编 号

本科生毕业设计（论文）

题目：基于 AIR 的地图应用程序开发

数字媒体 学 院 数字媒体技术 专 业

学　　号 0305070113

学生姓名 钟云男

指导教师 夏鸿斌　副教授

二〇一一年六月

摘 要

随着现代互联网技术的高速发展，人们可以很容易地从互联网上获得地理信息服务．目前，谷歌、雅虎等都提供了在线地理服务,并提供了如 Google Maps、Yahoo！ Mmaps 等开放的 API 集合，使得应用程序开发者可以很方便地开发和集成他们自己的地理信息应用程序．

Adobe AIR 环境使开发人员能使用 HTML、JavaScript、ActionScript 构建 web 应用程序，这些应用程序可以作为独立的客户端应用程序运行并且不受浏览器的约束．Adobe AIR 作为 Flash Platform 的一个关键组件，为跨设备和平台应用程序提供了一个一致、灵活的开发环境，是设计人员和开发人员能够完全释放自己的创意．

本文主要工作是基于 AIR 环境设计开发了在线地图应用程序，提供给用户目的地的地理信息在线查询．本应用程序除了可以在线搜索目的地的地理信息外，还为用户提供了离线查询、编辑的功能．该应用程序很适合打算旅游、出差等有出行计划的人们．

本文主要是在 Adobe AIR 的基础之上，通过 Flex 3.0 的集成环境，再与 Yahoo! Maps AJAX API 的结合进行开发的，对于如何构建富互联网应用架构和开发具有一定的指导意义．

关键词：地图应用程序；AIR；Yahoo！Maps；在线地理服务

ABSTRACT

With the rapid development of modern Internet technology, People can easily access geographic information from the Internet. Currently, Google, Yahoo and so on provide an online geographic services, and provide set of open API, such as Google Maps, Yahoo! Mmaps. which makes application developers develop and integrate their own GIS applications more easily.

Adobe AIR environment allows developers to use HTML, JavaScript, ActionScript to build web applications. These applications can be used as stand-alone client application and not be bound by the browser. Adobe AIR as a key component for applications of Flash Platform across devices and platforms provide a consistent and flexible development environment for designers and developers to fully release their own creative.

This article is about to buid a online maps based on the work environment of AIR applications to provide geographic information of destination online to users. This not only provides the fiction of geographic information online searching, but also provides users with offline query editing. The application is intended for tourism, travel and other people who have travel plans.

This article is based on Adobe AIR develop environment, using Flex 3.0 integrated environment, and then with a combination of Yahoo! Maps AJAX API to develop, and this has the guiding significance for how to build rich Internet application architecture and development.

Keywords: Map application; Adobe AIR; Yahoo！ Maps; Online Map Service

目 录

第 1 章 绪论

诗人 E.E.卡明斯曾经写过：

Somewhere I've never travelled，glad than any experience！

是的，几乎没有人不喜欢旅行，到陌生的地方总是可以带给我们很多莫名和未知的惊喜．但是旅行也需要很度考虑的东西，所谓衣食住行，这些都必须在出门之前有所考虑．当今社会，信息技术发达，对于目的地的诸多信息，我们都可以通过 Internet 检索，活得有关目的地的．

诸多信息，目的地附近的酒店、餐厅以及交通方式的选择都可以通过网络资源而活得．然而地图，首当其冲，仍是必不可少的必备工具．只是如今，电子信息技术的高速发展，我们可以不用再另备一份纸质的地图，而可以用电子地图取而代之了．

网络电子地图是一种以互联网为传输介质的新型电子地图产品．它将电子地图的特点与网络特性融于一体，具有远程地图信息传输、传播广泛便捷、适时动态更新、人机交互性强、充分利用超媒体结构等诸多特点，越来越受到人们的青睐，并日趋平民化、大众化，拥有具大发展潜力．国际上比较大的网站也都开辟了网络地图专栏，出现了一批专门的通用地图网站或专题地图网站．美国、加拿大等国家还专门设立国家地图集网站，将国家地图集的主要内容通过互联网发布，实现免费共享，用户可以上网查询检索、浏览阅读、打印或下载所需要的图幅或其他信息．

传统地图对于地图制作者而言，必须设法将三维球体尽可能精确地转换到二维平面上，必须系统地而不是任意地定义地图图幅，必须根据比例尺大小以及制图目的对地图所能表达的内容进行取舍和符号化．对于地图使用者而言，虽然可以把地图视为现实世界的模型，但只能通过视觉感官获取地图信息，而不能像与现实世界的交互那样可以调动多种感觉和运动器官．制作和使用过程中存在着一系列固有的约束．在这个时期地图认知的研究都是针对传统地图而言的．电子地图特别是新颖的富有创意的网络地图产品改变了地图用户的思维方式和认知模式．尤其是对地图的设计者提出了新的更高的要求，即如何设计网络地图，才能更块、更好地满足用图者的多样化需求．

目前互联网上的电子地图从功能来讲主要有两种：一种是阅读型的网络电子地图．主要用于单一的地图显示和阅读，只提供放大、缩小、漫游等功能， 没有空间信息数据库的支持，不能用于查询分析，即该地图不能被“点击”．如微软的 encarta 网站提供的地图，分析功能很少；另一种是交互型的电子地图．交互型电子地图是在数据库和软硬件资源支持下，能实现对电子地图各种操作，例如显示阅读、属性查询、空间检索分析、热点链接、打印输出、发送邮件等．交互型的电子地图是当前主要研究方向之一，在数据组织管理、功能操作、用户界面友好等诸多方面，体现了电子地图技术水平，已成为网络电子地图的主流．目前大多数网络电子地图产品均属于这一类．提供网络地图的网站也很多，如：百度地图、黄页地图（香港）、搜狗、中大在线地理信息网、台湾电子地图服务网、中原地图、澳门旅游电子地图、动态地图网（台湾）、TOM 地图搜索引擎、微软地图网站等．

地图应用就是一条已经被聚焦的虫子．从导航测绘公司到各类网站，形形色色的或创业公司或成熟公司已经在地图应用上投入了相当的技术和精力．同时，因为技术门坎的原因，地图网站以及相关的产业价值链的竞争秩序已经俨然成形．

当前，地图应用程序在许多基于地理信息的在线服务中应用非常广泛．无论是交通行驶的方向导航，还是寻找最近的酒店，或者在居住地进行鸟瞰，只要向在线地理服务如：google maps，yahoo maps 等发送请求信息，它就会告诉你在哪里转弯，目标方位或者着陆地点信息．这些地理信息在线服务提供一个开放的 API 集合，使得应用程序开发者可以开发和集成他们自己的地理信息应用程序．

使用电子地图可以远程地图信息传输，即使活得信息更新，方便快捷．伴随着人们对旅行、出游兴趣增加，对地图应用的普遍看好，地图 API 和地图应用程序开发面临着挑战和机遇．预计在不远的未来，地图应用更加成熟和丰富，到那时，人们使用电子地图就会像每日使用电子邮件或 IE 一样稀松平常的应用．

第 2 章 系统分析

2.1 需求分析

本地图应用程序将使用户可以运用在线地图 API 来搜索具体的目的地，添加去往某地的路线标记，以及离线状态下查看和编辑曾经保存过的目的地信息.

本应用程序尤其适用与即将去到一个陌生的地方旅行的旅行者．我们可以在出发之前就将要去的目的地的信息查好，并作好标记，保存在程序里，到达目的地之后，即使没有网络，没有有地图，只需要从程序的离线模式下调出已经查好的信息，就可以轻松找到目的地的信息．对旅行者来说，非常方便而且非常使用．另如果可以有便携的手机、PDA 等移动设备，就更加方便了.

尤其是当 Yahoo！ Maps V3.8 支持台湾地区之后，对于我们就更方便了．虽然雅虎奇摩地图，已经将台湾地图做得很完善了，但是对于刚到台湾，或者临时去台湾出差，旅游就不是很方便，而且，在去之前，已经查过的信息，我们就会很希望将其保存下来，以便下次还可以再用，而不需要重新再查过，那样，既方便又快捷，即使是在网络信号覆盖较弱或者临时到达地方还不是立刻连接上网络地方，也可以立刻使用．减少了旅途中的不少麻烦．尤其是独自旅行，最担心的就是迷路，对目的地不熟悉．本程序可以轻松解决这个问题.

另对于有些方向感不强的人来说，本地图程序，也在一定程度上解决了他们的后顾之忧，本程序还提供平面，卫星，混合模式的三种地图，也可以分别以这三种模式对地图进行保存.

2.2 可行性分析

技术可行性：目前几个大型的网站，如 Yahoo!奇摩地图，百度地图，搜狗地图，都有自己的地图应用，同时在线地理服务，如 Google maps 和 yahoo！Maps 提供了一个开放地理信息的 API 合集，可以将这些 API 集成到自己的地图应用程序中去.

Adobe Integrated Runtime (AIR™)允许程序员利用现有的 web 开发技能(包括 Flash, Flex, HTML, JavaScript, Ajax)优势，建立和配置跨平台（或跨操作系统）的桌面 RIA（Internet Applications）应用，所以从技术方面来考虑，是可以实现的.

经济可行性：时间方面大概要花费 2 个多月的时间，回报方面，目前没有任何的回报，但是，有了本系统用户可以轻松搜索地理信息，即使是在离线的情况下，也可以搜索到目的地的信息，本系统很适合旅行着，在旅行之前查好信息，到达目的地之后，即使不能联网也可以检索信息，所以从长远的角度来看，本系统是值得开发.

操作可行性： 本系统操作很简单，跟大部分网站的地图应用操作是差不多的，只需要在检索地址栏里输入要搜索的目的地名就可以了，将已检索过的另存为名称就好了.

结论：通过以上分析，此系统可以也有必要进行开发！

第 3 章 开发环境介绍

3.1 简介 Adobe AIR

3.1.1 Adobe AIR

Adobe AIR （AIR 是 Adobe Integrated Runtime 的缩写）：开发代号为 Apoll．AIR 是针对网络与桌面应用的结合所开发出来的技术，可以不必经由浏览器而对网络上的云端程式做控制．

图 3-1 Adobe AIR 标志

Adobe 集成运行时（AIR）是一个跨平台的运行环境为构建富 Internet 应用程序使用 Adobe Flash，Adobe 的 Flex，HTML 或 Ajax 的，可以作为一个桌面应用程序部署．从 Adobe 的 Adobe 集成运行时系统，已在发展，从过去几年．开发人员现在能够构建应用程序跨平台，利用其现有的 HTML 技能，Flash 和 Flex 和部署在几乎所有的作业系统中．

使用 AIR 开发人员可以创建应用程序相结合的好处 Web 应用程序，如：网络和用户连接，内容丰富的媒体，易于开发和广泛的接触与桌面的优势与其他应用程序，本地资源访问的互动：如申请离线访问信息，丰富的交互体验.

