《地理信息网络服务》实验报告

姓名： 武成龙

班级： 地理信息科学19-1班

学号： 07192329

中国矿业大学环境与测绘学院

2022年5月3日

**上机1 网站制作与发布**

**1.上机目的**

掌握HTML语言的基础知识，并能构建一个简单网站；初步掌握CSS、JavaScript技术的应用；掌握Web服务器的配置与应用方法。

**2.上机软件环境**

操作系统：Win10等；

编辑软件：WebStorm、VSCode等；

**3.上机内容**

（1）进行网页制作，尽可能多地使用到HTML各种标记，网站包含网页页面数>=3页；页面之间应有所关联；

（2）使用层叠样式表（CSS）进行网页格式的控制；

（3）使用Javascript技术提高页面的可交互性与动态性；

（4）配置Web服务器，发布个人网站。

**4.上机要求**

（1）以大学中学习的一门课程为主题进行网站制作；

    （2）独立实验，按时完成上机任务；

（3）认真编写上机实验报告，排版规范、条理清晰；

（4）各网页应主题突出、美观大方、色调和谐、内容积极传播正能量；

**5.上机步骤**

# 使用技术及平台

框架：Bootstrap v3.4.1

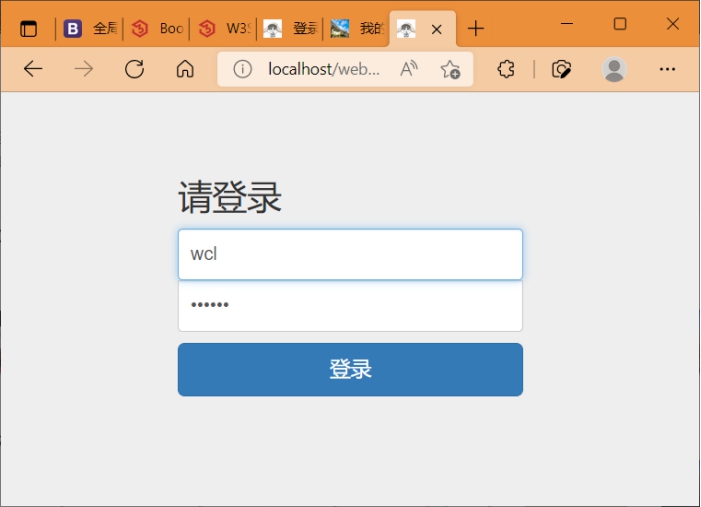
操作系统：Windows10

编辑软件：WebStorm

# 整体构思

总体分为登录页、主页和课程资料页。

从登录页登录成功之后跳转到主页，主页有对网站的介绍。通过主页可以跳转到课程资料页。各个页面之间可以互相跳转，共6个页面。



登录页



主页



课程资料页（小屏幕）



课程资料页（中等屏幕）

创新点：

1.全部使用Bootstrap 这一 HTML、CSS 和 JS 框架，用于开发具有响应式布局的网页。开发的网站和应用能在 Bootstrap 的帮助下通过同一份源码快速、有效地适配手机、平板和 PC 设备。

2.在课程资料页左侧添加了一个固定导航栏，其中以胶囊形式显示当前链接，存放了关键内容的链接，可以让用户快速定位到需要的内容。并且添加了一个“回到顶部“的功能，单击即可回到网页顶部，方便用户跳转到其他网页，大大提高了可读性和灵活性。代码如下：

<div class="col-sm-2 ">   
    <!-- Nav pills -->   
    <ul class="nav nav-pills flex-column sticky-top" role="tablist">   
        <h2 class="mt-1 text-center">快速参考</h2>   
        <hr>   
        <li class="nav-item">   
            <a class="nav-link active" data-bs-toggle="pill" href="#home">主题词</a>   
        </li>   
        <li class="nav-item">   
            <a class="nav-link" data-bs-toggle="pill" href="#menu1">概念的提取与扩展</a>   
        </li>   
        <hr>   
        <a class="text-center" href="#top" >返回顶部</a>   
    </ul>   
</div>

3.在矿图介绍页面，添加了图片轮播和视频。代码如下：

<!-- 轮播 -->   
<div id="demo" class="carousel slide" data-bs-ride="carousel">   
    <!-- 指标/点 -->   
    <div class="carousel-indicators">   
        <button type="button" data-bs-target="#demo" data-bs-slide-to="0" class="active"></button>   
        <button type="button" data-bs-target="#demo" data-bs-slide-to="1"></button>   
        <button type="button" data-bs-target="#demo" data-bs-slide-to="2"></button>   
    </div>   
   
    <!-- 幻灯片/轮播 -->   
    <div class="carousel-inner">   
        <div class="carousel-item active">   
            <img src="./pic/lib%20(1).jpg" alt="1" class="d-block" style="width:100%">   
        </div>   
        <div class="carousel-item">   
            <img src="./pic/lib%20(2).jpg" alt="2" class="d-block" style="width:100%">   
        </div>   
        <div class="carousel-item">   
            <img src="./pic/lib%20(3).jpg" alt="3" class="d-block" style="width:100%">   
        </div>   
    </div>   
   
