

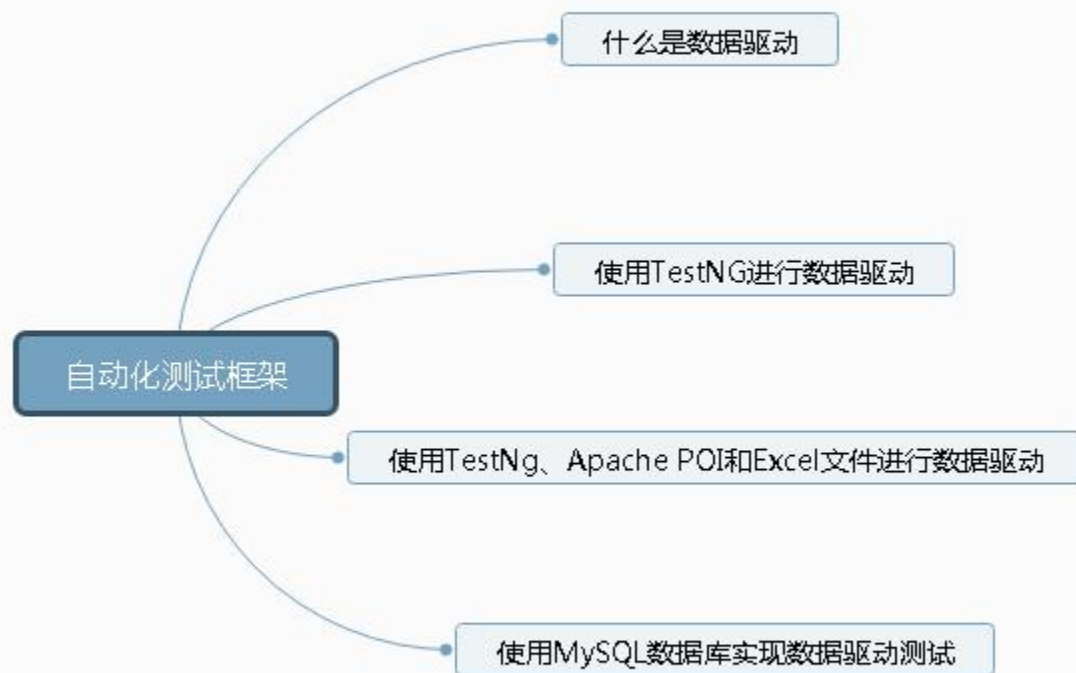
WebDriver API 详解二

讲师：芳姐

金斧子

| 5 金斧子五周年
ANNIVERSARY

科技提升投资品质



一、什么是数据驱动

相同的测试脚本使用不同的测试数据来执行，测试数据和测试行为进行了完全的分离，这样的测试脚本设计模式称为数据驱动。例如，测试网站的登录功能，自动化测试工程师想验证不同的用户名和密码在网站登录时的系统影响结果，就可以使用数据驱动模式来进行自动化测试。

实施数据驱动测试的步骤如下：

- (1) 编写测试脚本，脚本需要支持程序对象、文件或者数据库读入测试数据。
- (2) 将测试脚本使用的测试数据存入程序对象、文件或者数据库等外部介质中。
- (3) 运行脚本，循环调用存储在外部介质的测试数据。
- (4) 验证所有的测试结果是否符合期望的结果。

二、使用TestNG进行数据驱动

测试逻辑:

- (1) 打开erp登录首页
- (2) 输入用户名和密码
- (3) 点击登录按钮
- (4) 登录成功页面是否包含用户姓名信息, 包含则认为测试执行成功, 否则认为测试执行失败

Java代码:

```
public class DdataProviderTest {  
  
    private static WebDriver driver;  
    @DataProvider(name = "user")  
    public static Object[][] words()  
    {  
        return new Object[][] {  
            {"defang1", "123456", "德芳理财"},  
            {"defang2", "123", "德芳客服"},  
            {"defang3", "123", "德芳运维"}  
        };  
    }  
}
```

```
@Test(dataProvider = "user")
    public void test(String user1,String user2,String SearchResult)
    {
        System.setProperty("webdriver.chrome.driver",
"D:\\Selenium_Automated\\driver\\chromedriver.exe");
        driver = new ChromeDriver();
        //设定等待时间为10秒
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        driver.get("http://10.1.2.211:8080/login.jsp");

        // 文本框内输入用户名
        driver.findElement( By.name( "username" ) ).sendKeys( user1 );
        // 文本框内输入密码
        driver.findElement( By.name( "password" ) ).sendKeys( user2 );
        // 点击登录
        driver.findElement( By.className( "submit_wrap" ) ).click();
        try {
            Thread.sleep(3000);

        } catch
            (InterruptedException e) {
                e.printStackTrace();
            }
    }
```

```
//判断搜索结果页面是否包含测试数据中期望的关键词
Assert.assertTrue(driver.getPageSource().contains(SearchResult));
driver.quit();

}

}
```