Adobe AIR 的这不是另一种编程语言，它只是一个运行时是因为它允许现有的 Flash，ActionScript 或 HTML 和 JavaScript 代码将用于建设一个更加传统的桌面程序一样．对于 Flex 开发商什么可以做，在 Flex 中，你可以在 AIR 应用程序建于 Flex 和 Flash 的同样的事情或 HTML 的．

Adobe AIR 的这不是另一种编程语言，它只是一个运行时即使如此 allowAdobe AIR 是即将从网络到桌面，并针对 Web 开发人员．它的主要用途是使 Web 应用程序和 RIA 的部署到桌面上．

运行时本身就是作为一个软件安装在一个不同的数它支持的操作系统和桌面的安装和运行这是在 Flex，Flash 和 HTML 构建的应用程序架构．

有一个全套的开发工具是由 Adobe 提供构建这些应用程序．这些开发工具有些是免费的，其中一些被纳入现有的商业工具，例如 Flex 生成器，Flash 和 Dreamweaver．

Adobe AIR 运行时可能是一个相对较新的平台，但它其实嵌入三个高度成熟和稳定的跨平台技术权力 AIR 应用程序．这些如下：

WebKit 的：渲染 HTML 的 AIR 应用程序内的内容使用．WebKit 是一个开源，跨平台的浏览器，是基础层苹果的 Safari 浏览器． WebKit 是以其 W3C 的大力支持如 HTML，XHTML 的文档对象模型（DOM）的标准，层叠样式表（CSS）和 ECMAScript 的．然而，

它也提供增强功能的支持（使创造的圆角使用 CSS）．制定办法，因为 Adobe AIRWebKit 的开发，你可以自由地利用这些非标准的扩展优势，而无需担心关于浏览器的兼容性．

3.1.2 Adobe AIR 技术的特点

从 Adobe 官方的宣传来看，其特点有下列几点：

1、本地运行-类似桌面应用程序．

2、跨平台-类似 java 技术，在不同的操作系统上有对应的虚拟机支持，目前已经有 windows 和 mac，linux．

3、开发是基于现有的 web 技术，如 Flash / Flex / ActionScript / HTML / JavaScript / CSS / Ajax /PDF，对于开发人员，不需要学习 c、c++、java 之类的底层开发语言，不需要学习具体操作系统底层 API 的开发；这降低了开发门槛，使现有的做 web 开发的技术人员，依赖其原本就很熟悉的开发模式，稍加训练就可以开发良好丰富的富客户端应用．

3.2 Yahoo！Maps API 简介

Yahoo! Maps API 能协助程序员很容易地将 Yahoo! 地图嵌入您的网站或桌面应用程式中．Yahoo! Maps API 支持 GeoRSS 标准，开发者可以扩增地图 API 功能来与其他 API 整合．

目前 Yahoo! 奇摩已提供涵盖台湾地图的 Yahoo! Maps AJAX API V3.8 版本，程序员可以使用 DHTML 和 JavaScript® 程序语法将 Yahoo! Maps 嵌入自己的网站中．你可以使用自己喜欢的编辑器或 IDE 来编写 JavaScript 程序．

Yahoo! Maps AJAX API 支援下列浏览器版本，包括 Firefox 2, Internet Explorer 6 or 7, Opera 9 或者是 Safari 3， 更新版本也同时支持．

Yahoo!Maps 使用的街道网络和其他矢量信息主要来自 Navteq、Tele Atlas，以及公有领域信息源．目前针对美国、加拿大、波多黎各，以及维京群岛和大部分欧盟国家，还有专用的街道网络数据可以使用．国界线、城市，以及水域则映射自世界其他地区．

中等分辨率的卫星图像已经覆盖了全球范围，1-2 米分辨率的图像则仅覆盖美国及相邻地区，以及其他国家的部分城市．

3.3 集成开发环境 Flex 3.0

Flex Builder 是 Macromedia 的 IDE，用于 Flex 应用程序开发．Flex Builder 使设计者和开发者更高效地建立 Flex 应用程序．设计者可以快速建立应用程序界面、快速简单地分布引人注目的 Flex 用户界面并将其连接到后端数据源．开发者可以使用代码提示和调试工具，对 Flex 应用程序进行高效的的编码和调试． Flex Builder 3 是基于 Eclipse 的产品．其图标如图 3-2 所示．

图3-2 flex builder 标志

Flex 的目标是让程序员更快更简单地开发 RIA 应用．在多层式开发模型中，Flex 应用属于表现层．

Flex 采用 GUI 界面开发，使用基于 XML 的 MXML 语言．Flex 具有多种组件，可实现 Web Services，远程对象，drag and drop，列排序，图表等功能；FLEX 内建动画效果和其它简单互动界面等．相对于基于 HTML 的应用在每个请求时都需要执行服务器端的模板，由于客户端只需要载入一次，FLEX 应用程序的工作流被大大改善．FLEX 的语言和文件结构也试图把应用程序的逻辑从设计中分离出来．

第 4 章 系统设计

本应用程序主要是为了实现电子地图的功能．如同大多数地图应用一样，只要在搜索框里输入要搜索的目的地名称，就可以查到相应目的地信息，并在显示框里显示出来．同时本地图应用程序还允许用户通过拖拽鼠标在地图显示区域标记目的地的信息并进行保存；当用户将所标记的内容保存后，标记过及查询的内容将被保存在本地数据库中，这样即使在离线的情况下也可以查看目的地信息．

所以本应用程序主要有两个检索模式：在线和离线．当网络连接可用是，应用程序同过网络地图 API 获得地理信息，离线时程序通过本地数据库获得地理信息．这就需要对获得的数据进行处理．

但不论是在线模式还是在离线模式下，地图的地理信息都是在显示窗口中显示，因而本应用程序主要分为两大块的设计：数据处理和构建用户界面．

4.1 数据处理

根据系统功能设计，本应用程序的数据主要是分为两个部分：在线和离线．

在线部分，主要是通过与 Yahoo！ Maps API 通信来实现．使用 JavaScript 与一个在线 API 进行通信，搜索具体地点并添加目的地信息——在本地图应用程序中将使用在线地理信息服务 API 是 Yahoo！Maps AJAX 提供的 API．使用这个服务过程将涉及如何使用 HTML 控件，以及如何实现 AIR 应用程序与 JavaScript 之间进行代码桥接．由于程序和 Yahoo ！Maps AJAX API 间的通信依赖于 Internet 连接，因此程序的运行将受到网络资源可访问性和可用性的限制．

当程序的网络资源可用时，将生成一个 LocationVO 值对象，用于记录储存关于具体地理方位的信息，以及要显示的图形内容．

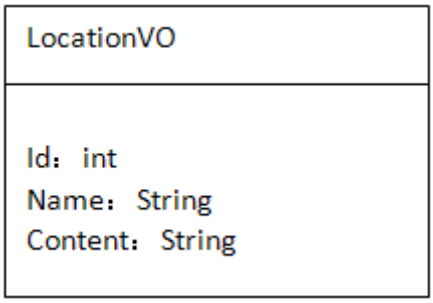

图 4-1 LocationVO 类的 uml 图

还会在创建一个 MarkerVO 值对象，用于记录关于目的地的标注信息．其作用不仅要记录下目的地标记在屏幕上的坐标，同时还要保存用户标记输入的有关目的地的名称．

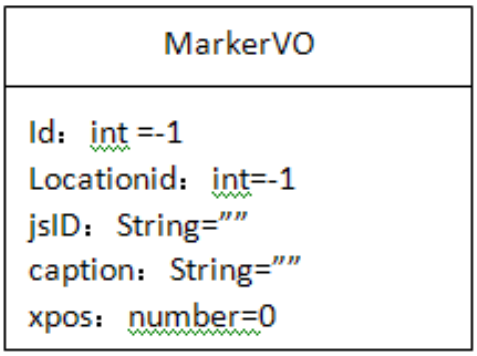

图 4-2 MarkerVO 类的 uml 图

本应用程序将创建两个 LocationVO 和 MarkerVO 值对象类．LocationVO 用于存贮关于具体地理方位的信息；MarkerVO 类用于存贮关于目的地信息．

本应用程序希望用户直接键入目的地名称时，就可以直接找到目的地位置，而不必知道目的地在地图上的经度、纬度等具体地理，坐标．然后我希望像大多数在先地理应用程序一样，可以通过可是标记来指明特定地理方位的目的地信息．这就需要将数据进行归一处理．因此，这两个类都应该继承接口 IDBEntry，通过 IDBEntry 将对数据经行预处理．参照很多网站地图搜索，在搜索框内键入地点名称，点击搜索按钮就可以检索到目的地．

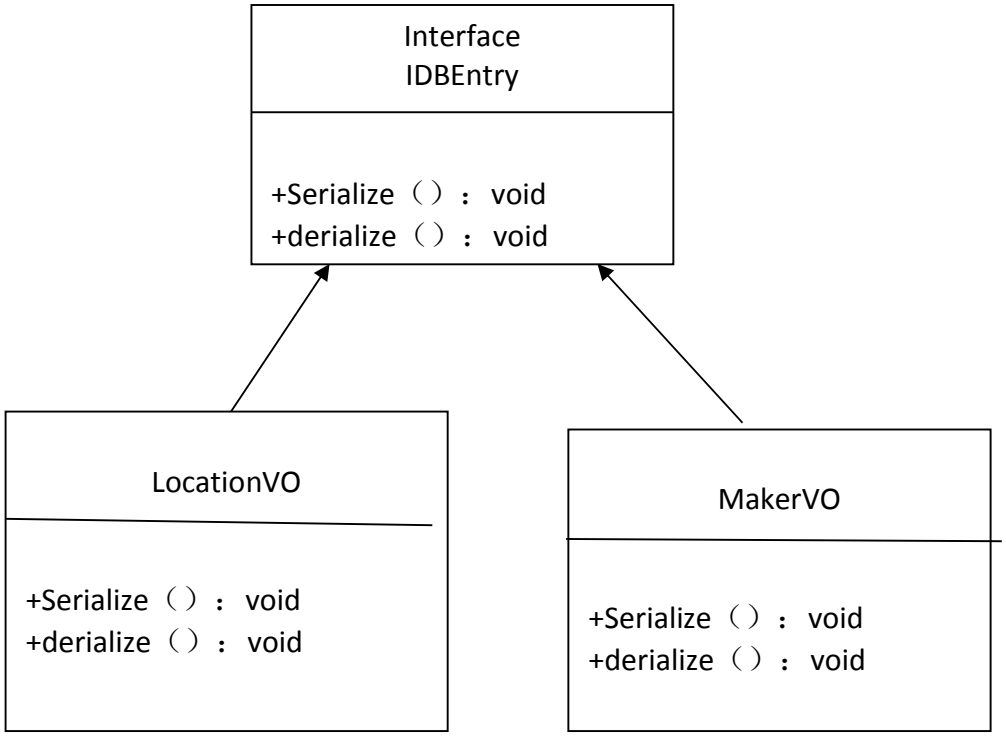

图 4-3 值对象继承接口 IDBEntry

离线部分，主要是通过与与本地数据库进行通信，并保存、查看和编辑地理位置信息——地图应用程序将利用 AIR 的数据库 API 来插入新的记录，从一个本地数据库中提取和跟新以前已经保存了的数据记录．由于保存的数据记录可以从本地数据库中获得，通信不受网络资源的限制，因此可以在没有 Internet 连接的情况下对数据进行访问．

