《网络地理信息系统》实习报告

——POI 在线管理系统

班　　级：114163

学　　号：20161001584

姓　　名：温梦甜

指导老师：郭明强

2019 年 01 月 10 日

目录

一、系统背景

1.1 地理信息系统——GIS

地理信息系统（Geographic Information System，GIS）是种特定的十分重要的空间信息系统，它是在计算机软硬件系统的支持下，以采集、存储、管理、检索、分析和描述空间物体的定位分布及与之相关的属性数据，并以回答用户的问题为主要任务的计算机系统。

任何事物都有时空属性，都与空间信息相关，地理信息无处不在。GIS 以空间数据为基础，航天、地面、地表、地下等都是 GIS 所研究的领域。我们在工作和生活中所接触到的各种地图制图工具及在线地图产品，如百度地图、Google 地图等，仅仅是 GIS 的冰山一角。其实，GIS 从最初的地图制图发展至今，已广泛应用于国土、气象、矿产、农林、市政等专业领域，以处理和分析各个行业面临的业务问题并辅助其进行决策。

GIS 究竟能做什么呢?

以大众应用为例。GIS 首先就是用于收集地理信息，人类的绝大部分活动都与地理位置有关，比如想和朋友找个餐厅吃饭，周末想找个电影院看电影，到一个陌生的城市出差找宾馆……这些都涉及地理信息。通过 GIS 能有效地把这些信息都存储起来，怎么存呢?用 Excel 吗?那怎么和地图关联起来?文本文件或数据库都可以，我们只要按要求把信息录入就可以了。收集到的地理信息，在电脑中只是一堆表格数据，那怎么为人们所看到呢?这就是所谓的“可视化”了，各种图表是信息可视化的产物，那地理信息可视化的产物就是“地图”，当然地图远比 GIS 出现得要早，这位前辈是 GIS 重要的代言人，GIS 可以方便地将收集到的信息展示在地图上。而空间分折其实离我们也并不遥远，像大众点评这样的应用已经相当普及，我们可以很方便地找到周边的餐馆。还有地图导航，通过 GPS 装置收集你的地理位置之后，在地图上找到正确的位置示,再进一步的实现查询、搜索等功能。

上述仅仅是 GIS 在大众应用中的一个缩影。GIS 发展至今，紧跟 IT 相关技术的步伐，从单机桌面工具到互联网 Web 在线应用，再到移动端便携应用；在各类应用需求的驱动下，从简单的制图到二维 GIS 应用，从 2.5D 到 3D 的进步，甚至全空间真三维的突破……GIS 在短短几十年中迅速发展、蜕变。GIS 应用渗透到各行各业，分别在横向与纵向逐步扩大应用的广度和深度，成为我们创建智慧城市和智慧地球的中坚力量。

1.2 网络地理信息系统——WebGIS

网络地理信息系统（WebGIS）是指基于 Internet 平台，客户端应用软件采用网络协议，运用在 Internet 上的地理信息系统，即将 GIS 这门学科所能提供的功能通过互联网展现给用户。顾名思义，WebGIS 就是展现于网络上的 GIS，是 GIS 与 Web 融合的产物。GIS 通过 Web 功能得以扩展，使得 GIS 冲破专业圈子，真正成为大众化的 GIS。如今，网络已成为日常生活中不可或缺的工具，人们可以在网上订餐、购物、查找路线信息、实现定位分析等。地理信息已普及大众，越来越多的人开始使用地理信息服务，享受地理信息所带来的便利与乐趣。

随着技术的不断发展，GIS 经历了单机应用向网络应用发展的过程。从本世纪开始，Internet 进入了爆发式增长阶段，网络的铺设以及网速的提升都有了大幅度增加，这为 WebGIS 的发展提供了坚实的大环境。网络环境中的 GIS 应用从 C/S （Client/Server，客户机/服务器）模式向 Internet 环境下的 B/S （Browser/Server, 浏览器/服务器）模式发展，并逐步成为 GIS 应用的主流。相比 C/S 模式，B/S 模式的 WebGIS 具有部署方便、使用简单、便于推广等天然优势，为地理信息服务的发展奠定了基础。于是，WebGIS 应用需求剧增，基于 B/S 模式的 GIS 系统越来越多地开始提供服务，并且随着 RIA (富客户端) 技术，Ajax (动态网页) 技术的涌现和成熟，WebGIS 具有更好的视觉效果与交互效果，越来越受到广大用户的喜爱。

网络的大发展为人类创造了极大的物质财富和精神财富，手指轻轻一点便可轻易获取各种信息资源。互联网与 GIS 的融合，成为 GIS 应用的催化剂，标志着 GIS 迎来一个新的时代，GIS 真正走向大众化，GIS 应用全面融入人们的工作与生活，并彰显出巨大的活力。WebGIS 激活了 GIS 大众应用的市场，互联网早已敏锐地嗅到了商机，大量资本与外界力量进驻 WebGIS，互联网巨头纷纷跨界布局地图领域，Google 地图、百度地图等服务提供商的大规模扩张便是最好的证明。同时，移动互联网进入爆发增长期，移动互联网成功的关键是为用户提供优质便捷的生活服务，地图则是实现移动端增值服务的最佳入口。因此，当移动互联网遇上无处不在的地理信息位置服务，LBS 应用市场需求旺盛，移动端必将涌现出更多意想不到的特色应用。随着终端定位能力、网络及资费等外部条件的成熟，位置服务可能会在很多应用上成为标配，更有希望基于位置信息维度重新组织互联网上的海量信息，创新地理信息价值。如今，GIS 早已融入人们的日常生活，网络在线地图不再限于导航，人们可以

通过地图快速获取周围的景点、 餐馆信息，甚至能在同一种应用中实现订餐、订房、支付等一站式服务，有了移动互联网的支撑,地图所承载的应用会更加丰富、多元化，WebGIS 的应用将会更加宽泛和深入。

随着新互联网技术的发展，广义网络 GIS 被赋予了更多的内容。我们所讨论的 WebGIS 通常为狭义的网络 GIS，即仅仅是基于 B/S 模式，并通过 Web 浏览器访问的 WebGIS。WebGIS 的应用非常广泛，可以应用到几乎所有的领域，主要分为行业应用与大众应用。行业应用通常为传统专业领域的应用，如地矿、国土、公安、市政、应急防灾等领域；大众应用则主要为互联网方向服务于人们日常生活的 GIS 应用，诸如百度地图等网络地图产品（见图 1.1），以及旅游、 餐饮、购物、公交出行等各类 WebGIS 应用系统。而公众接触最多的也就是这些大众应用类产品，只是很多时候我们并不清楚这些就是 WebGIS 应用而已。

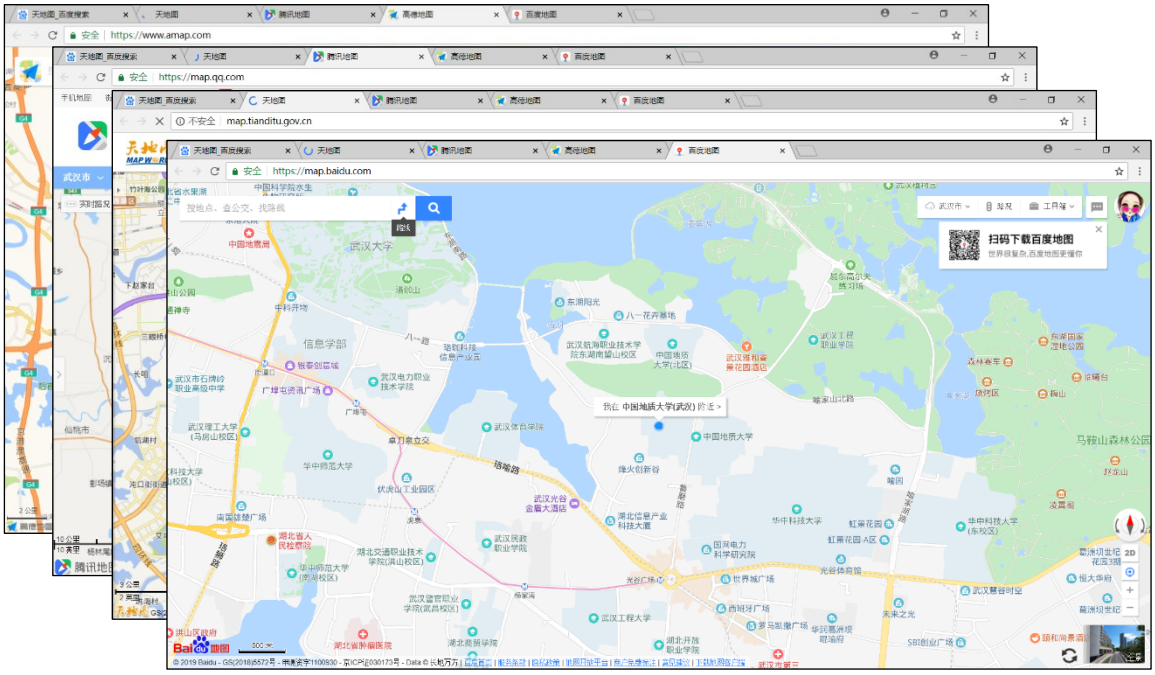


图 1.1 百度地图等在线地图

得益于互联网的发展，WebGIS 快速发展前进，开发工具与平台也呈现出百花齐放之势。目前，涌现出大量用于 WebGIS 二次开发的产品，主要包括开放 API、开源与商业 WebGIS 开发平台等。互联网方向，有百度地图 API、天地图 API、高德地图 API、腾讯地图 API，Google 地图 API 等；行业应用方向，有很多诸如 GeoServer 的开源 WebGIS 平台，还有中地数码、超图，Esri 等 GIS 厂商提供的专业 WebGIS 开发平台产品，如 MapGIS IGServer，ArcGIS forServer 等相关产品。