测试结果:

Downloader

in NIO DdataProviderTest

▶ [Icons] All 3 tests passed - 39s 1

▼ [✓] Default Suite	39s 105ms	Starting ChromeDriver 2.33.506120 (e3e53437346286c0bc2d2dc9aa4915ba81d9023f) on port 11712 Only local connections are allowed. log4j:WARN No appenders could be found for logger (org.apache.http.client.protocol.RequestAddCookie). log4j:WARN Please initialize the log4j system properly. log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info. 一月 26, 2018 6:48:31 下午 org.openqa.selenium.remote.ProtocolHandshake createSession 信息: Detected dialect: OSS
▼ [✓] Selenium_Automated	39s 105ms	
▼ [✓] DdataProviderTest	39s 105ms	
[✓] test[defang1, 123456, 德芳理财]	16s 895ms	
[✓] test[defang2, 123, 德芳客服] (1)	9s 990ms	
[✓] test[defang3, 123, 德芳运维] (2)	12s 220ms	

代码解释:

测试脚本会自动打开三次浏览器，分别输入三组不同词作为用户名进行登录，并且三次的结果均断言成功。

```
@DataProvider(name = "user")
public static Object[][] words()
{
    return new Object[][] {
        {"defang1","123456","德芳理财"},
        {"defang2","123","德芳客服"},
        {"defang3","123","德芳运维"}
    };
}
```

使用@DataProvider注解定义当前方法中的返回对象作为测试脚本的测试数据集，并且将测试数据集命名为“user”。

```
@Test(dataProvider = "user")
public void test(String user1,String user2,String SearchResult)
```

上述代码表示测试方法中的三个参数分别使用user测试数据集中的每个一维数组中的数据进行赋值。此测试方法会被调用三次，分别使用测试数据集中的三组数据，如下显示：

第一次调用使用的参数值：

```
user1="defang1"  
user2="123456"  
SearchResult="德芳理财"
```

第二次调用使用的参数值：

```
user1="defang2"  
user2="123"  
SearchResult="德芳客服"
```

第三次调用使用的参数值：

```
user1="defang3"
```


其中， `searchWord1` 和 `searchWord2` 分别作为登录框输入的用户名进行输入，`SearchResult` 用来 判断登录成功之后， 页面是否包含期望的用户名关键词。

`Assert. assertTrue(driver. getPageSource(). contains(SearchResult));`
此代码行用于判断登录成功的 页面 中 是否 包含 测试 数据 中 期望 的 关键词。

三、使用TestNg、Apache POI和Excel文件进行数据驱动

测试逻辑：

- (1) 打开erp登录首页
- (2) 输入用户名和密码
- (3) 点击登录按钮
- (4) 登录成功页面是否包含用户姓名信息，包含则认为测试执行成功，否则认为测试执行失败

测试环境准备：

在pom.xml配置文件里面进行配置

<dependency>

 <groupId>org.apache.poi</groupId>

 <artifactId>poi</artifactId>

 <version>3.16</version>

</dependency>

<dependency>

 <groupId>org.apache.poi</groupId>

 <artifactId>poi-examples</artifactId>

 <version>3.16</version>

</dependency>

pom.xml

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-excelant</artifactId>
    <version>3.16</version>
</dependency>
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>3.16</version>
</dependency>
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml-schemas</artifactId>
    <version>3.16</version>
</dependency>
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-scratchpad</artifactId>
    <version>3.16</version>
</dependency>
```

Java测试代码

```
@DataProvider(name="testData")
public static Object[][] data() throws IOException
{
    //调用类中的静态方法getTestData, 获取Excel文件的测试数据
    return getTestData("D:\\Selenium_Automated\\testdata", "testdata.xlsx",
"testdata");
}
@Test(dataProvider="testData")
public void testSearch(String user1,String user2,String Result) {
    //打开erp首页
    driver.get(baseUrl);

    //使用数据库测试数据库表中每个数据行的前两列数据作为登录用户名和密码

    // 文本框内输入用户名
    driver.findElement(By.name("username")).sendKeys(user1);
    // 文本框内输入密码
    driver.findElement(By.name("password")).sendKeys(user2);
    // 点击登录
    driver.findElement(By.className("submit_wrap")).click();
}
```