    <!-- 左右控件/图标 -->   
    <button class="carousel-control-prev" type="button" data-bs-target="#demo" data-bs-slide="prev">   
        <span class="carousel-control-prev-icon"></span>   
    </button>   
    <button class="carousel-control-next" type="button" data-bs-target="#demo" data-bs-slide="next">   
        <span class="carousel-control-next-icon"></span>   
    </button>   
</div>   
   
</div>   
<div class="row">   
    <video  controls="controls" autoplay="autoplay">   
        <source src="./video/lib.mp4" type="video/ogg" />   
    </video>

## 课程资料页结构

### 为什么要学习文献检索？

• 你能解决的问题

• 对科研人员的帮助

• 文献检索课简介

### 中国矿业大学图书馆简介

• 图书馆历史沿革

• 馆舍及服务模式

• 文献资源

### 基础知识

• 文献类型

• 信息与文献

### 计算机检索语言

• 主题词

• 概念的提取和扩展

# 发布网站

使用 IIS 创建虚拟目录，就可以通过在浏览器输入URL（http://localhost/webgis/）进入登录页面了。

# 实验小结

在编写代码过程中遇到了些问题：

• 登录页跳转功能的实现。由于连接数据库比较麻烦，所以采用了JS函数判断输入的值是否为预设值，若是则跳转到主页。

• 配置IIS之后不能正常打开图片，是因为图片引用格式不对。

经过这次实验，对HTML、CSS、JS的理解更深刻了，为后面的前端开发打下了基础。

**上机2 基于地图API的简单WebGIS应用**

**1.上机目的**

熟悉地图API（如百度地图API、高德地图API、腾讯地图API、天地图API等），并结合JavaScript\Html\CSS等技术实现WebGIS功能。理解基于地图API的WebGIS系统的实现原理。

**2.上机软件环境**

（1） Windows操作系统等；

（2） 百度地图API/高德地图API/腾讯地图API/天地图API等。

**3.上机内容**

（1）以中国矿业大学南湖校区为中心加载地图；

（2）查找出中国矿业大学南湖校区内的教室\超市信息；

（3）中国矿业大学各主要餐饮点到达环测学院大楼不同交通方式的线路分析（可选）。

**4.程序关键代码与界面**

## 加载地图

代码如下：

//创建地图实例   
var map = new BMapGL.Map("container");   
//设置中心点   
var centerPoint = new BMapGL.Point(117.151767, 34.221332);   
//地图初始化，同时设置地图展示级别   
map.centerAndZoom(centerPoint, 16);   
//开启鼠标滚轮缩放   
map.enableScrollWheelZoom(true);

界面如下：



以中国矿业大学南湖校区为中心

## 搜索超市/博学楼

使用 searchNearby 函数根据中心点、半径与检索词发起周边检索。代码如下：

// 本地关键词搜索   
function searchKey(){   
    // 清空之前所有搜索记录   
    map.clearOverlays();   
    // 清空路径规划结果   
    var result = document.getElementById("routeResult");   
    if (result!=null)   
        result.innerHTML = '';   
    // 获取关键词   
    var keyWord = document.getElementById("search").value;   
    var options = {   
        renderOptions: {   
            map: map,   
            autoViewport: true,   
            panel: "searchResult"   
        }   
    }   
    var local =  new BMapGL.LocalSearch(map,options);   
    local.searchNearby(keyWord, centerPoint, 900);   
}

界面如下：



搜索超市结果

## 路径规划

用户输入起点和终点，选择出行模式为步行，骑行或驾车（校内无公交）。选择规划方式为自动规划时，先进行搜索，把返回的第一个POI结果作为路线的起点和终点，输出结果；选择选点规划时，用户在本城市内选择合适的起点和终点，然后输出结果。结果列表将容器中进行展示，对驾车路线规划无效。

