

目录

- [[#什么是 JdbcTemplate|什么是 JdbcTemplate]]
- [[#使用|使用]]
 - [[#使用#引入依赖|引入依赖]]
 - [[#使用#在 spring 配置文件配置数据库连接池|在 spring 配置文件配置数据库连接池]]
 - [[#在 spring 配置文件配置数据库连接池#将数据参数写信写入jdbc.properties|将数据参数写信写入jdbc.properties]]
 - [[#在 spring 配置文件配置数据库连接池#在xml配置文件中引入jdbc.properties|在xml配置文件中引入jdbc.properties]]
 - [[#在 spring 配置文件配置数据库连接池#在xml配置文件中创建druid数据库连接池|在xml配置文件中创建druid数据库连接池]]
 - [[#使用#配置 JdbcTemplate 对象，注入 DataSource|配置 JdbcTemplate 对象，注入 DataSource]]
 - [[#使用#创建数据库表以及创建对应的类|创建数据库表以及创建对应的类]]
 - [[#创建数据库表以及创建对应的类#创建数据库表t_man|创建数据库表t_man]]
 - [[#创建数据库表以及创建对应的类#实体类|实体类]]
 - [[#创建数据库表以及创建对应的类#Dao接口、实现类以及Service类|Dao接口、实现类以及Service类]]
 - [[#使用#开启组件扫描并将对应的类加上注解|开启组件扫描并将对应的类加上注解]]
 - [[#使用#JdbcTemplate 操作数据库（添加） |JdbcTemplate 操作数据库（添加）]]
 - [[#使用#查询某个值|查询某个值]]
 - [[#使用#根据条件查询返回某个对象|根据条件查询返回某个对象]]
 - [[#使用#查询对象集合|查询对象集合]]

什么是 JdbcTemplate

Spring 框架对 JDBC 进行封装，使用 [JdbcTemplate](#) 方便实现对数据库操作。

使用

引入依赖

我这里将之前的项目中的依赖也放进来了，有的可能用不到。

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.16</version>
  </dependency>
  <dependency>
    <groupId>commons-logging</groupId>
    <artifactId>commons-logging</artifactId>
    <version>1.2</version>
  </dependency>
  <dependency>
    <groupId>aopalliance</groupId>
    <artifactId>aopalliance</artifactId>
    <version>1.0</version>
  </dependency>
  <dependency>
    <groupId>cglib</groupId>
    <artifactId>cglib</artifactId>
    <version>3.3.0</version>
  </dependency>

  <dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>1.9.7</version>
  </dependency>

  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aspects</artifactId>
    <version>5.3.16</version>
  </dependency>
  <dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>druid</artifactId>
    <version>1.1.9</version>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.19</version>
  </dependency>
</dependencies>
```

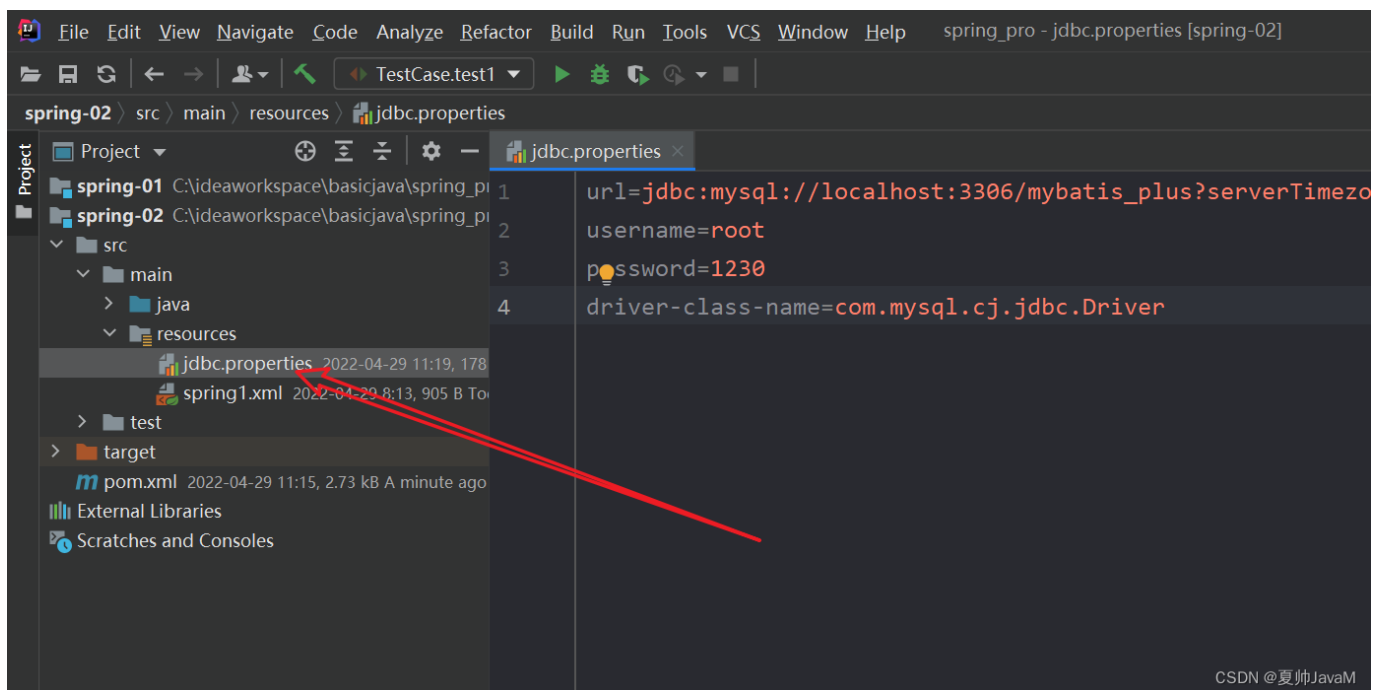
```