1.3 常见的开源 GIS 平台

WebGIS 市场需求旺盛，更多的人开始关注 WebGIS 应用，很多开发者投身于 GISer 大军，催生了众多开源 GIS 项目，推动 WebGIS 的普及，如表 1.1 所示。

表 1.1 部分开源 GIS 项目列表

类型	开源 GIS 项目	说明
桌面工具	QGIS、uDig、GRASS	主要用于制图、即在桌面端加载数据并对数据进行编辑
服务器	GeoServer、MapServer、Geodjango	GeoServer 基于 J2EE 框架，MapServer 核心部分基于 C 语言
数据库	PostGIS/PostgreSQL、MySQL Spatial	主要用于存储空间数据
客户端	QGIS、OpenLayers、OpenScales、Worldkit	作为客户端开发框架
工具集	JTS、GEOS(几何拓扑操作库)、Shapely、GDAL /OGR(栅格矢量数据操作库)、Proj4（地图投影库）	
中间件	GeoTools、MapTools	GeoTools 是一款基于 Java 的开源 GIS 工具集，允许用户对地理数据进行基本操作，空间分析功能一般基于中间件实现，或基于 OGC WPS 实现

1. uDig

uDig 是一个开源的桌面应用程序框架（见图 1.2），是构建在 Eclipse RCP 和 GeoTools（一个开源的 Java GIS 工具包）上的桌面 GIS。uDig 作为一款开源桌面 GIS 软件，基于 Java 和 Eclipse 平台，可以进行 shp 格式地图文件的编辑和查看；是一个开源空间数据查看器与编辑器，对 OpenGIS 标准、WebGIS、网络地图服务器和网络功能服务器有特别的加强。

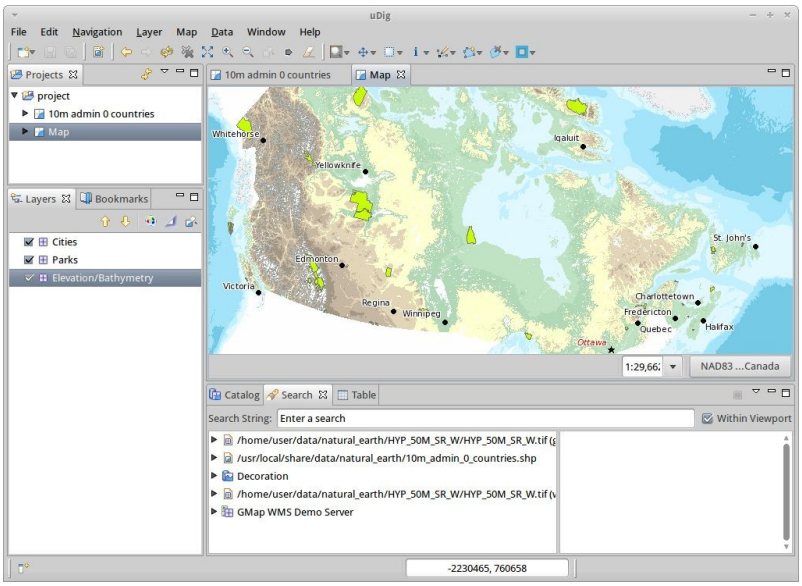


图 1.2 uDig 界面

2. QGIS

QGIS 是一个用户界面友好的桌面 GIS，可以运行在 Linux、UNIX、Mac OSX 和 Windows 等平台之上。QGIS 是基于 Qt（跨平台的图形工具软件包），使用 C++语言开发的一个用户界面友好、跨平台的开源版桌面地理信息系统，如图 1.3 所示。

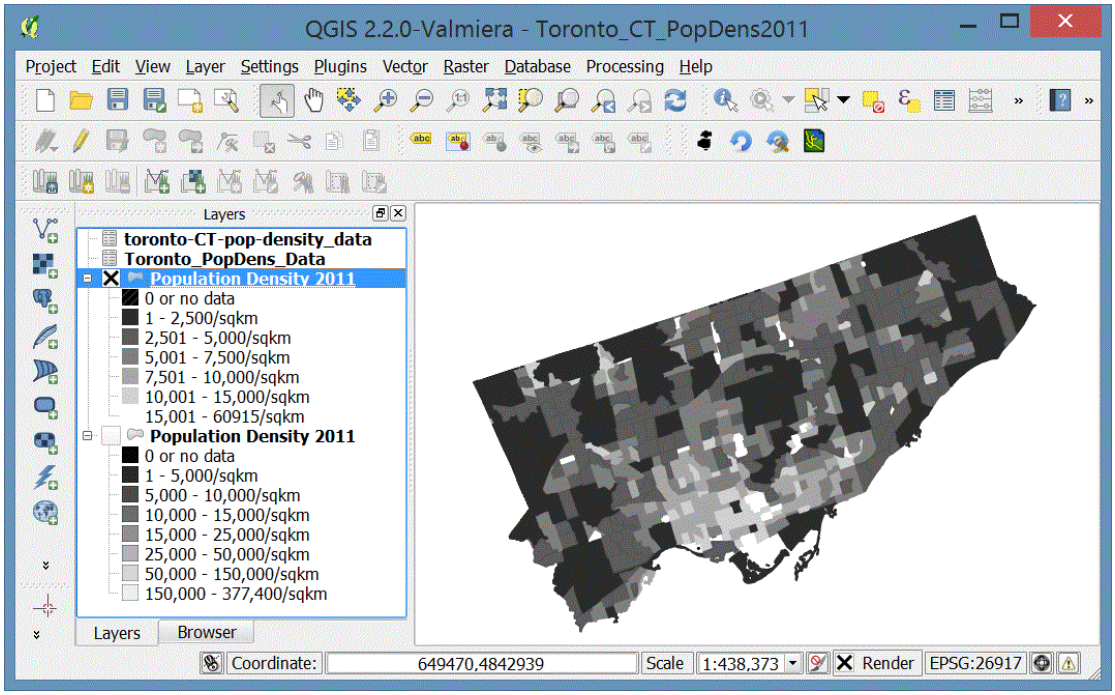


图 1.3 QGIS 界面

3. GeoServer

GeoServer 是 OpenGIS Web 服务器规范的 J2EE 实现，利用 GeoServer 可以方便地发布地图数据，允许用户对数据进行更新、删除、插入操作，通过 GeoServer 可以比较容易地在用户之间迅速共享地理信息。GeoServer 是社区开源项目，可以直接通过社区网站（如中文社区网站 http://www.opengeo.cn/）下载相关资料，如图 1.4 所示。

GeoServer 支持 OGC 标准规范的系列服务，支持 PostgreSQL、MySQL 等数据库，以及 ArcSDE、ShapeFile 等中间件和文件资源，能够将网络地图输出为 JPEG、PNG、KML 等多种图片和数据格，可以运行在任何基于 J2EE/Servlet 的容器之上，支持多种客户端框架，如 Openlayers 等。