所以通过 QueryVO，DBSerivice 类和 DBResultEvent 类对数据库进行读取操作．

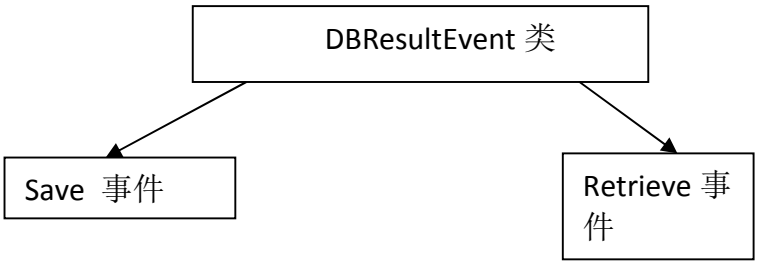

图 4-4 DBResultEvent 类的事件分类

小结：

当总的说来，数据处理部分需要实现的部分分为两块：

在线：

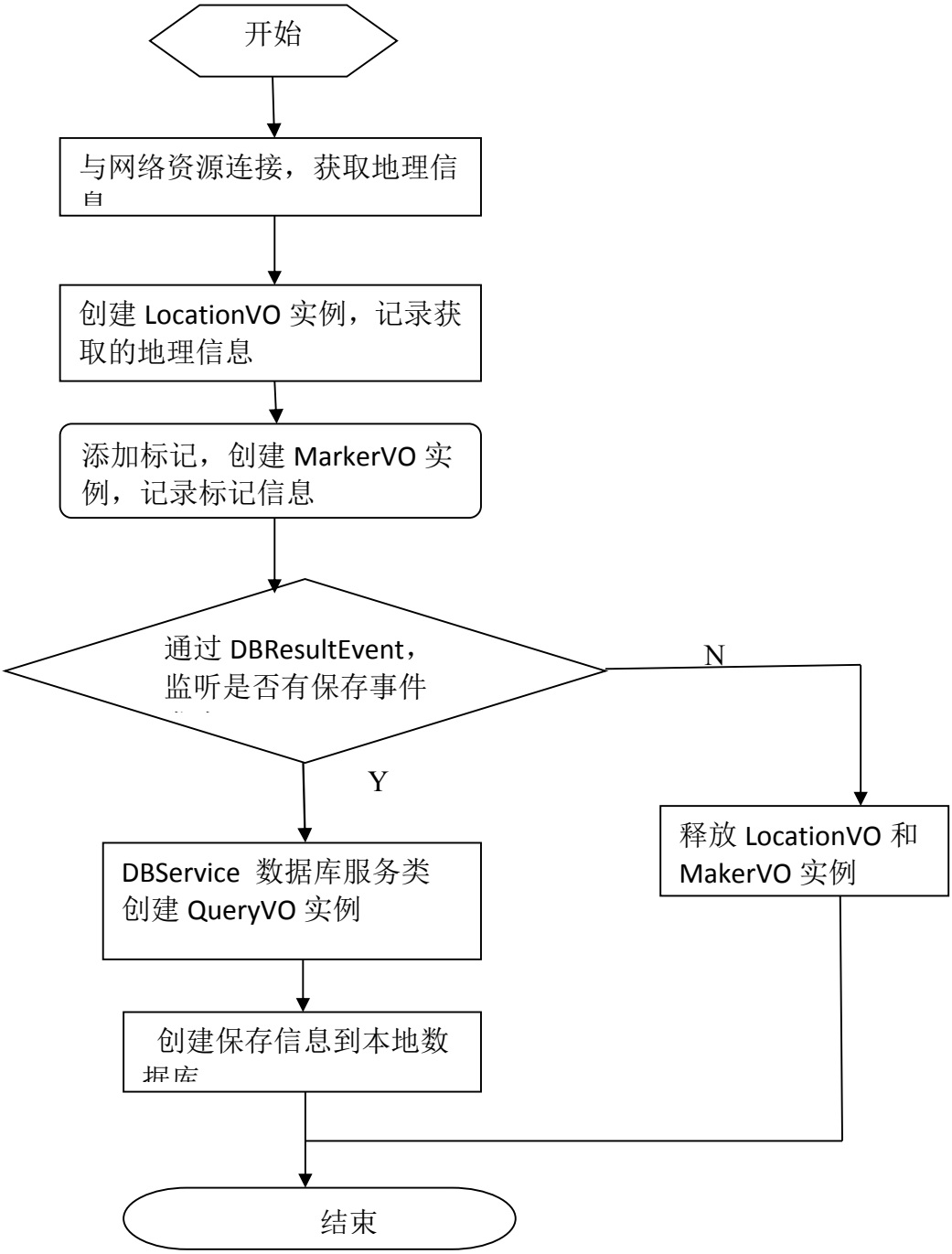

图 4-5 在线时数据处理流程图

离线时：

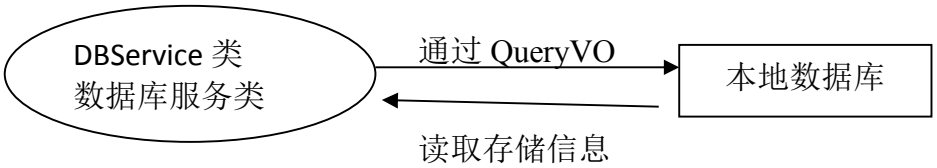

图 4-6 离线线时数据处理流程图

4.2 构建用户界面

地图应用程序的用户受当前网络资源是否可访问的影响．本程序支持使用在线地理信息数据的能力，当然这依赖于 Internet 连接．此外本程序也支持使用已保存地理信息数据的能力，这不需要 Internet 连接．因此，地图应用程序需要 3 个主要视图：

模式选择控制视图——模式控制视图根据 Internet 连接是否可用，确定用户是使用本地保存的数据，还是使用由在线地图 API 提供的数据．

地理信息显示视图——显示视图根据模式控制视图更新自己的显示效果．当用户倾向于或不得不在离线的状态下工作，显示视图将根据用户选定的具体地理方位，渲染所有已保存过的图形和记号．如果是在线状态下工作，显示视图将提供封装好的 API，支持用户定位和下载各种地理图形数据．

位置查询控制视图——位置查询控制基于模式控制来更新自身界面的控件，使用户可以添加、搜索和编辑具体的地理方位．

图 4-7 用户界面显示视图

4.2.1 模式选择控制视图

模式控制简单说来就是对程序是在线连接还是离线进行控制．设计两种模型：在线和离线．创建一个类 ModeTypes，封装这两种模型．然后通过通过 flex 页面布局，添加单选框以选择模式，并同将模式类型传递给参数 mode，其他模块根据 mode 的值来判断是是在线模式还是离线模式．同时通过 bindable 标签同步监听模式之间的转换，更新 mode 的值．Bindable 元数据标签是最经常用到的一个元数据标签，因为它使程序组件之间的数据同步变得很容易．

另外还需要用 isAvailable 属性来标记网络是否连接，网络资源是否可用．isAvailable 为 true 时网络是可用的，false 则不可用．isAvailable 的值将在最后模块整合时，通过 Locator.mxml 中的 checkConnection() 检测返回．

设计实现如下图：

图4-8 模式选择控制视图

4.2.2 地理图形显示视图

如同许多地图网站一样，能够在显示框中显示地图图形，通过 Yahoo！Maps API 类型选择控件可以实现三种方式显示地图信息：

1、普通地图模式：

图4-9 普通模式下地图显示

2、卫星模式

图4-10 卫星模式下地图显示

3、地形模式（可以显示等高线）

图4-11 地形模式下地图显示

在 flex IDE 中创建一个显示控件，用于显示地理图形，同时创建主视图类 LocationDisplay．主视图类首先要建立一个 ViewStack 容器，用于显示地图的地理信息，同时主视图类还要检测判断当前的模式是在线还是离线模式，从而确定 ViewStack 容器是选择 HTML 控间显示，还是 Image 控件显示．

在在线模式下，主视图类 LocationDisplay 通过与 HTML 文档交互，获得地理信息，并将其在主视图中显示出来．同时还要将当前具体地理方位数据生成地理图像的快照，存储于图片缓存类中，如果保存，则会将该快照作为 LocationVO 实例存入存储到数据 库中．

当在离线的模式下，主类视图就通过读取数据库缓存信息，显示出已保存过的内容．

另外还要实现通过拖拽就可以实现地图标记的功能，通过 flex 框架的的鼠标拖放，当标记图形拖到地图显示区域是，就会新建一个标记类，在视图上出现标记对话框，如图所示：

图4-12　弹出标记窗口

并将标记信息记录下来，当下次在路过的时候，就显示出用户标记的信息．

如下图：

图4-13 显示用户标记信息

通过 flex 页面布局构建一个标记框，当在用户拖拽标记图形放到图形显示窗口上时，就创建一个 MarkerVO 实例，并弹出标记对话框，记录下当前信息．并通过 flash 事件侦听器，记录下标记的信息内容．

4.2.3 位置查询控制视图

查询控制也是通过 Flex 的页面布局 mxml 文档中将用到 flash 事件侦听器，数据处理时创建的数据服务 DBservice，对数据进行管理的 LocatorDataManager 类以及 flex 框架的 Binding 和 ArrayCollection 等组件．

位置控制也要对 mode 值进行检测，判断当前模式是在线模式还是离线模式．

在在线模式下，用户输入目的的名称，并查询，就可以在显示视图中显示出目的地的地图信息．如图：

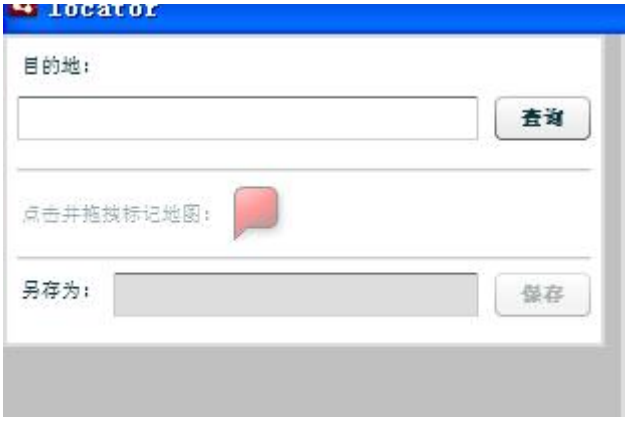

图4-14 在线模式下位置查询

当用户在文本框中输入要查询的目的的信息是，就要通知其他监听模块，并向 Yahoo！Maps API 发送请求.