主要代码如下：

function autoNevigation(){   
    // 获取POI   
    // 清空之前所有搜索记录   
    map.clearOverlays();   
    // 获取关键词   
    var startKeyWord = document.getElementById("start").value;   
    var endKeyWord = document.getElementById("end").value;   
    var startPOI;   
    var endPOI;   
    var startOptions = {   
        onSearchComplete: function (results) {   
            // 判断状态是否正确   
            if (startLocal.getStatus() == BMAP\_STATUS\_SUCCESS) {   
                // 返回第一个搜索结果作为路径规划的起点   
                startPOI = results.getPoi(0);   
                console.log(startPOI);   
                var endOptions = {   
                    onSearchComplete: function (results) {   
                        // 判断状态是否正确   
                        if (endLocal.getStatus() == BMAP\_STATUS\_SUCCESS) {   
                            // 返回第一个搜索结果作为路径规划的终点   
                            endPOI = results.getPoi(0);   
                            // console.log(endPOI);   
                            var select = document.getElementById("routeSelect").value;   
                            if (select == 'walking')   
                                walkingRouteSearch(startPOI,endPOI);   
                            else if (select == 'riding')   
                                ridingRouteSearch(startPOI, endPOI);   
                            else   
                                drivingRouteSearch(startPOI,endPOI);   
                        }   
                    }   
                }   
                var endLocal =  new BMapGL.LocalSearch(map,endOptions);   
                endLocal.searchNearby(endKeyWord, centerPoint, 1000);   
            }   
        }   
    }   
    // 本地半径查询   
    var startLocal =  new BMapGL.LocalSearch(map,startOptions);   
    startLocal.searchNearby(startKeyWord, centerPoint, 1000);   
    // 骑行规划   
    function ridingRouteSearch(startPOI, endPOI){   
        // 清空搜索结果   
        var result = document.getElementById("searchResult");   
        if (result!=null)   
            result.innerHTML = '';   
        var riding = new BMapGL.RidingRoute(map, {   
            renderOptions: {   
                map: map,   
                panel: "routeResult",   
                autoViewport: true   
            },   
        });   
        riding.search(startPOI, endPOI);   
    }   
    // 步行规划   
    function walkingRouteSearch(startPOI, endPOI){   
        // 清空搜索结果   
        var result = document.getElementById("searchResult");   
        if (result!=null)   
            result.innerHTML = '';   
   
        var walking = new BMapGL.WalkingRoute(map, {   
            renderOptions: {   
                map: map,   
                panel: "routeResult",   
                autoViewport: true   
            },   
        });   
        walking.search(startPOI, endPOI);   
    }   
    //   
    function drivingRouteSearch(startPOI, endPOI){   
        // 清空搜索结果   
        var result = document.getElementById("searchResult");   
        if (result!=null)   
            result.innerHTML = '';   
        var driving = new BMapGL.DrivingRoute(map, {   
            renderOptions:{   
                map: map,   
                panel: "routeResult",   
                autoViewport: true   
            }   
        });   
        driving.search(startPOI,endPOI);   
    }   
}

界面如下：



一食堂步行到达环测学院大楼



二食堂骑行到达环测学院大楼



三食堂驾车到达环测学院大楼

# 实验小结

前期尝试了天地图API，发现不够大众化，超市搜索结果不够全面；同时腾讯地图API封装做得不好，调用比较麻烦；最后选择了百度地图API。利用API开发应先把功能写好，再考虑美化设计。最后，开发路径规划功能时遇到了回调地狱问题，这也是调用第三方API的一个弊端。

本程序最大的特点就是不限制路径规划的输入，三种出行模式和两种规划模式都可以输出结果。

**上机3 利用ASP.NET制作留言板**

**1.上机目的**

了解并掌握ASP.NET的基本开发方法。学会ASP.NET Web应用程序或ASP.NET网站的建立方法。掌握HTML、ASP.NET、ACCESS数据库的基本知识，并以此为基础构建一个基于Access数据库的简易留言板Web应用。

**2.上机软件环境**

OS：Win10等；

编辑软件、调试工具：Visual Studio，常用的浏览器等；

数据库：Access等；

**3.上机内容**

（1）学会基于Visual Studio创建Web应用程序并且连接Accesss数据库的方法；

（2）编写操作数据库的代码，能够实现对Accesss数据库的CRUD操作；

（3）根据留言板应用的需求，能够合理对前端表现层进行规划，即通过HTML中的元素、CSS等工具能够创建符合需求的页面；

（4）ASP.NET的数据库基本使用方法；

（5）使用上述四项上机内容或者ASP.NET原生的工具、代码创建一个简易的留言板，至少包含留言、展示留言的功能。

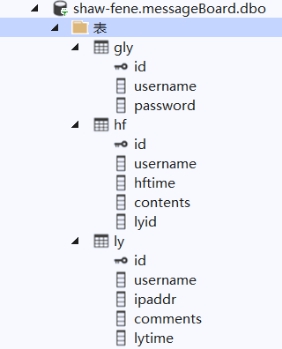
**4.上机步骤**

# 数据库部分

## 创建数据库

使用 SQL Server 代替 Access，表结构不变。其中要注意：

* 设置 id 为主键
* 设置 id 为标识列
* 设置lyid 为外键，对应留言的 id 。
* 时间的类型为 datetime ，如 2022-04-11 09:05:06 。





数据表结构

## 连接数据库

web.config

VS 数据库连接工具测试：菜单>工具>连接到数据库>

选择 SQL Server 数据源，然后选择 SQL Server 数据库文件，点击测试连接。

将连接字符串放在web.config文件中，代码如下：

<connectionStrings>  
 <add name="sqlConnect" connectionString="Data Source=SHAW-FENE;Initial Catalog=messageBoard;Integrated Security=True"/>  
 </connectionStrings>