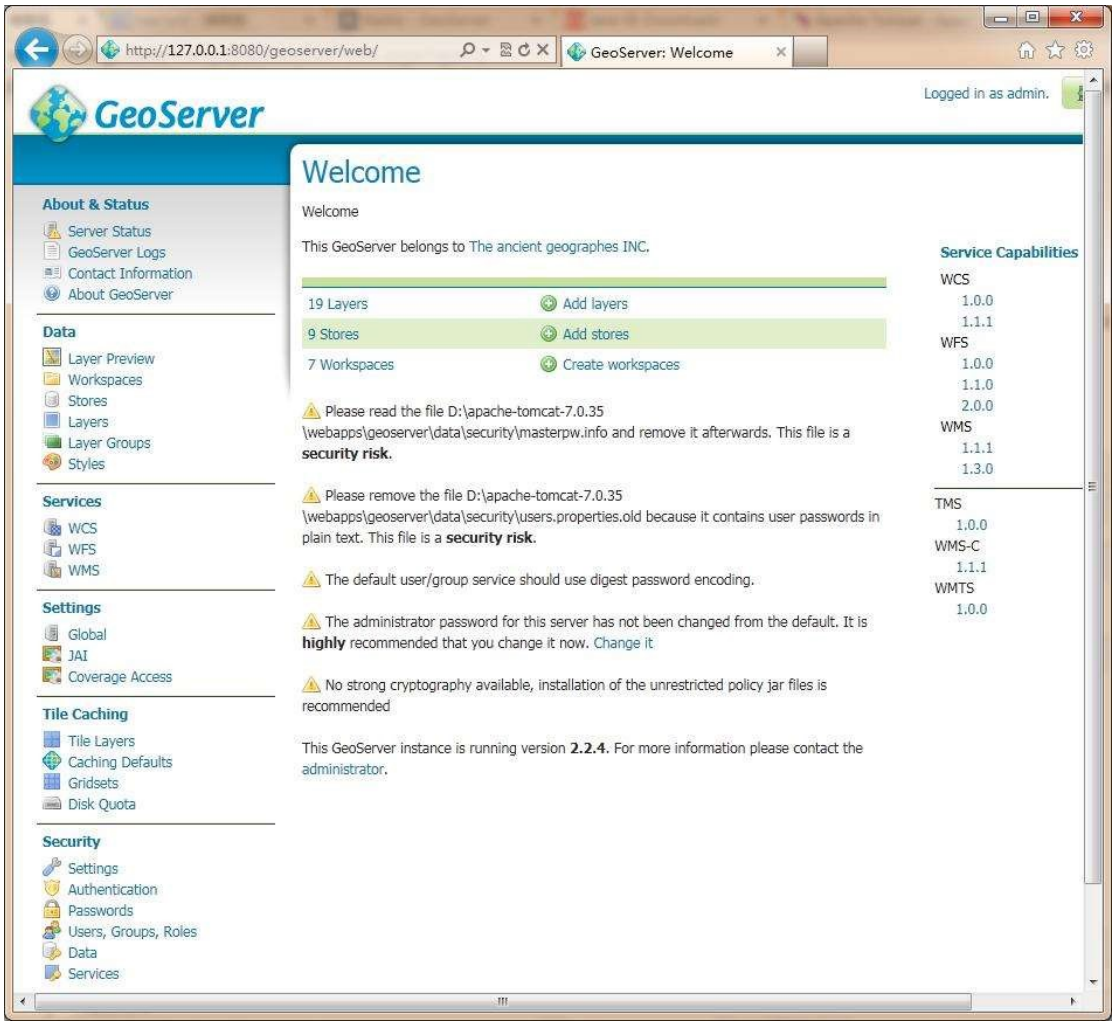


图 1.4　GeoServer 界面

4. MapServer

MapServer 是美国明尼苏达大学在 20 世纪 90 年代利用 C 语言开发的开源 WebGIS 项目。MapServer 是一套基于胖服务器端/瘦客户端模式的实时地图发布系统，客户端发送数据请求时，服务器端实时地处理空间数据，并将生成的数据发送给客户端 MapServer 时核心部分是 C 语言编写的地图操作模块，它的许多功能都依赖一些开源或免费的库。MapServer 遵循 OGC 系列规范，可以集成 PostGIS 和开源数据库 PostgreSQL，并对地理空间数括进行存储和 SQL 查询操作，同时还支持其他客户端 API 实现地理空间数据的传输与表达。

5. OpenLayers

OpenLayers 是一个专为 WebGIS 客户端开发提供的 JavaScript 类库包，用于实现地图数据的网络访问。它访问地理空间数据的方法都符合行业标准，支持各种公开的和私有的数据

标准和资源。OpenLayers 采用纯面向对象的 JavaScript 方式开发，同时借用了 Prototype 框架和 Rico 库的一些组件。OpenLayers 是一个开源的项目，其设计之意是为互联网客户端提供强大的地图展示功能，包括地图数据显示与相关操作，具有灵活的扩展机制。目前，OpenLayers 已经成为一个拥有众多开发者和帮助社区的成熟、流行的框架。目前 OpenLayers 2 已经升级为 OpenLayers 3，可以从其官方网站 (http://openLayers.org/)下载相关资源，如图 1.5 所示。

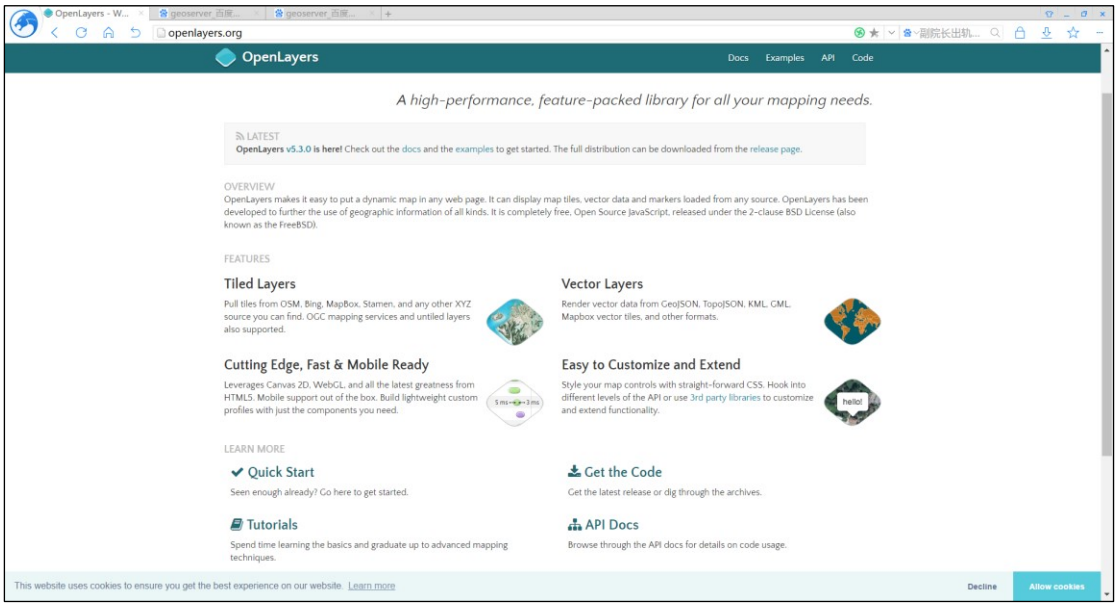


图 1.5 OpenLayers 官网

6. OpenScales

OpenScales 是一个基于 Flex 的优秀的前台地图框架，可以用来开发各种网络版、手机版和桌面版地图程序。

OpenScales 是基于 ActionScript 3 和 Flex 编写的,能够支持各种标准的地图服务，如 WMS、WFS、WMTS、OSM 等。它是开源的、免费的客户端开发框架，基于 LGPL 开源协议，它在 FlashPlayer 中运行，可以在各个浏览器中使用，具有很好的跨平台特性。OpenScales 作为一个开源的 GIS 客户端框架，具有非常大的应用潜力，可以从其官方网站 (http://www.openscales.org/)下载相关资源。OpenScales 可以看成对 OpenLayers 的 ActionScript 翻译，所以在学习 OpenScales 时可以适当参考 OpenLayers 的官方教程。相比之下，虽然 OpenLayers 的教程也是英文的，不过非常详细，提供的示例也远比 OpenScales 的官方教程丰富。

1.4 兴趣点——POI

POI 是“Point of Interest”的缩写，中文可以翻译为“兴趣点”。兴趣点（POI）是地理信息系统中的一个术语，泛指一切可以抽象为点的地理对象，尤其是一些与人们生活密切相关的地理实体，如学校、银行、餐馆、加油站、医院、超市等。兴趣点的主要用途是对事物或事件的地址进行描述，能在很大程度上增强对事物或事件位置的描述能力和查询能力，提高地理定位的精度和速度。

每个 POI 包含四方面信息，名称、类别、坐标、分类，全面的 POI 讯息是丰富导航地图的必备资讯，及时的 POI 兴趣点能提醒用户路况的分支及周边建筑的详尽信息，也能方便导航中查到你所需要的各个地方，选择最为便捷和通畅的道路来进行路径规划，因此，导航地图 POI 多少状况直接影响到导航的好用程度。

二、需求分析

兴趣点数据的准确性和实时性，对于 LBS 的可用性至关重要。由于城市建设快速发展，导致兴趣点也随着地形地貌、业务单位规划的变更而相应地变化，这就要求兴趣点数据能得到不断的丰富和更新。据不完全统计，每年兴趣点的变更量约占兴趣点总量的 20%左右，由此必须建立高效可行的兴趣点动态管理和定期维护更新机制，以满足城市管理的需要。

兴趣点数据的更新一直以来就被认为是一项耗费大、周期长的工作，易于变化也是兴趣点数据的一大特点，这给兴趣点数据的更新带来了很大的困难，传统的周期性更新方式已不能适用于对兴趣点数据的更新。

Google 和 Go2Map 采用的兴趣点数据更新方法是一种较新的模式。Google 和 Go2Map 都推出了各自的地图社区服务。地图社区服务主要通过一种开放式论坛的形式提供。用户可以对自己所知道的位置进行标注，发布到服务器上，与其他用户进行共享。这种方法将用户这个庞大的群体纳入到了数据生产者行列，为兴趣点数据更新提供了新的途径和方法。

