



《Web 程序设计》实验指导书

《Web 程序设计》课程组 编著

上海海洋大学海洋智能信息实验教学示范中心

实验一 访问数据库

一、实验目的

- 1、理解 JDBC 的工作原理
- 2、掌握 JDBC 的步骤和相关 API

二、实验环境

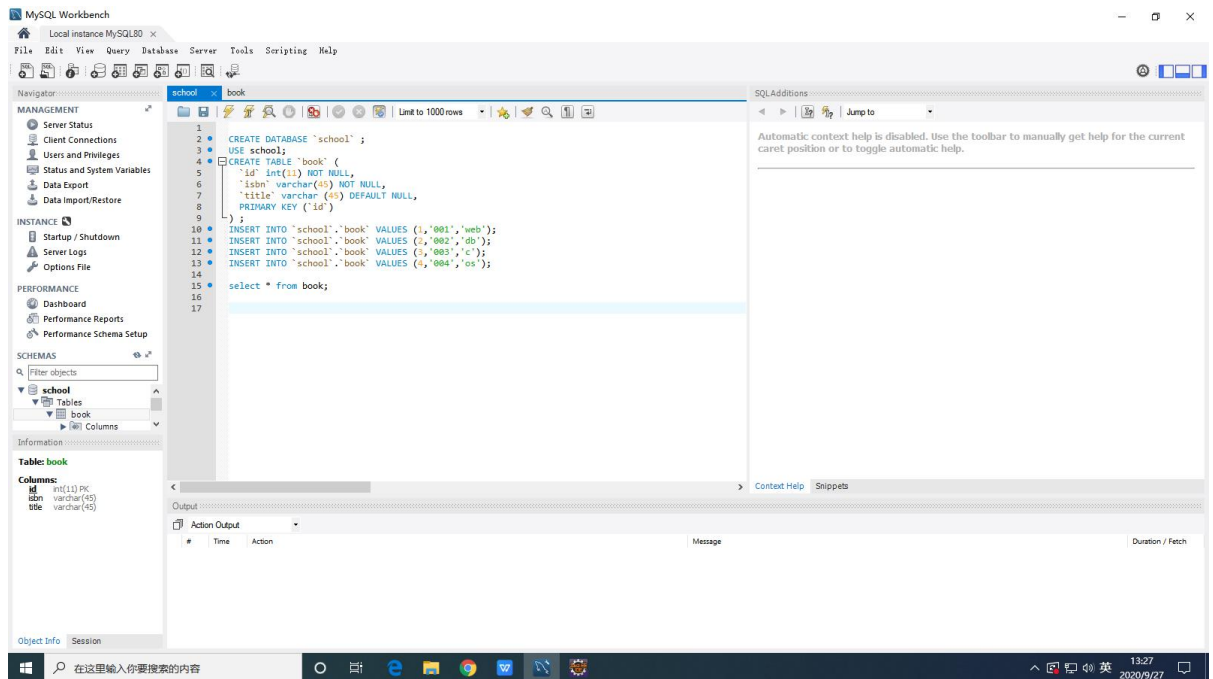
- 1、Java、MySQL 等。
- 2、Eclipse 集成开发环境

三、实验内容

1. 建立 mysql 数据库

新建 school 数据库，包含 book 表，该表结构如下。

Field	Type	Null	Key
id	int(11)	NO	PRI
isbn	varchar(45)	NO	
title	varchar(45)	YES	



2.在 jsp 页面中直接访问数据库数据

- 步骤：
- 1) 加载 jdbc 驱动
 - 2) 创建数据库连接
 - 3) 执行 sql 语句（注意有无参数语句的选择）
 - 4) 获得查询结果
 - 5) 关闭连接等

请参考泛雅平台资料中的课件、MySQLTest 和 7-1 项目等。

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" import="java.sql.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>第一次实验-JDBC</title>
</head>
<body>
欢迎使用本主页
<%
    //加载 jdbc 驱动
    try{
        Class.forName("com.mysql.cj.jdbc.Driver"); // mysql-connector-java 6
中
    }catch(ClassNotFoundException e){
        System.out.println("加载数据库驱动时抛出异常，内容如下：");
        e.printStackTrace();
    }
    //创建数据库连接
    Connection
conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/school?server
Timezone=GMT%2B8&useSSL=false", "root", "mysql");
    String sql="select * from book";
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery(sql);
    //如果有数据，rs.next()返回 true
    while(rs.next()){

        System.out.println("id:"+rs.getString(1)+";ISBN:"+rs.getString(2)+";Tit
le:"+rs.getString(3));
    }
    rs.close();
    stmt.close();

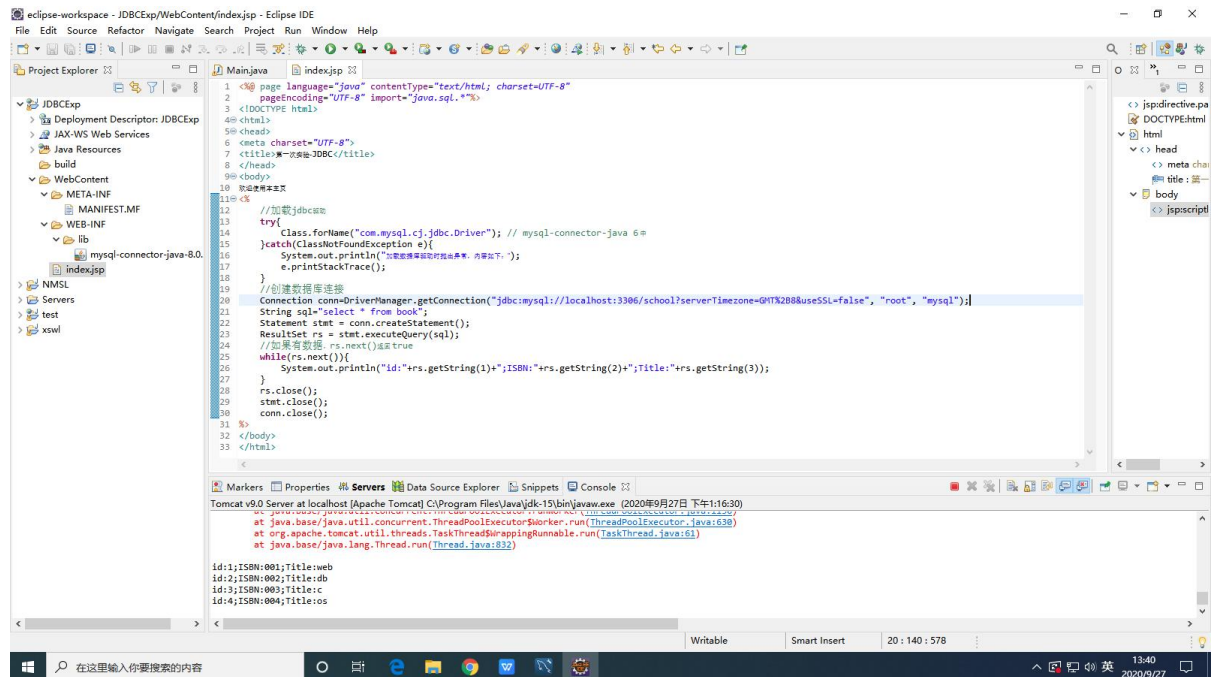
```

```

conn.close();
%>
</body>

</html>

```



四、实验练习：

1)请编写代码，实现根据 isbn 查询某书的信息

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" import="java.sql.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>第一次实验-JDBC</title>
</head>
<body>
欢迎使用本主页
<%
    //加载 jdbc 驱动
    try{
        Class.forName("com.mysql.cj.jdbc.Driver"); // mysql-connector-java 6
    }catch(ClassNotFoundException e){
        System.out.println("加载数据库驱动时抛出异常，内容如下：");

```

```

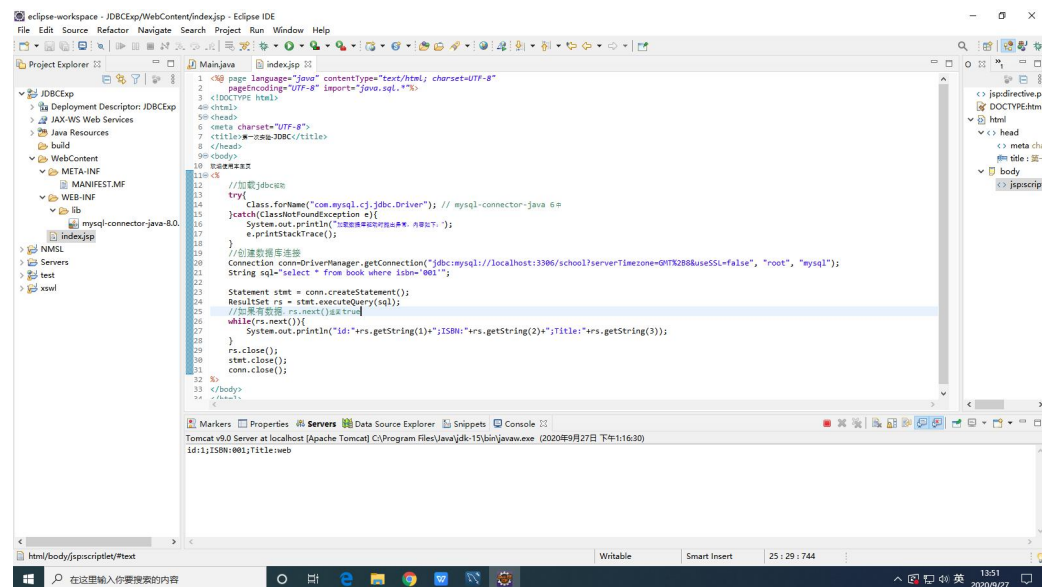
        e.printStackTrace();
    }
    //创建数据库连接
    Connection
conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/school?server
Timezone=GMT%2B8&useSSL=false", "root", "mysql");
    String sql="select * from book where isbn='001'";

    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery(sql);
    //如果有数据, rs.next()返回 true
    while(rs.next()){

        System.out.println("id:"+rs.getString(1)+"ISBN:"+rs.getString(2)+"Tit
le:"+rs.getString(3));
    }
    rs.close();
    stmt.close();
    conn.close();

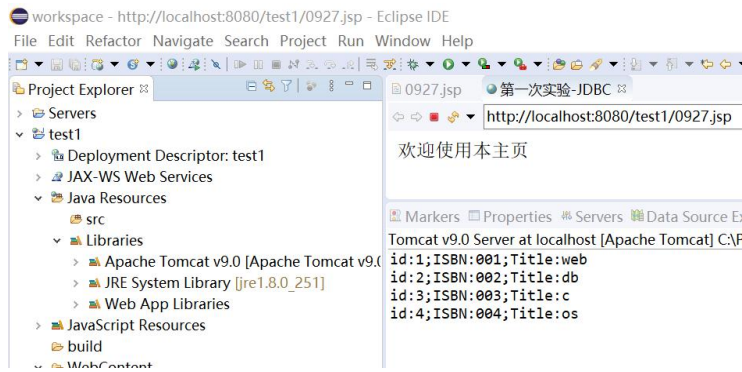
%>
</body>
</html>

```



2)请编写代码,实现图书信息的增加、删除和修改操作。

原来:



增加: 5, 005, python

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" import="java.sql.*"%>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>第一次实验-JDBC</title>
```

```
</head>
```

```
<body>
```

```
欢迎使用本主页
```

```
<%
```

```
    //加载jdbc驱动
```

```
    try{
```

```
        Class.forName("com.mysql.cj.jdbc.Driver"); // mysql-connector-java 6
```

```
    中
```

```
    }catch(ClassNotFoundException e){
```

```
        System.out.println("加载数据库驱动时抛出异常，内容如下：");
```

```
        e.printStackTrace();
```

```
    }
```

```
    //创建数据库连接
```

```
    Connection
```

```
conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/school?server
Timezone=GMT%2B8&useSSL=false", "root", "mysql");
```

```
    //String sql="insert into book (id,isbn,title) values(5,'005','python)";
```

```
    Statement stmt = conn.createStatement();
```

```
    //stmt.execute(sql);
```

```
    //System.out.println("插入到数据库成功");
```

```
    String sql2="select * from book";
```

```
    ResultSet rs = stmt.executeQuery(sql2);
```

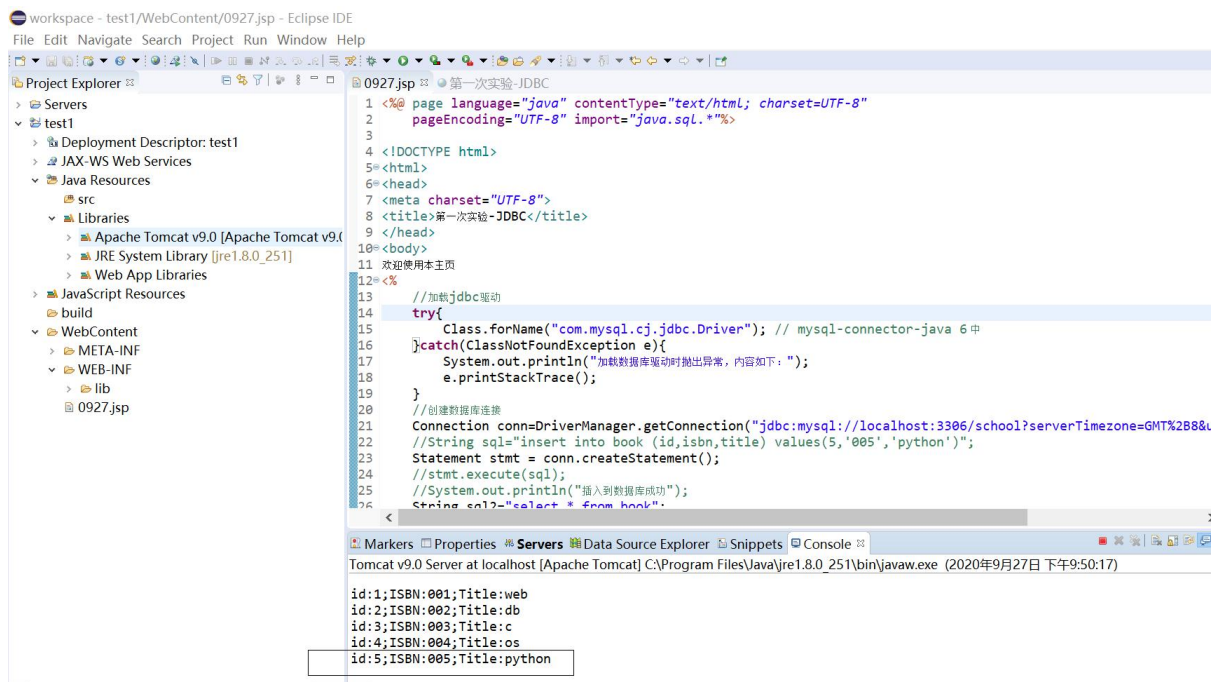
```
    //如果有数据，rs.next()返回 true
```

```
    while(rs.next()){
```

```

        System.out.println("id:"+rs.getString(1)+";ISBN:"+rs.getString(2)+";Title:"+rs.getString(3));
    }
    rs.close();
    stmt.close();
    conn.close();
%>
</body>
</html>

```



修改：将 5, 005, python 改为 5,005,AI

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" import="java.sql.*"%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>第一次实验-JDBC</title>
</head>
<body>
欢迎使用本主页
<%
    //加载jdbc驱动

```

```
try{
    Class.forName("com.mysql.cj.jdbc.Driver"); // mysql-connector-java 6
中
}catch(ClassNotFoundException e){
    System.out.println("加载数据库驱动时抛出异常，内容如下：");
    e.printStackTrace();
}
//创建数据库连接
Connection
conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/school?server
Timezone=GMT%2B8&useSSL=false", "root", "mysql");
//添加
//String sql="insert into book (id,isbn,title) values(5,'005','python')";
Statement stmt = conn.createStatement();
//stmt.execute(sql);
//System.out.println("插入到数据库成功");

//修改数据的代码
String sql3 = "update book set title='AI' where id = 5";
stmt.execute(sql3);
System.out.println("修改数据库成功");

String sql2="select * from book";
ResultSet rs = stmt.executeQuery(sql2);

//如果有数据，rs.next()返回 true
while(rs.next()){
    System.out.println("id:"+rs.getString(1)+";ISBN:"+rs.getString(2)+";Tit
le:"+rs.getString(3));
}
rs.close();
stmt.close();
conn.close();
%>
</body>
</html>
```



```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8" import="java.sql.*"%>
3
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="UTF-8">
8   <title>第一次实验-JDBC</title>
9 </head>
10 <body>
11   欢迎使用本主页
12 <%
13   //加载jdbc驱动
14   try{
15     Class.forName("com.mysql.cj.jdbc.Driver"); // mysql-connect
16   }catch(ClassNotFoundException e){
17     System.out.println("加载数据库驱动时抛出异常, 内容如下: ");
18     e.printStackTrace();
19   }
20   //创建数据库连接
21   Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/school?server
22   //添加
23   //String sql="insert into book (id,isbn,title) values(5,'005','
24   Statement stmt = conn.createStatement();
25   //stmt.execute(sql);
26   //System.out.println("插入到数据库成功");

```

Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0_251\bin\jav
id:1;ISBN:001;Title:web
id:2;ISBN:002;Title:db
id:3;ISBN:003;Title:c
id:4;ISBN:004;Title:os
id:5;ISBN:005;Title:AI

删除：将 5, 005, AI 删除

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8" import="java.sql.*"%>

```

```

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>第一次实验-JDBC</title>
</head>
<body>

```

欢迎使用本主页

```

<%
//加载jdbc驱动
try{
    Class.forName("com.mysql.cj.jdbc.Driver"); // mysql-connector-java 6
中
}catch(ClassNotFoundException e){
    System.out.println("加载数据库驱动时抛出异常, 内容如下: ");
    e.printStackTrace();
}
//创建数据库连接
Connection
conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/school?server

```

```
Timezone=GMT%2B8&useSSL=false", "root", "mysql");
    //添加
    //String sql="insert into book (id,isbn,title) values(5,'005','python)";
    Statement stmt = conn.createStatement();
    //stmt.execute(sql);
    //System.out.println("插入到数据库成功");
    //修改数据的代码
    //String sql3 = "update book set title='AI' where id = 5";
    //stmt.execute(sql3);
    //System.out.println("修改数据库成功");

    //删除数据
    String sql4 = "delete from book where id = 5";
    stmt.execute(sql4);
    System.out.println("修改数据库成功");

    String sql2="select * from book";
    ResultSet rs = stmt.executeQuery(sql2);