DBHelper.cs

将原有的命名空间注释，使用新的命名空间。

更换变量 connStr 中字符串。

并且将所有原有的类替换为 SqlClient 中的类。

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Data;  
//using System.Data.OleDb;  
using System.Configuration;  
using System.Data.SqlClient;//SQL server数据提供程序  
  
namespace lyban.App\_Code  
{  
 public class DBHelper  
 {  
 private static readonly string connStr = ConfigurationManager.ConnectionStrings["sqlConnect"].ConnectionString;  
 private SqlConnection conn;//负责打开或连接数据库文件  
 private SqlCommand cmd;//对数据库下达行动查询指令，以及执行SQL Server的存储过程  
 // 构造函数  
 public DBHelper() { }  
  
 public SqlConnection getConn()  
 {  
 conn = new SqlConnection(connStr);  
 return conn;  
 }

# 前后端交互部分

* 回复成功之后清空回复框。
* 优化按钮显示样式。

## 留言

显示数据库中已有留言

* + init.ashx 修改 SQL 语句等
* public void ProcessRequest(HttpContext context)  
  {  
   DBHelper dh = new DBHelper();  
   // 选择留言表的所有数据  
   string sql = "select \* from ly";  
   SqlDataReader reader = dh.queryDb(sql);  
   List<message> lylist = new List<message>();  
   // 判断表是否为空  
   if (reader.HasRows)  
   {  
   while (reader.Read())  
   {  
   message tmp = new message();  
   tmp.id = (int)reader[0];  
   tmp.username = (string)reader[1];  
   tmp.ipaddr = (string)reader[2];  
   tmp.comments = (string)reader[3];  
   tmp.lytime = Convert.ToDateTime(reader[4]);  
   lylist.Add(tmp);  
   }  
   // 关闭数据库连接，释放资源  
   dh.closeDb();  
   // 把留言转换成JSON格式  
   string res = JsonConvert.SerializeObject(lylist);  
   context.Response.Write(res);  
   }  
   else   
   {  
   context.Response.Write("error");  
   }  
  }

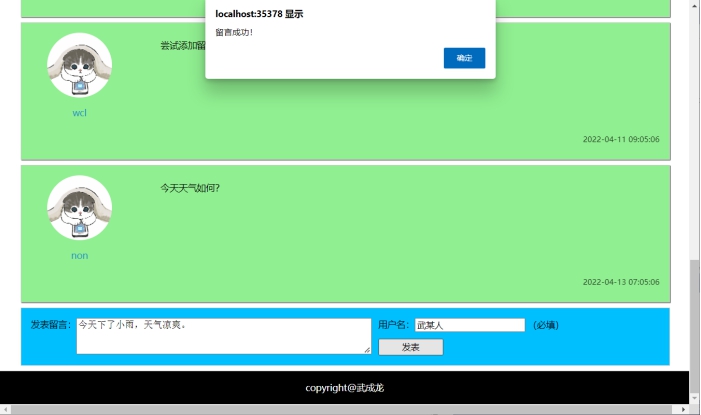
存放新留言到数据库中

* + handleMsg.ashx 修改 SQL 语句等
* public void ProcessRequest(HttpContext context)  
  {  
   if (context.Request["username"] != null && context.Request["comments"] != null)  
   {  
   // 假如用户名和留言不为空，就创建一个 message 对象。  
   message ly = new message();  
   ly.username = context.Request["username"];  
   ly.comments = context.Request["comments"];  
   ly.ipaddr = context.Request.UserHostAddress=="::1"?"127.0.0.1":context.Request.UserHostAddress;  
   ly.lytime = DateTime.Now;   
   DBHelper dh = new DBHelper();  
   // 定义SQL语句  
   string sql = "insert into ly (username,ipaddr,comments,lytime) values ( '" +  
   ly.username + "','" + ly.ipaddr + "','" + ly.comments + "','" + ly.lytime.ToString() + " ')";  
   // 执行SQL语句  
   int res =dh.ExeCurd(sql);  
   if (res > 0)  
   {  
   context.Response.Write("success");  
   }  
   }  
  }

留言成功后清空留言窗口

* + index.js
* function sendMsg(msg) {  
   $.ajax({  
   type:"POST",  
   url: "handleMsg.ashx",  
   data: msg,  
   success: function (res,status,xhr) {  
   if (res = "success") {   
   alert("留言成功！");  
   //添加留言  
   var comments = $(".comments").eq(0).clone();  
   comments.find(".uname").find("label").text($("#uname2").val());  
   comments.find(".ucontents").html($("#ucontents2").val());  
   comments.find("#lytime").text(msg.lytime);  
   $(".contents").prepend(comments);  
   //清空留言窗口内容  
   var message = $("#message").find("#ucontents2");  
   message.val("");  
   }   
   }  
   })  
  }