针对 POI 的数据管理需求，本系统主要包括以下功能模块：

（1）POI 查询：可以通过类别、名称进行 POI 的查询，可以直观掌握目前

数据库中 POI 数据，便捷地获取想要的 POI 信息，如坐标、名称、类别、详细信息等；

（2）地图显示：将查询到的 POI 在地图上进行标注，帮助用户快速定位，支持放大、缩小、拖拽、跳转等功能；

（3）POI 管理：用户可以根据实际情况更新 POI 数据，可以进行添加、删除、更新操作，以保证 POI 的准确性。

三、系统架构设计

POI 在线管理系统的整体架构如图 3.1 所示。

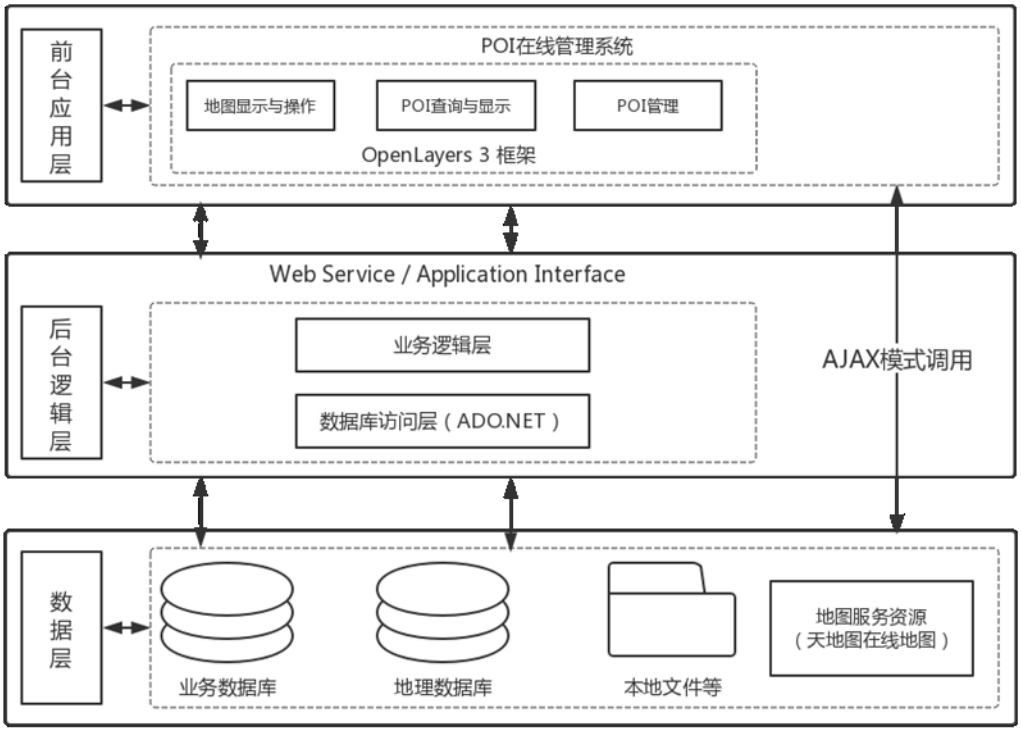


图 3.1　系统体系架构图

该系统以公共地图数据服务，天地图 POI 的业务数据为基础，客户端使用 OpenLayers3 框架，后台结合 ASP.NET 体系框架实现，构建一个涵盖地图放大、缩小、拖拽、复位等基础功能以及 POI 查询和管理等功能的 WebGIS 系统。

四、功能设计

POI 在线管理系统提供的 POI 信息可以方便用户的查询、定位，使用户能够掌握 POI 的

位置信息，实现快速定位；除此之外，在线 POI 的更新、删除等功能能够实现多用户协作，更新 POI 数据。其具体的功能模块如图 4.1 所示。

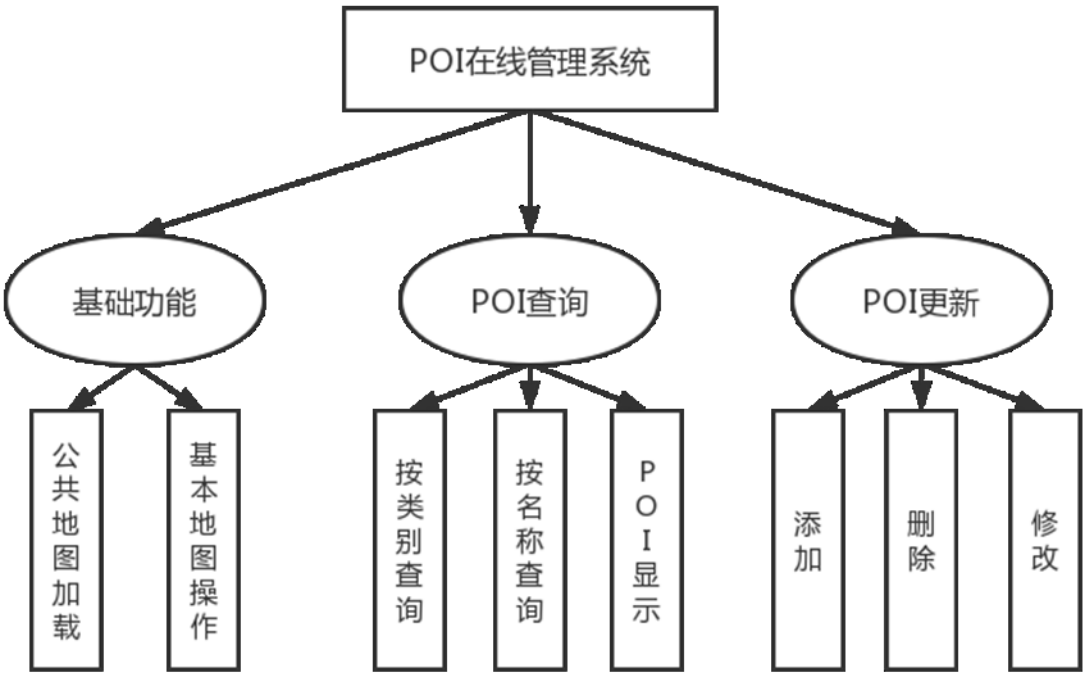


图 4.1　系统功能模块

4.1 基础功能模块

1. 公共地图加载

本系统选择天地图在线地图，默认加载天地图的瓦片地图图层和天地图注记图层，以帮助用户实现初步定位。在此基础上，以天地图的瓦片地图和注记图层为底图，动态叠加查询到的 POI 矢量图层。

2. 基本底图操作

加载公共地图之后，实现对地图的基本操作功能，比如地图的缩放控制等。本系统通过加载地图导航控件实现对地图的基本控制功能。

4.2 POI 查询模块

本模块实现 POI 数据的查询与显示，通过列表展示查询到的 POI 记录，同时将其展示在下方的公共地图上，实现多种形式的查询和显示，让用户的查询更加高效。

1. 按类别查询

在系统的主页面上,通过下拉框展示当前数据库中所有的POI类型以供用户进行选择，点击查询即可展示当前查询类型的POI数据。

2. 按名称查询

在系统的主页面上，通过文本框可以供用户输入 POI 名称，点击查询即可展示当前类型和名称的POI数据，

3. POI显示

在系统的主页面上，点击查询之后，一方面通过数据列表来展示POI的坐标、名称、类别、备注等信息；另一方面将查询到的POI在地图上标示出来，便于用户快速精准定位。

4.3 POI更新模块

1. 添加

在系统的主页面上，点击添加按钮，即可在地图上找到需要添加的POI的位置，单击此位置即可弹出信息输入框，便于用户输入该POI的详细信息，最终点击提交按钮，将其信息添加入数据库中，从而实现POI的添加。

2. 删除

在系统的主页面上查询到需要删除的 POI 点，在列表中点击“删除”按钮即可将此条POI记录删除。

3. 修改

在系统的主页面上查询到需要删除的POI点，在列表中点击“编辑”按钮，列表中的该条记录则会变为可编辑状态，此时用户可以更新该条记录的信息，最后点击“更新”按钮即可实现POI数据的修改。

五、数据库设计

由于本系统只存储POI点数据，因此只需要一张表即可完成POI数据的存储。POI点数据的表如表5.1所示。表名：point；说明：记录POI点的详细信息。

表 5.1　POI 点信息表

列名	说明	数据类型
ID	POI 的序号，主键	int
pnt_x	POI 的 x 坐标（投影坐标）	float
pnt_y	POI 的 y 坐标（投影坐标）	float
pnt_type	POI 的类别	nvarchar(10)
pnt_name	POI 的名称	nvarchar(20)
pnt_attribute	POI 的备注	nvarchar(50)

六、系统实现

POI 在线管理系统基于 OpenLayers3 开发库实现，主要功能包括地图的基础操作、POI 查询显示和 POI 更新等功能。本系统采用 JavaScript 的客户端方式，结合 ASP.NET 开发模式实现。

本系统的开发环境如下：

- 操作系统：Windows 10
- 开发工具：Microsoft Visual Studio 2017
- Web 服务器：Internet 信息服务（IIS）管理器 10.0 版本
- WebGIS API：OpenLayers 3
- 数据库：Microsoft SQL Server 2014
- 浏览器：Internet Explorer，Chrome，Microsoft Edge、猎豹浏览器等。

该系统的客户端使用 jQuery 的 JavaScript 框架，并采用了 HTML5 技术，改进了系统的可用性及用户体验，让客户端的呈现更加炫彩夺目，交互更加友好。系统的后台数据服务采用 ADO.NET 实现数据库交互，通过 Ajax 技术实现客户端与后台的数据交互，使用 JSON 格式进行数据传输。