同时在离线情况下，文本框变为下拉选项框，下拉选项中则会列出本地数据库中存储的已保存的数据信息内容，如图所示：

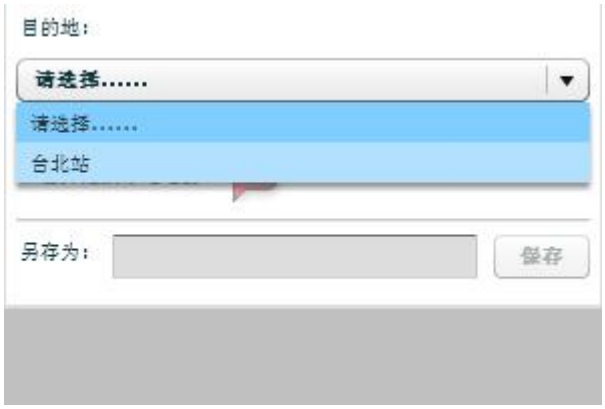

图4-15 离线模式下位置查询

同时在离线情况下还需要监听保存按钮的鼠标事件．当 save 保存事件发生时，就将创建的 LocationVO 对象的 id 值进行修改．如果 LocationVO 对象中的 id 值为-1，则表明 LocationVO 对象中的地理方位数据没有进行保存.

第 5 章 系统实现

5.1 处理数据

地图应用程序数据来自两个相互独立的资源—一个是与在线 API 交互的 HTML 文档，另一个是本地数据库.尽管由着两个资源服务提供的数据可能是不同的，但是地图应用程序本身将使用单一类型的模型.

要实现 Yahoo! Maps AJAX API 与本地数据库间的交互，需要使用不同的控制逻辑代码，但这些控制逻辑鼻血知道如何处理同一类型的数据.数据在两个数据源和地图应用程序之间传递需要进行归一化预处理.

5.1.1 对数据经行预处理

地图应用程序将表现两种具体类型的地理数据：地理方位和目的地.应用程序同过对数据的处理和表现，使得用户在查看使用地理方位表达的目的地信息是，可以直接找到目的地的位置，而不必知道目的地在地图上的经度、纬度等具体地理坐标.正如大多数的在线了地理信息应用程序一样，我们可以用可视标记来指明特定地理方位的目的地信息.因此，将先处理由数据库和在线地理信息服务提供的两种主要数据：地理方位和目的地标记.

首先创建一个 IDBEntry 接口，保存在 IDBEntry.as 文件中，该接口中的 serialize（）和 deserialize（）方法分别用于对字符串属性进行序列化和反序列化.序列化用于对数据的属性值进行预处理，是数据可以正确的输入数据库.对数据库的数据进行预序列化将使值对象还原为用户最初输入的格式.基于数据预处理需求，用用程序在录入和提取数据库的任何数据时都要用 IDBEntry 接口的序列化和反序列化方法进行预处理.

IDBEntry 接口：

```
public interface IDBEntry
{
    function serialize():void;
    function deserialize():void;
}
```

5.1.1 保存地理信息的值对象的实现

创建 LocationVO 类和 MarkerVO 类，这两个类是两个值对象，他们在与 the Yahoo！Maps AJAX API 和本地数据库之间交互时被具体实现.LocationVO 类用于存贮关于具体地理方位的信息. 它的 x 和 y 属性表示目的地在屏幕上的坐标. jsID 属性与通过 Yahoo！Maps AJAX API 添加的标记相关，caption 属性用于表示用户输入的有关目的地的名称.

LocationVO 类：

```
public class LocationVO extends EventDispatcher  implements IDBEntry
{
    public var id:int;
    public var name:String;
    public var content:String;
     public function serialize():void
    {
```

```
            name = escape( name );
            content = escape( content );
        }
        public function deserialize():void
        {
            name = unescape( name );
            content = unescape( content );
        }
```

其 name 属性与用户的输入信息有关，content 用于记录要显示的图形内容.

MarkerVO 类：

```
    public class MarkerVO implements IDBEntry
    {
        public var id:int;
        public var locationId:int;
        public var jsId:String;
        public var caption:String;
        public var x:Number;
        public var y:Number;
    public function serialize():void
        {
            caption = escape( caption );
        }
        public function deserialize():void
        {
            caption = unescape( caption );
        }
```

其中，LocationID 与本地数据库记录有关，而 jsID 与 Yahoo ！Maps API 标记相关，caption 记录用户用户输入的有关目的地的名称．以上两个类都实现了 IDBEntry 接口中的连个 serialize()、deserialize()函数，对数据经行预处理.

5.1.2 构建数据模型

由于地图应用程序可以在没有 Internet 连接时工作，因此需要使用多个视图来表现网络资源是否可以可以访问．通过在多个显示类中注册事件处理程序，来监听对公用数据模型属性的更新事件，就可以确保数据显示的一致性.

在工作目录下创建 LocatorDataManagement．as 文件．LocationVO 和 MarkerVO 对象建立了对数据库中的数据表记录的一一映射，并将各自的属性与数据库表中的记录建立了关联，这里使用单实例模式来确保所有客户端在任何时候都使用相同的数据进行工作，LocatorDateManagement 就承担了管理这些数据的角色．同时，与当前具体地理方位和目的地数据相关联是实际地址的描述信息——如具体地理方位的实际名称，如：“ＸＸ公司”这与在地理编码中实际的地址完全不同．currentAddress 属性的值将根据用户输入的描述信

息进行更新，并用于根据在线地理信息服务或本地数据库提供的数据在地图上确定当前位置.

```
public class LocatorDataManager extends EventDispatcher
  {
      private static var _instance:LocatorDataManager = new LocatorDataManager();
      private var _currentAddress:String;
      private var _location:LocationVO;
      private var _markers:ArrayCollection;       // MarkerVO[]

      public static function getInstance():LocatorDataManager
      {
          return _instance;
      }

      public function LocatorDataManager()
      {
          if( _instance != null )
          {
              // singleton warning...
          }
      }

      public function refresh():void
      {
          currentAddress = "";
          location = new LocationVO();
          markers = new ArrayCollection();
      }

      public function addMarker( marker:MarkerVO ):void
      {
          _markers.addItem( marker );
      }

      [Bindable]
      public function get currentAddress():String
      {
          return _currentAddress;
      }
      public function set currentAddress( str:String ):void
      {
          _currentAddress = str;
```

```
            }

            [Bindable]
            public function get location():LocationVO
            {
                return _location;
            }
            public function set location( value:LocationVO ):void
            {
                _location = value;
            }
            [Bindable]
            public function get markers():ArrayCollection
            {
                return _markers;
            }
            public function set markers( arr:ArrayCollection ):void
            {
                _markers = arr;
            }
        }
    }
```

5.1.3 与数据库交互

Adobe AIR 中包含 SQL 关系数据库引擎，支持数据库文件的本地保存．DBService.as 将负责创建地图应用程序的数据库文件和必要的数据库表，同时可以通过运行查询对具体地理方位和目的地的星系进行保存和提取．

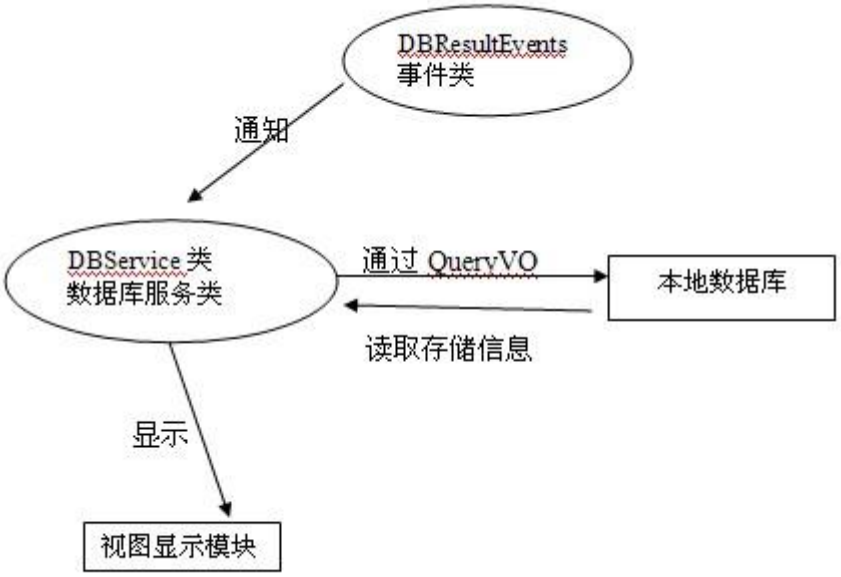

图 5-1 与数据库交互

5.1.3.1DBService 服务类：

在 DBService.as 的代码中，创建了 DBService 数据服务类，它通过运行 SQL 查询语句与数据库进行交互．DBService 类将由主应用程序类进行实例化，它将为各用户界面视图从数据库中提取具体数据的访问点．在创建 DBServic 实例的同时，将创建一个 SQLConection 实例，SQLConnection 实例负责管理与本地数据库文件的连接，并作为 sqlConnection 属性提供给 flash.data.SQLStatement 对象：

```
public function DBService()
        {
            _connection = new SQLConnection();
            _connection.addEventListener( SQLEvent.OPEN, onConnectionOpen );
            _connection.addEventListener( SQLEvent.CLOSE, onConnectionClose );
            _connection.addEventListener( SQLErrorEvent.ERROR,
onConnectionError );

            _queryQueue = new Array();
            _dbFile = File.applicationStorageDirectory.resolvePath( DB );
            invalidate();
        }
```

在 SQLConnection 被创建的同时，如果事先没有进行创建，将在应用程序的存储目录中自动创建一个名为 Locator.db 的数据库文件．与只允许读访问的应用程序目录不同，可以对存储目录同时进行读写操作，存储目录课改过 flash．Filesystem．file 类的静态属性 applicationStorageDirectory 进行访问．

在 Windows 环境下，应用程序的存储目录位于 C:\Document and Settings\<username>\Application Data\locator\local store