```
//判断搜索结果页面是否包含测试数据中期望的关键词
Assert.assertTrue(driver.getPageSource().contains(Result));
}
@BeforeMethod
public void beforeMethod() {

    System.setProperty("webdriver.chrome.driver",
"D:\\Selenium_Automated\\driver\\chromedriver.exe");
    driver = new ChromeDriver();
    //设定等待时间为5秒
    driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
}

@AfterMethod
public void afterMethod() {
    //关闭打开的浏览器
    driver.quit();
}
```

```
//从Excel文件获取测试数据的静态方法
public static Object[][] getTestData(String filePath,String FileName,String sheetName) throws
IOException{
    //根据参数传入的数据文件路径和文件名称，组合出Excel数据文件的绝对路径，声明一个File文件对象
    File file = new File(filePath + "\\ " + FileName);
    //创建FileInputStream对象用于读取Excel文件
    FileInputStream inputStream = new FileInputStream(file);
    //声明workbook对象
    Workbook Workbook = null;
    //获取文件名参数的扩展名，判断是.xlsx文件还是.xls文件
    String fileExtensionName = FileName.substring(FileName.indexOf("."));
    //判断文件类型如果是.xlsx，则使用XSSFWorkbook对象进行实例化
    //判断文件类型如果是.xls，则使用ssfworkbook对象进行实例化
    if (fileExtensionName.equals(".xlsx")) {
        Workbook = new XSSFWorkbook(inputStream);
    } else if (fileExtensionName.equals(".xls")) {
        Workbook = new HSSFWorkbook(inputStream);
    }
    //通过sheetName参数，声称Sheet对象
    Sheet Sheet = Workbook.getSheet(sheetName);
    //获取Excel数据文件Sheet1中数据的行数，getLastRowNum()方法获取数据的最后一行行号
    //getFirstRowNum()方法获取数据的第一行行号，相减之后得出数据的行数，Excel文件的行号和列号都
    是从0开始
    int rowCount = Sheet.getLastRowNum() - Sheet.getFirstRowNum();
```

```
//创建名为records的list对象来存储从Excel数据文件读取的数据
List<Object[] > records = new ArrayList<Object[] >();
//循环遍历Excel数据文件的所有数据，除了第一行，第一行是数据列名称
for (int i = 1; i < rowCount + 1; i++) {
    //使用getShow方法获取行对象
    Row row = Sheet.getRow(i);
    //声明一个数组，存储Excel数据文件每行中的3个数据，数组的大小用getLastCellNum()方法进行动态声明，实现测试数据个数和数组大小一致
    String fields[] = new String[row.getLastCellNum()];
    for (int j = 0; j < row.getLastCellNum(); j++) {
        //使用getCell()和getStringCellValue()方法获取Excel文件中的单元格数据
        ////先设置getCell的类型，然后就可以把纯数字作为String类型读进来了：
        row.getCell(j).setCellType(Cell.CELL_TYPE_STRING);
        fields[j] = row.getCell(j).getStringCellValue();
        //函数返回指定单元格的字符串内容

    }
    //将fields的数据对象存入records的list中
    records.add(fields);
}
//定义返回值，即Object[][]
// 将存储测试数据的List转换为一个Object的二维数组
Object[][] results = new Object[records.size()][];
// 设置二维数组每行的值，每行是一个Object对象
for (int i = 0; i < records.size(); i++) {
    results[i] = records.get(i);
}
return results;
```

TestDataByExcelFile / getTestData()

TestDataByExcelFile

1 test passed - 8s 877ms

Test Suite	Duration	Log Output
Default Suite	8s 877ms	"D:\Program Files\Java\jdk1.8.0_101\bin\java" ...
Selenium_Automated	8s 877ms	Starting ChromeDriver 2.33.506120 (e3e53437346286c0bc2d2dc9aa4915ba81d9023f) on port 47788
TestDataByExcelFile	8s 877ms	Only local connections are allowed.
testSearch[defang1, 123456, 德芳理]	4s 442ms	log4j:WARN No appenders could be found for logger (org.apache.http.client.protocol.RequestAddCookies). log4j:WARN Please initialize the log4j system properly. log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info. 一月 31, 2018 3:27:35 下午 org.openqa.selenium.remote.ProtocolHandshake createSession 信息: Detected dialect: OSS

Tests Passed: 1 passed (5 minutes ago)

三.使用mysql数据库实现数据驱动测试

#测试逻辑:

- (1) 打开erp登录首页
- (2) 输入用户名和密码
- (3) 点击登录按钮
- (4) 登录成功页面是否包含用户姓名信息, 包含则认为测试执行成功, 否则认为测试执行失败

测试环境准备:

- (1) 已经安装好mysql数据库
- (2) 在pom.xml配置文件里面配置好MySQL驱动的信息