6.1 系统框架

根据系统的架构设计、功能设计与数据库设计，在集成开发环境（VS2017）中进行系统的具体开发。按照该系统的功能模块划分，采用 HEML、JavaScript（jQuery）等 UI 技术搭建系统主框架。

（1）在 VS2017 中新建一个 ASP.NET 窗体应用程序（WebGISProject），在解决方案资源管理器中 WebGISProject 项目下新建两个文件夹，分别命名为 libs 和 pages，分别用来存放开发库和新建的页面。将 OpenLayers3 的开发库、样式文件以及第三方控件都复制到 libs 文件夹下；在 pages 文件夹下新建 Web 窗体，如图 6.1 所示。

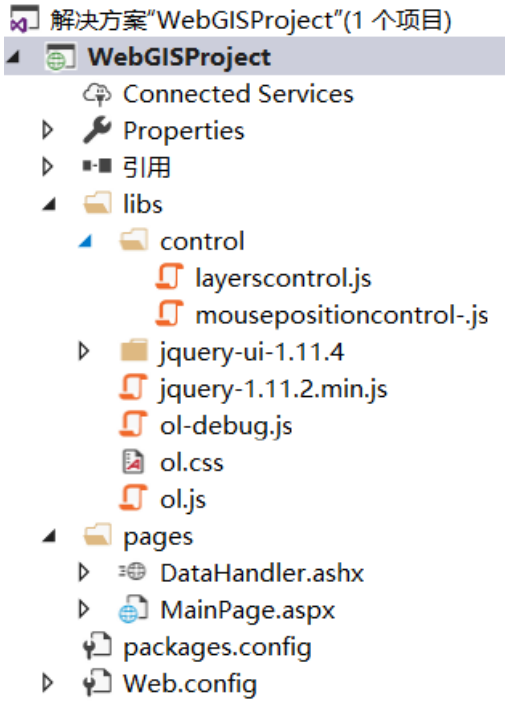


图 6.1 系统目录结构

（2）新建的“MainPage.aspx”是系统的主界面。该页面的搭建效果如图 6.2 所示。上述“MainPage.aspx”页面主要用 form 和 div 层来搭建和实现，其结构如图 6.3 所示。

整个框架使用层叠的 div 层来搭建，地图的 div 层为“mapCon”，用于加载地图；查询功能键、数据源以及数据列表对应的 div 层为“function”，添加 POI 时的“图形属性信息设置”对应的 div 层为“dialog-confirm”。

具体的搭建方法参见 MainPage.aspx 源码，在此不再详细阐述。

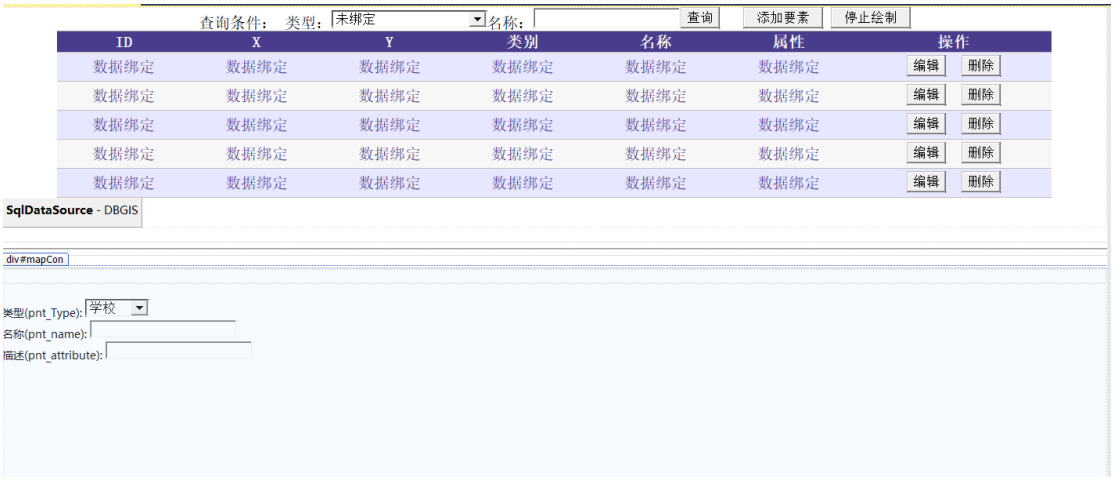


图 6.2 MainPage.aspx 界面效果

```
<body >
    <form id="form1" runat="server">
        <%--查询等功能键、数据源以及数据列表--%>
        <div id="function" class="auto-style1">
            <asp:Label ID="Label1" runat="server" Text="查询条件:        "></asp:Label>
            <asp:Label ID="Label2" runat="server" Text="类型: "></asp:Label>
            <asp:DropDownList ID="ddl_type" runat="server" Height="20px" Width="198px"
                style="z-index:990" AppendDataBoundItems="True" OnLoad="ddl_type_Load">
            </asp:DropDownList>
            <asp:Label ID="Label3" runat="server" Text="名称: "></asp:Label>
            <asp:TextBox ID="txt_name" runat="server" Height="20px"></asp:TextBox>
            <asp:Button ID="btn_Query" runat="server" Text="查询" style="text-align: center" OnClick="btn_Query_Click" />
             
            <input id="Add" type="button" value="添加要素" onclick="AddPoint()"/>
            <input id="StopAdd" type="button" value="停止绘制" onclick="CancelAdd()"/>

            <asp:GridView ID="gvGIS" ...>...</asp:GridView>
            <asp:SqlDataSource ID="DBGIS" ...>...</asp:SqlDataSource>
        </div>
    </form>
    <hr />
    <form>
        <%--地图容器--%>
        <div id="mapCon" style="width: 100%; position:absolute;  display:block" >
            <%--鼠标位置控件--%>
            <div id="mouse-position"></div>
        </div>
    </form>
    <%--图形属性信息设置对话框--%>
    <div id="dialog-confirm">...</div>
</body>
```

图 6.3 MainPage.aspx 页面设计

6.2 数据库查询

本系统中的数据库查询、更新和删除操作通过 ADO.NET 绑定的数据源来实现前后台的数据请求与交互。POI 添加，也即数据的插入操作，前台采用 Ajax 模式请求，即使用 jQuery 的方法发送数据请求，后台则用 DataHandler.ashx 处理前台发送的数据请求，并直接利用 ADO.NET 进行数据插入操作。

（1）在 aspx 页面中添加 GridView 控件，命名为 gvGIS，并配置 SQL 数据源。将 SQL 数据源命名为 DBGIS，为其生成 INSERT、UPDATE 和 DELETE 语句，便于后期各项操作，