通过程序与数据库的连接，SQL 查询语句以 flash．Data．SQLStatement 对象的方式运行，DBService 类通过队列管理查询语句，它可以使用共有方法 addQuery（）向队列添加查询语句．

```
public function addQuery( query:QueryVO, run:Boolean = true ):void
        {
            _queryQueue.push( query );
            if( !_isRunningQuery && run ) runQuery();
        }

        public function runQuery():void
        {
            if( _queryQueue.length > 0 ) executeNextQuery();
        }
```

上述代码中，QueryVO 实例 Query 作为 addQuery（）方法的一个参数，它将代表要被添加到队列中的查询语句对象．QueryVO 对象包含三个属性：type、query 和 itemClass，这三个属性分别表示查询语句的种类，如（insert、select 等），查询语句字符串，以及结

果数据向那个类映射．SQLStatement 类具有将数据库查询的结果数据映射到一个具体类的能力，使相关的类可以无缝地提取数据库中的数据，而不必手动解析和替换属性数据．为了将数据映射到一个具体的类，要用到 SQLStatement 类的 itemCLass 属性．QueryVO 对象将引用 SQLStatement 类，用于 QueryVO 对象进行更新的查询．

```
public class QueryVO
{
    public var type:String;
    public var query:String;

    public function QueryVO( type:String, query:String )
    {
        this.type = type;
        this.query = query;
    }
}
```

所以与目的地的具体地理方位相关的查询类型，都将在 DBResultEvent 类中被罗列出来．在 SQL 查询语句的执行完成后，DBService 类使用和 QueryVO 实例的 type 属性相同的事件类型来触发 DBResultEvent 时间实例．DBResultEvent 的 Result 属性有 SQL 查询返回的数据予以赋值．

以下是一个将查询语句加入运行队列中的 retrieveAllLocations 的公有方法：

```
public function retrieveAllLocations():void
    {
        var sqlSelect:String = "SELECT DISTINCT name FROM locations;";
        addQuery( new QueryVO(DBResultEvent.LOCATION_RETRIEVED_ALL,
                              sqlSelect) );
    }
```

retrieveAllLocations（）方法将一个心得 QueryVO 对象加入队列，同时将 QueryVO 对象的 type 属性值设为 DBResultEvent 的 LOCATION_RETRIEVED_ALL 类型，并将查询语句值设为包含在 sqlSelect 中的查询语句，查询语句将罗列出本地数据库 Locator.db 中名为 Locations 的数据库表中所有那么字段（这里的数据库 Locator．Db 是前面 DBService 类进行初始化时创建的）．在接收到 SQL 查询语句查询结果时，立即触发一个心得 DBResultEvent 事件，该事件消息包含了传递给 QueryVO 类的 type 类型及查询结果．

```
private function onQueryResult( evt:SQLEvent ):void
    {
        var result:SQLResult = _statement.getResult();
        if( _currentQuery.type != FALL_THROUGH )
            dispatchEvent( new DBResultEvent( _currentQuery.type, result ) );

        executeNextQuery();
```

```
        }
```

由于 SQLite 中包含的所有查询数据类型，数据表属性的类型以及各种约束．Adobe AIR 中自带大量的 SQL 引擎．

在 DBService 类构造函数调用 invalidate（）方法，将使用 CREATE IF NOT EXISTS 查询语句在本地数据库中创建两个表（表事先未进行创建）．表格分别被命名为 locations 和 markers –用于保存应用程序获得的具体地理方位和目的地信息，并在执行后继数据库查询时使用，在共有的 saveLocation（）和 saveMarker（）方法中对数据库的更新操作是，分别使用 LocationVO 类和 MarkerVO 对象各自的 id 属性来插入一条心得记录或更新一个已有的记录．

```
    private function invalidate():void
        {
            _statement = new SQLStatement();
            _statement.sqlConnection = _connection;
            _statement.addEventListener(SQLEvent.RESULT, onCreationQueryResult);
            _statement.addEventListener(SQLErrorEvent.ERROR,
onCreationQueryError);
            var sqlCreateLoc:String = "CREATE TABLE IF NOT EXISTS locations ("
+
                                        "id INTEGER PRIMARY
KEY," +                                 "name TEXT NOT NULL,"
+
                                        "content TEXT NOT NULL);";
            var sqlCreateMark:String = "CREATE TABLE IF NOT EXISTS markers ("
+
                                        "id INTEGER PRIMARY KEY," +
                                        "jsId TEXT NOT NULL," +
                                        "locationId TEXT NOT NULL,"
+
                                        "caption TEXT NOT NULL," +
                                        "x FLOAT NOT NULL," +
                                        "y FLOAT NOT NULL);";
            addQuery( new QueryVO( CREATE_TABLE_LOC, sqlCreateLoc ), false );
            addQuery( new QueryVO( CREATE_TABLE_MARK, sqlCreateMark ),
true );
        }
```

5.1.3.2 DBResultEvent 类

DBResultEvent 类主要是当用户在在地图界面上添加标记信息时，用于监听 Save 事件和 Retrieve 事件．在很多时候，用户添加了信息，发现并不是自己所需要的，或者说并不想保存信息，所以在标记框中会显示保存按钮和取消按钮，当用户点击这两个按钮是就会触发保存事件和取消事件．

当某个事件发生时，DBResultEvent 监听到之后，通过_result 属性发送到 DBService 类，从而调用数据服务类中相对应的函数，创建相应的 QueryVO 实例.

DBResultEvent 类：

```
public class DBResultEvent extends Event
{
    private var _result:SQLResult;

    public static const LOCATION_SAVED:String = "locationSaved";
    public static const LOCATION_RETRIEVED:String = "locationRetrieved";
    public static const LOCATION_RETRIEVED_ALL:String = "locationRetrievedAll";
    public static const MARKER_SAVED:String = "markerSaved";
    public static const MARKER_RETRIEVED:String = "markerRetrieved";
    public static const MARKER_RETRIEVED_ALL:String = "markerRetrievedAll";

    public function DBResultEvent( type:String, sqlResult:SQLResult )
    {
        super( type );
        _result = sqlResult;
    }

    public function get result():SQLResult
    {
        return _result;
    }
```

5.1.3.3 QueryVO 值对象：

DBService 类和本地数据库的交互主要是通过创建一个个 QueryVO 实例来实现的：

QueryVO 类：

```
public class QueryVO
{
    public var type:String;
    public var query:String;

    public function QueryVO( type:String, query:String )
    {
        this.type = type;
        this.query = query;
    }
}
```

5.2 系统各功能模块的实现

5.2.1 在线模式和离线模式的选择

地图应用程序使用户可以根据具体地理方位和典型地标物经行导航. 如果有了 Internet 连接，可以通过在线的地理数据服务或已保存的数据来提取具体地理方位．如果没有 Internet 连接，则只能访问已保存的数据.

首先建立 ModeType 类，包含两种模式类型：在线模式和离线模式

```
public class ModeType
{
    public static const ONLINE:String = "online";//在线模式
    public static const OFFLINE:String = "offline";//离线模式
}
```

然后建立 ModeControls.mxml

ModeControls 组件允许用户通过 RadioButton 控件来选择地图应用程序在离线或在线两种模式下工作．当 mode 属性被更新时，将触发相应的时间给任何监听变化的程序模块．当成功连接到 Internet 时，用户可以选择工作模式，并相应地讲 isAvailable 属性值设置为 true．在 creationComplete 事件处理程序中，isvalidateAvailability()函数被绑定到 isAvailable 属性上．当更新 isAvailable 属性时，invlidateAvailability()方法被激活，该方法讲确定基于网络资源可用性的可用模式和用户选择的工作模式：

```
private function invalidateAvailability( args:* ):void
        {
            if( !_isAvailable && _mode == ModeType.ONLINE )
            {
                mode = ModeType.OFFLINE;
            }
            else if( !(_mode != ModeType.OFFLINE && _isAvailable ) )
            {
                mode = _isAvailable ? ModeType.ONLINE : ModeType.OFFLINE;
            }
        }
```

与在线服务两桶状态用另外的程序模块进行监控，该模块负责通知 ModeControls 组件是否发生状态的改变．ModeControls 组件是否可用由 isAvailable 属性的值以及当前状态的视图来确定．invalidateAvailability()方法确保如果用户之前是离线工作，并且网络资源的状态发生改变，监听的模块将不会得到 mode 属性的更新通知.

5.2.2 显示目的地的地理信息图形

与查看地理位置列表的方式恰好相反，地图应用程序讲通过图形的方式来表现具体的地理方位，它还允许用户标注具体地理方位．根据网络资源是否可用和用户偏好的工作模式，地图应用程序提供两种地图显示方案．如果是在线工作，将使用 HTML 控件与在线地

图 API 进行通信，并通过 HTTP 控件对地理图形进行渲染．如果是离线工作，则使用 Image 组件对保存的地理数据进行图形表示．

LocationDisplay.mxml 文件中的代码，是用于显示地理方位图形，并与 HTML 控件包含的 DOM 对象和方法进行交互的主视图类．主视图类通过 invalidateMode()方法检验描述模式的 mode 属性的值，来确定在 ViewStack 容器内选择 HTML 或 Image 控件之一进行显示．

```
private function invalidateMode( arg:* ):void
    {
        switch( _mode )
        {
            case ModeType.ONLINE:
                if( _isMapAvailable )
                    map.domWindow.showDefault();
                displayCtrl.selectedChild = onlineCtrl;
                break;
            case ModeType.OFFLINE:
                displayCtrl.selectedChild = offlineCtrl;
                if( image == null ) return;
                image.source = null;
                markerLayer.removeAllChildren();
                break;
        }
    }
```

在线工作状态下，若用户要保存当前具体地理方位数据，则将生成当前 HTML 控件中显示的地图图像快照，该快照通过 ImageFileBuffer 类的静态 bufferImage()方法被保存到应用程序的存储目录中．返回的路径被保存为具体地理方位的 content 值，LocationVO 对象可以保存 content 值．实际上，LocationVO 对象作为数据源被绑定到 Image 控件的 source 属性上．

```
private function onMapPrepared():void
    {
        if( _data.location.content != "" )
        {
            // 重写图形文件
            ImageFileBuffer.bufferImage( map,
                                    _data.location.content.replace( /.png/, '' ) );
        }
        else
        {
            // 新建图形文件
            _data.location.content = ImageFileBuffer.bufferImage( map,
```

```
                                                    ( new Date().getTime() ).toString() );
                }
                map.domWindow.reset();
                dispatchEvent( new Event( LocationDisplay.PREPARED ) );
            }
```