```
<dependency>
```

```
    <groupId>mysql</groupId>
```

```
    <artifactId>mysql-connector-java</artifactId>
```

```
    <version>6.0.6</version>
```

```
</dependency>
```

- (3) 创建一个测试数据库

```
create database testdb;
```

- (4) 执行如下sql语句创建测试数据表

```
CREATE TABLE testdata (Movie_Name char(30),Movie_Property char(30),Expect_result char(30));
```

- (5) 插入测试数据

```
INSERT into testdata (Movie_Name,Movie_Property,Expect_result) VALUES ("defang2","123","德芳运维");
```

Java代码：

```
public WebDriver driver;
String baseUrl = "http://10.1.2.211:8080/login.jsp";
//使用注解DataProvider，将数据集合命名为“testData”
@DataProvider(name = "testData")
public static Object[][] words() throws IOException {
    //调用类中的静态方法getTestData，获取MySQL数据库中的测试数据
    return getTestData("testdata");
}

@Test(dataProvider = "testData")
public void testSearch (String user1,String user2,String Result) {
    //打开erp首页
    driver.get(baseUrl);

    //使用数据库测试数据库表中每个数据行的前两列数据作为登录用户名和密码

    // 文本框内输入用户名
    driver.findElement(By.name("username")).sendKeys(user1);
    // 文本框内输入密码
    driver.findElement(By.name("password")).sendKeys(user2);
    // 点击登录
    driver.findElement(By.className("submit_wrap")).click();
}
```

```
//判断搜索结果页面是否包含测试数据中期望的关键词
Assert.assertTrue(driver.getPageSource().contains(Result));
@BeforeMethod public void beforeMethod(){
    System.setProperty("webdriver.chrome.driver",
"D:\\Selenium_Automated\\driver\\chromedriver.exe");
    driver = new ChromeDriver();
}
@AfterMethod public void afterMethod(){
    driver.quit();
}
public static Object [][] getTestData(String tablename) throws IOException{
    //声明MySQL数据库的驱动
    String driver = "com.mysql.jdbc.Driver";
    //声明本地数据库的IP地址和数据库名称
    String url = "jdbc:mysql://10.1.2.71:3306/testdb";

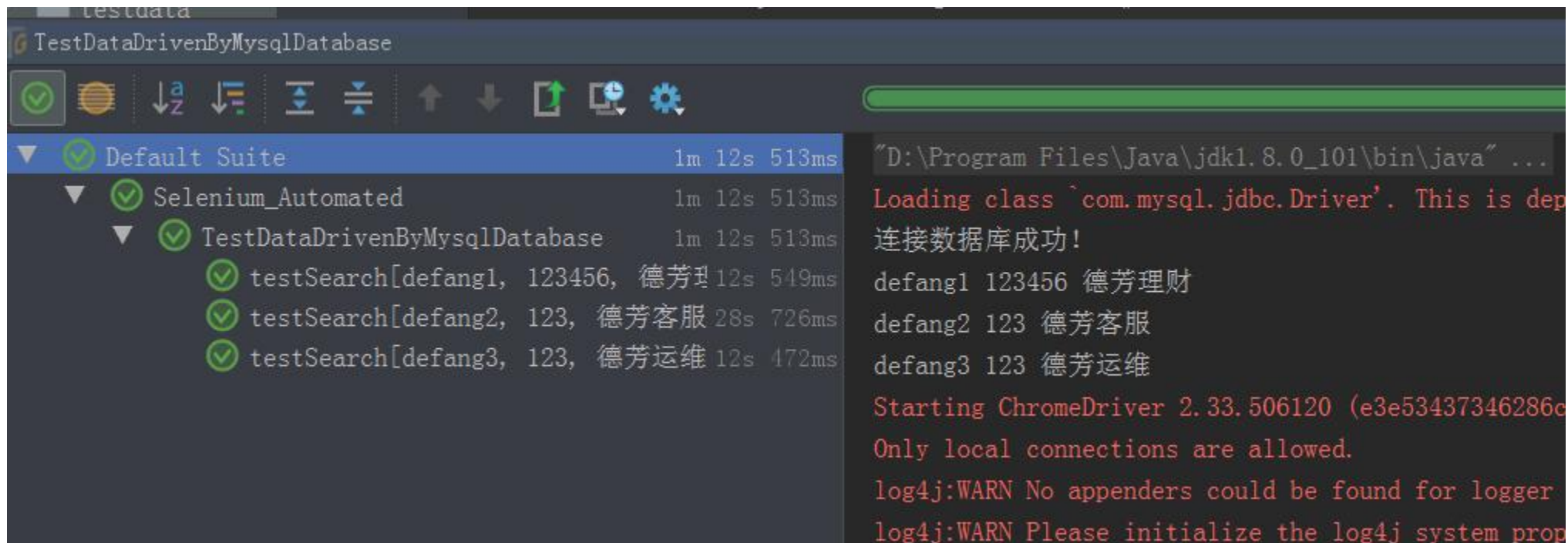
    //声明数据库的用户名
    String user = "root";
    //声明数据库用户名的登录密码
    String password = "testjfz";
    //声明存储测试数据的list对象
    List<Object[] > records = new ArrayList<Object[] >();
    try {