    //如果有数据，rs.next()返回 true
    while(rs.next()){

        System.out.println("id:"+rs.getString(1)+";ISBN:"+rs.getString(2)+";Tit
le:"+rs.getString(3));
    }
    rs.close();
    stmt.close();
    conn.close();
%>
</body>
</html>
```

The screenshot shows the Eclipse IDE interface. The Project Explorer on the left shows a project named 'test1' with a 'WebContent' folder containing '0927.jsp'. The main editor displays the following Java code:

```
25 //stmt.execute(sql);
26 //System.out.println("插入到数据库成功");
27 //修改数据的代码
28 //String sql3 = "update book set title='AI' where id = 5";
29 //stmt.execute(sql3);
30 //System.out.println("修改数据库成功");
31
32 //删除数据
33 String sql4 = "delete from book where id = 5";
34 stmt.execute(sql4);
35 System.out.println("修改数据库成功");
36
37 String sql2="select * from book";
38 ResultSet rs = stmt.executeQuery(sql2);
39
40 //如果有数据,rs.next()返回true
41 while(rs.next()){
42     System.out.println("id:"+rs.getString(1)+"ISBN:"+rs.getString(2)
43 }
44 rs.close();
45 stmt.close();
46 conn.close();
47 %>
48 </body>
49 </html>
50
```

The Console window at the bottom shows the output of the application:

```
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe
修改数据库成功
id:1;ISBN:001;Title:web
id:2;ISBN:002;Title:db
id:3;ISBN:003;Title:c
id:4;ISBN:004;Title:os
```

五、实验报告要求:

实验报告模板如下（见附件 1 信息学院实验报告模板）：

实验二 Servlet 和 MVC 开发模式

一、实验目标

- 1、理解 JSP 和 Servlet 关系
- 2、掌握 servlet 的结构、创建方法及使用
- 3、理解 MVC（模型、视图和控制器）设计模式
- 4、掌握和实现 JSP+Servlet+JavaBean 模型

二、实验环境

- 1、Java、MySQL、Servlet 等。
- 2、Eclipse 集成开发环境

三、实验内容

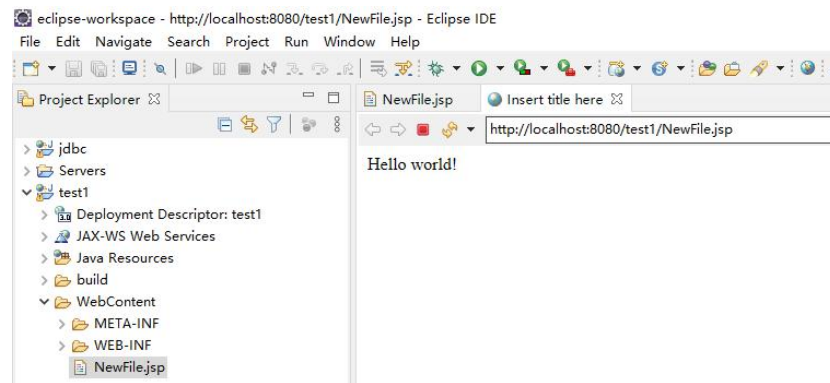
2.1 jsp 与 servlet 的关系：查看 eclipse 的 workspace 中的 metadata-plugins-org.eclipse.wst.

server.core-tmp0-work-catalina-localhost-“project 名”文件夹下的 .java 文件。了解 jsp 文件转译为 servlet，然后编译为 class 字节文件，最后运行的过程。注意 jsp (login.jsp) 与 .java(login_jsp.java) 的对应关系，阅读并了解该 java 文件的构成。

```

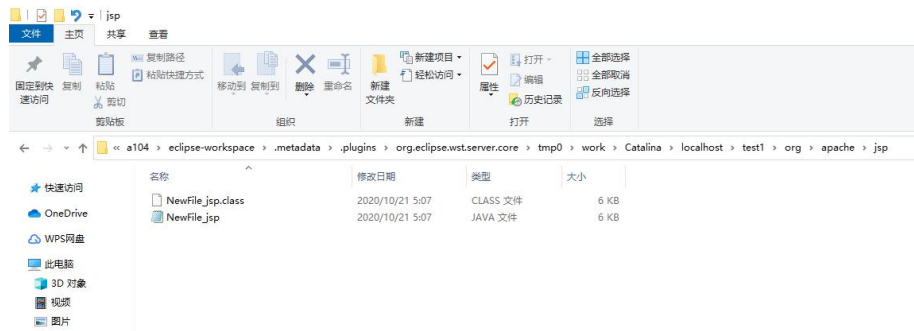
- Eclipse IDE
Project Run Window Help
NewFile.jsp FirstServlet.java web.xml web.xml NewFile1.jsp FirstServlet.java show.jsp
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="ISO-8859-1">
7 <title>Insert title here</title>
8 </head>
9 <body>
10   Hello world!
11   <form action="FirstServlet">
12     <input type="submit" value="submit">
13
14   </form>
15
16 </body>
17 </html>
18 </html>

```



C:\Users\a104\eclipse-workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\work\Catalina\localhost\test1\org\apache\jsp

Jsp 本质是.java 文件



```
Generated by the Jasper component of Apache Tomcat
* Version: Apache Tomcat/9.0.37
* Generated at: 2020-10-20 21:07:08 UTC
* Note: The last modified time of this file was set to
* the last modified time of the source file after
* generation to assist with modification tracking.
*/
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class NewFile_jsp extends org.apache.jasper.runtime.HttpJspBase
implements org.apache.jasper.runtime.JspSourceDependent,
org.apache.jasper.runtime.JspSourceImports {

    private static final javax.servlet.jsp.JspFactory _jspxFactory =
        javax.servlet.jsp.JspFactory.getDefaultFactory();

    private static java.util.Map<java.lang.String,java.lang.Long> _jspx_dependants;

    private static final java.util.Set<java.lang.String> _jspx_imports_packages;

    private static final java.util.Set<java.lang.String> _jspx_imports_classes;

    static {
        _jspx_imports_packages = new java.util.HashSet<>();
        _jspx_imports_packages.add("javax.servlet");
        _jspx_imports_packages.add("javax.servlet.http");
        _jspx_imports_packages.add("javax.servlet.jsp");
        _jspx_imports_classes = null;
    }

    private volatile javax.el.ExpressionFactory _el_expressionfactory;
    private volatile org.apache.tomcat.InstanceManager _jsp_instancemanager;
}
```

2.2 servlet 的创建-以显示 helloworld 为例

(1) 在 project 的 src 文件夹中定义类实现 HttpServlet 接口，重写 service 方法；在 web.xml 文件中配置该 servlet；运行测试。

```
package com.my.servlet;

import java.io.IOException;

import javax.servlet.*;
import javax.servlet.http.*;

public class FirstServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    // TODO Auto-generated constructor stub
    public FirstServlet() {
        super();
    }

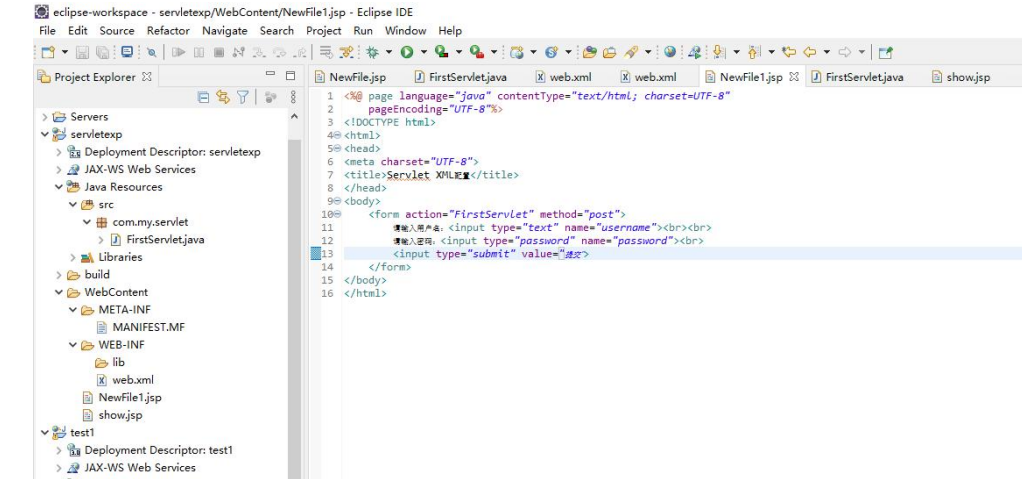
    // TODO Auto-generated method stub
    protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        System.out.println("Hello world!");
        response.getWriter().println("Hello world!");
    }
}
```

```
Tomcat/9.0.37 Server at localhost/apache-tomcat/9.0.37/Program Files/Java/jdk-11.0.10/javaxswa (2020年10月21日 上午9:59:59)
10月 21, 2020 5:19:56 上午 org.apache.coyote.AbstractProtocol start
** Server startup in 100ms **
10月 21, 2020 5:19:56 上午 org.apache.catalina.startup.Catalina start
** Catalina start **
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
```

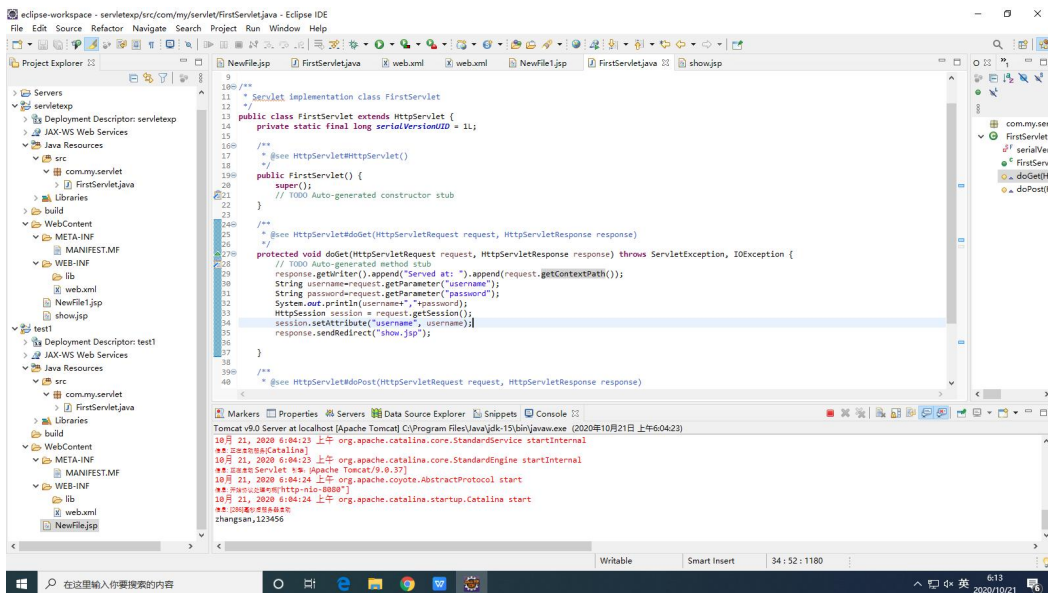
```
<? page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    Hello world!
    <form action="FirstServlet">
        <input type="submit" value="submit">
    </form>
</body>
</html>
```

(2)实现 HttpServlet 抽象类

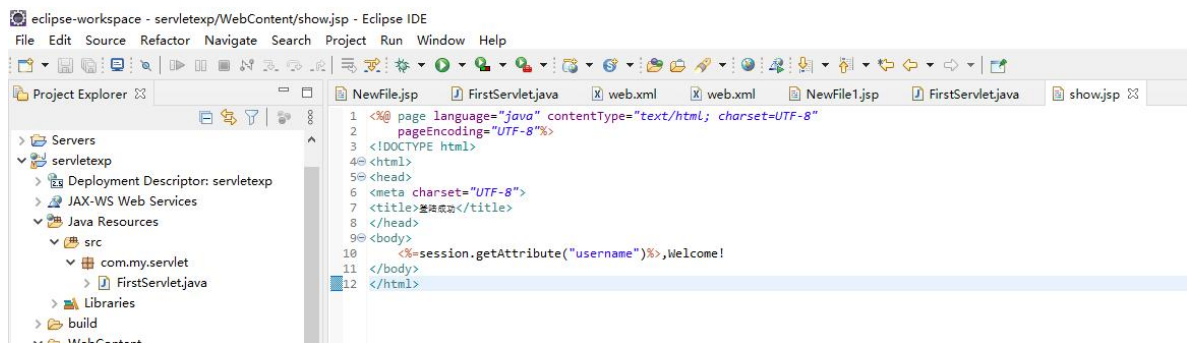
在 project 中创建 QueryServlet（自动继承 HttpServlet 类），需要重写 doGet 和 doPost 方法，使用注解的方式配置 QueryServlet，不需要在 web.xml 中配置。



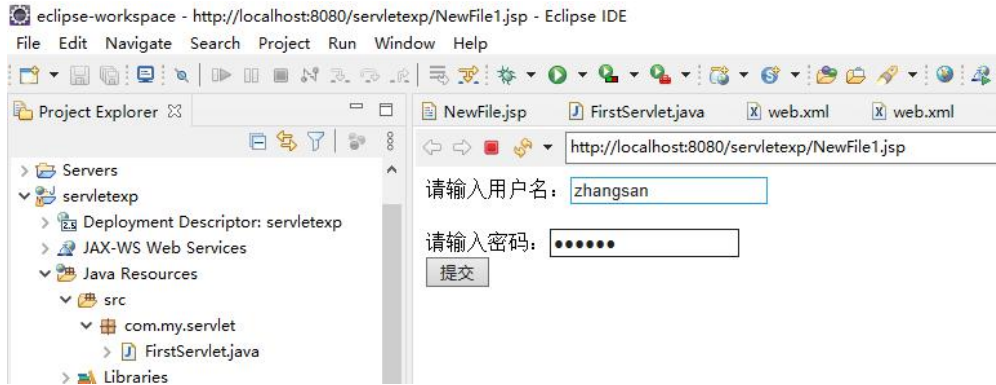
```
1 <% page language="java" contentType="text/html; charset=UTF-8"
2 pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="UTF-8">
7 <title>Servlet XML练习</title>
8 </head>
9 <body>
10 <form action="FirstServlet" method="post">
11 请输入用户名: <input type="text" name="username"><br><br>
12 请输入密码: <input type="password" name="password"><br>
13 <input type="submit" value="提交">
14 </form>
15 </body>
16 </html>
```



```
9 /**
10  * Servlet implementation class FirstServlet
11  */
12 public class FirstServlet extends HttpServlet {
13     private static final long serialVersionUID = 1L;
14
15     /**
16      * @see HttpServlet#HttpServlet()
17      */
18     public FirstServlet() {
19         super();
20     }
21
22     /**
23      * // TOOO Auto-generated method stub
24      */
25     @Override
26     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
27
28     }
29
30     /**
31      * // TOOO Auto-generated method stub
32      */
33     @Override
34     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
35         String username=request.getParameter("username");
36         String password=request.getParameter("password");
37         System.out.println(username+" "+password);
38         HttpSession session = request.getSession();
39         session.setAttribute("username", username);
40         response.sendRedirect("show.jsp");
41     }
42
43     /**
44      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
45      */
46 }
```



```
1 <% page language="java" contentType="text/html; charset=UTF-8"
2 pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="UTF-8">
7 <title>登陆成功</title>
8 </head>
9 <body>
10 <%=session.getAttribute("username")%>,Welcome!
11 </body>
12 </html>
```



四、实验练习：

1、计算三角形面积：输入三角形三条边的长度，计算并显示三角形的面积。

设计分析：

界面(JSP)： 输入三条边（input.jsp）

显示三角形面积（show.jsp）

模型(JavaBean)： 判断三条边是否能组成三角形，计算三角形面积(Triangle.java)

控制器(Servlet)： 从 input.jsp 接收三条边输入的数据，创建响应的 JavaBean 实例，验证输入合法性后再计算三角形面积发给 show.jsp

input.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <form action="Triangle" method="post">
        请输入三角形的边a: <input type="text" name="a"><br><br>
        请输入三角形的边b: <input type="text" name="b"><br><br>
```



```
        请输入三角形的边c: <input type="text" name="c"><br><br>
        <input type="submit" value="计算面积">
    </form>
</body>
</html>
```

show.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>计算结果</title>
</head>
<body>
    <%=session.getAttribute("area")%>
</body>
</html>
```

Triangle.java

```
package com.my.servlet;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class Triangle
 */
@WebServlet("/Triangle")
```

```

public class Triangle extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Triangle() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
    response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at:
    ").append(request.getContextPath());

        String string_a=request.getParameter("a");
        String string_b=request.getParameter("b");
        String string_c=request.getParameter("c");
        System.out.println(string_a+","+string_b+","+string_c);
        double a = 0, b = 0, c = 0;
        if (string_a == null | string_b == null | string_c==null)
        {
            string_a = "0";
            string_b = "0";
            string_c = "0";
        }
    }
}

```