为了在具体地点上设置目的地标记，Flex 框架的鼠标拖放（drag-and-drop）API 被 dropCanvas 容器用来辨识在图形显示上的鼠标操作并对其作出反应．LocationDisplay 组件负责将这些目的地标记添加到地图显示界面中，并将标记的数据添加到 LocatorDataManager 单实例上．

```
public function addMarker( marker:MarkerVO ):void
        {
            _markers.addItem( marker );
        }

        [Bindable]
        public function get currentAddress():String
        {
            return _currentAddress;
        }
        public function set currentAddress( str:String ):void
        {
            _currentAddress = str;
        }
        [Bindable]
        public function get location():LocationVO
        {
            return _location;
        }
        public function set location( value:LocationVO ):void
        {
            _location = value;
        }
        [Bindable]
        public function get markers():ArrayCollection
        {
            return _markers;
        }
        public function set markers( arr:ArrayCollection ):void
        {
            _markers = arr;
        }
```

```
        }
```

HTML 控件的 Location 属性在设置 htmlLocation 属性的过程中被更新．在离线工作状态下，选定的视图将显示任何与具体方位关联的已保存过的地图．在线工作状态下，地图将通过与 Yahoo !Maps AJAX API 的通信返回．与 API 间的通信将 HTML 控件载入的 HTML 文档中包含的 JavaScript 方法完成．点号标识符（dot-notation）被用来辅助 HTML 控件的 domWindow 属性从载入网页中提取 JavaScript 的属性和方法．

当 LocationDataManager 的 currentAddress 属性被更新时，LocationDisplay 实例的 invalidateAddress()方法将被激活，同时，基于用户已选定的工作模式，与 HTML 文件对象模型（DOM）的通信也将被实施．

```
        private function invalidateAddress( arg:* ):void
                {
                    switch( _mode )
                    {
                        case ModeType.ONLINE:
                            map.domWindow.locateAddress( unescape(

_data.currentAddress ) );
                            map.domWindow.returnMarkers = onHTMLMarkers;
                            map.domWindow.returnClean = onMapPrepared;
                        break;
                        case ModeType.OFFLINE:
                            markerLayer.removeAllChildren();
                            dispatchEvent(new
Event(LocationDisplay.REQUEST_LOCATION));
                            break;
                    }
                }
```

能够在 ActionScript 和 JavaScript 之间进行桥接通信是 AIR API 提供的卓越能力．AIR 程序可以直接调用 JavaScript 方法，甚至支持使用 JavaScript 调用 ActionScript 方法．通过使用这种类型的交互功能，用户可以直接执行在 Flash 播放器下渲染的 AJAX 操作．

5.2.3 HTML 文档

在地理应用程序中，负责渲染 HTML 地理文件的内容的是 mx.controls.HTML 控件．Location 属性既可以是网络服务器提供的可访问 HTML 文档的 URL，也可以是系统文件的路径．要创建的 HTML 文档将与地图应用程序一起被打包，然后将被复制并由应用程序存储目录导入，该 HTML 文档主要用于与 Yahoo！Maps AJAX API 进行通信．

要使用 Yahoo！Maps AJAX API，需要通过注册来获得一个应用程序 ID．注册的地址是：https://developer.yahoo.com/wrsegapp/index.php ．在活得 id 之后，就要将其嵌入到自己的应用程序中去．

```
        <head>
            <style type="text/css">
```

```
                #mapContainer {
                    height: 600px;
                    width: 800px;
                }
            </style>
            <script type="text/javascript" src="http://tw.api.maps.yahoo.com/ajaxymap?v=3.8&appid=hYWJ3zHV34FfaCDFA0oImEXpkNtMhfOxL1sf9mrKVMOyzGdtmenFDiuV_IdkX7OwIFY-"></script>
    </head>
```

其中，v=3.8 是版本，appid 是注册后的 ID．因为 Yahoo!Maps API 是开放的，只要不用于商业用途，其实注册获得 ID 是比较容易的．

在 Yahoo 地图 API 中，封装了大量的地理信息，只需要向其中添加 API 控件，就可以实现地图普通地图，卫星地图，地形地图的显示．

```
ar markers = new Array();
                var map = new YMap( document.getElementById( 'mapContainer' ),

YAHOO_MAP_REG );
                map.addPanControl( )
                map.addTypeControl();
                map.addZoomLong();
                YEvent.Capture( map, EventsList.onEndGeoCode, onGeoCodeResult );
                showDefault();
```

Ymap.html 文件将被载入 LocationDisplay 组件的 HTML 控件之后，程序将会调用 Yahoo！ Maps AJAX API 来创建新的 YMap 对象，并将一个默认的地理方位集合全部放大，在线工作时，地图应用程序将通过与 YMap 控件进行交互来显示具体地理方位和目的地标记．

```
public static const MAP_HEIGHT:int = 600;
                public static const MAP_WIDTH:int = 800;
                public static const CHANGE:String = "change";
                public static const COMPLETE:String = "complete";
                public static const PREPARED:String = "prepared";
            public static const REQUEST_LOCATION:String = "requestLocation";
    <mx:Canvas width="{MAP_WIDTH}" height="{MAP_HEIGHT}">
        <mx:ViewStack id="displayCtrl"
            width="100%" height="100%">
            <mx:Canvas id="onlineCtrl"
                width="100%" height="100%">
                <mx:HTML id="map"
                    width="{MAP_WIDTH}" height="{MAP_HEIGHT}"
                    complete="onMapLoadComplete();"
                    locationChange="onMapLocationChange();"
```

```
                    />
                </mx:Canvas>
```

以上是在在 LocationDisplay 组件中加载 html 的代码，其中 MAP_HEIGHT 和 MAP_WIDTH 是显示地图的长和宽.

Ymap.html 文件中的 JavaScript 将通过多个访问点，与 Yahoo！Maps AJAX API 进行通信，本地图应用程序调用了 API 的拖动、下载地图图形以及添加和释放标记等功能.对所有这些方法和属性的访问 HTML 控件的 domWindow 属性负责实施.

```
function getAllVisibleMarkers()
        {
            var bounds = map.getBoundsLatLon();
            var arr = new Array();
            for( i = 0; i < markers.length; i++ )
            {
                updateMarker( markers[i] );
                if( inBounds( markers[i], bounds ) ) arr.push( markers[i] );
            }
            returnMarkers( arr );
        }
```

returnMakers() 方法在 JavaScript 中被激活，并由 LocationDisplay 组件的 onHTMLMarkets()方法负责处理：

```
private function onHTMLMarkers( value:Object, ...args ):void
        {
            if( value == null ) return;
            var jsMarkers:* = value;
            var markers:ArrayCollection = _data.markers;
            for( var i:int = 0; i < jsMarkers.length; i++ )
            {
                var jsMarker:Object = jsMarkers[i];
                var marker:MarkerVO;
                var xyPt:Object = jsMarker._xy;
                for( var j:int = 0; j < markers.length; j++ )
                {
                    marker = markers.getItemAt(j) as MarkerVO;
                    if( marker.jsId == jsMarker.id )
                    {
                        marker.locationId = _data.location.id;
                        marker.x = xyPt.x;
                        marker.y = xyPt.y;
                    }
                }
            }
```

```
            map.domWindow.clean();
        }
```

在上述代码中，JavaScript 的 Array 对象作为 onHTMLMakers 方法的值参数被传递．在 onHTMLMakers 方法中，LocatorDataManager 实例包含所有 MarkerVO 对象将根据 array 中 JavaScript 对象的属性进行更新．

5.2.4 对目的地添加标记信息

前面创建的 MarkerVO 类包含使用具体地理方位描述的地理信息属性，而不包含使用地理坐标描述的信息．可以将这些具体的地理方位保存在值对象中，它们是目的地的图形表示．用户还可以在图形上添加标注，甚至可以通过提取保存过的地理方位历史数据来经行图形显示．

Marker.mxml 是目的地的标记显示界面．您也许会注意到它使用了位于工作目录 assets 文件夹下的 marker_red.png 图形文件．用于标记目的地．

图 5-2 用于做标记的图形文件

Marker 对象使用类型为 MarkerVO 的对象实例，并在鼠标移动到其上方时，以类似 tooltip 的工具栏提示的方式显示值对象相应的 caption 标题属性．Caption 标题值的复制操作在 MarkerForm 窗体中进行．MarkerForm 提供两个选择来添加标记的标题．当用户选择保存标记的标题是，MarkerVO 值对象将被更新，并且界面将继续显示标记实例．如果取消保存，则标记将从显示界面移除．

```
<mx:VBox width="100%" height="100%"
        paddingLeft="5" paddingRight="5" paddingTop="5" paddingBottom="5">
        <mx:Label text="添加标记信息:" />
        <mx:TextArea id="captionField"
            width="100%" height="100%"
            change="{_markerData.caption = captionField.text}"
            />
        <mx:HBox width="100%" height="30" horizontalAlign="center">
            <mx:Button id="saveButton"
                label="保存" click="dispatchEvent( new Event( SAVE ) );"
                />
            <mx:Button id="cancelButton"
                label="取消" click="dispatchEvent( new Event( CANCEL ) );"
                />
        </mx:HBox>
    </mx:VBox>
```

```
</mx:Panel>
```

当用户将标记图标通过拖拽的方式添加到显示目的地主要地理信息的时候，程序就会自动跳出标记框．用户将可以标记框中添加标记信息并保存．MarkerForm 窗体同时还通过 JavaScript 脚本和 flash.events.Event 事件来实现保存和取消事件的监听．以下代码正是实现了事件监听：

```
<mx:Metadata>
        [Event(name="save", type="flash.events.Event")]
        [Event(name="cancel", type="flash.events.Event")]
    </mx:Metadata>
    <mx:Script>
        <![CDATA[
            import com.aircmr.locator.data.MarkerVO;
            [Bindable]
            private var _markerData:MarkerVO;
            public static const SAVE:String = "save";
            public static const CANCEL:String = "cancel";
            private function onCreationComplete():void
            {
                stage.focus = captionField;
            }
            override public function get data():Object
            {
                return _markerData;
            }
            override public function set data(value:Object):void
            {
                _markerData = ( value as MarkerVO );
                super.data = value;
            }
        ]]>
    </mx:Script>
```

5.2.5 搜索具体地点

之前我创建了用于显示具体地理方位和目的地的相关模型和视图类．到目前为止，地图应用程序仍缺少用于导航和保存这些地理方位和目的地的用户控件．这些控件同样也基于应用程序的当前工作模式——在线或离线．

LocationControls 类中包含了用于地理方位导航和添加目的地标记的控件．根据当前应用程序正在运行的模式，用户可以输入或选择一个具体的地理地点，然后将请求发送给 Yahoo！Maps AJAX API 或本地数据库．如果是在线工作模式，将显示一个 input 输入栏和 Sumit 按钮用于输入；如果是离线工作模式，将显示一个 ComboBox 控件和一个位于本地数据库包含的具体地理方位的列表．

具体由 LocationControls 中的 <mx:ViewStack id="locationCtrl"/> 和 <mx:Canvas id="offlineCtrl" />实现．LocationControls 组件的主要功能是触发用户交互相关事件，并通知其他监听程序模块．