        //设定驱动
        Class.forName(driver);
```

```
//如果数据库链接可用，打印数据库连接成功的信息
    if(!conn.isClosed())
        System.out.println("连接数据库成功！");
//创建statement对象
Statement statement = conn.createStatement();
//使用函数参数拼接要执行的sql语句，此语句用来获取数据表的所有数据行
String sql = "SELECT * FROM " + tablename;
//声明ResultSet对象，存取执行sql语句返回的数据结果集
ResultSet rs = statement.executeQuery(sql);
//声明一个ResultSetMetadata对象
ResultSetMetaData rsMetaData = rs.getMetaData();
//调用ResultSetMetadata对象的getColumnCount方法获取数据行的列数
int cols = rsMetaData.getColumnCount();
//使用next方法遍历数据结构集中的所有数据行
while (rs.next()) {
    //声明一个字符型数据，数组大小使用数据行的列个数进行声明
    String fields[] = new String[cols];
    int col = 0;
    //遍历所有数据行中的所有列数据，并存储在字符数组中
    for (int colIdx = 0; colIdx < cols; colIdx++) {
        fields[col] = rs.getString(colIdx+1);
        col++;
    }
    //将每一行的数据存储在字符数据后，存储在records中
    records.add(fields);
    //输出数据行中的前三列内容，用于验证数据库内容是否正确取出
    System.out.println(rs.getString(1) + " " + rs.getString(2) + " " +
```

```
//关闭数据结果集对象
    rs.close();
    //关闭数据库链接
    conn.close();
} catch (ClassNotFoundException e) {
    System.out.println("未能找到MySQL的驱动类!");
    e.printStackTrace();
} catch (SQLException e) {
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
}

//定义函数返回值，即Object[][]
//将存储测试数据的list转换成为一个object的二维数组
Object[][] results = new Object[records.size()][];
//设置二维数组每行的值，每行是一个Object对象
for (int i = 0; i < records.size(); i++) {
    results[i] = records.get(i);
}
return results;
```



The screenshot shows the Selenium IDE interface with a test suite named 'Default Suite' and a test case 'Selenium_Automated'. The test case contains three test steps, all of which passed successfully. The test steps are: 'testSearch[defang1, 123456, 德芳理财]' (12s 549ms), 'testSearch[defang2, 123, 德芳客服]' (28s 726ms), and 'testSearch[defang3, 123, 德芳运维]' (12s 472ms). The test case 'Selenium_Automated' passed in 1m 12s 513ms. The test suite 'Default Suite' also passed in 1m 12s 513ms. The test data is stored in a database named 'TestDataDrivenByMysqlDatabase'.

Test Suite	Test Case	Test Step	Duration	Status
Default Suite	Selenium_Automated	testSearch[defang1, 123456, 德芳理财]	12s 549ms	Pass
Default Suite	Selenium_Automated	testSearch[defang2, 123, 德芳客服]	28s 726ms	Pass
Default Suite	Selenium_Automated	testSearch[defang3, 123, 德芳运维]	12s 472ms	Pass
Default Suite	Selenium_Automated		1m 12s 513ms	Pass
Default Suite			1m 12s 513ms	Pass

Log output:

```
"D:\Program Files\Java\jdk1.8.0_101\bin\java" ...  
Loading class `com.mysql.jdbc.Driver'. This is deprecated.  
连接数据库成功!  
defang1 123456 德芳理财  
defang2 123 德芳客服  
defang3 123 德芳运维  
Starting ChromeDriver 2.33.506120 (e3e53437346286c...  
Only local connections are allowed.  
log4j:WARN No appenders could be found for logger  
log4j:WARN Please initialize the log4j system prop
```

作业

- 1、Java代码实现TestNg实现数据驱动
- 2、Java代码实现TestNg+Excel实现数据驱动
- 3、Java代码实现MySQL数据库实现数据驱动

Thanks!

科技提升投资品质