```

HttpSession session = request.getSession();
try
{
    a = Double.valueOf(string_a).doubleValue();
    b = Double.valueOf(string_b).doubleValue();
    c = Double.valueOf(string_c).doubleValue();
    if (a+b>c && b+c>a && c+a>b)
    {
        double p = (a+b+c)/2.0;
        double area = Math.sqrt(p*(p-a)*(p-b)*(p-c));
        System.out.println(area);
        session.setAttribute("area", area);
    }
    else
    {
        System.out.println("输入数据有误");
        session.setAttribute("area", "输入数据有误");
    }
} catch (NumberFormatException e) {
    System.out.println("请重新输入");
    session.setAttribute("area", "请重新输入");
}

response.sendRedirect("show.jsp");
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
 */

```

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

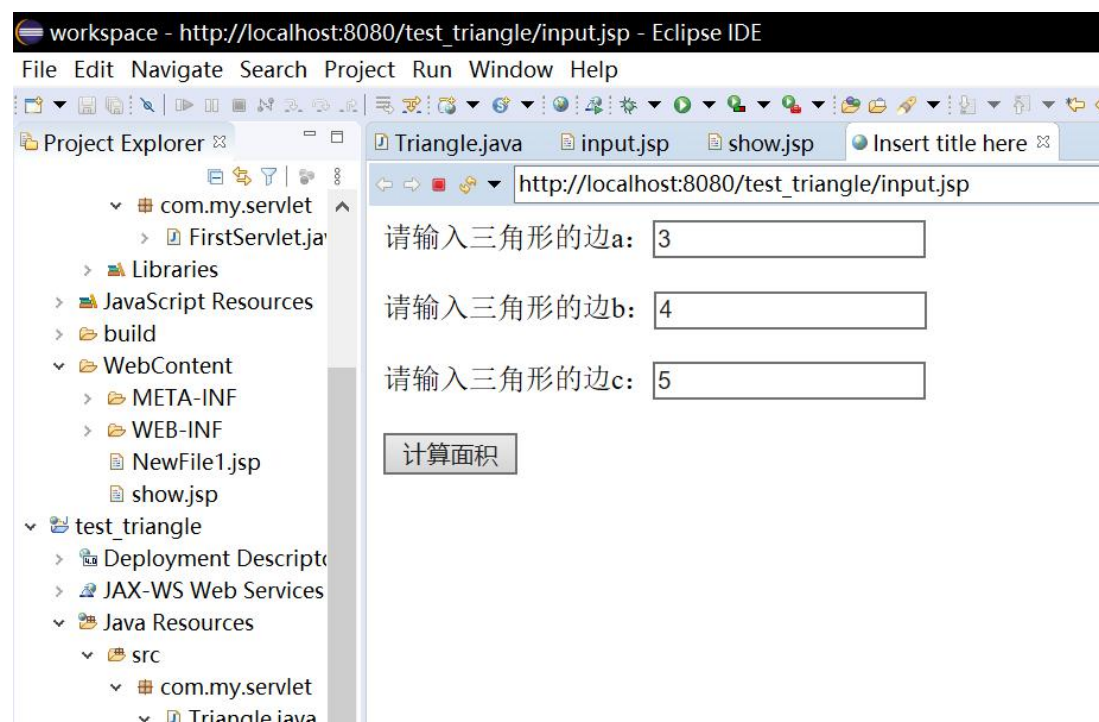
    // TODO Auto-generated method stub

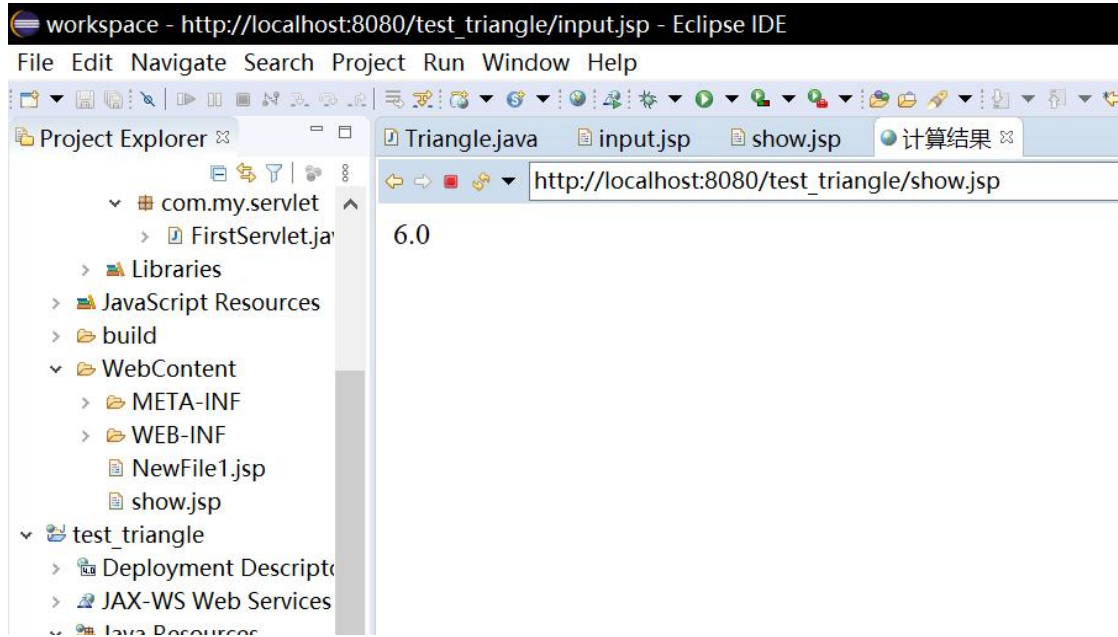
    doGet(request, response);

}

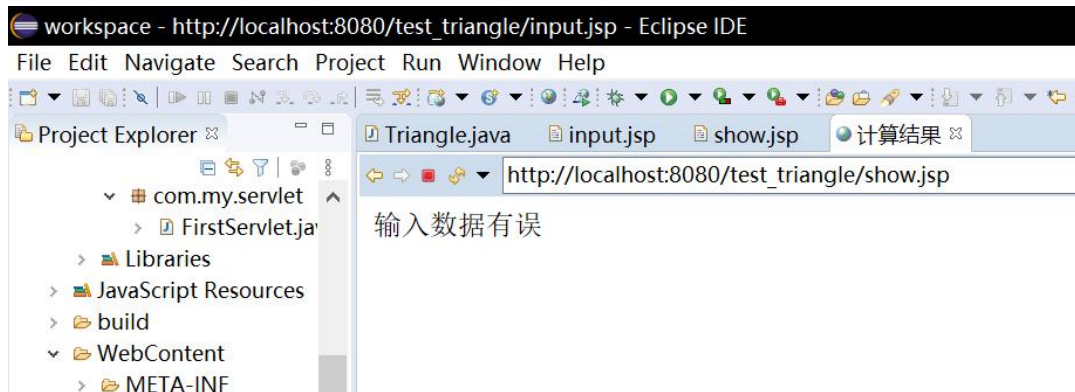
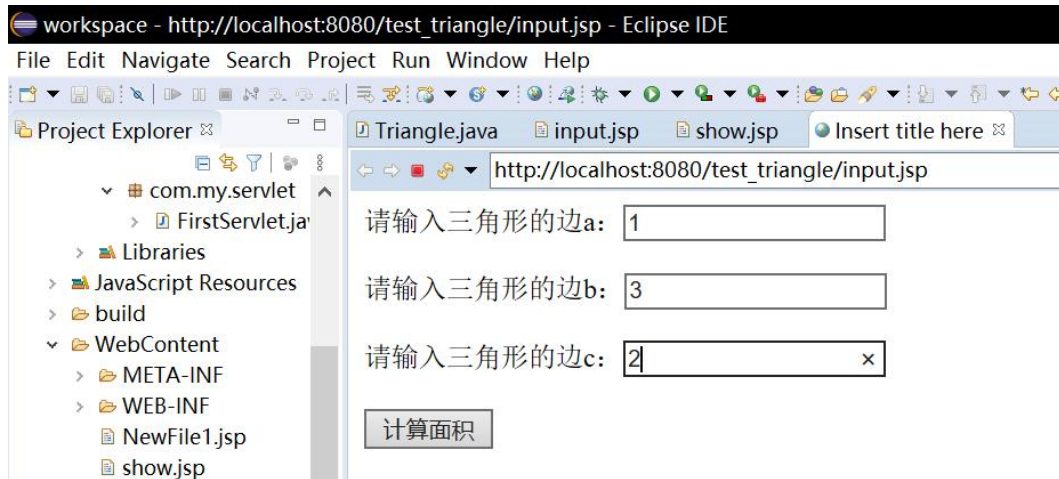
}
```

运行截图：





输入不符合 $a+b>c$ 的边长



五、实验报告要求：

实验报告模板如下（见附件 1 信息学院实验报告模板）：

实验三 Spring (1)

一、实验目标

- 理解 Spring 的思想
- 理解 IoC/DI 的原理
- 掌握搭建 Spring 开发环境的方法
- 掌握实现依赖注入的方法
- 掌握使用 XML 和注解进行注入

二、实验环境

- 1、Java、MySQL、Spring 等。
- 2、Eclipse 集成开发环境

三、实验内容

1. 小鸟类 (Bird) 有 color 属性、fly() 方法，其中 fly() 方法用于输出 “**颜色的小鸟在天上飞”。测试类 (Test)，使用 Spring 实现在 Test 中调用小鸟类的 fly() 方法的程序。

实现步骤：

- (1) 新建一个 Dynamic Web Project，工程名为：springProject1



- (2) 将 Spring 中的基础包和 commons.logging 包复制到 lib 目录中，并发布到类路径下，见下图：

```
commons-logging-1.2.jar
spring-beans-4.3.6.RELEASE.jar
spring-context-4.3.6.RELEASE.jar
spring-core-4.3.6.RELEASE.jar
spring-expression-4.3.6.RELEASE.jar
```

- (3) 在 src 目录下创建一个 com.my.entity 包，并在包中创建小鸟类 Bird，定义 color 属性和 fly() 方法

```
src
├── com.my.entity
│   └── Bird.java
└── ...

public class Bird {
    private String color;
    public void fly() {
```

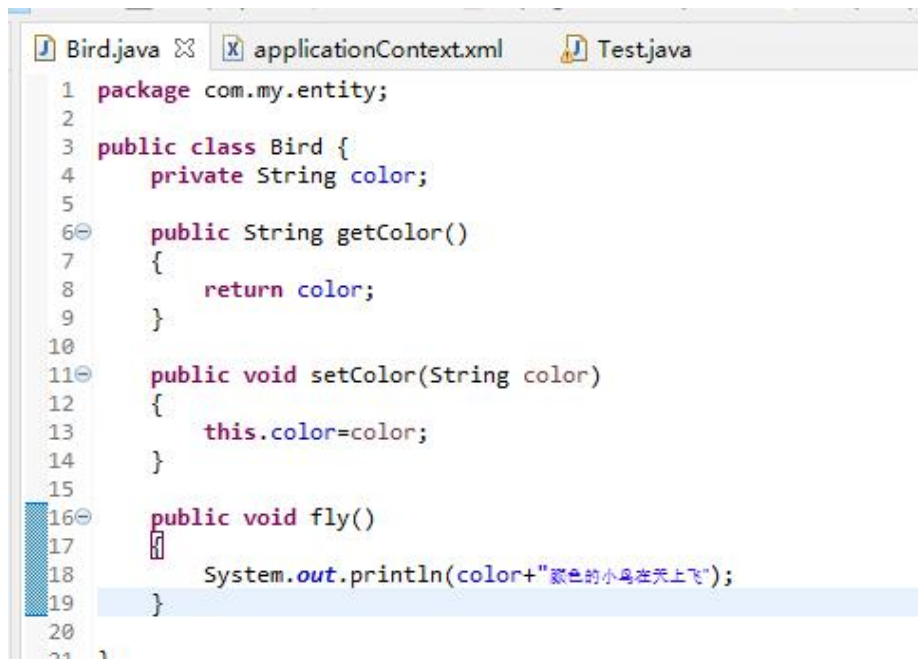
- (4) 在 com.my.entity 包中创建 Spring 的配置文件 applicationContext.xml，并在配置文件中分别创建 id 为 blueBird 的 Bean (color="blue") 和 id 为 blackBird 的 Bean (color="black")。

```
<bean id="blueBird" class="com.my.entity.Bird">
    <property name="color">
        <value>蓝色</value>
    </property>
</bean>

<bean id="blackBird" class="com.my.entity.Bird">
    <property name="color" value="黑色" />
</bean>
```

- (5) 在 com.my.entity 包中创建测试类 Test，并在类中编写 main() 方法，并分别实现 blueBird 和 blackBird 类的 fly() 方法。

```
ApplicationContext applicationContext = new ClassPathXmlApplicationContext("com/my/entity/applicationContext.xml");
```



```
Bird.java applicationContext.xml Test.java
1 package com.my.entity;
2
3 public class Bird {
4     private String color;
5
6     public String getColor()
7     {
8         return color;
9     }
10
11    public void setColor(String color)
12    {
13        this.color=color;
14    }
15
16    public void fly()
17    {
18        System.out.println(color+"颜色的小鸟在天上飞");
19    }
20
21 }
```



```
Bird.java applicationContext.xml Test.java
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://www.springframework.org/schema/beans
5         http://www.springframework.org/schema/beans/spring-beans-4.3.xsd">
6
7 <bean id="blueBird" class="com.my.entity.Bird">
8     <property name="color">
9         <value>蓝色</value>
10    </property>
11
12 </bean>
13
14 <bean id="blackBird" class="com.my.entity.Bird">
15     <property name="color" value="黑色">
16    </property>
17 </bean>
18 </beans>
```

```
Bird.java applicationContext.xml Test.java
1 package com.my.entity;
2
3 import org.springframework.context.ApplicationContext;
4
5 import org.springframework.context.support.ClassPathXmlApplicationContext;
6
7 public class Test {
8     public static void main(String[] args) {
9         ApplicationContext ac = new ClassPathXmlApplicationContext("com/my/entity/applicationContext.xml");
10        @SuppressWarnings("unused")
11        Bird blueBird = (Bird)ac.getBean("blueBird");
12        blueBird.fly();
13        Bird blackBird = (Bird)ac.getBean("blackBird");
14        blackBird.fly();
15    }
16 }
17
18
```

Markers Properties Servers Data Source Explorer Snippets Console

```
<terminated> Test [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (2020年11月4日 上午6:05:14 - 上午6:05:14)
11月 04, 2020 6:05:14 上午 org.springframework.context.support.ClassPathXmlApplicationContext prepareRefresh
刷新 org.springframework.context.support.ClassPathXmlApplicationContext@7de26db8: startup date [Wed Nov 04 06:05:14 2020]
11月 04, 2020 6:05:14 上午 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
Loading XML bean definitions from class path resource [com/my/entity/applicationContext.xml]
蓝色颜色的小鸟在天上飞
黑色颜色的小鸟在天上飞
```

2. 使用 XML 配置方式，实现如下 Spring 工程：

- (1) 新建一个 Dynamic Web Project，工程名为：springProject2；
- (2) 将 Spring 中的基础包和 commons.logging 包复制到 lib 目录中，并发布到类路径下；
- (3) 在 src 目录下创建一个 com.my.entity 包，并在包中创建班级类 (Classes)、学院类 (College) 和学生类 (Student)；
 - Classes：定义班级编号(classNo)、班级名称(className) 属性，并使用 toString 方法返回班级类 Classes 的基本信息：

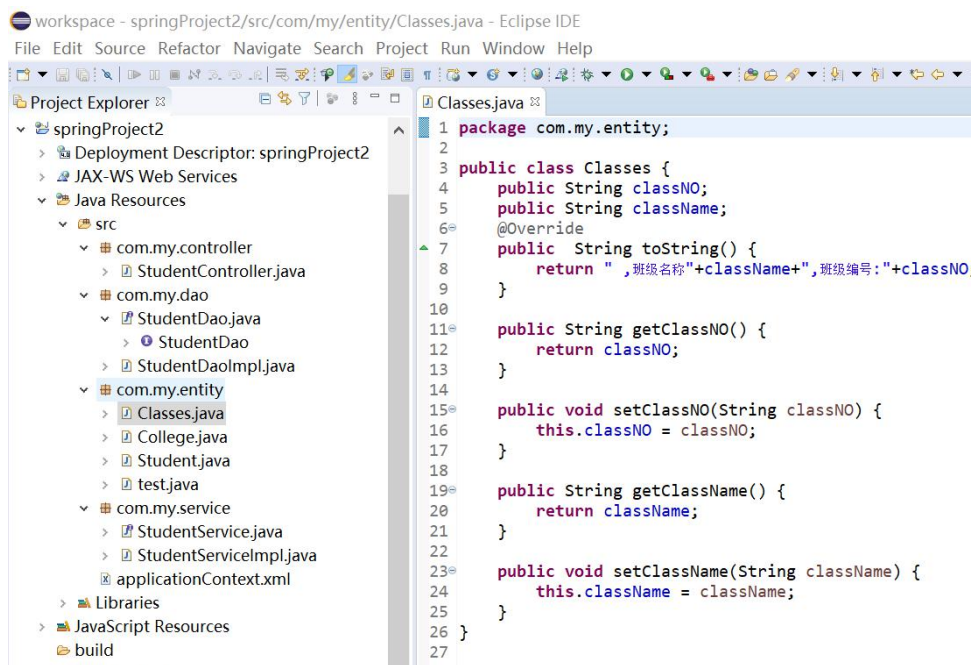
班级编号: ***, 班级名称: ***

- **College:** 定义学院名称(collegeName)属性, 并使用 toString 方法返回学院类 College 的基本信息:

学院名称:***

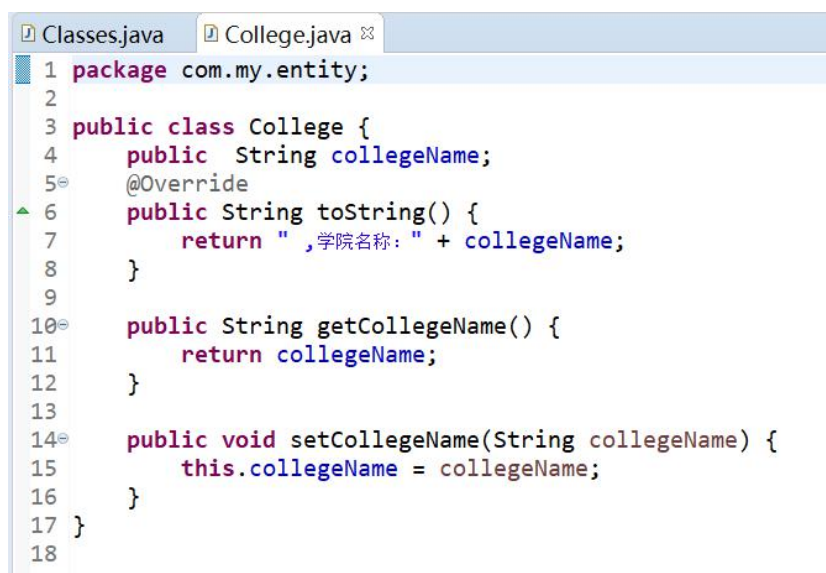
- **Student:** 定义学生的学号(studentNo)、姓名(name)、年龄(age)、性别(sex)、学生所在的学院(College—学院类)、班级(Classes—班级类)和学生的兴趣爱好(interesting—字符串列表) 属性, 并定义一个 show()方法, 打印学生的基本信息, 如:

学生编号:***, 学生姓名:***, 学生性别:***, 学生年龄:***, 学院名称:***, 班级编号:***, 班级名称:***, 兴趣爱好:***

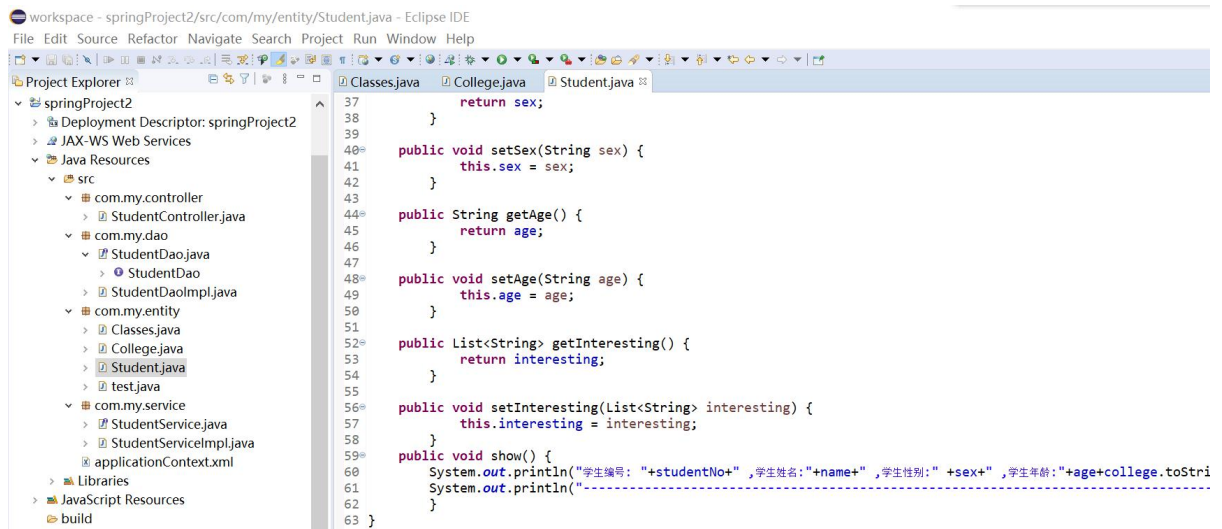


```
workspace - springProject2/src/com/my/entity/Classes.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
springProject2
  Deployment Descriptor: springProject2
  JAX-WS Web Services
  Java Resources
    src
      com.my.controller
        StudentController.java
      com.my.dao
        StudentDao.java
        StudentDaoImpl.java
      com.my.entity
        Classes.java
        College.java
        Student.java
        test.java
      com.my.service
        StudentService.java
        StudentServiceImpl.java
    applicationContext.xml
  Libraries
  JavaScript Resources
  build

Classes.java
1 package com.my.entity;
2
3 public class Classes {
4     public String classNO;
5     public String className;
6     @Override
7     public String toString() {
8         return " ,班级名称"+className+", 班级编号:"+classNO;
9     }
10
11     public String getClassNO() {
12         return classNO;
13     }
14
15     public void setClassNO(String classNO) {
16         this.classNO = classNO;
17     }
18
19     public String getClassName() {
20         return className;
21     }
22
23     public void setClassName(String className) {
24         this.className = className;
25     }
26 }
27
```

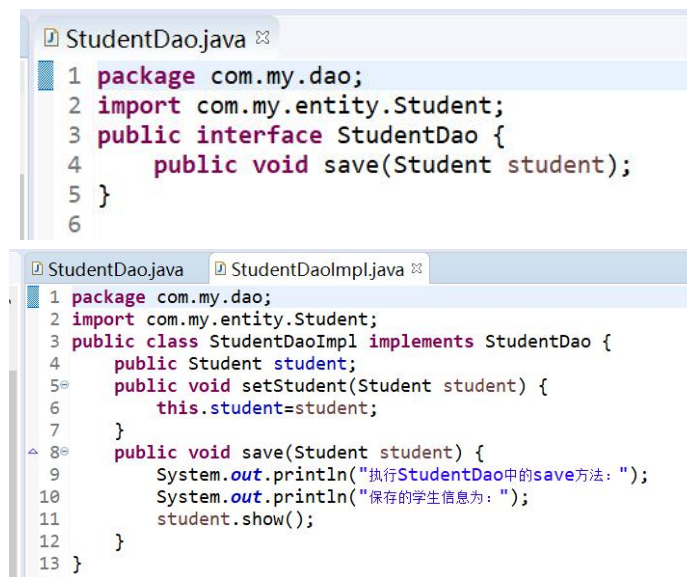


```
Classes.java College.java
1 package com.my.entity;
2
3 public class College {
4     public String collegeName;
5     @Override
6     public String toString() {
7         return " ,学院名称: " + collegeName;
8     }
9
10     public String getCollegeName() {
11         return collegeName;
12     }
13
14     public void setCollegeName(String collegeName) {
15         this.collegeName = collegeName;
16     }
17 }
18
```



- (4) 在 src 目录下创建一个 com.my.dao 包，并在包中创建一个接口 StudentDao（含 public void save(Student student)的方法）和该接口的实现类 StudentDaoImpl，并将 save 方法，具体为：

```
System.out.println("执行 StudentDao 中的 save 方法：");
System.out.println("保存的学生信息为：");
学生编号：***，学生姓名：***，学生性别：***，学生年龄：***，学院
名称：***， 班级编号：***，班级名称：***，兴趣爱好：***（此处为调用 Student 中的 show()方法）
```



- (5) 在 src 目录下创建一个 com.my.service 包，并在包中创建一个接口 StudentService（含 public void save(Student student)的方法）和该接口的实现类 StudentServiceImpl，该类中定义 StudentDao 属性和实现 save 方法，具体为：

```
System.out.println("执行 StudentService 中的 save 方法：");
调用 StudentDao 中的 save 方法
```

```

StudentService.java
1 package com.my.service;
2 import com.my.entity.Student;
3 public interface StudentService {
4     public void save(Student student);
5 }
6

StudentService.java StudentServiceImpl.java
1 package com.my.service;
2 import com.my.dao.StudentDao;
3
4 public class StudentServiceImpl implements StudentService {
5     public StudentDao studentdao;
6     //依赖注入
7     public void setStudentDao(StudentDao studentdao){
8         this.studentdao=studentdao;
9     }
10    public void save(Student student) {
11        // TODO Auto-generated method stub
12        System.out.println("执行StudentService中的save方法: ");
13        this.studentdao.save(student);
14    }
15 }
16

```

- (6) 在 src 目录下创建一个 com.my.controller 包，并在包中创建 StudentController 类，该类中定义 StudentService 属性和 save 方法，具体为：

System.out.println("执行 StudentController 中的 save()方法: ");
调用 StudentService 中的 save()方法

```

StudentController.java
1 package com.my.controller;
2 import com.my.entity.Student;
3
4 public class StudentController {
5     public StudentService studentService;
6     public void setStudentService(StudentService studentService){
7         this.studentService=studentService;
8     }
9     public void save(Student student) {
10        System.out.println("执行StudentController中的save()方法: ");
11        this.studentService.save(student);
12    }
13 }
14

```

- (7) 在 src 中创建 Spring 的配置文件 applicationContext.xml，并在配置文件中分别创建：

➤ 3 个班级 Bean:

班级编号：201801，班级名称：计科 1 班
 班级编号：201802，班级名称：计科 2 班
 班级编号：201803，班级名称：计科 3 班

➤ 1 个学院 Bean:

学院名称：信息学院

➤ 6 个学生 Bean:

学号	姓名	性别	年龄	班号	班名	学院名	兴趣爱好		
1801101	张三	男	18	201801	计科 1	信息学院	足球	游戏	

1801102	李四	女	19	201801	计科 1	信息学院	舞蹈	唱歌	书法
1801201	王五	男	20	201802	计科 2	信息学院	篮球	跳绳	
1801202	赵六	女	18	201802	计科 2	信息学院	美术	舞蹈	
1801301	孙七	男	19	201803	计科 3	信息学院	羽毛球	游泳	跑步
1801302	周八	女	20	201803	计科 3	信息学院	唱歌	舞蹈	

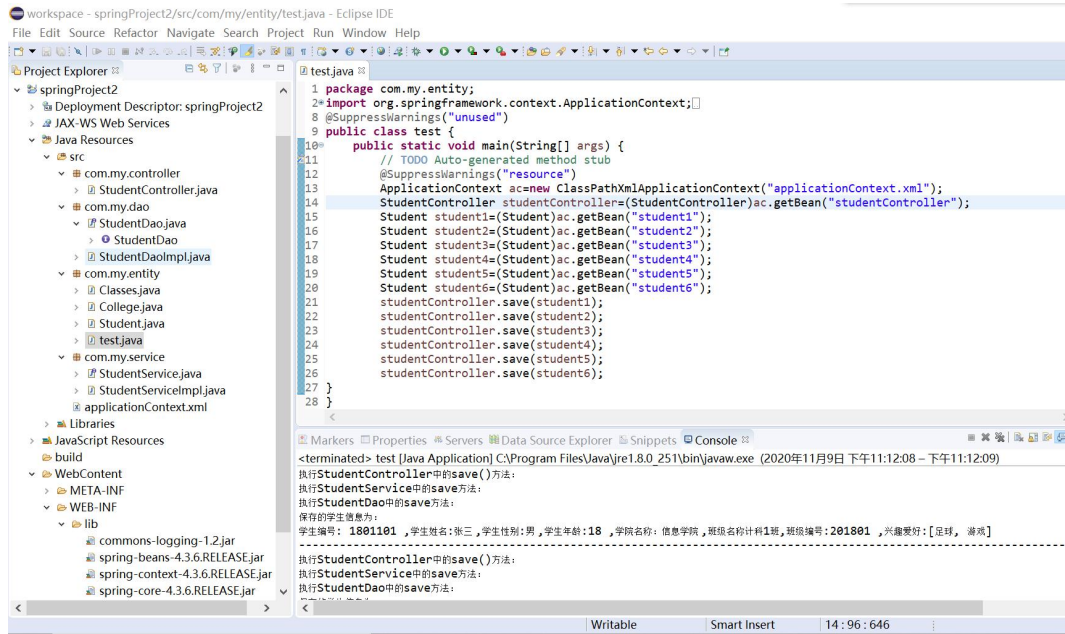
- StudentDao 的 Bean，标识符为 studentDao
- StudentService 的 Bean，标识符为 studentService
- StudentController 的 Bean，标识符为 studentController

```

25
26 <bean id="student1" class="com.my.entity.Student">
27 <property name="studentNo"><value>1801101</value></property>
28 <property name="name"><value>张三</value></property>
29 <property name="sex"><value>男</value></property>
30 <property name="age"><value>18</value></property>
31 <property name="colLege" ref="colLege1"></property>
32 <property name="classes" ref="class1"></property>
33 <property name="interesting">
34 <list><value>足球</value><value>游戏</value></list>
35 </property>
36 </bean>
37
38 <bean id="student2" class="com.my.entity.Student">
39 <property name="studentNo"><value>1801102</value></property>
40 <property name="name"><value>李四</value></property>
41 <property name="sex"><value>女</value></property>
42 <property name="age"><value>19</value></property>
43 <property name="colLege" ref="colLege1"></property>
44 <property name="classes" ref="class1"></property>
45 <property name="interesting">
46 <list> <value>舞蹈</value><value>唱歌</value><value>书法</value></list>
47 </property>
48 </bean>
49
50 <bean id="student3" class="com.my.entity.Student">
51 <property name="studentNo"><value>1801201</value></property>

```

(6) 在 com.my 包中添加测试类 Test，并在类中编写 main() 方法，并将表格中的 6 个学生信息保存（模拟输出到控制台）



3. 将 springProject2 复制一份为 springProject3, 将其改为注解方式实现 (提示: 可以只改动 studentDao、studentService、studentController 和 applicationContext.xml 的内容)。

workspace - springProject3/src/com/my/dao/StudentDaoImpl.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

```
1 package com.my.dao;
2*import org.springframework.stereotype.Repository;
4 @Repository("studentDao")
5 public class StudentDaoImpl implements StudentDao {
6     public Student student;
7     public void setStudent(Student student) {
8         this.student=student;
9     }
10    public void save(Student student) {
11        System.out.println("执行StudentDao中的save方法: ");
12        System.out.println("保存的学生信息为: ");
13        student.show();
14    }
15 }
16
```

```
1 package com.my.service;
2*import javax.annotation.Resource;
8 @Service("studentService")
9 public class StudentServiceImpl implements StudentService {
10    @Resource(name="studentDao")
11    public StudentDao studentdao;
12    //依赖注入
13    public void setStudentDao(StudentDao studentdao){
14        this.studentdao=studentdao;
15    }
16    public void save(Student student) {
17        // TODO Auto-generated method stub
18        System.out.println("执行StudentService中的save方法: ");
19        this.studentdao.save(student);
20    }
21 }
22
```

```
1 package com.my.controller;
2*import javax.annotation.Resource;
7 @Controller("studentController")
8
9 public class StudentController {
10    @Resource(name="studentService")
11    public StudentService studentService;
12    public void setStudentService(StudentService studentService){
13        this.studentService=studentService;
14    }
15    public void save(Student student) {
16        System.out.println("执行StudentController中的save()方法: ");
17        this.studentService.save(student);
18    }
19 }
20
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2<beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xmlns:context="http://www.springframework.org/schema/context"
5     xsi:schemaLocation="http://www.springframework.org/schema/beans
6     http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
7     http://www.springframework.org/schema/context
8     http://www.springframework.org/schema/context/spring-context-4.3.xsd">
9
```

```
applicationContext.xml
94 </property><property name="age">
95 <value>20</value>
96
97 </property>
98
99 <property name="college" ref="college1"></property>
100 <property name="classes" ref="class3"></property>
101 <property name="interesting">
102 <list><value>唱歌</value><value>舞蹈</value></list>
103 </property>
104 </bean>
105
106 <context:component-scan base-package="com.my.dao"></context:component-scan>
107 <context:component-scan base-package="com.my.service"></context:component-scan>
108 <context:component-scan base-package="com.my.controller"></context:component-scan>
109
110 </beans>
</pre>
<pre>
Design Source
Markers Properties Servers Data Source Explorer Snippets Console
<terminated> test (1) [Java Application] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe (2020年11月9日 下午11:31:34 - 下午11:31:35)
执行StudentService中的save方法:
执行StudentDao中的save方法:
保存的学生信息为:
学生编号: 1801202 , 学生姓名: 赵六 , 学生性别: 女 , 学生年龄: 18 , 学院名称: 信息学院 , 班级名称计科2班 , 班级编号: 201802 , 兴趣爱好: [美术, 舞蹈]
-----
执行StudentController中的save()方法:
执行StudentService中的save方法:
执行StudentDao中的save方法:
保存的学生信息为:
学生编号: 1801301 , 学生姓名: 孙七 , 学生性别: 男 , 学生年龄: 19 , 学院名称: 信息学院 , 班级名称计科3班 , 班级编号: 201803 , 兴趣爱好: [羽毛球, 游泳, 跑步]
-----
执行StudentController中的save()方法:
执行StudentService中的save方法:
执行StudentDao中的save方法:
</pre>
```

四、实验报告要求:

实验报告模板如下（见附件 1 信息学院实验报告模板）：

实验四 Spring (2)

一、实验目标

- 了解 AOP 编程的作用、了解切面、切点的概念
- 掌握使用 AspectJ 实现 AOP 的方法（使用 XML 配置文件和注解方式）
- 掌握 Spring 模板类 JdbcTemplate 的各种方法
- 掌握通过配置 XML 或注解实现 Spring 事务管理的方法

二、实验环境

- 1、Java、MySQL、Spring 等。
- 2、Eclipse 集成开发环境

三、实验内容

(1)按照下列步骤完成相关代码的编写(模板代码见附件 springProject 动态 Web 工程)。其主要功能是使用 Spring JDBC 对 MySQL 数据库中的内容进行增、删、改、查的操作。

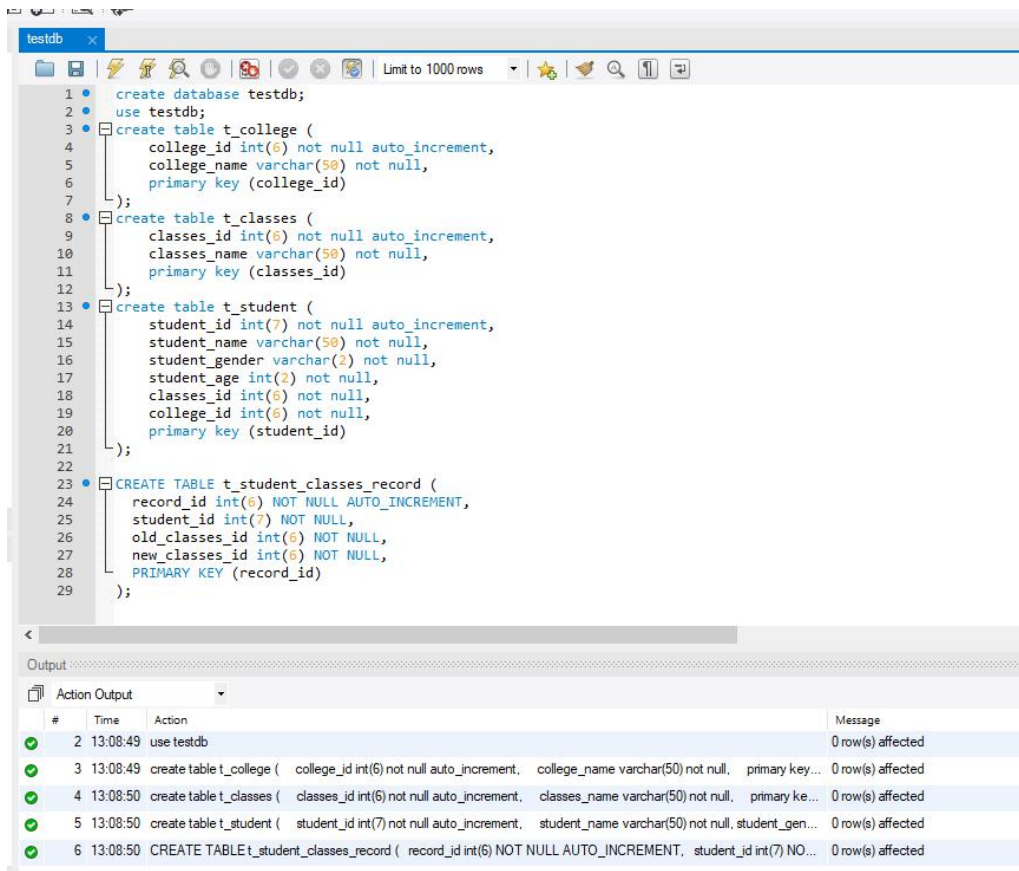
1. 在 MySQL 数据库中创建一个名为 testdb 的数据库，在数据库中新建三张表，分别为:t_college、t_classes、t_student。具体 sql 语句为：

```
create database testdb;
use testdb;
create table t_college (
    college_id int(6) not null auto_increment,
    college_name varchar(50) not null,
    primary key (college_id)
);
create table t_classes (
    classes_id int(6) not null auto_increment,
    classes_name varchar(50) not null,
    primary key (classes_id)
);
create table t_student (
    student_id int(7) not null auto_increment,
```

```

student_name varchar(50) not null,
student_gender varchar(2) not null,
student_age int(2) not null,
classes_id int(6) not null,
college_id int(6) not null,
primary key (student_id)
);

```



2. 新建一个 Dynamic Web Project，取名为 springProject，将 Spring 的基础包、AOP 相关包、Spring JDBC 包、Spring 事务管理包和 MySQL 的数据库驱动包添加到 springProject 工程中的 lib 文件夹中，并发布到类路径下，见图 1：