LocationControls 组件负责发送地址请求，在线时，这个地址来源于 addressField 的 text 属性或 locationCB 实例的 selectedLabel 属性．

离线时，LocationControls 组件触发事件，请求活得在本地数据库中保存的所有具体地点．

在 Save 按钮的 Click 事件处理程序中，LocationControls 类触发 save 事件，提示用户将保存当前地理方位和目的地的信息．

```
private function onLocationSave():void
        {
            // if we have changed the name, mark for new entry...
            if( saveField.text != _data.location.name )
            {
                _data.location.id = -1;
                _data.location.name = saveField.text;
            }
            dispatchEvent( new Event( LocationControls.SAVE ) );
        }
```

因为数据库中已保存的地理方位具有唯一正整数类型 ID ，所以若当前 LocationVO 对象中 id 的属性值为-1，则表明 LocationVO 对象对应的地理方位数据之前没有进行本地存储．onLocationSave 处理器中的 if 语句检查用户是否正在使用数据库中的数据，并且是否已选择一个与数据相关联的名字－－这用于提示其他程序模块；谁将处理与数据库的通信来添加新的记录，而不是覆盖任何存在的数据．

MarkerBank 组件负责显示 Marker 标记实例，它支持在具有拖放能力的显示对象上对标记进行拖放操作－－在本应用程序实例中，显示对象是 LocationDisplay 组件．尽管本实例中 MarkerBank 组件当前智能显示一个使用通用图片的标记，但它可以通过更新从而具有能够处理更多类型标记的能力，其他类型的标记可以根据目的地类型采用不同的图片表示－－例如餐厅位置或者旅馆位置.

MarkerBank：

```
<mx:Script>
        <![CDATA[
            import com.aircmr.locator.data.MarkerVO;

            import mx.containers.Canvas;
            import mx.core.DragSource;
            import mx.managers.DragManager;

            private function onMarkerStart( evt:MouseEvent ):void
```

```
                {
                    var pt:Point = marker.localToGlobal( new Point( evt.localX,

evt.localY ) );
                    var source:DragSource = new DragSource();
                    source.addData( new MarkerVO(), "marker" );
                    var bmp:BitmapData = new BitmapData( marker.width,
marker.height );
                    bmp.draw( marker );
                    DragManager.doDrag( this, source, evt, marker,
                                                    pt.x - 5, evt.localY - marker.height );
                }
            ]]>
        </mx:Script>

        <mx:Label text="点击并拖拽标记地图:" />
        <ui:Marker id="marker" mouseDown="onMarkerStart(event);" />
```

在成功构建了值对象和提供地理方位及目的地交互所使用的视图类之后，将创建主程序文件，将负责与本地数据库进行通信，并处理由视图类触发的事件，同时该文件还负责与数据库服务类的交互.

5.2.6 模块组合

在主程序文件的基本布局是水平并排显示 LocationControls 和 LocationDisplay 两个视图，而 ModeControls 视图则位于显示列表的垂直下方．这些视图和组件上所有被触发的事件将由 Locator 处理，以更新数据模型．Locator 的事件处理还负责向“数据库查询”小节中创建的数据库服务类发送数据请求.

在应用程序完成初始化工作并被添加到显示视图列表后， onAppComplete() 事件处理程序函数将创建 DBService 的实例．如果数据库当前不存在的话，DBService 将在应用程序的存储中创建一个本地数据库．同时将原来位于应用程序资源文件夹下的 ymap．html 文件复制到应用程序存储目录中：

```
private function checkHTMLFile():void
                {
                    var tFile:File = File.applicationDirectory.resolvePath( MAP_HTML );
                    tFile.copyTo(
                       File.applicationStorageDirectory.resolvePath( MAP_HTML ), true );

                    locationDisplay.htmlLocation = "app-storage:/" + MAP_HTML;
                }
```

在进行上述操作的同时，Locator 还将 isAvailabilable 属性与一个负责更新 ModeControl s 实例属性的函数进行绑定．ModeControls 类允许用户在具有 I nternet 连接的时候选择离线或在线工作模式．在没有 Internet 连接的状态下，则限制用户只能访问离线数据．这种

限制依赖于对 ModeControls 实例的 isAvailable 属性更新，isAvailable 属性的值由如下 networkChange 事件的 checkConnection() 事件处理程序负责更新，而 networkChange 事件则在<mx:Windowed Application> 应用程序根标记中被声明：

```
private function checkConnection():void
        {
            modeControl.mode = ModeType.OFFLINE;
            var request:URLRequest = new URLRequest();
            request.method = URLRequestMethod.HEAD;
            request.url = CONN_URL;
            var loader:URLLoader = new URLLoader();
            loader.addEventListener( HTTPStatusEvent.HTTP_STATUS,

onHTTPStatus );
            loader.addEventListener( IOErrorEvent.IO_ERROR, onStatusError );
            loader.addEventListener( SecurityErrorEvent.SECURITY_ERROR,

onStatusError );
            loader.load( request );
        }
```

上述代码首先发出了一个基于 HTTP 的 HEAD 请求，用于确定 Internet 连接对用程序是否可用（如果返回的状态代码是 HTTP__STATUS 小于 400，则说明网络可用）. HTTP__STATUS 值在 onHTTPStatus() 事件处理程序中被检测，，如果小于 400，则将 isAvailable 属性设为 true；如果大于等于 400， 则设为 false . 如果请求发错误，则 onStatusError() 事件处理程序直接将 isAvailable 属性设为 false . isAvailable 属性对 ModeControls 组件的 mode 属性进行驱动，并反过来只是其他组件的显示状态.

```
private function onHTTPStatus( evt:HTTPStatusEvent ):void
        {
            isAvailable = evt.status < 400;
        }
        private function onStatusError( evt:IOErrorEvent ):void
        {
            isAvailable = false;
        }
```

于本应用程序而言，Locator 只对于网络连接是否可用进行一次检测，而不是连续的检测其可用性.

除了对网络的变化进行检测外，Locator 还负责处理由 LocationControls 和 LocationDisplay 组件发出的大部分事件，这两个组件需要与 DBService 数据库服务类进行通信. 尽管大部分对 DBService 的调用只是一些对返回结果进行处理的简单请求，但需要对 DBService 中涉及如何保存具体地理方位和地理表示的调用做深入分析. 当

LocationControls 组件的 save 事件被触发时，程序通知 LocationDisplay 组件运行相应的操作进行保存操作．此时，saveLocation() 方法被激活．

```
public function saveLocation():void
        {
            _service.addEventListener( DBResultEvent.LOCATION_SAVED,
onLocationSaved );
            _service.saveLocation( LocatorDataManager.getInstance().location );
        }

private function onLocationSaved( evt:DBResultEvent ):void
        {
            LocatorDataManager.getInstance().location.id =
                                                    evt.result.data[0].id;
            _service.addEventListener( DBResultEvent.MARKER_SAVED,
saveNextMarker );
            _markerIndex = 0;
            saveNextMarker();
        }
```

上述代码中，当前 LocationDataManager 中包含的 LocationVO 对象被传递给 DBService 实例，用以向数据库中添加或更新方位记录．当记录添加成功后，onLocationSaved 方法被激活，它将根据返回的 result.data 结果数据更新 LocationVO 对象的 id 属性．由于数据库中每个具体地理方位的记录具有唯一的 ID，因此可以从 SQLResult 对象中提取出 ID，并根据 ID 更新地理方位当前的属性．在 onLocationSaved() 方法中，saveNextMarker() 方法被调用，用来想 DBService 实例 _service 发送 LocationDataManager 中包含的各个 MarkerVO 标记对象值：

```
private function saveNextMarker( evt:DBResultEvent = null ):void
        {
            var markers:ArrayCollection =
                                    LocatorDataManager.getInstance().markers;
            if( evt != null )
            {
                markers.getItemAt( _markerIndex ).id = evt.result.data[0].id;
                _markerIndex++;
            }
            if( _markerIndex < markers.length )
            {
                markers.getItemAt( _markerIndex ).locationId =
                                    LocatorDataManager.getInstance().location.id;
                _service.saveMarker( markers.getItemAt( _markerIndex )
```

```
                                                                    as
MarkerVO );
                        }
                        else
                        {

_service.removeEventListener( DBResultEvent.MARKER_SAVED,

saveNextMarker );
                            hideProgress();
                        }
                    }
```

在所有标记被成功地加入数据库后，包含在 LocationDataManager 内的每个 MarkerVO 对象的 id 属性将根据 DBResultEvent 事件返回的结果数据进行跟新．就像对地理方位 ID 的更新需要检验 SQLResult 返回的 ID 一样，对标记 ID 更新与否也需要通过检验已输入本地数据库 marks 表中每个标记的 ID 来确定．如果具体地理方位和标记的 ID 在输入应用程序之后没有被更新，那么 DBService 类将视这些地理方位和标记实例为新的记录，会立即复制这些地理方位和标记的数据并写入数据库中．

```
        public function MarkerVO( id:int = -1,
                                  locationId:int = -1,
                                  jsId:String = "",
                                  caption:String = "",
                                  xpos:Number = 0, ypos:Number = 0 )
            {
                this.id = id;
                this.locationId = locationId;
                this.jsId = jsId;
                this.caption = caption;
                this.x = xpos;
                this.y = ypos;
            }
```

以上是将前面几节中各个功能模块的进行整合，下面的代码则是通过 flex 的页面布局对系统的三种视图进行整合，完成最终用户界面：

```
    <mx:VBox width="100%" height="100%">
            <mx:HBox width="100%" height="100%">
                <ui:LocationControls id="locationControl"
                    width="300"
                    styleName="displayHolder"
                    requestAddress="onAddressRequest();"
                    requestLocations="onLocationsRequest();"
                    save="onSaveLocation();"
```

```
            />
        <ui:LocationDisplay id="locationDisplay"
            width="100%" height="100%"
            change="onDisplayChange();"
            complete="onDisplayComplete();"
            prepared="saveLocation();"
            requestLocation="onLocationRequest();"
            styleName="displayHolder"
            />
    </mx:HBox>
    <ui:ModeControls id="modeControl"
        width="100%" height="34"
        styleName="displayHolder"
        modeChanged="onModeChanged();"
        />
</mx:VBox>
```