        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>5.3.16</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifactId>
        <version>5.3.16</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
        <version>5.3.16</version>
    </dependency>
</dependencies>

```

在 spring 配置文件配置数据库连接池

将数据参数写信写入jdbc.properties



jdbc.properties内容

```

url=jdbc:mysql://localhost:3306/mybatis_plus?
serverTimezone=GMT%2B8&characterEncoding=utf-8&useSSL=false
username=root
password=1230
driver-class-name=com.mysql.cj.jdbc.Driver

```

在xml配置文件中引入jdbc.properties

```
<context:property-placeholder location="classpath:jdbc.properties"></context:property-placeholder>
```

在xml配置文件中创建druid数据库连接池

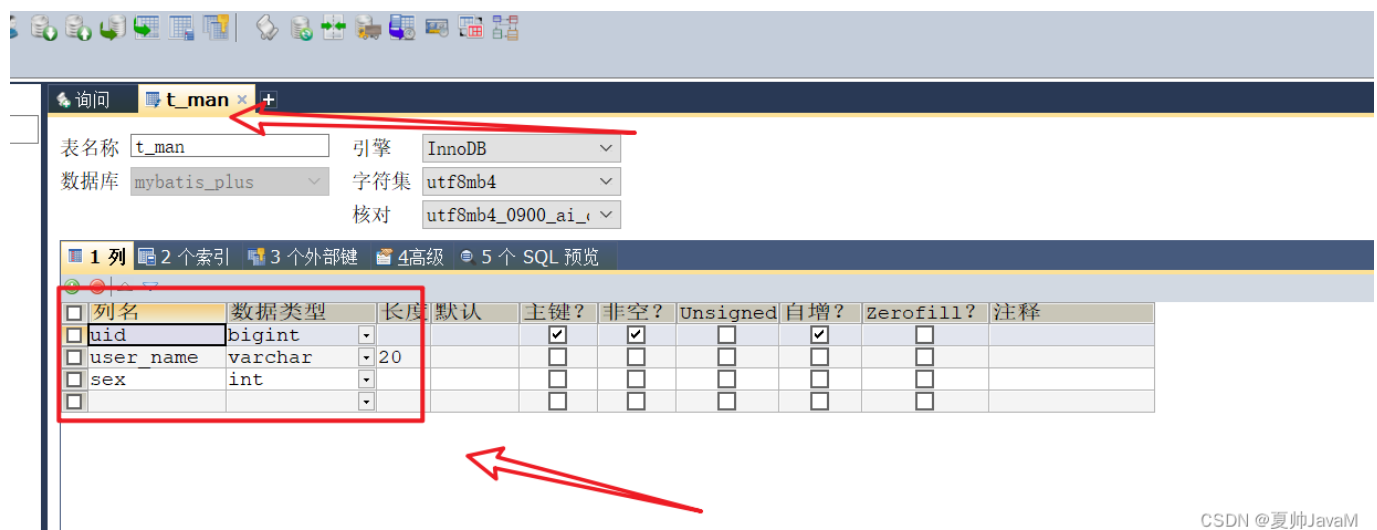
```
<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
    <property name="username" value="${username}"></property>
    <property name="password" value="${password}"></property>
    <property name="url" value="${url}"></property>
    <property name="driverClassName" value="${driver-class-name}"></property>
</bean>
```

配置 JdbcTemplate 对象，注入 DataSource