- aopalliance-1.0.jar
- aspectjweaver-1.8.10.jar
- commons-logging-1.2.jar
- mysql-connector-java-8.0.18.jar
- spring-aop-4.3.6.RELEASE.jar
- spring-aspects-4.3.6.RELEASE.jar
- spring-beans-4.3.6.RELEASE.jar
- spring-context-4.3.6.RELEASE.jar
- spring-core-4.3.6.RELEASE.jar
- spring-expression-4.3.6.RELEASE.jar
- spring-jdbc-4.3.6.RELEASE.jar
- spring-tx-4.3.6.RELEASE.jar

图 1. springProject 所需 jar 包

3. 在 src 文件夹中新建 com.my.entity 包, 里面有三个类: Classes、College、Student。三个类分别对应 testdb 数据库中的 t_classes、t_college、t_student 三张表, 每个类中的私有属性为数据库中对应表格的字段, 并添加 setter、getter 方法, 修改 toString 方法, 具体见如下:

com.my.entity.Classes:

```
package com.my.entity;
public class Classes {
    private int classes_id;
    private String classes_name;
    .....
    public String toString() {
        return "班级编号: "+classes_id+", 班级名称: "+classes_name;
    }
}
```

com.my.entity.College:

```
package com.my.entity;
public class College {
    private int college_id;
    private String college_name;
    .....
    public String toString() {
        return "学院编号: "+college_id+", 学院名称: "+college_name;
    }
}
```

com.my.entity.Student:

```
package com.my.entity;
public class Student {
    private int student_id;
    private String student_name;
    private String student_gender;
    private int student_age;
    private int classes_id;
    private int college_id;
    .....
    public String toString() {
        return "学生编号: "+student_id+", 学生姓名: "+student_name+",
        学生性别: "+student_gender+", 学生年龄: "+student_age;
    }
}
```

4. 在 src 文件夹中新建 applicationContext.xml 文件，创建如下 Spring Bean:

- 学院 Bean: 信息学院 (information_college)
工程学院 (engineering_college)
爱恩学院 (aien_college)

college_id	college_name
100001	信息学院
100002	工程学院
100003	爱恩学院

图 2. 学院 Bean

- 班级 Bean: 18 计科 1 班~18 计科 3 班 (classes_jk1~classesjk3)
18 工业 1 班~18 工业 3 班 (classes_gy1~classes_gy3)
18 工商 1 班~18 工商 3 班 (classes_gs1~classes_gs3)

classes_id	classes_name
201801	18计科1
201802	18计科2
201803	18计科3
201811	18工业1
201812	18工业2
201813	18工业3
201821	18工商1
201822	18工商2
201823	18工商3

图 3. 班级 Bean

- 学生 Bean: 计科 9 名学生 (student_jk1~student_jk9, 每个班 3 人)
工业 9 名学生 (student_gy1~student_gy 9, 每个班 3 人)
工商 9 名学生 (student_gs1~student_gs9, 每个班 3 人)

student_id	student_name	student_gender	student_age	classes_id	college_id
1801101	张1	男	18	201801	100001
1801102	张2	女	17	201801	100001
1801103	张3	男	18	201801	100001
1801201	张4	男	18	201802	100001
1801202	张5	女	17	201802	100001
1801203	张6	男	18	201802	100001
1801301	张7	男	18	201803	100001
1801302	张8	女	17	201803	100001
1801303	张9	男	18	201803	100001
1802101	李1	男	18	201811	100002
1802102	李2	女	17	201811	100002
1802103	李3	男	18	201811	100002
1802201	李4	男	18	201812	100002
1802202	李5	女	17	201812	100002
1802203	李6	男	18	201812	100002
1802301	李7	男	18	201813	100002
1802302	李8	女	17	201813	100002
1802303	李9	男	18	201813	100002
1803101	王1	男	18	201821	100003
1803102	王2	女	17	201821	100003
1803103	王3	男	18	201821	100003
1803201	王4	男	18	201822	100003
1803202	王5	女	17	201822	100003
1803203	王6	男	18	201822	100003
1803301	王7	男	18	201823	100003
1803302	王8	女	17	201823	100003
1803303	王9	男	18	201823	100003

图 4. 学生 Bean

具体信息见附件中 src 文件夹中的 applicationContext.xml 文件。注意 beans 元素中的约束信息的使用。

5. 在 src 文件夹中新建 com.my.dao 包，里面有三个接口：IClassesDao、ICollegeDao、IStudentDao，每个接口中都含有 5 个操作数据库对应表的方法（含添加、修改、删除、查询全部记录、查询单个记录），具体为：

com.my.dao.IClassesDao:

```

package com.my.dao;
import java.util.List;
import com.my.entity.Classes;
public interface IClassesDao {
    //添加班级
    public int addClasses(Classes classes);
    //修改班级
    public int updateClasses(Classes classes);
    //删除班级
    public int deleteClasses(int classes_id);
    //查找班级表中所有班级信息
    public List<Classes> findAllClasses();
}

```

```

//查找班级表中对应的classes_id班级信息
public Classes findClassesById(int classes_id);
}

```

com.my.dao.ICollegeDao:

```

package com.my.dao;
import java.util.List;
import com.my.entity.College;
public interface ICollegeDao {
    //添加学院
    public int addCollege(College college);
    //修改学院
    public int updateCollege(College college);
    //删除学院
    public int deleteCollege(int college_id);
    //查找学院表中所有学院信息
    public List<College> findAllCollege();
    //查找学院表中对应的college_id学院信息
    public College findCollegeById(int college_id);
}

```

com.my.dao.IStudentDao:

```

package com.my.dao;
import java.util.List;
import com.my.entity.Student;
public interface IStudentDao {
    //添加学生信息
    public int addStudent(Student student);
    //修改学生信息
    public int updateStudent(Student student);
    //删除学生信息
    public int deleteStudent(int student_id);
    //查找学生表中的所有学生信息
    public List<Student> findAllStudent();
    //查找学生表中对应student_id学生信息
    public Student findStudentById(int student_id);
}

```

6. 在 com.my.dao 包，创建三个接口（IClassesDao、ICollegeDao、IstudentDao）对应的实现类（ClassesDaoImpl、CollegeDaoImpl、StudentDaoImpl），要求使用 JdbcTemplate，实现接口中的相应方法。

参考案例：(com.my.dao.CollegeDaoImpl)

```

public class CollegeDaoImpl implements ICollegeDao {

```

```

private JdbcTemplate jdbcTemplate;
.....
@Override
public int addCollege(College college) {
    String sql = "insert into t_college values (?,?)";
    int result = this.jdbcTemplate.update(sql, college.getCollege_id(),
college.getCollege_name());

    return result;
}
.....
}

```

CollegeDaoImpl:

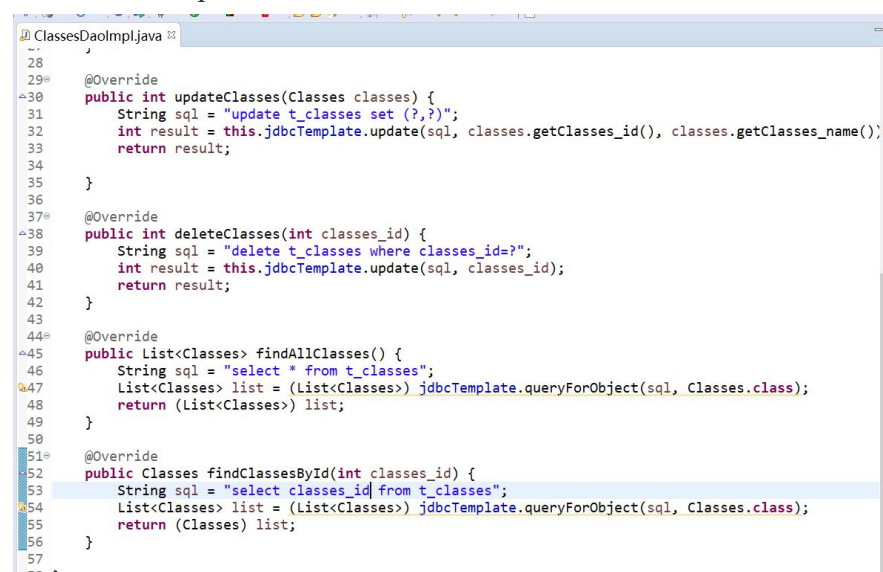


```

ClassesDaoImpl.java  CollegeDaoImpl.java
29 @Override
30 public int updateCollege(College college) {
31     //Insert code
32     String sql = "update t_college set (?,?)";
33     int result = this.jdbcTemplate.update(sql, college.getCollege_id(), college.getCollege_name());
34     return result;
35 }
36
37 @Override
38 public int deleteCollege(int college_id) {
39     //Insert code
40     String sql = "delete t_college where college_id=?";
41     int result = this.jdbcTemplate.update(sql, college_id);
42     return result;
43 }
44
45 @Override
46 public List<College> findAllCollege() {
47     //Insert code
48     String sql = "select * from t_college";
49     List<College> list = (List<College>) jdbcTemplate.queryForObject(sql, College.class);
50     return (List<College>) list;
51 }
52
53 @Override
54 public College findCollegeById(int college_id) {
55     //Insert code
56     String sql = "select college_id from t_college";
57     List<College> list = (List<College>) jdbcTemplate.queryForObject(sql, College.class);
58     return (College) list;
59 }
60

```

ClassesDaoImpl:



```

ClassesDaoImpl.java
28
29 @Override
30 public int updateClasses(Classes classes) {
31     String sql = "update t_classes set (?,?)";
32     int result = this.jdbcTemplate.update(sql, classes.getClasses_id(), classes.getClasses_name());
33     return result;
34 }
35
36
37 @Override
38 public int deleteClasses(int classes_id) {
39     String sql = "delete t_classes where classes_id=?";
40     int result = this.jdbcTemplate.update(sql, classes_id);
41     return result;
42 }
43
44 @Override
45 public List<Classes> findAllClasses() {
46     String sql = "select * from t_classes";
47     List<Classes> list = (List<Classes>) jdbcTemplate.queryForObject(sql, Classes.class);
48     return (List<Classes>) list;
49 }
50
51 @Override
52 public Classes findClassesById(int classes_id) {
53     String sql = "select classes_id from t_classes";
54     List<Classes> list = (List<Classes>) jdbcTemplate.queryForObject(sql, Classes.class);
55     return (Classes) list;
56 }
57

```

StudentDaoImpl:

```

ClassesDaoImpl.java  CollegeDaoImpl.java  StudentDaoImpl.java
34=  @Override
35=  public int updateStudent(Student student) {
36=      //Insert code
37=      String sql = "update t_student set student_name=?,student_gender=?,student_age=?,classes_id=?";
38=      int result = this.jdbcTemplate.update(sql,
39=          student.getStudent_name(),
40=          student.getStudent_gender(), student.getStudent_age(),
41=          student.getClasses_id(), student.getCollege_id(),student.getStudent_id());
42=      return result;
43=  }
44=  @Override
45=  public int deleteStudent(int student_id) {
46=      //Insert code
47=      String sql = "delete from t_student where student_id=?";
48=      int result = this.jdbcTemplate.update(sql, student_id );
49=      return result;
50=  }
51=  @Override
52=  public List<Student> findAllStudent() {
53=      //Insert code
54=      String sql = "select * from t_student";
55=      return jdbcTemplate.query(sql, new BeanPropertyRowMapper(Student.class));
56=  }
57=  @Override
58=  public Student findStudentById(int student_id) {
59=      List<Student> list = jdbcTemplate.query("select * from t_student where student_id = ?", new C
60=          if(list!=null && list.size()>0){
61=              Student student = list.get(0);
62=              return student;
63=          }
64=          else
65=          {return null;}

```

7. 修改 applicationContext.xml 中的内容, 添加 Spring JDBC 数据源的配置、JDBC 模板的配置, 并定义 IStudentDao 的 Bean、IClassesDao 的 Bean 和 ICollegeDao 的 Bean, 完成这三个 Bean 中 JdbcTemplate 的注入。

参考:

<!--1.配置数据源 -->

.....

```

<bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <!--数据库驱动 -->
    <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
    <!--连接数据库的url -->
    <property name="url"
value="jdbc:mysql://localhost:3306/testdb?serverTimezone=UTC" />
    <!--连接数据库的用户名 -->
    <property name="username" value="root" />
    <!--连接数据库的密码 -->
    <property name="password" value="123456" />
</bean>

```

<!--2.配置JDBC模板 -->

```

<bean id="jdbcTemplate"
class="org.springframework.jdbc.core.JdbcTemplate">
    <!--默认必须使用数据源 -->
    <property name="dataSource" ref="dataSource" />
</bean>

```

<!--3.定义id为IStudentDao的Bean -->

```

<bean id="studentDao" class="com.my.dao.StudentDaoImpl">
    <!--将 jdbcTemplate注入到 IStudentDao实例中 -->

```



```

    <property name="jdbcTemplate" ref="jdbcTemplate" />
</bean>

```

.....

```

ClassesDaoImpl.java  CollegeDaoImpl.java  StudentDaoImpl.java  applicationContext.xml
313 <!--连接数据库的用户名 -->
314 <property name="username" value="root" />
315 <!--连接数据库的密码 -->
316 <property name="password" value="mysql" />
317 </bean>
318
319 <!--2.配置JDBC模板 -->
320 <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
321 <!--默认必须使用数据源 -->
322 <property name="dataSource" ref="dataSource" />
323 </bean>
324
325 <!--3.定义id为IStudentDao的Bean -->
326 <bean id="studentDao" class="com.my.dao.StudentDaoImpl">
327 <!--将 jdbcTemplate注入到 IStudentDao实例中 -->
328 <property name="jdbcTemplate" ref="jdbcTemplate" />
329 </bean>
330
331 <!--4.定义id为IClassesDao的Bean -->
332 <bean id="classesDao" class="com.my.dao.ClassesDaoImpl">
333 <!--将 jdbcTemplate注入到 IClassesDao实例中 -->
334 <property name="jdbcTemplate" ref="jdbcTemplate" />
335 </bean>
336
337 <!--5.定义id为ICollegeDao的Bean -->
338 <bean id="collegeDao" class="com.my.dao.CollegeDaoImpl">
339 <!--将 jdbcTemplate注入到 ICollegeDao实例中 -->
340 <property name="jdbcTemplate" ref="jdbcTemplate" />
341 </bean>
342
343 </beans>

```

8. 新建 com.my.test 包，在包中新建 Test 类，在其中添加一个 main 方法，完成下面内容的测试：

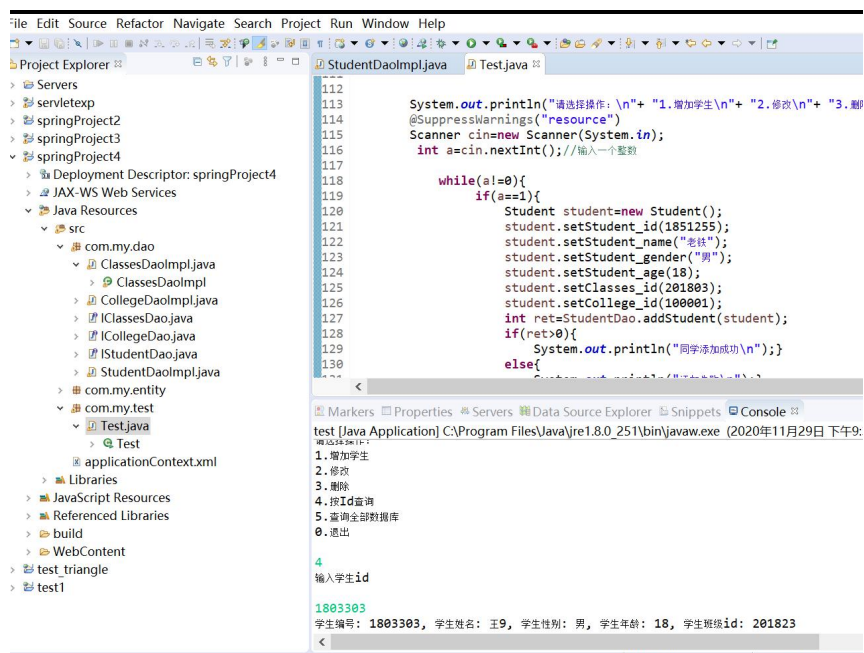
- 1) 将图 2-图 4 中的学院、班级、学生 Bean 使用 com.my.dao 中的实现类的添加方法，分别将其内容添加到 MySQL 数据库中的对应表中 (t_college、t_classes、t_student)
- 2) 使用 com.my.dao 中的实现类中的其他数据库操作方法（修改、删除、查询），检验这些方法是能够正常使用。

注：也可以使用注解的方式实现。

```

StudentDaoImpl.java  Test.java
118 while(a!=0){
119     if(a==1){
120         Student student=new Student();
121         student.setStudent_id(1851255);
122         student.setStudent_name("老铁");
123         student.setStudent_gender("男");
124         student.setStudent_age(18);
125         student.setClasses_id(201803);
126         student.setCollege_id(100001);
127         int ret=StudentDao.addStudent(student);
128         if(ret>0){
129             System.out.println("同学添加成功\n");}
130         else{
131             System.out.println("添加失败\n");}
132     }
133
134     if(a==2){//修改
135         Student student=new Student();
136         student.setStudent_id(1851255);
137         student.setStudent_name("老铁");
138         student.setStudent_gender("女");
139         student.setStudent_age(21);
140         student.setClasses_id(201802);
141         student.setCollege_id(100003);
142         int ret=StudentDao.updateStudent(student);
143         if(ret>0){
144             System.out.println("信息更新成功\n");
145         }
146         else{
147             System.out.println("更新失败\n");
148         }
149     }

```



(2) 在 (1) 的基础上，增加一个学生专业分班的操作（假设学生大三将按照培养方案选定专业选修方向（如计科班学生大三分为硬件班和软件班；工业班学生大三分为传统 IE 班和现代 IE 班；工商班分为企业管理班和财务管理班等；）），同时记录学生专业分班前后的班级信息。

参考步骤：

1. 创建一张学生专业分班记录表（t_student_classes_record），具体 sql 语句见如下：

```
CREATE TABLE t_student_classes_record (
    record_id int(6) NOT NULL AUTO_INCREMENT,
    student_id int(7) NOT NULL,
    old_classes_id int(6) NOT NULL,
    new_classes_id int(6) NOT NULL,
    PRIMARY KEY (record_id)
);
```

2. 在 IStudentDao 接口中添加分班方法：

```
//学生分班
```

```
public int divisionOfClasses(int student_id, String
new_classes_name);
```