结果展示

* 
* 

添加留言



数据表变化

## 登录

修改样式

添加清空文本函数 clearText() ，绑定在“取消”按钮上。



登录页面

## 回复

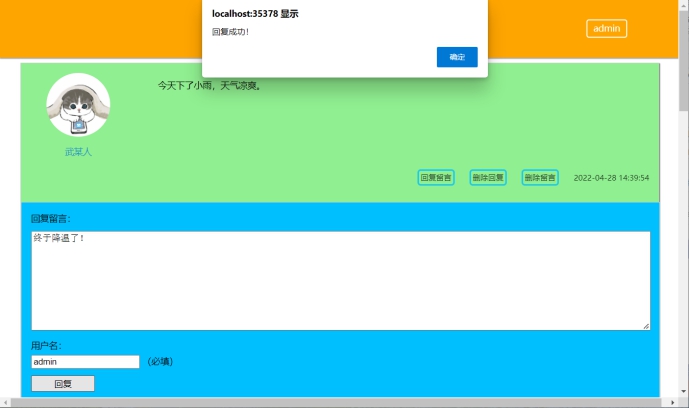
存放新回复到数据库中

* + handleHf.ashx 修改 SQL 语句等

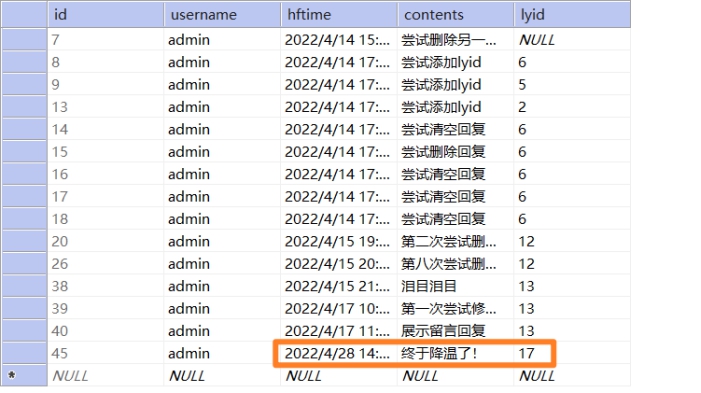
将回复放在 reply-container 块中，方便删除。回复成功后清空回复窗口。

* + manager.js
* function sendHf(msg) {  
   $.ajax({  
   type: "POST",  
   url: "handleHf.ashx",  
   data: msg,   
   success: function (res, status, xhr) {  
   if (res = "success") {  
   alert("回复成功！");  
   //显示回复  
   $("#hf").parent().find(".ucontents").append("<div class='reply-container'><p style='font-size:12px;opacity:0.8;'>"  
   + msg.hfContents + "</p>" + "<label style='font-size:12px;opacity:0.5;'>" + msg.hftime + "</label></div>");  
   //回复存值  
   $("#hf").parent().find(".reply-container").data("lyid", msg.lyid);  
   //清空回复  
   $("#hf").find("#hfContents").val("");  
   //隐藏回复区  
   $("#hf").css("display", "none");  
   }  
   }  
   })  
  }

结果展示

* 
* 

回复留言



数据库变化

## 删除留言

改变点击函数，传递 this 参数。

* + Manager.aspx
  + <label class="rp" onclick="delReplyHandle(this)">删除回复</label>

类似 sendHf.ashx 用 ajax 传递信息给函数，打开数据库连接处理数据。

manager.js

* + //删除留言  
    function delLyHandle(e) {  
     if (confirm("确定要删除留言吗?") == true) {  
     //找到对应的元素  
     const id = $(e).parent().parent().parent().find(".left").find(".h").html();  
     deleteMsg({ ID:id }, e)  
     }  
    }  
    //删除留言  
    function deleteMsg(msgId, e) {  
     $.ajax({  
     type: "POST",  
     url: "deleteMsg.ashx",  
     data: msgId,  
     success: function (res, status, xhr) {  
     if (res = "success") {  
     $(e).parent().parent().parent().remove();  
     alert("删除成功! ");  
     }  
     }  
     })  
    }

deleteMsg.ashx

* + public void ProcessRequest(HttpContext context)  
    {  
     if (context.Request["ID"] != null)  
     {  
     // 假如ID不为空  
     string delID = context.Request["ID"];  
     DBHelper dh = new DBHelper();  
     // 定义SQL语句  
     string sql = "DELETE FROM ly WHERE id = " + delID;  
     // 执行SQL语句  
     int res = dh.ExeCurd(sql);  
     if (res > 0)  
     {  
     context.Response.Write("success");  
     }  
     }  
     else  
     {  
     context.Response.Write("error");  
     }  
      
    }

结果展示

* 
* 

删除“花落了”留言

## 删除回复

改变点击函数，传递 this 参数。

* + Manager.aspx
  + <label class="rp" onclick="delLyHandle(this)">删除留言</label>

类似 sendHf.ashx 用 ajax 传递信息给函数，打开数据库连接处理数据。

manager.js

* + function delReplyHandle(e) {  
     if (confirm("确定要删除所有回复吗? ") == true) {  
     //找到对应的元素  
     var lyid = $(e).parent().parent().parent().find(".reply-container").data("lyid");  
     deleteHf({ lyid }, e);  
     }  
    }  
    //删除回复  
    function deleteHf(lyid, e) {  
     $.ajax({  
     type: "POST",  
     url: "deleteHf.ashx",  
     data: lyid,  
     success: function (res, status, xhr) {  
     if (res = "success") {  
     $(e).parent().parent().parent().find(".reply-container").text("");  
     alert("删除成功! ");  
     }  
     }  
     })  
    }