```
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="dataSource"/>
</bean>
```

创建数据库表以及创建对应的类

创建数据库表t_man




表名称: t_man 引擎: InnoDB 数据库: mybatis_plus 字符集: utf8mb4 核对: utf8mb4_0900_ai_ci

列名	数据类型	长度	默认	主键?	非空?	Unsigned	自增?	Zerofill?	注释
uid	bigint			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
user_name	varchar	20		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
sex	int			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

CSDN @夏帅JavaM

实体类



```
@Data
public class Man {
    private Integer uid;
    private String userName;
    private Integer sex;
}
```

加入这个标签可以自动提供getter以及setter方法
必须

- 1.在idea加入插件lombok
- 2.在pom中加入以下依赖

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.2</version>
</dependency>
```

CSDN @夏帅JavaM

Dao接口、实现类以及Service类

分别为 ManDao、ManDaoImpl、ManService 里面目前没什么方法

开启组件扫描并将对应的类加上注解

```
<context:component-scan base-package="com.csdn.dao,com.csdn.service">
</context:component-scan>
```

```
@Repository
public class ManDaoImpl implements ManDao {
}
```

CSDN @夏帅JavaM

```
@Service
public class ManService {
}
```

CSDN @夏帅JavaM

JdbcTemplate 操作数据库 (添加)

我们是在测试方法中调用Service对象的方法，然后Service调用Dao的实现类的方法来实现的，因此我们需要现在Dao以及Service中增加方法并且在Service中注入Dao对象，在Dao中需要注入JdbcTemplate来实现方法。