该方法中有以下 5 步具体操作：

- 1) 根据 `student_id` 获取学生信息
- 2) 获取学生的原班级信息
- 3) 获取学生的新班级信息（如果 `t_classes` 表中没有新班级信息则添加后获取）
- 4) 将学生的新、老班级信息加入学生专业分班记录表（`t_student_classes_record`）中
- 5) 修改学生表中的 `classes_id`，改为新班级的 `classes_id`;

3. 在分班方法中的 5 个步骤应该是作为一个事务提交的（即学生的分班信息应该在分班专业表和学生表以及班级表是一致的），所以这里要使用 AOP 方式添加一个 Spring 的事务管理。在 `applicationContext.xml` 文件中，定义事务管理器、编写通知，编写 aop，具体见如下：

```
<!-- 事务管理器，依赖于数据源 -->
<bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"/>
</bean>
<!-- 编写通知：对事务进行增强（通知），需要编写对切入点和具体执行事务细节 -->
<tx:advice id="txAdvice" transaction-manager="transactionManager">
    <tx:attributes>
        <tx:method name="*" propagation="REQUIRED" isolation="DEFAULT"
read-only="false"/>
    </tx:attributes>
</tx:advice>
<!-- 编写aop,让spring自动对目标生成代理，需要使用AspectJ的表达式 -->
<aop:config>
    <!--切入点 -->
    <aop:pointcut expression="execution(* com.my.dao.*(..))" id="txPointCut" />
    <!--切面 -->
    <aop:advisor advice-ref="txAdvice" pointcut-ref="txPointCut"/>
</aop:config>
```

4. 在 `Test` 类中的 `main` 方法中，添加一个对分班方法的测试，验证一下基于 AOP 的事务管理是否起作用。

参考：可以在分班方法的第 4、5 步之间添加：

```
//用于检测基于AOP的事务管理是否生效
//int i=1/0;
```

注：也可以使用注解的方式实现。

```
TransactionManager.java
114 student_dao.addStudent(student_gs3);
115 student_dao.addStudent(student_gs4);
116 student_dao.addStudent(student_gs5);
117 student_dao.addStudent(student_gs6);
118 student_dao.addStudent(student_gs7);
119 student_dao.addStudent(student_gs8);
120 student_dao.addStudent(student_gs9);
121
122 Scanner cin=new Scanner(System.in);
123
124 System.out.println("输入学生id\n");
125 int s_id=cin.nextInt();
126 System.out.println("输入新班级: \n");
127 String c_id=cin.next();
128
129 int ret=StudentDao.divisionOfClasses(s_id, c_id);
130 if(ret>0){
131     System.out.println("分班成功\n");
132 }
133 else{
134     System.out.println("分班失败\n");
135 }
136
137 }
138 }
139
```

Markers Properties Servers Data Source Explorer Snippets Console

<terminated> TransactionManager [Java Application] C:\Program Files\Java\jre1.8.0_251\bin
输入学生id

1803303
输入新班级:

201801
学生信息学生编号: 1803303, 学生姓名: 王9, 学生性别: 男, 学生年龄: 18

四、实验报告要求:

实验报告模板如下（见附件 1 信息学院实验报告模板）：

实验五 Mybatis 实验

一、实验目标

- 掌握 MyBatis 原理与工作流程
- 学会搭建 MyBatis 开发环境
- 学会使用工具类简化 MyBatis 的开发
- 掌握 MyBatis 配置文件和映射文件的方法
- 理解结果映射
- 掌握在 MyBatis 中添加增删改查的操作
- 掌握动态查询的方法
- 掌握一对一、一对多、多对一、多对多查询的方法

二、实验环境

- 1、Java、MySQL、Spring、MyBatis 等。
- 2、Eclipse 集成开发环境

三、实验内容

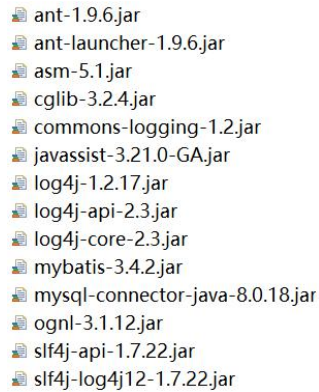
1、在 MySQL 中创建表 1 所示的商品数据表 goods，在 Eclipse 中创建一个 MyBatis 项目，**查询出全部商品**。

表 1 商品数据表 goods

商品编号 (id)	商品名称 (goodsname)	商品单价 (price)	商品数量 (quantity)
1	电视机	5000	100
2	电冰箱	4000	200
3	空调	3000	300
4	洗衣机	3500	400

解题思路：

- (1) 在 MySQL 中创建 goods 表，并插入表 1 中的数据；
- (2) 新建动态 Web 工程，添加 MyBatis 所需的 jar 包，并发布到类路径下；



- ant-1.9.6.jar
- ant-launcher-1.9.6.jar
- asm-5.1.jar
- cglib-3.2.4.jar
- commons-logging-1.2.jar
- javassist-3.21.0-GA.jar
- log4j-1.2.17.jar
- log4j-api-2.3.jar
- log4j-core-2.3.jar
- mybatis-3.4.2.jar
- mysql-connector-java-8.0.18.jar
- ognl-3.1.12.jar
- slf4j-api-1.7.22.jar
- slf4j-log4j12-1.7.22.jar

图 1. MyBatis 所需 jar 包

- (3) 在 src 文件夹中创建 log4j.properties 和 jdbc.properties，分别添加如下内容：

log4j.properties:

```
# Global logging configuration
log4j.rootLogger=ERROR, stdout
# MyBatis logging configuration...
log4j.logger.com.ssm=TRACE
# Console output...
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%5p [%t] - %m%n
```

jdbc.properties: (需根据数据库修改下列信息)

```
jdbc.driver=com.mysql.cj.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/db_mybatis?serverTimezone=UTC
jdbc.username=root
jdbc.password=123456
```

- (4) 在 src 文件夹中创建 mybatis-config.xml 文件，添加相关内容；
- (5) 在 src 文件夹中创建 com.ssm.utils 包，并在其中添加 MyBatisUtil.java，添加如下内容：

```
package com.ssm.utils;
import java.io.Reader;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
public class MyBatisUtil {
    private MyBatisUtil() {}
    private static final String RESOURCE = "mybatis-config.xml";
```

```

private static SqlSessionFactory sqlSessionFactory = null;
private static ThreadLocal<SqlSession> threadLocal = new ThreadLocal<SqlSession>();
static {
    Reader reader = null;
    try {
        reader = Resources.getResourceAsReader(RESOURCE);
        SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();
        sqlSessionFactory = builder.build(reader);
    } catch (Exception e1) {
        e1.printStackTrace();
        throw new ExceptionInInitializerError("初始化MyBatis失败，请检查配置文件或数据库");
    }
}
public static SqlSessionFactory getSqlSessionFactory() {
    return sqlSessionFactory;
}
//获取SqlSession对象
public static SqlSession getSession() {
    SqlSession session = threadLocal.get();
    if(session==null) {
        session = (sqlSessionFactory != null)?sqlSessionFactory.openSession():null;
        threadLocal.set(session);
    }
    return session;
}
//关闭SqlSession对象
public static void closeSession() {
    SqlSession session = (SqlSession)threadLocal.get();
    threadLocal.set(null);
    if(session!=null) {
        session.close();
    }
}
}

```

- (6) 在 src 中新建 com.ssm.entity 包，并添加一个商品类 Goods（与 goods 表对应）；

```
test.java GoodsMapper.xml Goods.java
1 package com.ssm.entity;
2
3 public class Goods {
4     private int id;
5     private String goodsname;
6     private int price;
7     private int quantity;
8
9     public Goods() {}
10    public Goods(int id, String goodsname, int price, int quantity) {
11        this.id = id;
12        this.goodsname = goodsname;
13        this.price = price;
14        this.quantity=quantity;
15    }
16
17    public void show(){
18        System.out.println("商品编号:"+id+" 商品名字:"+goodsname+" 商品价格:"+price+" 商品数量:"+quantity);
19    }
20
21    public int getGoods_id() {
22        return id;
23    }
24    public void setGoods_id(int id) {
25        this.id = id;
26    }
}
```

- (7) 在 src 中新建 com.ssm.mapper 包，并添加一个 GoodsMapper.xml 映射文件，将其添加到 mybatis-config.xml 文件中；

```
per.xml - Eclipse IDE
Window Help
test.java GoodsMapper.xml Goods.java
1 <?xml version="1.0" encoding="utf-8" ?>
2 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
3 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
4 <mapper namespace="com.ssm.dao.IGoodsDao">
5     <select id="findAllGoods" resultType="com.ssm.entity.Goods">
6         SELECT
7             id,
8             goodsname,
9             price,
10            quantity
11        FROM goods
12    </select>
13 </mapper>
```

- (8) 在 src 中新建 com.ssm.dao 包，并添加一个 IGoodsDao 接口和实现类 GoodsDaoImpl；

```
test.java GoodsMapper.xml Goods.java IGoodsDao.java
1 package com.ssm.dao;
2
3 import java.util.List;
4
5
6
7 public interface IGoodsDao {
8     //查找学生表中的所有学生信息
9     public List<Goods> findAllGoods();
10 }
11
```



```

1 package com.ssm.dao;
2 import java.io.IOException;
13
14 public class GoodsDaoImpl implements IGoodsDao{
15     @Override
16     public List<Goods> findAllGoods() {
17         SqlSession session = null;
18         List<Goods> list = new ArrayList<Goods>();
19         try {
20             //1. 读取主配置文件mybatis-config.xml
21             String resource = "mybatis-config.xml";
22             Reader reader = Resources.getResourceAsReader(resource);
23             //2. 根据主配置文件mybatis-config.xml构建SqlSessionFactory对象factory
24             SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();
25             SqlSessionFactory factory = builder.build(reader);
26             //3. 根据SqlSessionFactory对象创建SqlSession对象session
27             session = factory.openSession();
28             //4. 调用SqlSession对象session的selectList方法执行查询数据库的操作，返回映射后的结果集合
29             list = session.selectList("com.ssm.dao.IGoodsDao.findAllGoods");
30         } catch (IOException e1) {
31             e1.printStackTrace();
32         } finally {
33             session.close();
34         }
35         return list;
36     }
37 }
38

```

(9) 在 src 中新建 com.ssm.test 包，并添加一个 Test 测试类，实现对 goods 表中所有商品的查询。

```

1 package com.ssm.test;
2 import java.util.List;
3
4 import com.ssm.dao.IGoodsDao;
5 import com.ssm.dao.GoodsDaoImpl;
6 import com.ssm.entity.Goods;
7
8 public class test {
9     public static void findAllGoods(){
10         IGoodsDao goodsDao=new GoodsDaoImpl();
11         List<Goods> sList=goodsDao.findAllGoods();
12         for(int i=0;i<sList.size();i++){
13             sList.get(i).show();
14         }
15     }
16     public static void main(String[] args) {
17         findAllGoods();
18     }
19 }

```

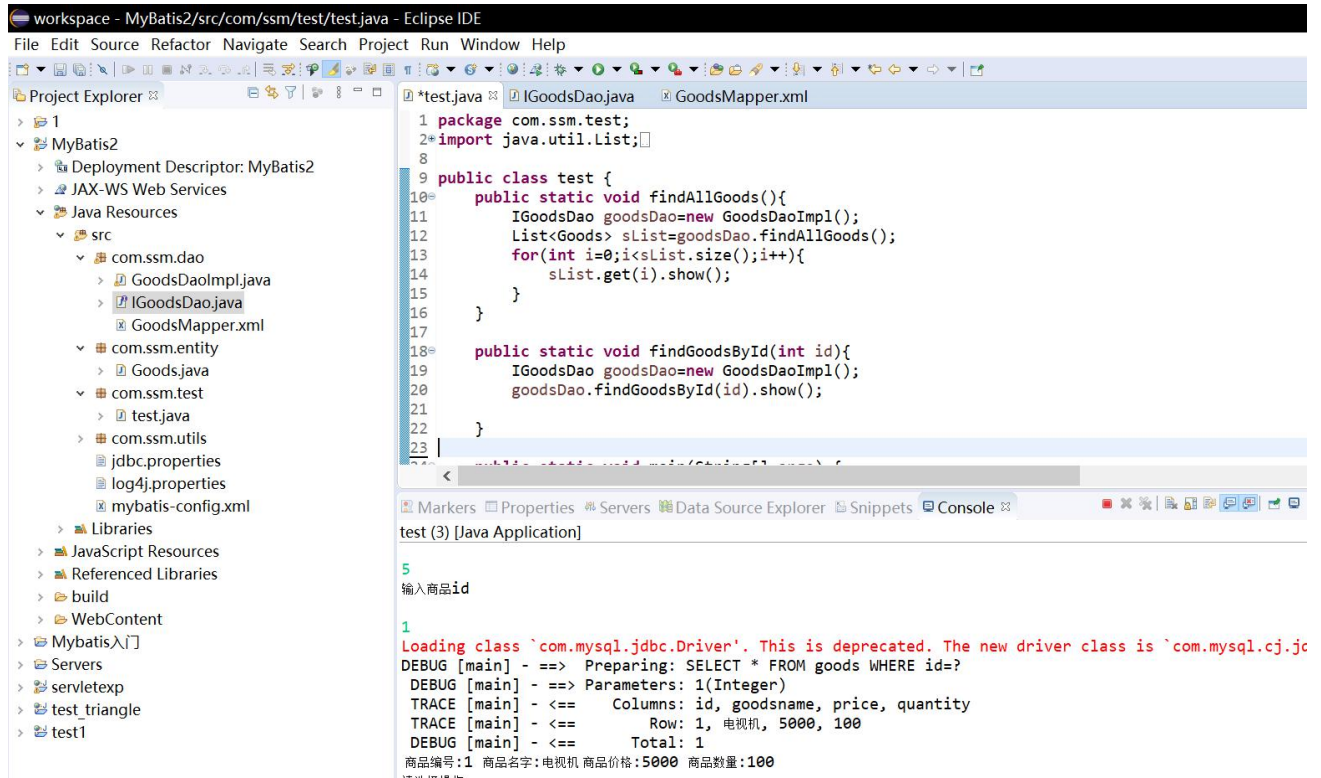
```

<terminated> test (2) [Java Application] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe (2020年12月7日 下午4:08:23 - 下午4:08:25)
DEBUG [main] - ==> Parameters: SELECT id, goodsname, price, quantity FROM goods
TRACE [main] - <==> Columns: id, goodsname, price, quantity
TRACE [main] - <==> Row: 1, 电视机, 5000, 100
TRACE [main] - <==> Row: 2, 电冰箱, 4000, 200
TRACE [main] - <==> Row: 3, 空调, 3000, 300
TRACE [main] - <==> Row: 4, 洗衣机, 3500, 400
DEBUG [main] - <==> Total: 4
商品编号:1 商品名字:电视机 商品价格:5000 商品数量:100
商品编号:2 商品名字:电冰箱 商品价格:4000 商品数量:200
商品编号:3 商品名字:空调 商品价格:3000 商品数量:300
商品编号:4 商品名字:洗衣机 商品价格:3500 商品数量:400

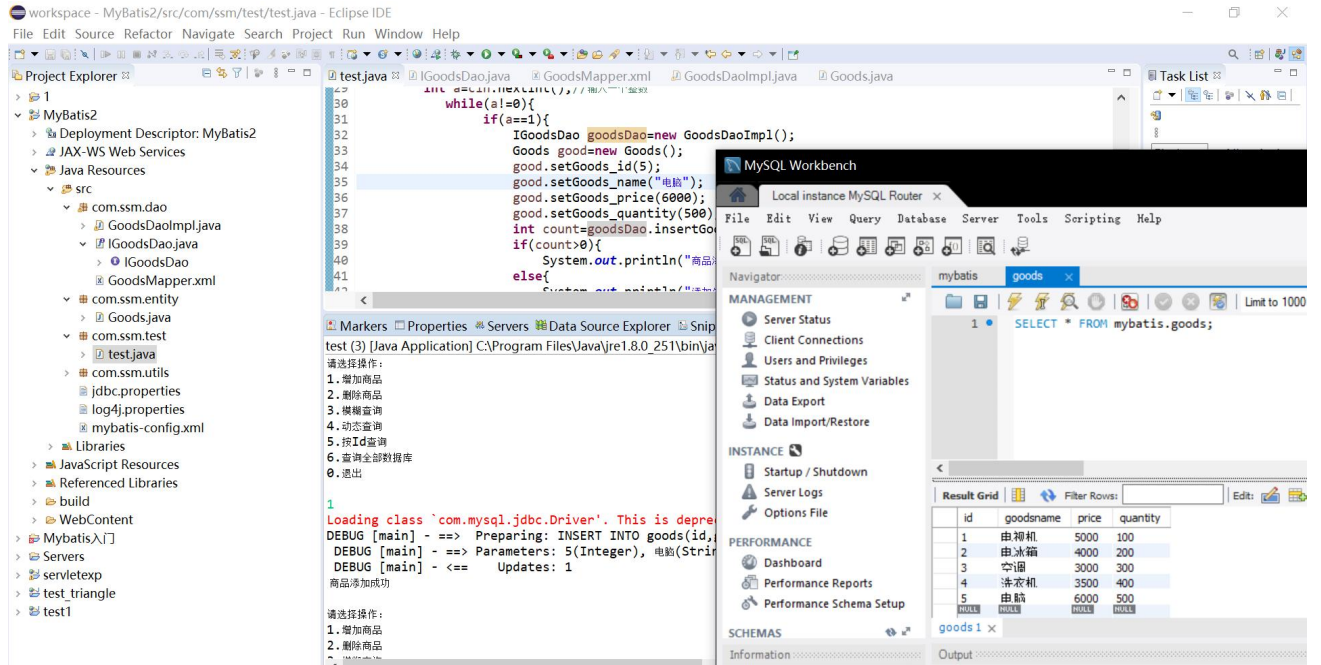
```

2、在第 1 题的基础上，实现按 ID 号查询商品；添加一个新商品；按 ID 号删除一个商品；按商品名称模糊查询商品；动态查询：多个条件为“商品名称、价格、数量组合”。