deleteHf.ashx

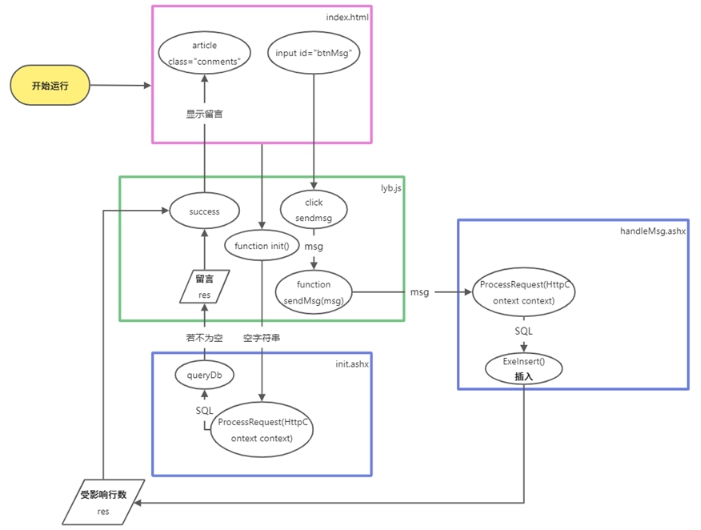
* + public void ProcessRequest(HttpContext context)  
    {  
     if (context.Request["lyid"] != null)  
     {  
     // 假如ID不为空  
     string delID = context.Request["lyid"];  
     DBHelper dh = new DBHelper();  
     // 定义SQL语句  
     string sql = "DELETE FROM hf WHERE lyid = " + delID;  
     // 执行SQL语句  
     int res = dh.ExeCurd(sql);  
     if (res > 0)  
     {  
     context.Response.Write("success");  
     }  
     }  
     else  
     {  
     context.Response.Write("error");  
     }  
      
    }

结果展示

* 
* 

删除回复

# 实验小结

以 index.html 加载函数和留言函数为例分析程序前后端交互过程：  


前后端交互过程

## 当运行程序时，会发生什么？

首先加载 index.html ，然后 index.js 开始执行 init() 函数，即初始化。把 XMLHttpRequest 对象（这里是空字符串）发送到 init.ashx 。

然后开始处理请求，创建 DBHelper 对象，其中构造函数为空。调用 queryDb() 方法，返回一个 SqlDataReader 对象。如果受影响行不为0，即表中有数据，则逐行把数据表读取到一个 message 类的对象 tmp 中，返回一个 res 变量。

最后回到init() 函数，执行回调函数 success ，把已有的留言添加到页面上。

## 如果点击登录，会发生什么？

跳转到 login.aspx ，按下登录按钮就会执行 loadButton() 函数，创建 DBHelper 对象执行查询。  
若用户名和密码正确，会把用户名存放在 Session 中，最后用 Response 重定向到 Manager.aspx 页面。

**上机实验4 基于OpenLayers的WebGIS功能开发**

**1.上机目的**

了解并掌握将地理空间数据发布为数据服务的方法，掌握开源WebGIS开发技术。学会使用GeoServer进行数据服务发布，使用OpenLayers进行空间查询等WebGIS功能开发。

**2.上机软件环境**

OS：Win10等；

开发环境：WebStorm、VSCode等；

应用软件：OpenLayers、Geoserver；

**3.上机内容**

（1）使用GeoServer将矢量数据发布为WMS/WFS图层；

（2）进行图层之间的叠加显示，进行图层显示控制；

（3）通过Openlayers调用发布的地图/数据服务，实现空间查询（包括地图查属性、属性查地图）功能。

**4.上机要求**

要求独立完成上机实验，报告撰写格式规范。人机交互顺畅，界面友好。报告内容包括：上机实验目的、软件环境、上机要求、关键代码与步骤、上机小结。

**5.关键代码与步骤**

# 发布服务

安装好 Java 和 GeoServer 之后将本地的 shapefile 文件发布到 GeoServer，结果如图：



发布服务

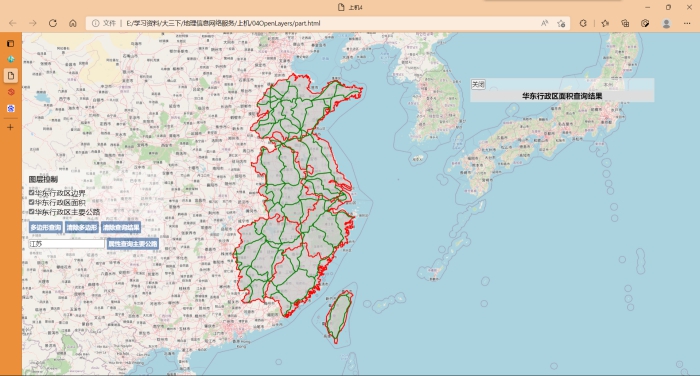
# 添加底图和访问服务

设置界面样式并且添加 OSM 底图，以及三个矢量图层（华东行政区面要素，华东行政区边界线要素，华东主要道路线要素）。使用网络墨卡托投影（EPSG:3857）。

部分代码如下：

// 添加底图   
var map = new ol.Map({   
 layers: [new ol.layer.Tile({   
 source: new ol.source.OSM()   
 })],   
 target: 'map',   
 view: new ol.View({   
 center: [117.145, 34.215],   
 zoom:3,   
 projection: 'EPSG:3857'   
 })   
});   
   