Dao接口

```
public interface ManDao {  
  
    int addEntity(Man man);  
}
```

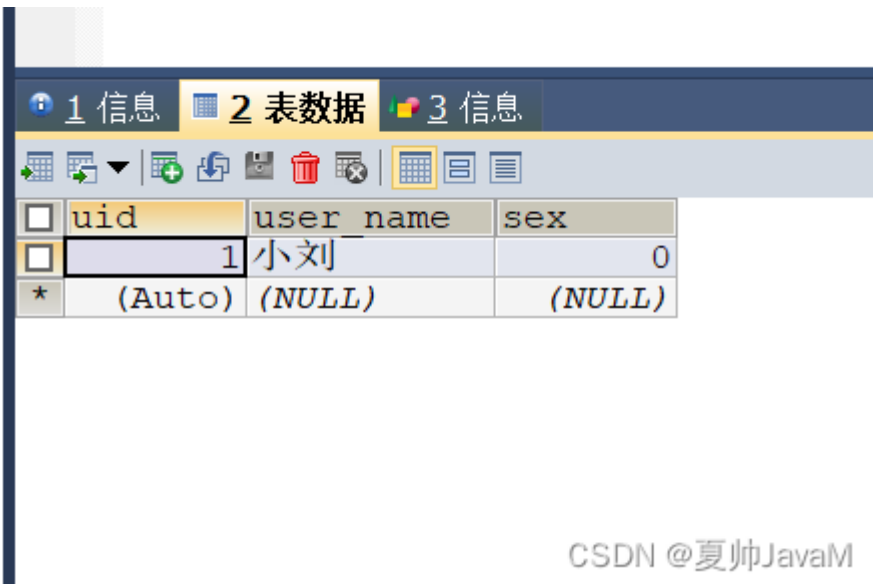
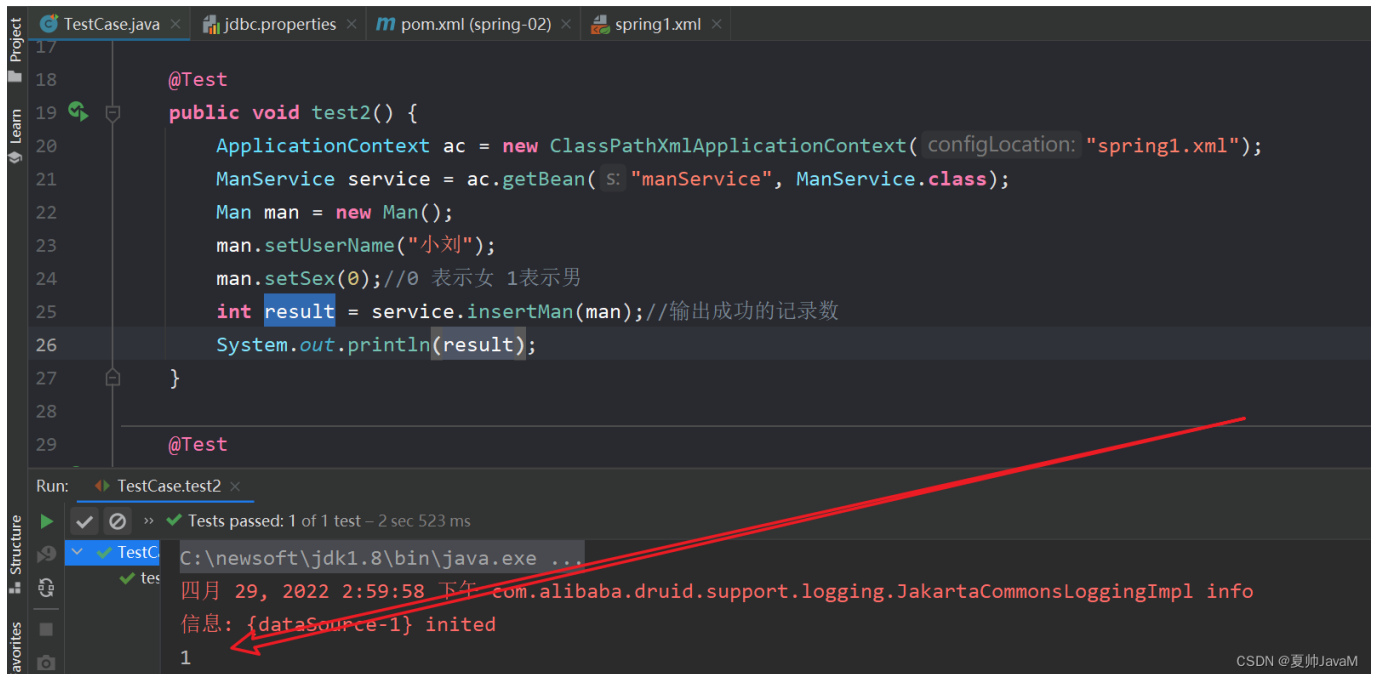
Dao实现类

```
@Repository  
public class ManDaoImpl implements ManDao {  
    @Autowired  
    private JdbcTemplate jdbcTemplate;  
  
    @Override  
    public int addEntity(Man man) {  
        String sql = "insert into t_man(user_name,sex) values(?,?)";  
        int update = jdbcTemplate.update(sql, man.getUserName(), man.getSex());  
        return update;  
    }  
}
```

Service类

```
@Service  
public class ManService {  
    @Autowired  
    private ManDao dao;  
    public int insertMan(Man man) {  
        return dao.addEntity(man);  
    }  
}
```

测试



修改与删除调用的都是差不多这里不再写。

查询某个值

查询id为1的 username

Dao接口中加入以下方法

```
String getNameById(int id);
```

Dao实现类实现上述方法

```

@Override
public String getNameById(int id) {
    String sql = "select user_name from t_man where uid = ?" ;
    return jdbcTemplate.queryForObject(sql,String.class,id);
}