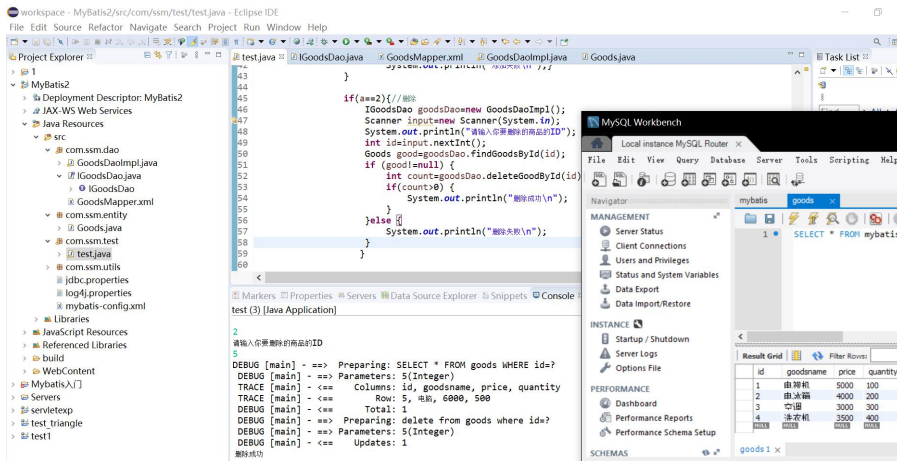
按 ID 号查询商品



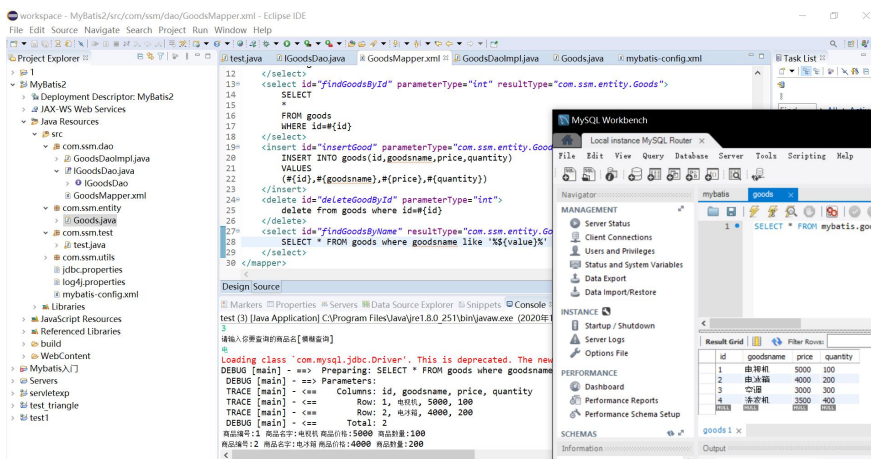
添加一个新商品



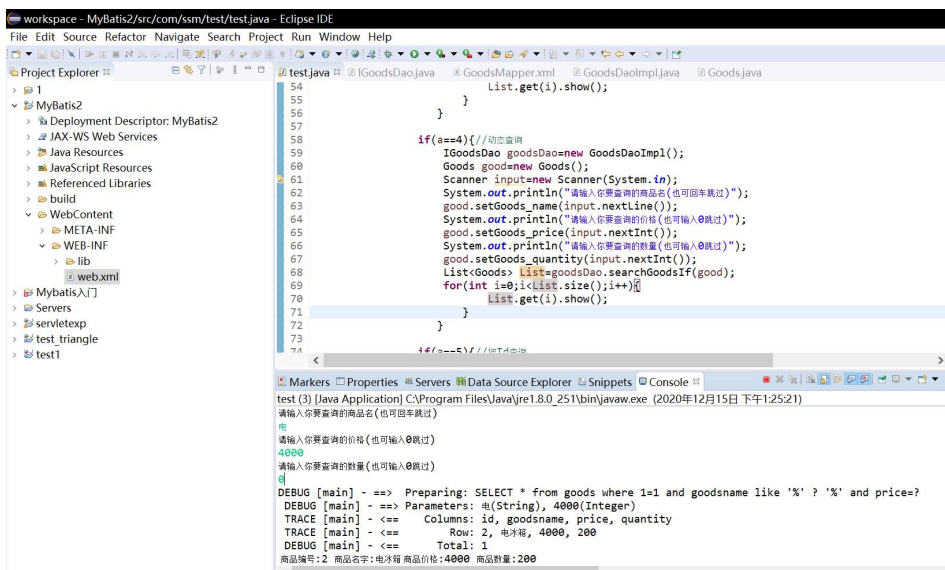
按 ID 号删除一个商品



按商品名称模糊查询商品



动态查询：多个条件为“商品名称、价格、数量组合”



3、修改第 1 题中的商品表，再添加一列：商品类别。（商品表的内容见表 2）

表 2 商品数据表 goods

商品编号 (id)	商品名称 (goodsname)	商品单价 (price)	商品数量 (quantity)	商品类别 (typeid)
1	电视机	5000	100	1
2	电冰箱	4000	200	2
3	空调	3000	300	2
4	洗衣机	3500	400	2

再建一个商品类别表，见表 3。

表 3 商品类别表

商品类别编号(tid)	商品类别名称
1	黑色家电
2	白色家电

要求：

(1) 查询商品表时同时输出其类别名称（多对一/一对一）

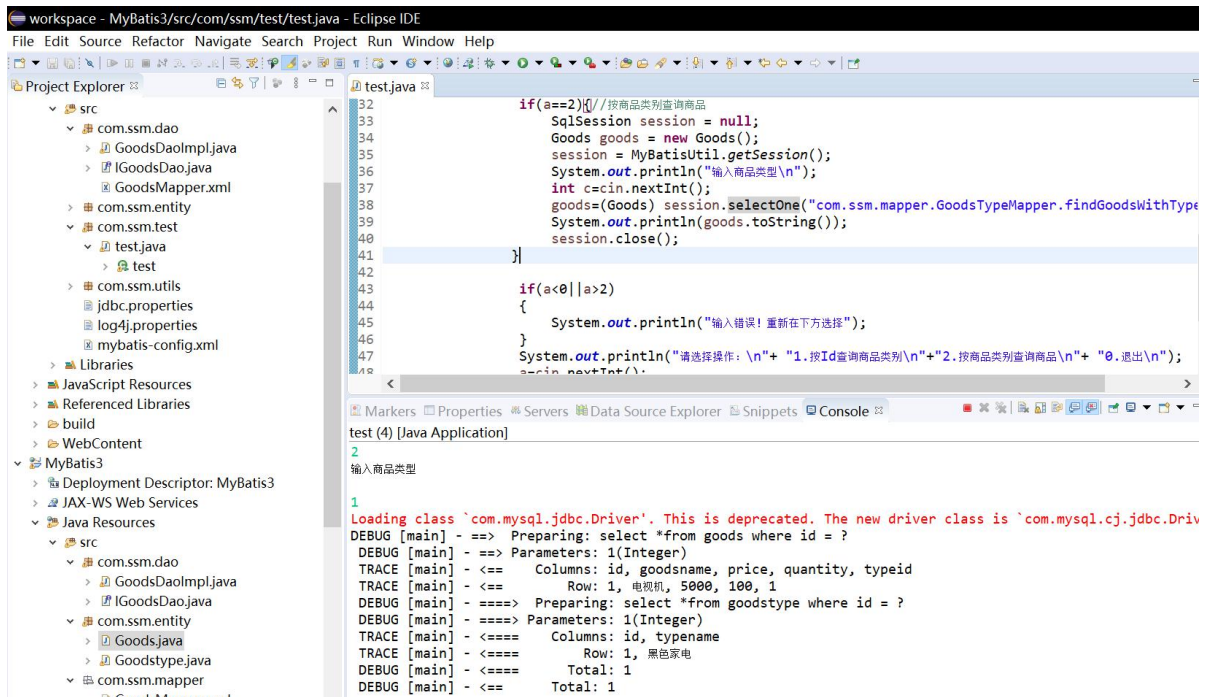
```

13 public class test {
14     public static void main(String[] args) {
15
16         System.out.println("请选择操作：\n"+ "1.按Id查询商品类别\n"+ "2.按商品类别查询商品\n"+ "0.退出\n");
17         @SuppressWarnings("resource")
18         Scanner cin=new Scanner(System.in);
19         int a=cin.nextInt();//输入一个整数
20
21         while(a!=0){
22             if(a==1){//按商品类别查询商品
23                 SqlSession session = null;
24                 Goodstype goodstype = new Goodstype();
25                 session = MyBatisUtil.getSession();
26                 System.out.println("输入商品id\n");
27                 int c=cin.nextInt();
28                 goodstype = session.selectOne("com.ssm.mapper.GoodsTypeMapper.findGoodsTypeWithGc");
29                 System.out.println(goodstype.toString());
30                 session.close();
31             }
32         }
33     }
34 }
    
```

```

test (4) [Java Application]
输入商品id
1
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'.
DEBUG [main] - ==> Preparing: select *from goodstype where id = ?
TRACE [main] - <== Parameters: 1(Integer)
TRACE [main] - <== Columns: id, typename
TRACE [main] - <== Row: 1, 黑色家电
DEBUG [main] - ===== Preparing: select *from goods where id = ?
DEBUG [main] - ===== Parameters: 1(Integer)
TRACE [main] - <===== Columns: id, goodsname, price, quantity, typeid
TRACE [main] - <===== Row: 1, 电视机, 5000, 100, 1
DEBUG [main] - <===== Total: 1
DEBUG [main] - <===== Total: 1
商品编号: 1 商品类别: 黑色家电
    
```

(2) 查询商品类别表时同时输出其商品集合（一对多）



4、在第3题的基础上，创建一个订单表和订单明细表，这里商品表与订单表的关系是多对多的关系，见表4和表5。

表4 订单表

订单编号 orderid	订购单位 buyer	交货日期
1	公司 A	2020-10-1
2	公司 B	2020-10-2
3	公司 C	2020-10-3
4	公司 D	2020-10-4

表5 订单明细表

编号 orderitemid	订单编号 orderid	商品编号 goodid
1	1	1
2	1	2
3	2	2
4	2	3

要求：

(1) 查询1号订单有哪些商品。

```

GoodsToTypeTest.java OrderMapper.xml goodsMapper.xml Order.java
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
3   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
4
5 <mapper namespace="GoodsMapper.OrderMapper">
6   <!-- 嵌套查询 -->
7   <select id="findOrderWithGoods" parameterType="Integer" resultMap="OrderWithGoodsresult">
8     select * from orders where orderId=#{id};
9   </select>
10
11 <resultMap type="com.ssm.entity.Order" id="OrderWithGoodsresult">
12   <id property="id" column="id"/>
13   <result property="buyer" column="buyer"/>
14   <result property="data" column="data"/>
15   <collection property="goods" column="orderid" ofType="com.ssm.entity.Goods"
16     select="GoodsMapper.goodsMapper.findGoodById">
17   </collection>
18 </resultMap>
19 </mapper>
20

```

```

GoodsToTypeTest.java OrderMapper.xml goodsMapper.xml Order.java
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
3   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
4 <mapper namespace="GoodsMapper.goodsMapper">
5   <!-- 根据用户编号获取用户信息 -->
6   <select id="findGoodById" parameterType="Integer" resultMap="GoodsResultMap">
7     select * from goods2 where id in(select goodid from orderDetail where orderId=#{id});
8   </select>
9   <resultMap type="com.ssm.entity.Goods" id="GoodsResultMap">
10    <id property="id" column="id" />
11    <result property="goodsname" column="goodsname" />
12    <result property="price" column="price" />
13    <result property="quantity" column="quantity" />
14  </resultMap>
15 </mapper>
16

```

```

GoodsToTypeTest.java OrderMapper.xml goodsMapper.xml Order.java
1 package com.ssm.test;
2
3 import java.io.IOException;
4
5 public class GoodsToTypeTest {
6   public static void main(String[] args) throws IOException {
7     // TODO Auto-generated method stub
8     String resource="mybatis-config.xml";
9     InputStream inputStream=Resources.getResourceAsStream(resource);
10    SqlSessionFactory sqlSessionFactory=new SqlSessionFactoryBuilder().build(inputStream);
11    SqlSession sqlSession=sqlSessionFactory.openSession();
12    Order ty=sqlSession.selectOne("GoodsMapper.OrderMapper.findOrderWithGoods",1);
13    System.out.println(ty.toString());
14    sqlSession.commit();
15    sqlSession.close();
16  }
17 }
18

```

<terminated> GoodsToTypeTest (2) [Java Application] D:\jdk\bin\javaw.exe (2020年12月6日 下午6:23:17)
 订单 [订单号: 0, 买家: 公司A, 日期: 2020-10-1, 购买商品: [Goods [id=1, goodsname=电视机, price=5000, quantity=100], Goods [id=2, good

(2) 查询在哪些订单里有 2 号商品。

```

GoodsToTypeTest.java
1 package com.ssm.test;
2
3 import java.io.IOException;
4
5 public class GoodsToTypeTest {
6   public static void main(String[] args) throws IOException {
7     // TODO Auto-generated method stub
8     String resource="mybatis-config.xml";
9     InputStream inputStream=Resources.getResourceAsStream(resource);
10    SqlSessionFactory sqlSessionFactory=new SqlSessionFactoryBuilder().build(inputStream);
11    SqlSession sqlSession=sqlSessionFactory.openSession();
12
13    Goods ty=sqlSession.selectOne("GoodsMapper.goodsMapper.findGoodsWithOrder",2);
14    System.out.println(ty);
15    sqlSession.commit();
16    sqlSession.close();
17  }
18 }
19

```

<terminated> GoodsToTypeTest (3) [Java Application] D:\jdk\bin\javaw.exe (2020年12月6日 下午7:12:33)
 Goods [id=2, goodsname=电冰箱, price=4000, quantity=200, order=[订单 [订单号: 1, 买家: 公司A, 日期: 2020-10-1], 订单 [订单号: 2,

四、实验报告要求：

实验报告模板如下（见附件 1 信息学院实验报告模板）：

实验六 Spring MVC 实验

一、实验目标

- 掌握 Spring MVC 框架的搭建
- 掌握 Spring MVC 的注解
- 掌握 Spring MVC 数据绑定的使用
- 理解 Spring MVC 的拦截器原理及作用

二、实验环境

- 1、Java、MySQL、Spring、MyBatis、Spring MVC 等。
- 2、Eclipse 集成开发环境

三、实验内容

1、搭建 Spring MVC 环境, 在前端页面输出欢迎页面“欢迎进入 Spring MVC 的学习与使用!” (要求使用配置式和注解式分别实现)。



配置式

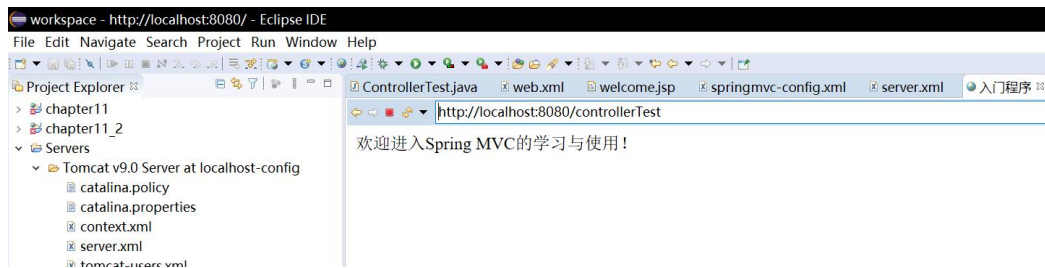
```
ControllerTest.java | web.xml | welcome.jsp | springmvc-config.xml | server.xml | 入门程序
1 package com.ssm.controller;
2 import javax.servlet.http.HttpServletRequest;
3 import javax.servlet.http.HttpServletResponse;
4 import org.springframework.web.servlet.ModelAndView;
5 import org.springframework.web.servlet.mvc.Controller;
6 public class ControllerTest implements Controller {
7     @Override
8     public ModelAndView handleRequest(HttpServletRequest arg0, HttpServletResponse arg1) throws Except
9         ModelAndView m=new ModelAndView();
10        m.addObject("msg","欢迎进入Spring MVC的学习与使用!");
11        m.setViewName("/WEB-INF/jsp/welcome.jsp");
12        return m;
13    }
14 }
```



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://www.springframework.org/schema/beans
5         http://www.springframework.org/schema/beans/spring-beans-4.3.xsd">
6     <!--配置处理器Handle, 映射"controllerTest"请求 -->
7     <bean name="/controllerTest" class="com.ssm.controller.ControllerTest" />
8     <!--处理器映射, 将处理器Handle的name作为url进行查找 -->
9     <bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping" />
10    <!--处理器适配器, 配置对处理器中handleRequest()方法的调用 -->
11    <bean class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter" />
12    <!--视图解析器 -->
13    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver" />
14 </beans>

```



注解式

```

1 package com.ssm.controller;
2 import javax.servlet.http.HttpServletRequest;
3
4 @Controller
5 @RequestMapping(value = "/controll")
6 public class ControllerTest {
7     @RequestMapping(value = "/annotationController")
8     public String handleRequest(HttpServletRequest arg0, HttpServletResponse arg1, Model model) throws
9         model.addAttribute("msg", "欢迎进入Spring MVC的学习与使用!");
10    return "welcome";
11 }

```

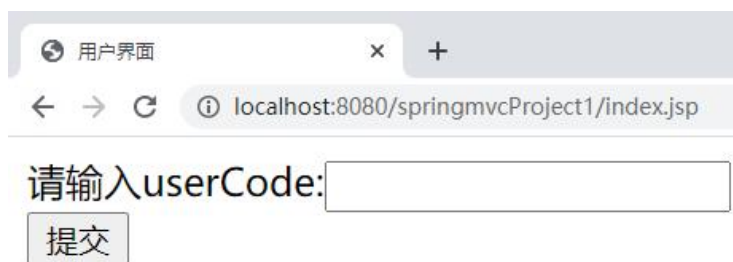
```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xmlns:context="http://www.springframework.org/schema/context"
5     xsi:schemaLocation="http://www.springframework.org/schema/beans
6         http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
7         http://www.springframework.org/schema/context
8         http://www.springframework.org/schema/context/spring-context-4.3.xsd">
9     <!--指定需要扫描的包 -->
10    <context:component-scan base-package="com.ssm.controller" />
11    <!-- 定义视图解析器 -->
12    <bean id="viewResolver"
13        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
14        <!-- 设置前缀 -->
15        <property name="prefix" value="/WEB-INF/jsp/" />
16        <!-- 设置后缀 -->
17        <property name="suffix" value=".jsp" />
18    </bean>
19 </beans>