// 添加三个矢量图层   
// 行政区图层   
var area = new ol.layer.Vector({   
 source: new ol.source.Vector({   
 format: new ol.format.GeoJSON(),   
 url: 'http://localhost:9080/geoserver/wfs?service=wfs&version=1.1.0&request=GetFeature&typeNames=area&outputFormat=application/json&srsname=EPSG:3857'   
 }),   
 style: function (feature, resolution) {   
 return new ol.style.Style({   
 fill: new ol.style.Fill({   
 color: 'rgba(200, 200, 200, 0.7)',   
 })   
 });   
 }   
});   
map.addLayer(area);

结果如图：



底图

# 功能实现

## 图层控制

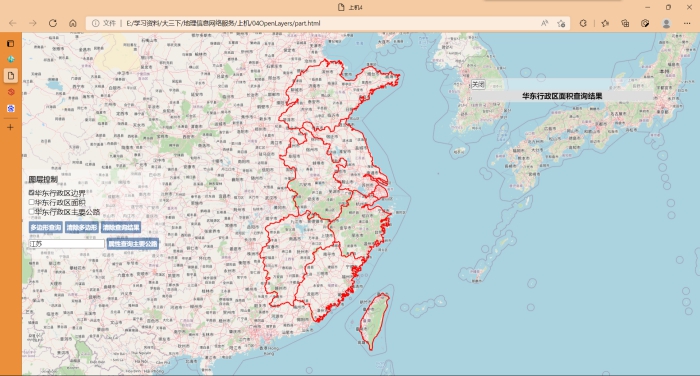
为复选框添加 id 方便图层控制。

css 样式如下：

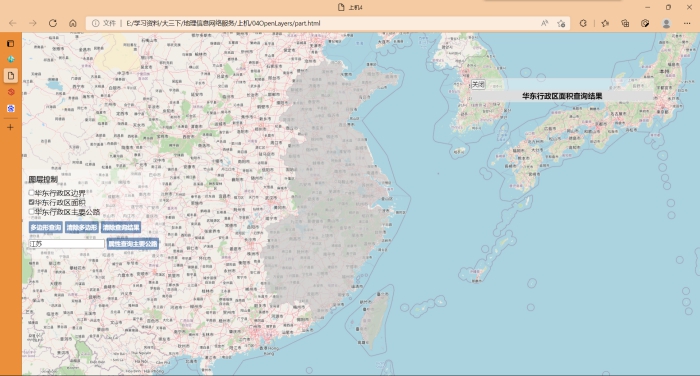
<div>   
 <input type="checkbox" class="layer-checkbox" id="layer-0" checked="checked">边界   
 <br>   
 <input type="checkbox" class="layer-checkbox" id="layer-1" checked="checked">面积   
 <br>   
 <input type="checkbox" class="layer-checkbox" id="layer-2" checked="checked">主要公路   
</div>

代码如下：

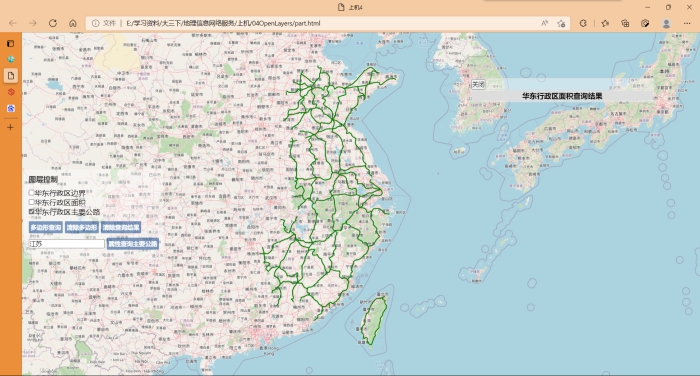
// 图层控制   
document.querySelectorAll(".layer-checkbox").forEach(item => {   
 item.addEventListener("click", e => {   
 const { id, checked } = e.target;   
 switch (id) {   
 case "layer-0":   
 boundary.set("visible", checked);   
 break;   
 case "layer-1":   
 area.set("visible", checked);   
 break;   
 case "layer-2":   
 road.set("visible", checked);   
 break;   
 }   
 });   
});