```

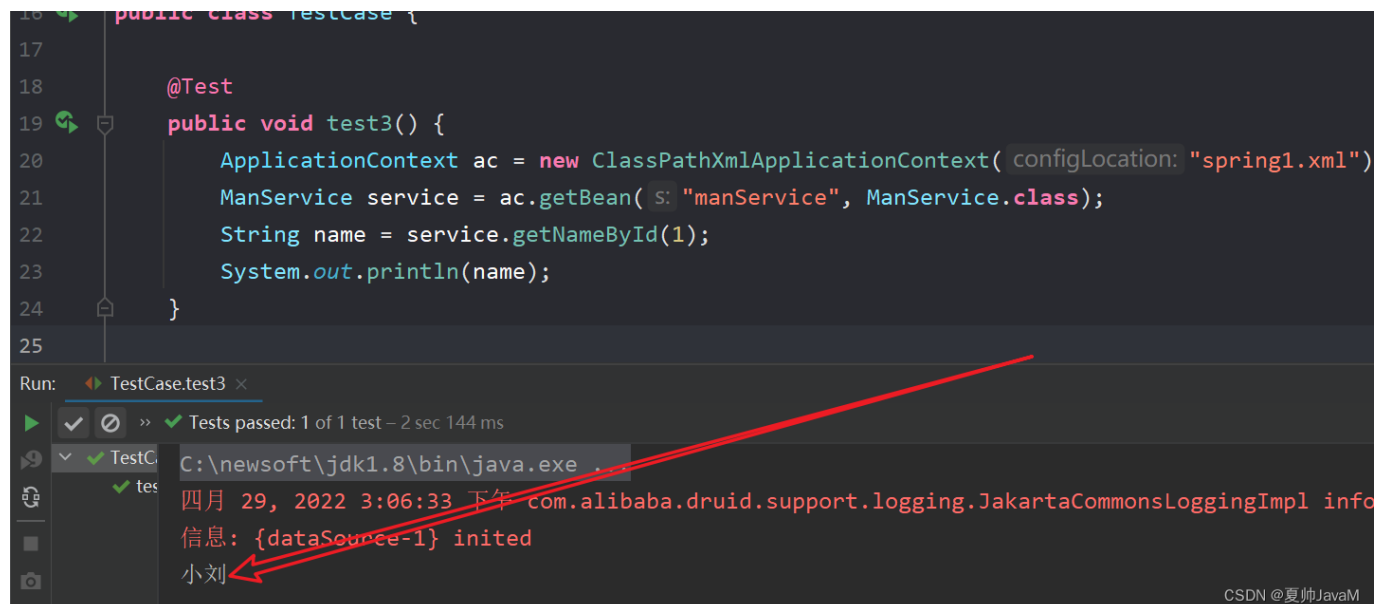
Service中调用

```

public String getNameById(int id) {
    return dao.getNameById(id);
}
}

```

测试



The screenshot shows an IDE with a Java test class and its execution output. The test class, `TestCase`, contains a `@Test` method `test3()` that creates a `ClassPathXmlApplicationContext` with `spring1.xml`, retrieves a `ManService` bean, and calls `getNameById(1)` to print the name. The output console shows the test passed and a log message: `com.alibaba.druid.support.logging.JakartaCommonsLoggingImpl info: {dataSource-1} inited`. A red arrow points from the log message to the text '小刘' (Xiao Liu) in the output, which is the name returned by the service.

```

16 public class TestCase {
17
18     @Test
19     public void test3() {
20         ApplicationContext ac = new ClassPathXmlApplicationContext( configLocation: "spring1.xml")
21         ManService service = ac.getBean( S: "manService", ManService.class);
22         String name = service.getNameById(1);
23         System.out.println(name);
24     }
25

```

Run: `TestCase.test3` ×

Tests passed: 1 of 1 test – 2 sec 144 ms

TestC: `C:\newsoft\jdk1.8\bin\java.exe`

四月 29, 2022 3:06:33 下午 com.alibaba.druid.support.logging.JakartaCommonsLoggingImpl info: {dataSource-1} inited

小刘

CSDN @夏帅JavaM

根据条件查询返回某个对象

加入如下方法

Dao中方法

```
Man getEntityById(int id);
```

实现类实现方法

```
@Override
public Man getEntityById(int id) {
    String sql = "select uid, user_name userName ,sex from t_man where uid =?";
    return jdbcTemplate.queryForObject(sql,new BeanPropertyRowMapper<Man>
(Man.class),id);
}
```

Service中调用

```
public Man getManById(int id) {
    return dao.getEntityById(id);
}
```

测试

```
public class Testcase {
    @Test
    public void test4() {
        ApplicationContext ac = new ClassPathXmlApplicationContext( configLocation: "spring1.xml");
        ManService service = ac.getBean( S: "manService", ManService.class);
        Man man = service.getManById(1);
        System.out.println(man);
    }
}
```

Debug: TestCase.test4 ×

Debugger Console

Tests passed: 1 of 1 test - 3 sec 139 ms

C:\newsoft\jdk1.8\bin\java.exe ...

Connected to the target VM, address: '127.0.0.1:63529', transport: 'socket'

四月 29, 2022 3:15:44 下午 com.alibaba.druid.support.logging.JakartaCommonsLoggingImpl info

信息: {dataSource-1} inited

Man(uid=1, userName=小刘, sex=0)

CSDN @夏帅JavaM

查询对象集合

对象 t_man @mybatis_plus (127....		
开始事务 备注 筛选 排序 导入 导出		
uid	user_name	sex
1	小刘	0
2	小王	1