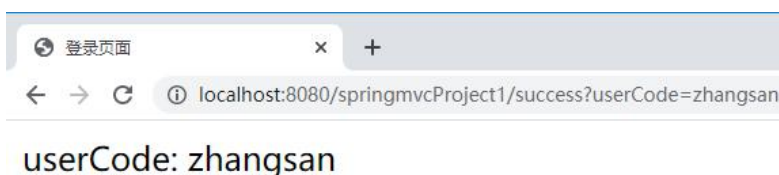
```



2、在1的基础上，完成 View 与 Controller 之间的参数传递，在用户界面（index.jsp）提供输入的 inputText 框中，输入用户编码，如下图所示。



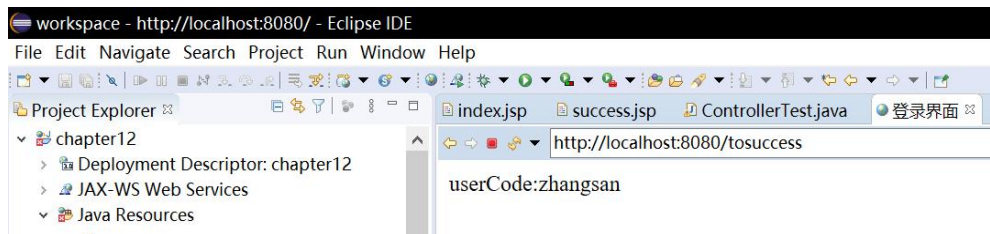
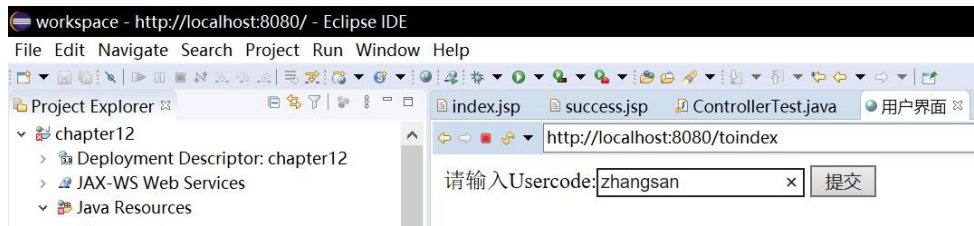
输入编码后单击“提交”按钮，跳转到如下图所示界面（WEB-INF/jsp/success.jsp），在该界面输出上一界面中输入并提交的用户编码。



```
index.jsp success.jsp ControllerTest.java 登录界面
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <title>用户界面</title>
8 </head>
9 <body>
10 <form action="{pageContext.request.contextPath}/tosuccess" method="post">
11 请输入Usercode:<input type="text" name="usercode">
12 <%
13   String usercode = request.getParameter("usercode");
14
15 %>
16
17 <input type="submit" value="提交"/>
18 </form>
19 </body>
20 </html>
```

```
index.jsp success.jsp ControllerTest.java 登录界面
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <title>登录界面</title>
8 </head>
9 <body>
10 userCode:<%=request.getSession().getAttribute("res")%>
11 </body>
12 </html>
```

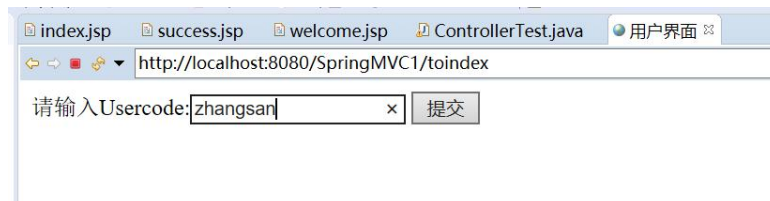
```
index.jsp success.jsp ControllerTest.java 登录界面
1 package com.ssm.controller;
2 import java.io.IOException;
12 @Controller
13 public class ControllerTest {
14     @RequestMapping(value = "/toindex")
15     public String toindex() {
16         return "index";
17     }
18     @RequestMapping(value = "/tosuccess")
19     public String tosuccess(HttpServletRequest request, HttpServletResponse response) throws IOException {
20         String usercode=request.getParameter("usercode");
21         HttpSession session = request.getSession();
22         session.setAttribute("res", usercode);
23         System.out.println(usercode);
24         return "success";
25     }
26 }
```

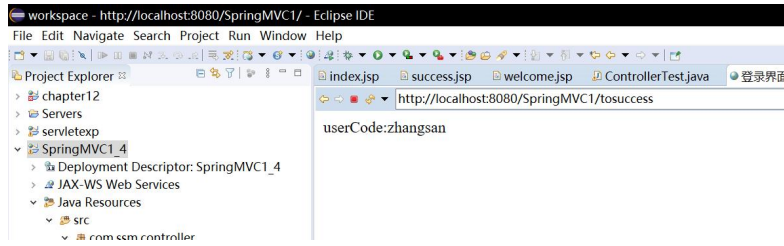


3、在 2 的基础上，继续练习 View 与 Controller 之间的参数传递（含默认数据类型、简单数据类型、POJO 类型、包装 POJO 类型、数组和集合）。

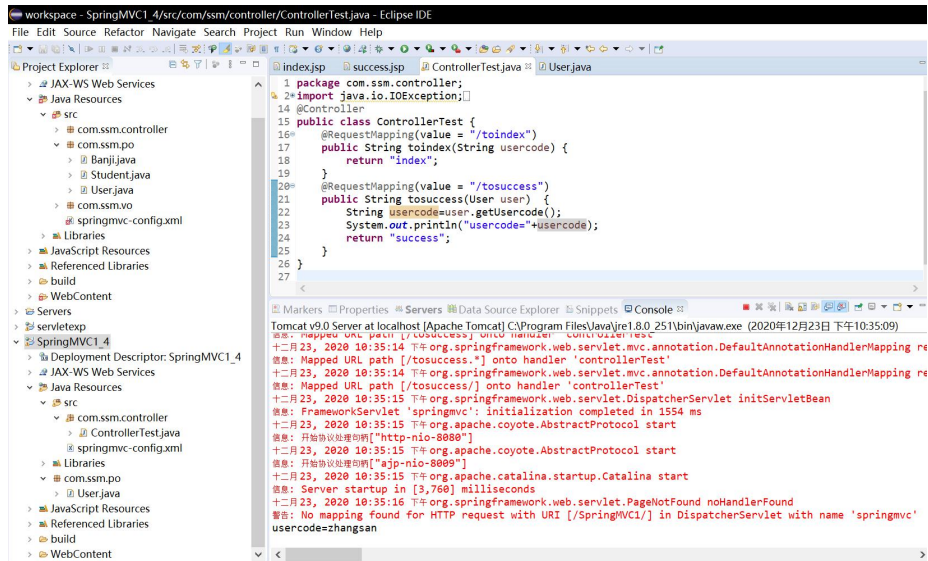
简单数据类型：

```
index.jsp success.jsp welcome.jsp ControllerTest.java 登录界面
1 package com.ssm.controller;
2 import java.io.IOException;
12 @Controller
13 public class ControllerTest {
14     @RequestMapping(value = "/toindex")
15     public String toindex(String usercode) {
16         System.out.println("userCode="+usercode);
17         return "index";
18     }
19     @RequestMapping(value = "/tosuccess")
20     public String tosuccess(HttpServletRequest request, HttpServletResponse response) throws IOException {
21         String usercode=request.getParameter("usercode");
22         HttpSession session = request.getSession();
23         session.setAttribute("res", usercode);
24         System.out.println(usercode);
25         return "success";
26     }
27 }
```

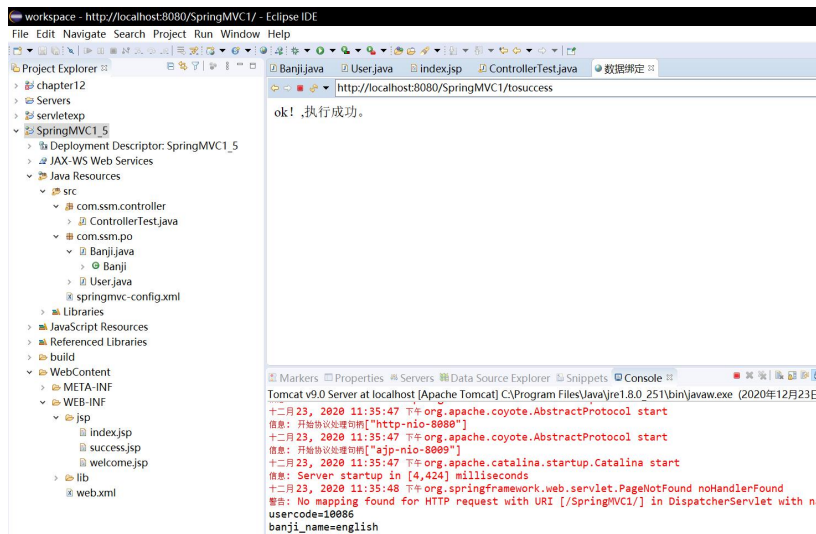
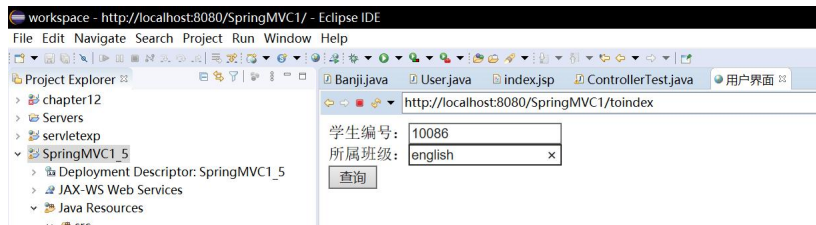


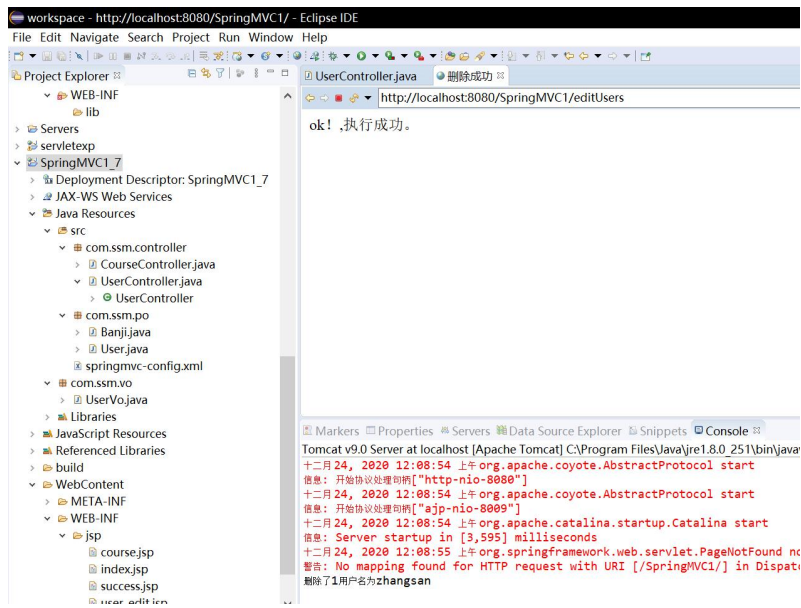


POJO 类型

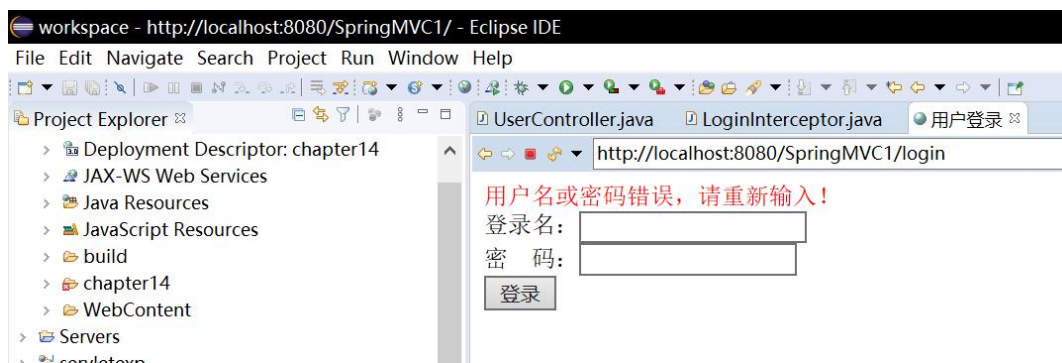
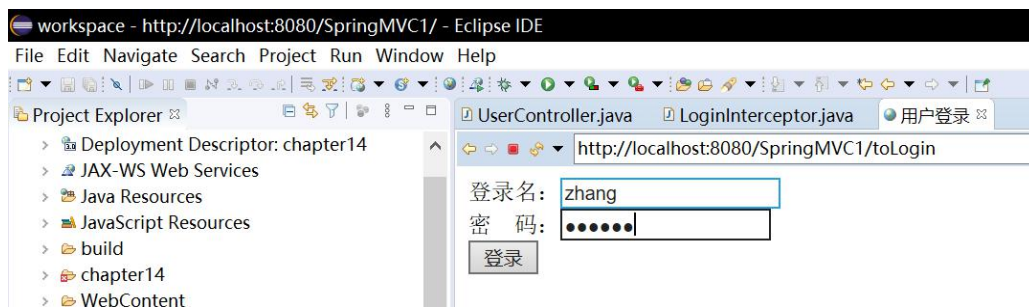


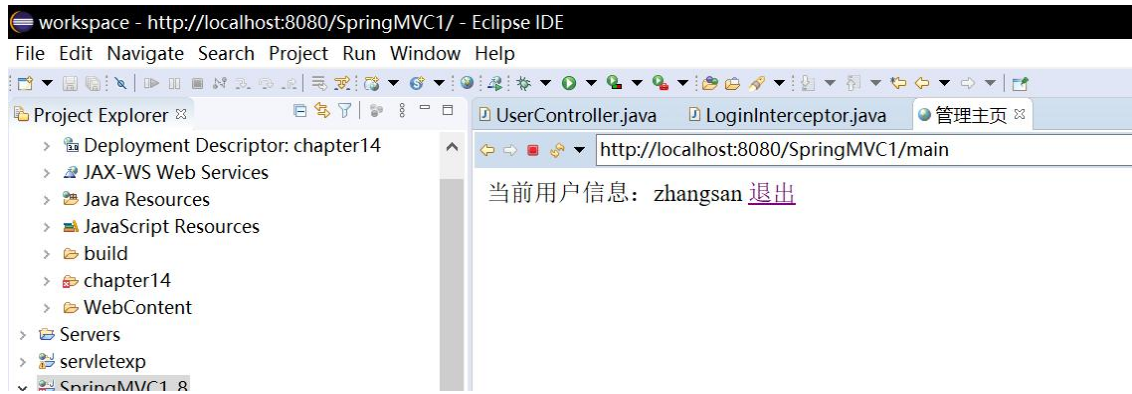
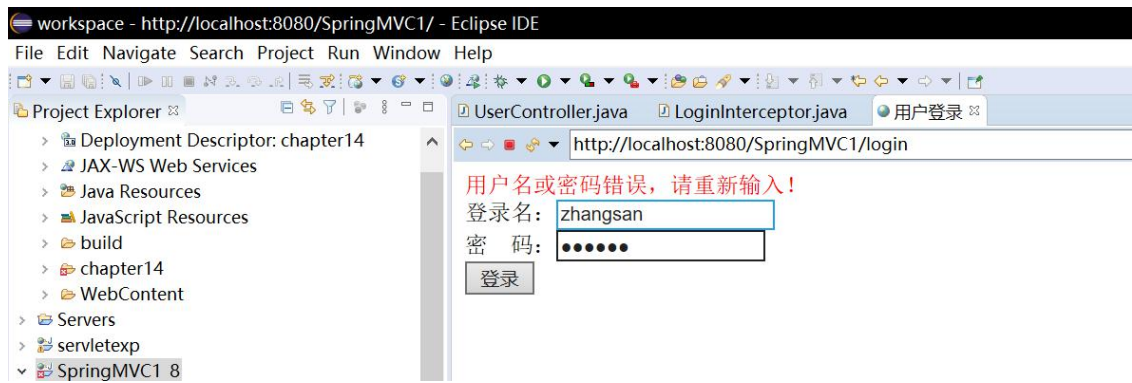
包装 POJO 类型





4、创建一个项目，内有学生类，实现学生信息的数据验证功能。（使用拦截器）





四、实验报告要求:

实验报告模板如下（见附件 1 信息学院实验报告模板）:

实验七 Web 程序设计综合实验

一、背景

为提高学校对志愿者服务管理的水平，请基于 SSM 框架设计、开发学生志愿者服务管理网站。主要功能大致包括：发布志愿者服务项目的招募信息、志愿者报名、评定志愿者的表现等。

二、综合实验要求

1、系统功能模块主要由以下四部分构成

- 1) 志愿者信息：增加、删除、修改和查询
- 2) 志愿活动信息：增加、删除、修改和查询
- 3) 成绩评定：增加、删除、修改和查询
- 4) 用户注册、登录、管理等（用户可以分为学生志愿者、项目的负责教师、系统管理员等）

说明：

- 不同角色，权限不同。例如：学生只能查询志愿者项目、选择或改选或退选志愿者项目；教师可以增删改查自己负责的志愿者服务项目，给出此项目中志愿者的成绩，对其他教师负责的志愿者服务项目无删改权限；系统管理员维护基本数据（学生志愿者、志愿服务负责的教师）
- 除基本功能之外，请尽可能完善。例如：志愿者项目的负责老师可以是多人，其中三位教师给出参与志愿服务项目的学生成绩，然后汇总取平均分；导入教师及学生名单；导出成绩单等。

3、网站中页面的布局、风格等适当统一，页面的布局可参考下图但不局限于此：

图片（应用系统名） 当前位置	
网站导航	当前网页的功能实现区
网站制作、联系等信息	

三、综合实验开展形式

本次综合实验是分组实验（每组 1-3 人），选出组长并进行分工。分组在“泛雅”（学习通）填写。

四、综合实验提交

1、系统代码：

2021 年 1 月 6 日系统实现演示检查。

演示检查后，请组长将项目文件夹及建立数据库的 sql 文件等打包压缩，并以组长的学号命名，提交至泛雅平台。

2、实验报告：每人一份，主要内容有：

- (1) 同组成员及分工
- (2) 简单说明采用的技术、框架
- (3) 描述实现的主要功能、数据库（表及其结构、视图等）
- (4) 找部分源代码（体现本组技术水平的）进行注释说明
- (5) 内容（4）相关的运行结果（截图粘贴）
- (6) 综合实验的总结（成果及不足）

电子版实验报告：提交至课程泛雅平台，提交截止时间：2021 年 1 月 12 日。

纸质版实验报告：提交截止时间：2021 年 1 月 12 日。

五、实验报告的格式要求：

见附件 1 信息学院实验报告模板。

附件：



实验报告

题目： _____

学院：信息学院

专业：

班级：

学号：

姓名：

年 月 日

一、实验目的（宋体四号加粗）

正文（正文 宋体小四，1.5 倍行距）

二、实验环境（宋体四号加粗）

三、实验内容（宋体四号加粗）

四、实验步骤（图文方式叙述）（宋体四号加粗）

五、实验结果及分析（遇到的问题与解决）（宋体四号加粗）

六、实验体会（宋体四号加粗）

