的SQL语句中。
- #符号**存在预编译的过程**，对问号赋值，防止SQL注入。
- \$符号是**直译**的方式，一般用在**order by \${列名}**语句中。

(7) MyBatis的缓存机制有了解吗？

答：MyBatis的缓存机制分为一级和二级缓存，分别介绍如下：

- **一级缓存（同一个SqlSession）：**

基于 HashMap 的本地缓存，其存储作用域为 Session，当 Session flush 或 close 之后，该 Session 中的所有缓存就将清空，默认打开一级缓存。

- **二级缓存（同一个SqlSessionFactory）：**

二级缓存与一级缓存其机制相同，默认也是采用 HashMap 的本地存储，不同在于其存储作用域为 Mapper(Namespace)，并且可自定义存储源。

(8) MyBatis的接口绑定是什么？有哪些实现方式？

答：接口绑定就是在MyBatis中任意定义接口,然后把接口里面的方法和SQL语句绑定,我们直接调用接口方法就可以。

接口绑定有两种实现方式：

- 通过注解绑定，就是在接口的方法上面加上 @Select、@Update等注解，里面包含SQL语句来绑定
- 通过xml里面写SQL来绑定,在这种情况下,要指定xml映射文件里面的namespace必须为接口的全路径名。

接口绑定方式的选择：

当SQL语句比较简单时候，用注解绑定；当SQL语句比较复杂时候，用xml绑定；实际的开发中，我们一般都使用xml绑定方式。

总结:

本小节中我们主要针对热门持久层框架MyBatis进行了交流与学习，相对于Spring框架，MyBatis在面试中考察的较少。建议我们没有项目可做的同学，使用Spring+MyBatis来搭建一个完整的服务，做一个简单的项目，毕竟学习还是得靠多多动手总结。下一小节开始将进入第三大章，主要介绍Redis和Kafka相关知识。

限于作者水平，文章中难免会有不妥之处。大家在学习过程中遇到我没有表达清楚或者表述有误的地方，欢迎随时在文章下边指出，我会及时关注，随时改正。另外，大家有任何话题都可以在下边留言，我们一起交流探讨。

讨论

评论

NJUPT_csy1#

还更新嘛？

发表于 2019-12-11 00:20:14

赞(0) 回复(1)

我是祖国的花朵  ： 本周还会更新的，加油

2019-12-11 08:11:49

赞(0) 回复(0)

请输入你的观点

回复

心非2#

大佬别鸽啊，求更

发表于 2019-12-12 01:54:05

赞(0) 回复(1)

我是祖国的花朵  ： 已经更新，加油~

2019-12-12 20:34:50

赞(0) 回复(0)

请输入你的观点

回复

Albattoss19973#

面试的时候经常问我连接池配置 我觉得也算高频了


发表于 2019-12-13 15:21:45

赞(2) 回复(2)

我是祖国的花朵  ： 嗯嗯，我们在Demo中给出了一个简单的配置。主要是驱动，用户名密码以及其余连接池属性的配置。

2019-12-15 16:24:30

赞(0) 回复(0)

四月hope 回复 我是祖国的花朵  ： 现在用mybatis用这样吗？用springboot整合mybatis没那么复杂啊

请输入你的观点

回复

9



从心出发

4#

老哥 请问为什么会报 sqlsession错误啊，然后 找了一下usermapper.xml也没发现错误

发表于 2019-12-16 09:17:10

赞(1) 回复(2)

我是祖国的花朵 : 嗯，可以给出具体点的使用场景和报错信息，一起看下。

2019-12-16 21:07:49

赞(0) 回复(0)

从心出发 回复 我是祖国的花朵 : 谢谢啦，我已经解决了

2019-12-17 11:49:17

赞(0) 回复(0)

请输入你的观点

回复



刘畅201904211606816

5#

打卡

发表于 2020-01-05 22:15:58

赞(0) 回复(0)



牧水s

6#

打卡

发表于 2020-02-14 22:08:24

赞(0) 回复(0)



小源20190118151956

7#

打卡

发表于 2020-02-18 22:43:38

赞(0) 回复(0)



赚多多

8#

你好，请问校招对框架的考察多吗，比如像mybatis这种常用框架需要掌握到什么程度

发表于 2020-05-11 21:36:51

赞(0) 回复(1)

我是祖国的花朵 : 重点掌握spring基础知识点，这个mybatis框架考察不多

2020-05-11 21:46:04

赞(0) 回复(0)

请输入你的观点

回复



ttbh

9#

大佬，是否可以写点java web和hibernate的相关内容呢

发表于 2020-05-25 16:37:03

赞(0) 回复(0)