如图 6.4 所示。

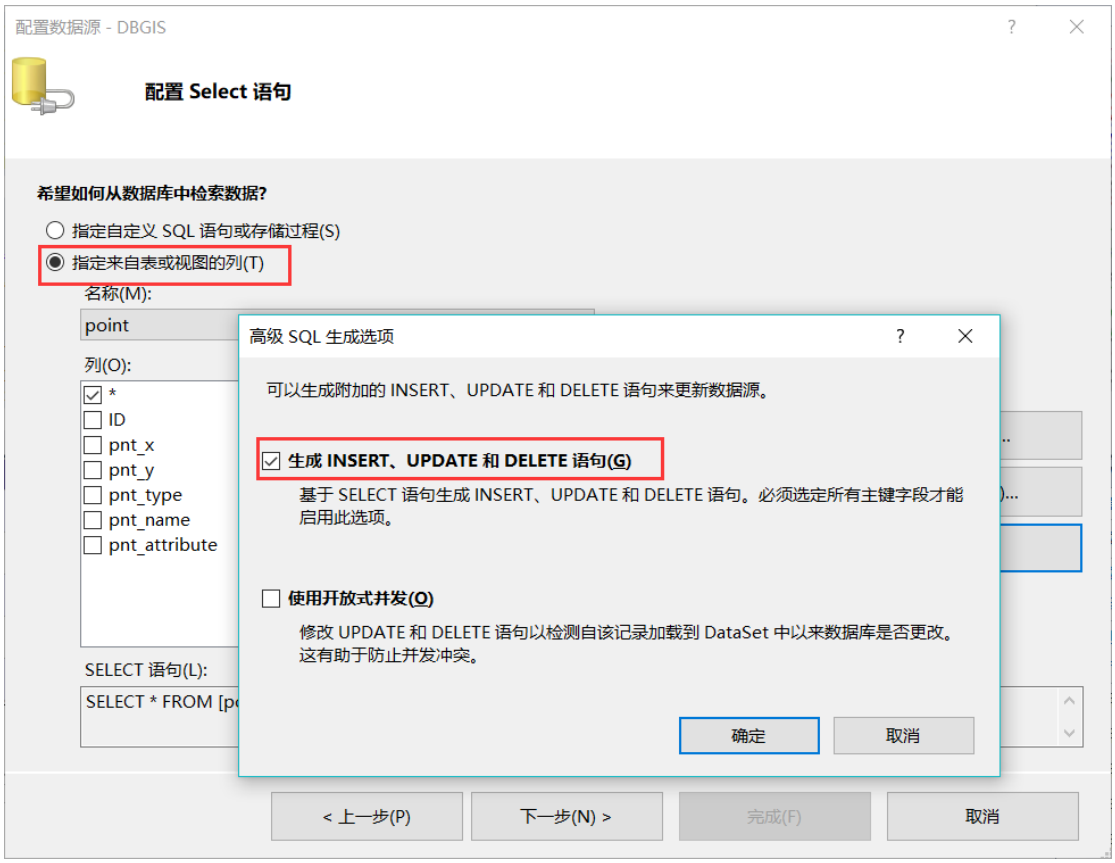


图 6. 4 配置数据源

配置完成之后在 aspx 页面中生成的代码如下所示：

```
<asp:SqlDataSource ID="DBGIS" runat="server"
ConnectionString="<%$ ConnectionStrings:GISConnectionString %>"
DeleteCommand="DELETE FROM [point] WHERE [ID] = @original_ID"
InsertCommand="INSERT INTO [point] ([pnt_x], [pnt_y], [pnt_type], [pnt_name],
[pnt_attribute]) VALUES (@pnt_x, @pnt_y, @pnt_type, @pnt_name, @pnt_attribute)"
SelectCommand="SELECT * FROM [point]"
UpdateCommand="UPDATE [point] SET [pnt_x] = @pnt_x, [pnt_y] = @pnt_y, [pnt_type]
= @pnt_type, [pnt_name] = @pnt_name, [pnt_attribute] = @pnt_attribute WHERE [ID]
= @original_ID" >
            <DeleteParameters>
                <asp:Parameter Name="original_ID" Type="Int32" />
            </DeleteParameters>
            <InsertParameters>
                <asp:Parameter Name="pnt_x" Type="Double" />
                <asp:Parameter Name="pnt_y" Type="Double" />
                <asp:Parameter Name="pnt_type" Type="String" />
                <asp:Parameter Name="pnt_name" Type="String" />
```

```
            <asp:Parameter Name="pnt_attribute" Type="String" />
        </InsertParameters>
        <UpdateParameters>
            <asp:Parameter Name="pnt_x" Type="Double" />
            <asp:Parameter Name="pnt_y" Type="Double" />
            <asp:Parameter Name="pnt_type" Type="String" />
            <asp:Parameter Name="pnt_name" Type="String" />
            <asp:Parameter Name="pnt_attribute" Type="String" />
            <asp:Parameter Name="original_ID" Type="Int32" />
        </UpdateParameters>
    </asp:SqlDataSource>
```

在 Web.config 中生成的连接数据库的语句如下：

```
<connectionStrings>
    <add name="GISConnectionString" connectionString="Data Source=.;Initial
Catalog=GIS;User ID=WMT;Password=wenmengtian"
      providerName="System.Data.SqlClient" />
</connectionStrings>
```

6.3 基本功能

基本功能包括公共地图的加载和地图的基本操作等。本系统选择天地图在线地图，默认加载天地图的瓦片地图图层和天地图注记图层，以帮助用户实现初步定位。在此基础上，以天地图的矢量地图和注记图层为底图，动态叠加查询到的 POI 矢量图层。加载公共地图之后，通过加载地图导航控件实现对地图的基本操作功能，比如地图的缩放控制等。

（1）在 js 代码中将 tileLayer 图层初始化为天地图的瓦片图层，将 markLayer 图层初始化为注记图层，另外再新建一个矢量图层，命名为 vector。初始化代码如下：

```
var tileLayer = new ol.layer.Tile({
        title: "天地图瓦片图层",
        source: new ol.source.XYZ({
            url: "http://t0.tianditu.com/DataServer?T=vec_w&x={x}&y={y}&l={z}",
            wrapX: false
        })
    });
    var markLayer = new ol.layer.Tile({
        title: "注记图层",
        source: new ol.source.XYZ({
```

```
                url: "http://t3.tianditu.com/DataServer?T=cva_w&x={x}&y={y}&l={z}",
                wrapX: false
            })
        });
var vector = new ol.layer.Vector({
            source: new ol.source.Vector(),
            style: new ol.style.Style({
                fill: new ol.style.Fill({
                    color: 'rgba(255, 255, 255, 0.7)'
                }),
                stroke: new ol.style.Stroke({
                    color: '#0099ff',
                    width: 2
                }),
                image: new ol.style.Circle({
                    radius: 7,
                    fill: new ol.style.Fill({
                        color: '#0099ff'
                    })
                })
            })
        });
        map.addLayer(vector);
```

代码说明：此处初始化地图图层是利用了 ol.source.XYZ 接口来加载天地图服务，用 ol.layer.Vector 接口来新建矢量图层。初始化地图之后，还要通过 addLayer 函数将这些图层加载到地图容器中去。

（2）本系统的地图操作功能主要包括缩放控制和鼠标坐标位置的显示。缩放控制为 OpenLayers3 的地图控件中的 collapsible 属性，将其设置为 true 即可，因此在此基础上只添加了鼠标位置控件来显示鼠标所在处的位置坐标。具体代码如下：

```
var mousePositionControl = new ol.control.MousePosition({//实例化鼠标位置控件
            //坐标格式
            coordinateFormat: ol.coordinate.createStringXY(4),
            //地图投影坐标系（若未设置则默认输出投影坐标系下的坐标）
            //projection: 'EPSG:4326',
            //坐标信息显示样式类名，默认是'ol-mouse-position'
            className: 'custom-mouse-position',
            //显示鼠标位置信息的目标容器
            target: document.getElementById('mouse-position'),
            //未定义坐标的标记
```

```
            undefinedHTML: ' '
        });
```

代码说明：此部分代码仅为实例化鼠标位置控件的部分，后续还应该将此控件插入到地图的控件中去。

6.4 POI 查询

POI 查询通过按类别和按名称两种方式进行查询，通过下拉框进行查询，如图 6.5 所示。

图 6.5　POI 查询工具条

首先要绑定下拉框选项，使其在加载的时候能够把数据库中的 pnt_type 的所有值都加载进去；然后实现“查询”按钮的响应事件，判断查询条件，然后更改 SELECT 语句，实现查询。

（1）绑定下拉框选项时，首先连接数据库，查询数据库中 pnt_type 的所有值，将其分别加载到字符串链表 types 中，然后将 types 绑定到下拉框列表。此部分通过 ASP.NET 的后台实现，具体代码如下：

```
//在加载dropdownlist控件时绑定下拉框选项
protected void ddl_type_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                types.Add("全部");
                string connStr = "Data Source=.;Initial Catalog=GIS;User
ID=WMT;Password=wenmengtian";
                SqlConnection conn = new SqlConnection(connStr);
                try
                {
                    if (conn.State == ConnectionState.Open)
                    {
                        conn.Close();
                    }
```

```
                conn.Open();//打开连接

                string selectStr = "select distinct pnt_type from point;";
                SqlCommand selectCmd = new SqlCommand(selectStr, conn);
                SqlDataReader dr = selectCmd.ExecuteReader();
                string ti = "";
                while (dr.Read())
                {
                    ti = dr["pnt_type"].ToString();
                    types.Add(ti);
                }
                ddl_type.DataSource = types;
                ddl_type.DataBind();

                dr.Close();
                conn.Close();
            }
            catch (SqlException)
            {
                Response.Write("<script>alert('初始化错误！');</script>");
            }
        }
    }
```

（2）查询按钮响应事件。由于页面中 GridView 控件 gvGIS 已经绑定了数据源 DBGIS，并且已经生成了 INSERT、SELECT、UPDATE、DELETE 语句，因此在查询的时候，只需要按照查询条件调整 SELECT 语句即可。具体代码如下：

```
//查询按钮响应事件
    protected void btn_Query_Click(object sender, EventArgs e)
    {
        string pnt_type = ddl_type.SelectedItem.Text;;
        string pnt_name = txt_name.Text;

            string selectStr = "";
            if(pnt_name==""||pnt_name=="全部")
            {
                if (pnt_type == "全部")//查询全部类型的全部点
                {
                    selectStr = "select * from point;";
                }
                else           //查询某个类型的全部点
```

```
                {
                    selectStr = "select * from point where pnt_type='" +
pnt_type + "';";
                }
            }else{
                if (pnt_type == "全部")    //查询全部类型中的某个点；
                {
                 selectStr = "select * from point where pnt_name='"+pnt_name+"';";
                }
                else                    //查询某个类型的某个点；
                {
                    selectStr = "select * from point where pnt_type='" +
pnt_type + "' and pnt_name='" + pnt_name + "';";
                }
            }
            this.DBGIS.SelectCommand = selectStr;
        }
```

6.5 POI 显示

POI 显示分为两部分，一部分是将查询到的数据列表中的 POI 记录依次绘制在地图上，另一部分是将新交互添加的 POI 点绘制在地图上。此处只对前者进行说明，后者在 POI 添加的部分会加以说明。