华东行政区边界图层



华东行政区面积图层



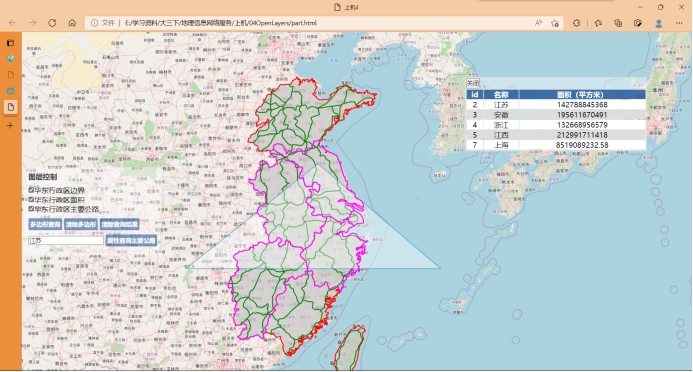
华东行政区主要道路图层

## 空间查询

对于华东行政区面积图层 area 进行多边形查询，并显示返回的结果，代码如下：

// 空间查询   
function spatialQuery() {   
 var featureRequest = new ol.format.WFS().writeGetFeature({   
 srsName: "EPSG:3857", // 坐标系   
 featureNS: "http://my.webgis.class/geoserver", // 命名空间   
 featurePrefix: "gislearn", // 工作区   
 featureTypes: ["area"], // 图层名   
 outputFormat: "application/json",   
 filter: ol.format.filter.intersects("the\_geom", feature.getGeometry(), "EPSG:3857")// 选择   
 });   
   
 fetch("http://localhost:9080/geoserver/wfs", {   
 method: "POST",   
 body: new XMLSerializer().serializeToString(featureRequest)   
 })   
 .then(function (response) {   
 // console.log(response, response.json());   
 return response.json();   
 })   
 .then(function (json) {   
 showResultList(json);   
 var features = new ol.format.GeoJSON().readFeatures(json);   
 if (features.length == 0) {   
 alert("没有数据");   
 } else {   
 spaceSource.clear();   
 spaceSource.addFeatures(features);   
 }   
 });   
}   
   
// 展示空间查询结果列表   
function showResultList(list) {   
 const { features } = list;   
 const propertyList = features.map(item => ({   
 id: item.properties.OBJECTID,   
 name: item.properties.NAME,   
 area: item.properties.Shape\_Area   
 }));   
 const tableDOM = document.querySelector(".result-container table");   
 tableDOM.innerHTML = `   
 <tr class="head">   
 <th>id</th>   
 <th>名称</th>   
 <th>面积（平方米）</th>   
 </tr>   
 `;   
 document.querySelector(".result-container").style.display = "block";   
 propertyList.forEach(item => {   
 const trDOM = document.createElement("tr");   
 trDOM.innerHTML = `   
 <tr>   
 <td>${item.id}</td>   
 <td>${item.name}</td>   
 <td>${item.area}</td>   
 </tr>   
 `;   
 tableDOM.appendChild(trDOM);   
 });   
}

结果如下：



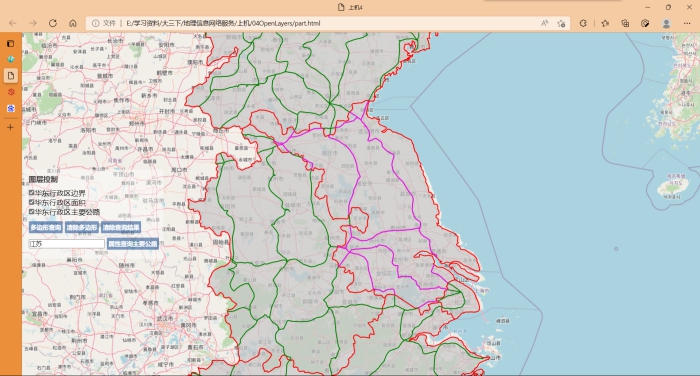
空间查询结果示例

## 属性查询

对于华东行政区主要道路图层 road 进行属性查询，默认查询江苏省内，代码如下：

// 属性查询   
function propertyQuery(queryStr) {   
 var featureRequest = new ol.format.WFS().writeGetFeature({   
 srsName: "EPSG:3857", // 坐标系   
 featureNS: "http://my.webgis.class/geoserver", // 命名空间   
 featurePrefix: "gislearn", // 工作区   
 featureTypes: ["road"], // 图层名   
 outputFormat: "application/json",   
 filter: ol.format.filter.like("NAME", `\*${queryStr}\*`)// 选择   
 });   
   
 fetch("http://localhost:9080/geoserver/wfs", {   
 method: "POST",   
 body: new XMLSerializer().serializeToString(featureRequest)   
 })   
 .then(function (response) {   
 return response.json();   
 })   
 .then(function (json) {   
 var features = new ol.format.GeoJSON().readFeatures(json);   
 spaceSource.clear();   
 spaceSource.addFeatures(features);   
 map.getView().fit(spaceSource.getExtent());   
 });   
}   
document.querySelector(".property-query").addEventListener("click", () => {   
 const str = document.querySelector("#property-inp").value;   
 propertyQuery(str);   
});

结果如下：



空间查询结果示例

**6.小结**

这次实验主要是了解 OpenLayers 体系结构和基本原理，熟悉 Map、Layer、Source、View 四个核心类及其方法函数。用华东行政区数据实现了空间查询和属性查询，不足之处是不够灵活，只能查询某一特定图层上满足要求的要素。