CSDN @夏帅JavaM

代码

Dao方法

```
List<Man> findAll();
```

实现类实现方法

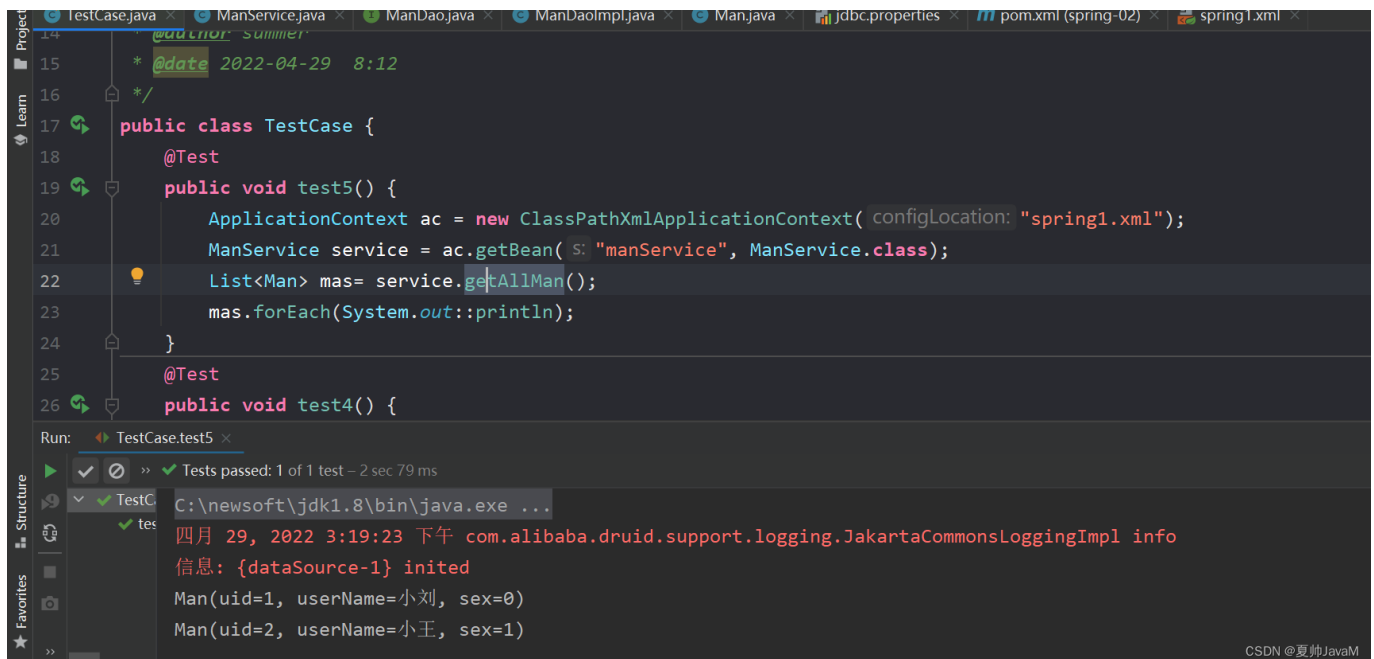
```
@Override
```

```
public List<Man> findAll() {  
    String sql = "select uid, user_name userName ,sex from t_man";  
    return jdbcTemplate.query(sql,new BeanPropertyRowMapper<Man>(Man.class));  
}
```

Service中方法调用

```
public List<Man> getAllMan() {  
    return dao.findAll();  
}
```

测试



The screenshot shows an IDE with several tabs open: TestCase.java, ManService.java, ManDao.java, ManDaoImpl.java, Man.java, jdbc.properties, pom.xml (spring-02), and spring1.xml. The TestCase.java file is open, showing a public class with two test methods: test5() and test4(). The test5() method is annotated with @Test and contains the following code:

```
public void test5() {  
    ApplicationContext ac = new ClassPathXmlApplicationContext( configLocation: "spring1.xml");  
    ManService service = ac.getBean( S: "manService", ManService.class);  
    List<Man> mas= service.getAllMan();  
    mas.forEach(System.out::println);  
}
```

The Run window at the bottom shows the execution of the test. It indicates that the test passed (Tests passed: 1 of 1 test - 2 sec 79 ms). The output of the test is as follows:

```
C:\newsoft\jdk1.8\bin\java.exe ...  
四月 29, 2022 3:19:23 下午 com.alibaba.druid.support.logging.JakartaCommonsLoggingImpl info  
信息: {dataSource-1} inited  
Man(uid=1, userName=小刘, sex=0)  
Man(uid=2, userName=小王, sex=1)
```

The bottom right corner of the screenshot contains the text "CSDN @夏帅JavaM".