POI 显示首先是要根据列表进行遍历，在新建的矢量图层的数据源上，依次添加点要素。该部分在前端以 js 代码实现，具体代码如下：

```
//添加表格中的点
        function DrawPoints() {
            ////添加矢量图层作为绘制图层
            var vectorSource = new ol.source.Vector({ wrapX: false });
            var vectorLayer = new ol.layer.Vector({
                source: vectorSource,
                style: new ol.style.Style({        //样式
                    fill: new ol.style.Fill({      //填充样式
                        color: 'rgba(255,255,255,0.2)'
                    }),
                    stroke: new ol.style.Stroke({
                        color: '#ffcc33',
                        width: 2
                    }),
```

```
                image: new ol.style.Circle({
                    radius: 7,
                    fill: new ol.style.Fill({
                        color: '#00ffff'
                    })
                })
            })
        });

        var p_x, p_y;
        var point;
        var coordinate;
        var gv = document.getElementById("<%= gvGIS.ClientID%>");
        for (var i = 1; i < gv.rows.length; i++) {
            p_x = gv.rows[i].cells[1].innerText;
            p_y = gv.rows[i].cells[2].innerText;
            coordinate = [parseFloat(p_x), parseFloat(p_y)];
            point = new ol.Feature({
                geometry: new ol.geom.Point(coordinate)
            })
            vectorSource.addFeature(point);//添加要素
        }
        map.addLayer(vectorLayer);
    }
    window.onload = DrawPoints;
```

6.6 POI 添加

POI 添加功能主要是通过 OpenLayers 的交互绘制控件 ol.interaction.Draw 来实现的，它可以实现在地图上通过鼠标进行交互，并获取点坐标。在绘制点之后，还要输入该 POI 的其他属性，并将其传入后台，存储进数据库中。

（1）交互添加 POI 的控件 ol.interaction.Draw 的具体使用代码如下：

```
var draw; //绘制对象
        var coordinates = null;// 当前绘制图形的坐标串
        var currentFeature = null; //当前绘制的几何要素
        var geostr = null;
        /**
        * 根据绘制类型进行交互绘制图形处理
        */
```

```
        //交互添加点
            function AddPoint() {
            draw = new ol.interaction.Draw({
                source: vector.getSource(), //绘制层数据源
                type: 'Point' //几何图形类型
            });
            map.addInteraction(draw);
            //添加绘制结束事件监听，在绘制结束后保存信息到数据库
            draw.on('drawend', drawEndCallBack,this);
        }
```

（2）输入 POI 其他属性的“图形属性信息设置”对话框页面设计如下：

```
<%--图形属性信息设置对话框--%>
    <div id="dialog-confirm" title="图形属性信息设置">
        <label>类型(pnt_Type):</label>
        <select id="pnt_Type" name="D1">
            <option value="学校" selected="selected" >学校</option>
            <option value="医院" >医院</option>
            <option value="景点" >景点</option>
            <option value="火车站" >火车站</option>
            <option value="其他" >其他</option>
        </select>
        <br />
        <label>名称(pnt_name):</label>
        <input type="text" value="" id="pnt_name" />
        <br />
        <label>描述(pnt_attribute):</label>
        <input type="text" value="" id="pnt_attribute" />
    </div>
```

对话框初始状态为不可见，当交互添加点之后才会通过 drawEndCallBack 函数将其展示在界面上，如图 6.6 所示。具体的初始化代码如下：

图 6.6　图形属性信息设置对话框

```
// 初始化信息设置对话框
      $("#dialog-confirm").dialog(
          {
              modal: true,  // 创建模式对话框
              autoOpen: false, //默认隐藏对话框
              //对话框打开时默认设置
              open: function (event, ui) {
                  $(".ui-dialog-titlebar-close", $(this).parent()).hide(); //隐藏
默认的关闭按钮
              },
              //对话框功能按钮
              buttons: {
                  "提交": function () {
                      submitData(); //提交几何与属性信息到后台处理
                      $(this).dialog('close'); //关闭对话框
                  },
                  "取消": function () {
                      $(this).dialog('close'); //关闭对话框
                      vector.getSource().removeFeature(currentFeature); //删除当前
绘制图形
                  }
              }
          });
```

在交互添加点之后会出发回调函数 drawEndCallBack，该函数的功能主要是获取当前添加的点的坐标，并将其 x 坐标和 y 坐标以“;”隔开，同时打开“图形属性信息设置”对话框。在打开该对话框并输入相应信息之后，点击提交按钮，则会触发 submitData 函数，从而

将该对话框中的信息获取到变量中去。如果该点的坐标不为空，则执行 SaveData 函数，将前台的点坐标、名称、类别、属性等信息通过 Ajax 技术传送到后台，从而在后台进行插入数据库的工作；如果该点的坐标为空，则给出提示并清除当前绘制的要素。具体代码如下：

```
/**
        * 绘制结束事件的回调函数，
        * @param {ol.interaction.DrawEvent} evt 绘制结束事件
        */
        function drawEndCallBack(evt) {
            var geoType = $("#pnt_Type:selected").val();//绘制图形类型
            $("#dialog-confirm").dialog("open"); //打开属性信息设置对话框
            currentFeature = evt.feature; //当前绘制的要素
            var geo = currentFeature.getGeometry(); //获取要素的几何信息
            coordinates = geo.getCoordinates(); //获取几何坐标
            geostr = coordinates.join(";");
        }
         /**
        * 将绘制的几何数据与对话框设置的属性数据提交到后台处理
        */
    function submitData() {
            var type = $("#pnt_Type option:selected").val(); //绘制图形类型
            var name = $("#pnt_name").val(); //名称
            var attribute = $("#pnt_attribute").val(); //详细信息
        if (coordinates != null) {
            saveData(type, geostr, name, attribute); //将数据提交到后台处理（保
存到数据库中）
                currentFeature = null;  //置空当前绘制的几何要素
                coordinates = null; //置空当前绘制图形的coordinates
            }
            else {
                alert("未得到绘制图形几何信息！");
                vector.getSource().removeFeature(currentFeature); //删除当前绘
制图形
            }
        }
        /**
        * 提交数据到后台保存
        * @param {string} type 绘制的几何类型
        * @param {string} geoData 几何数据
        * @param {string} attribute 属性数据
        * @param {string} name 名称
        */
        function saveData(type,geoData, name,attribute) {
```

```
            //通过ajax请求将数据传到后台文件进行保存处理
            $.ajax({
                url: 'DataHandler.ashx', //请求地址
                type: 'POST',  //请求方式为post
                data: { 'type': type, 'geo': geoData, 'name': name, 'att':
attribute }, //传入参数
                dataType: 'text', //返回数据格式
                success: function (response) {   //请求成功完成后要执行的方法
                    alert(response);
                },
                error: function (err) {
                    alert("执行失败");
                }
            });
        }
```

（3）在后台处理前台传送过来的数据需要新建一个 DataHandler.ashx 一般处理程序来进行处理。在后台进行数据处理时，需要先将前台的数据获取到变量中，然后打开数据库连接，通过 INSERT 语句将其插入到数据库中。具体处理代码如下：

```
            context.Response.ContentType = "text/plain";
            string type = context.Request.Form["type"];
            string geo = context.Request.Form["geo"];
            string name = context.Request.Form["name"];
            string attribute = context.Request.Form["att"];

            string[] arrTemp = geo.Split(';');
            double x = Convert.ToDouble(arrTemp[0]);
            double y = Convert.ToDouble(arrTemp[1]);

            string connStr = "Data Source=.;Initial Catalog=GIS;User
ID=WMT;Password=wenmengtian";
            SqlConnection conn = new SqlConnection(connStr);
            try
            {
                if (conn.State == ConnectionState.Open)
                {
                    conn.Close();
                }
                conn.Open();//打开连接

                string insertStr = "insert into point
```

```
(pnt_x,pnt_y,pnt_type,pnt_name,pnt_attribute)
values("+x+","+y+",'"+type+"','"+name+"','"+attribute+"');";
                SqlCommand insertCmd = new SqlCommand(insertStr, conn);
                int updateCount = insertCmd.ExecuteNonQuery();
                if(updateCount==1)
                {
                    context.Response.Write("数据保存成功！");
                }
                else
                {
                    context.Response.Write("数据保存失败！");
                }
            }
            catch (SqlException e)
            {
                string message = "数据保存失败" + e.Message;
                context.Response.Write(message);
            }
            finally
            {
                if(conn.State==ConnectionState.Open)
                    conn.Close();
            }
```

代码说明：以上代码是写在 ashx 文件的 ProcessRequest 函数中的。

七、系统部署

本系统使用 JavaScript 结合.NET 模式开发，因此需要 Internet 信息服务管理器（IIS）作为 Web 服务器发布站点。在部署站点之前，需要安装配置 IIS，可以通过控制面板的打开或删除 Windows 功能安装 IIS，具体操作步骤请参见相关资料。下面以 Windows 10 系统为例，发布 POI 在线管理系统。

系统详细发布步骤如下：

（1）在发布站点之前，首先验证 IIS 中的“应用程序池”是否包含 ASP.NET v4.5、ASP.NET v4.5 Classic，打开 Internet 信息服务（IIS）管理器，鼠标双击打开应用程序池，查看是否有图 7.1 中方框里的 ASP.NET v4.5 和 ASP.NET v4.5 Classic 项。如果没有，说明.NET Framework 没有完全安装，，这种情况下发布的网站是无法访问的，需要重新安装配置。