进行完以上各个部分后，最终实现的用户界面如下：

在线工作的界面：

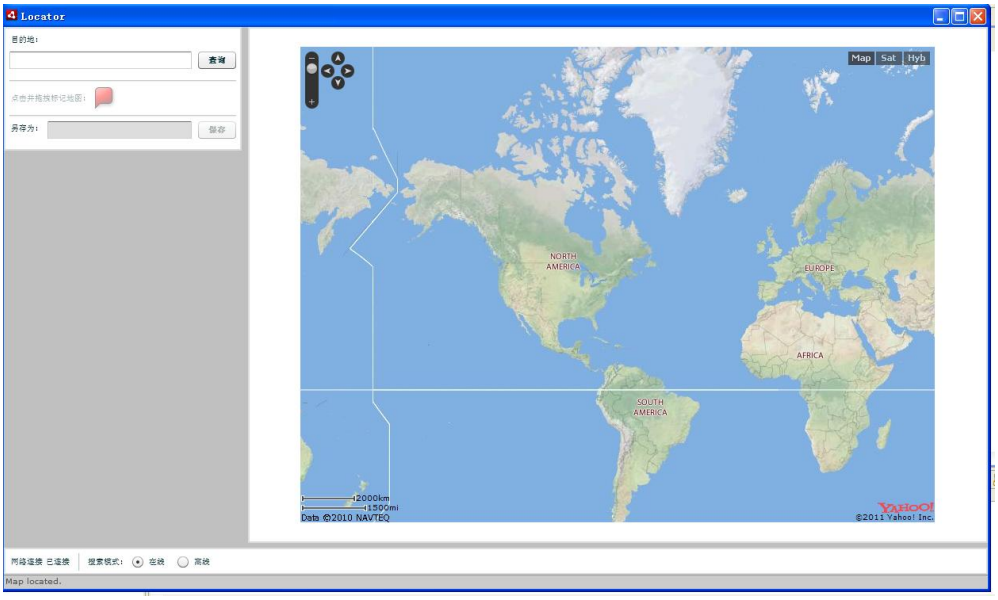

图 5-3 在线用户界面图

离线时的工作界面：

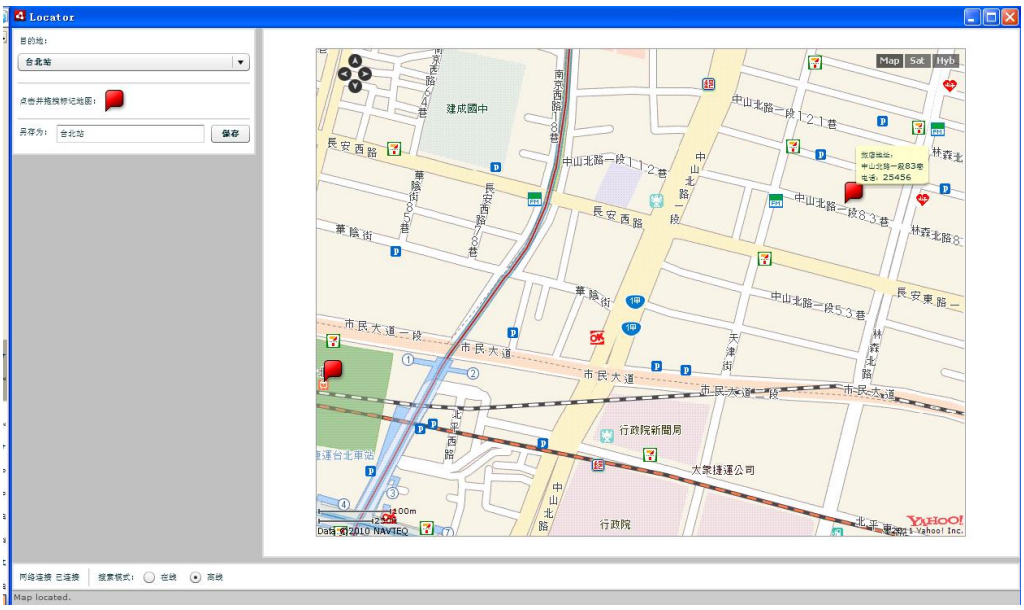
Locator
目的地：
台北站
点击并拖拽标记地图：
另存为：
台北站
保存
Map
Sat
Hyb
建成國中
長安西路
中山北路一段112巷
中山北路一段121巷
中山北路二段83巷
林森北路
華陰街
市民大道二段
市民大道
中山北路二段53巷
長安東路
天津街
北平西路
行政院新聞局
太業捷運公司
中山北路
行政院
YAHOO!
Data © 2010 NAVTEQ
© 2011 Yahoo! Inc.
网络连接 已连接
搜索模式：
在线
离线
Map located.

图5-4 离线时用户界面

第六章 部署应用程序

前面已经完成了地图应用程序的构建工作，现在对程序经行部署.

6.1 描述文件

Locator-app.xml 文件为地图应用程序的描述文件. 该文件中的代码段中的元素值在一定程度上是通用的，在实际使用时需要根据情况做相应的修改. 在安装 Locator 应用程序之后，应用程序目录将被置于在操作系统的程序目录下的 AIRCMR 文件夹中. 当数据库被创建后，yam.html 文件被复制到应用程序的存储目录中，两者都可以在 com.aircmr.Locator 文件夹中被找到，com.aircmr.Locator 对应与描述文件 id 元素的值.

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://ns.adobe.com/air/application/1.5">
    <id>com.aircmr.Locator</id>
    <filename>Locator</filename>
    <version>0.1</version>
    <name>Locator</name>
    <description>A destinations-based map appplication</description>
    <copyright>2007</copyright>
      <initialWindow>
        <title>Locator</title>
        <content>Locator.swf</content>
        <systemChrome>standard</systemChrome>
        <transparent>false</transparent>
        <visible>true</visible>
        <width>760</width>
        <height>540</height>
      </initialWindow>
    <installFolder>AIRDeveloper/Locator</installFolder>
    <programMenuFolder>AIRDeveloper</programMenuFolder>
</application>
```

6.2 编译和打包

通常，打包一个 AIR 应用程序时需要两个文件：主 SWF 文件和 XML 应用程序描述文件.

打开命令提示符，进入到应用程序的工作目录下，输入下面的命令：

```
> amxmlc Locator.mxml
```

在工作目录下创建一个自签名的数字证书：

```
> adt -certificate -cn Locator 1024-RSA certificate.pfx password
```

在保持命令提示符值向工作目录的前提下，输入以下命令：

```
> adt -package -storetype pkcs12 -keystore certificate.pfx Locator.air Locatorapp.
xml Locator.swf ymap.html
```

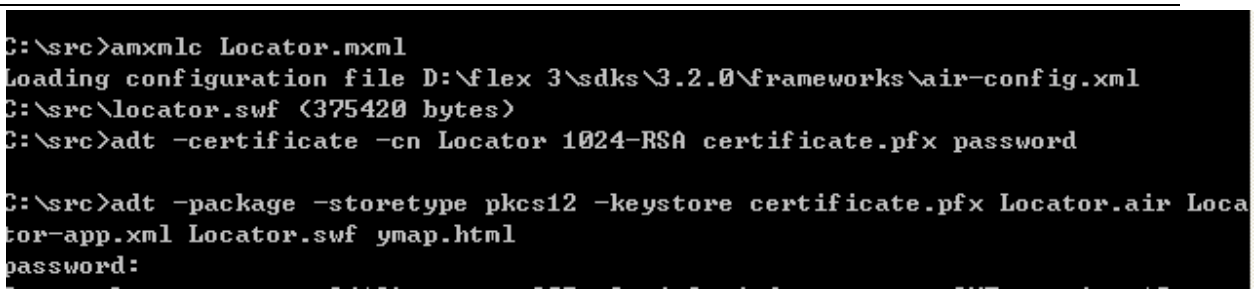

图 6-1 编译打包

输入在第二条命令的Password，在运行了这些命令后，工作目录下将会产生Locator.swf文件、certificate.pfx 文件和 Locator.air 安装文件.

当成功创建了安装文件后，进入工作目录，双击 AIR 安装文件就可以安装并运行 Locator 应用程序了.

当应用程序已正确显示与上面类似的两个界面时，程序后台就已创建了一个本地的数据库及其相应的数据表，该数据表位于在应用程序的存储目录中．在 Windows 环境下，存储目录的位置是 C:\Documents and Settings\<username>\Application Data\com.apolloir.Locator\LocalStore..

如果执行了一次具体地理方位的保存操作，则将在存储目录中将创建一个图片文件，图片文件使用文件创建事件的时间戳字符串进行命名.

第 7 章 结论与展望

7.1 结论

通过以上工作，基本实现了地图应用程序搜索目的地，添加标记和在离线情况下进行数据查询的功能．本程序支持 Yahoo！Maps API 和本地数据库进行交互．在构建程序过程中我对 AIR 包含的数据库引擎工作原理进行了研究，并对 HTML 控件和 ActionScript 与 JavaScript 之间的脚本桥接有了一定的了解．

7.2 不足之处及展望

但是程序并不是很智能化，还存在着很多不足，且仅用到了 Yahoo！Maps API 中很少的一部分，还有很大的扩展空间．随着 AIR 等新技术的出现，以及各个在线地理服务 API 的不断完善，以及大量程序开发者的不断努力，相信不久后，地图应用程序将得到更广更好的应用．

虽然我现在从事的工作与 flex 和地图不是很有关系，但是通过以上工作，我对新技术的接触，相信对我人生还是很有帮助的．

参考文献

[1] Marc Leuchner, Todd Anderson, Matthew Wright． Adobe AIR 范例精解:创建-修改-重用[M]．北京：清华大学出版社; 2009：293-342：

[2] 郭少瑞， 张鑫． Adobe AIR 权威指南 [M]． 北京：人民邮电出版社，2009：194-245．

[3] 张海藩．软件工程导论（第 4 版）[M]． 北京：清华大学出版社，2007： 25-43，52-54．

[4] 黄曦． 完全手册——Flex3．0 RIA 开发详解：基于 ActionScript3．0 实现[M]．北京：电子工业出版社，2008：113-120．．

[5] 董龙飞，肖娜． Adobe Flex 大师之路[M]．北京：电子工业出版社，2009：356-387．

[6] 井中月．Adobe AIR 应用开发实践[M]． 北京：人民邮电出版社，2008： 63-70．

[7] Joshua Noble，Todd Anderson．Flex 3 Cookbook[M]．Sebastopol：O'Reilly, 2008:189-204

[8] Greg Goralski, LordAlex Leon．Flex for Designers[M]．New York：Friends of ED, 2008:268-279．

[9] John Crosby．Creating Mashups with Adobe Flex and AIR [M]． America: SPRINGER A PR TRADE, 2008: 100-250．

[10] Dragos Georgita． Adobe AIR for JavaScript Developers[M]． America：O'REILLY & ASSOC INC，2008：54-81．

致　谢

本论文是在夏鸿斌老师的悉心指导下完成的，从论文选题、开发平台的选择、系统的设计、论文写作、初稿修改、直至最后定稿，夏老师都给了我很多的指导和帮助，在此我要向夏老师说声“谢谢，老师辛苦了！”，还有要感谢在这次设计中帮助过我的老师和同学们.