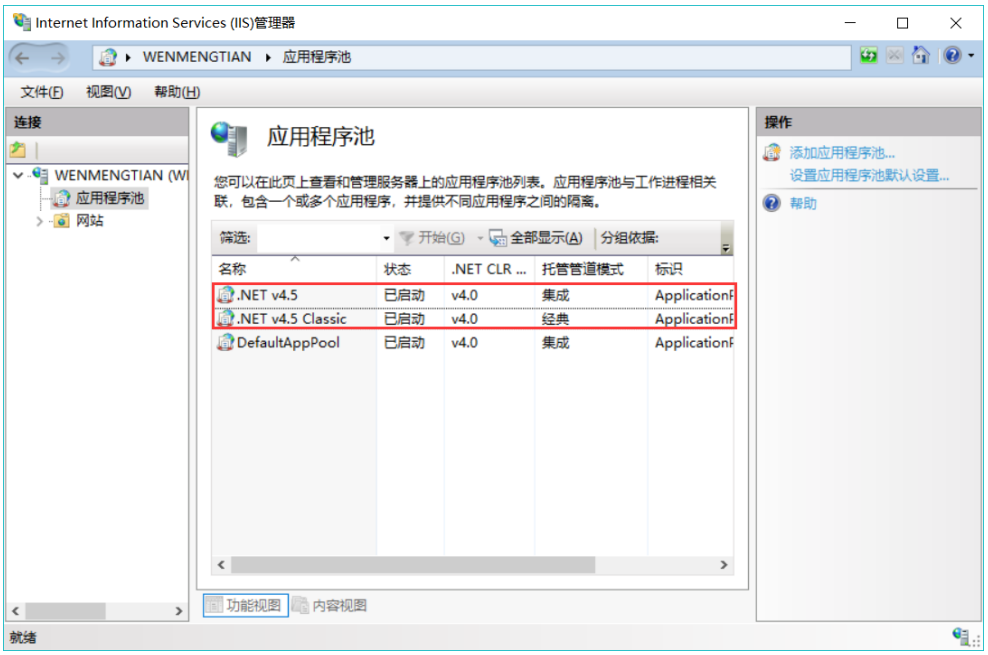


图 7.1 验证.NET Framework

（2）在 IIS 中，鼠标右键单击“网站”节点，在弹出菜单中选择“添加网站…”，然后在弹出的“添加网站”窗口中配置要发布的站点。即按照如图 7.2 所示配置站点，应用程序池选择 .NET v4.5，设置 IP 及网站名称。

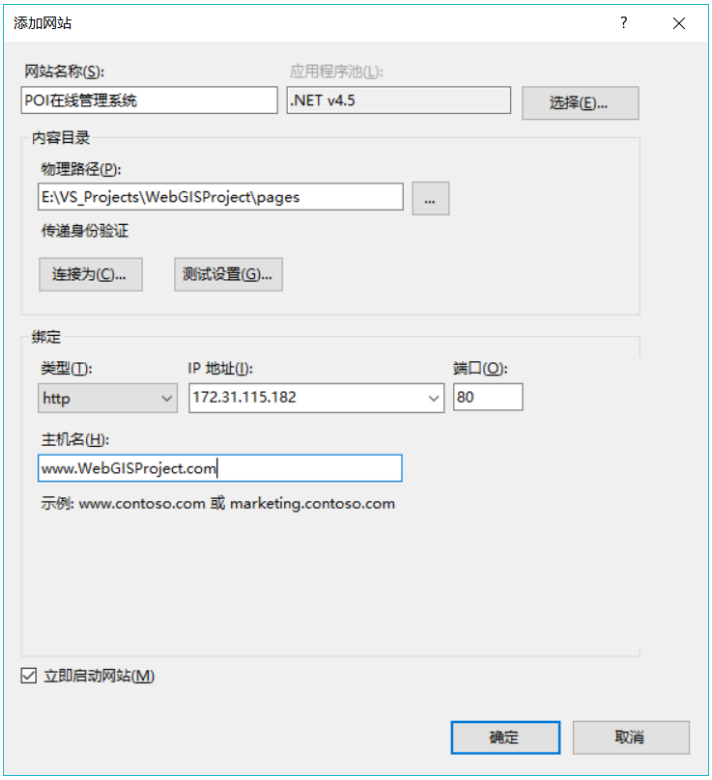


图 7.2 站点发布配置

（3）双击新建的网站，如图 7.3 所示，选择右侧操作窗口中“管理网站”下的“重新启动”项，然后单击“浏览网站”下的“浏览 172.31.115.180…”项，即可在浏览器中查看已发布的网站。

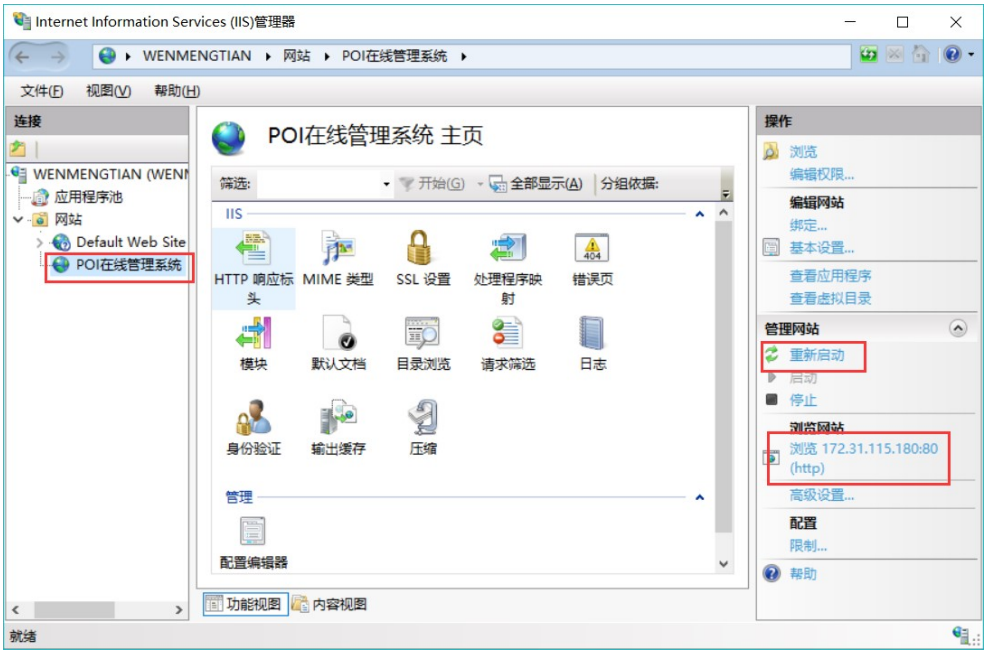


图 7.3　浏览网站方式

以上便是 POI 在线管理系统实现的整个过程。该系统以 OpenLayers3+jQuery 为核心，保证了系统的灵活性、可拓展性和可维护性。

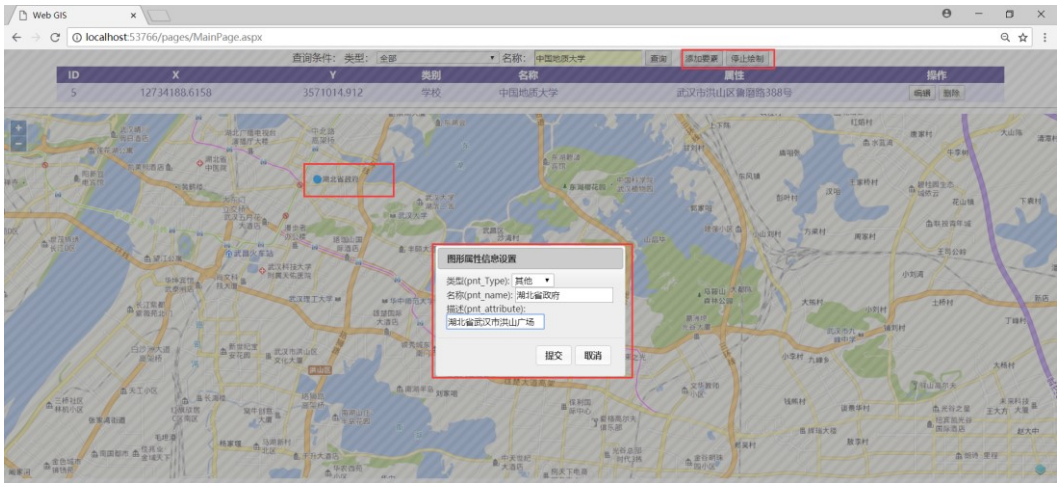
八、成果展示

（1）系统主页

（2）按类别查询

（3）按名称查询

（4）添加 POI



九、附录

9.1 WebGISProject

pages：主页面代码和一般处理程序代码

MainPage.aspx——主页面前端源代码

MainPage.aspx.cs——主页面后台源代码

MainPage.aspx.designer.cs——主页面前端控件代码

DataHandler.ashx——一般处理程序代码

DataHandler.ashx.cs——一般处理程序后台

libs：第三方库目录

9.2 数据库

GIS.mdf——数据库数据文件

GIS_log.ldf——数据库日志文件