基于 Google Map API 的图书查询系统的设计与实现

李慧
（北京邮电大学网络技术研究院，北京，100876）

摘要：GIS（Geographic Information System）以地理空间数据位操作对象，近年来得到了广泛地发展，但是，现有的 GIS 系统仅仅是基于室外环境或室内环境的，而没有将二者结合起来，此外，利用现有图书馆系统查询图书，系统返回的信息中没有图书的位置信息，用户需要根据索书号查询图书，比较耗时费力。本文将地图信息和图书信息相结合，基于 Google Map API，实现了一种图书查询系统，该系统将图书馆室内地图和室外地图无缝地结合到一起，并将图书查询结果在地图上显示出来，方便用户快速地找到图书。

关键词： GIS；Google Map API；图书查询
中图分类号：TP311

Design and Implementation of Book Searching System Base on Google Map API

Li Hui
(Institute of Network Technology,Beijing University of Posts and Telecommunications, Beijing 100876)

Abstract: Geographic Information System (GIS) has a fast development in recent years. But the current geographic systems are based only outdoor environment or indoor environment respectively. There is not a way to integrate them. Moreover, it is time costing to find the book's location with current library system, because the querying book results are only some text information, not including book's location information. To cope with these problems, this paper implemented a book searching system. The system integrated the library indoor maps and outdoor maps together based on Google Map APIs. The book querying results will show on the map, which will help users find the book more quickly.

Key words: GIS; Google Map API; Book Search

0 引言

随着 web 2.0 与 GIS 的发展，与地图相关的应用得到了快速发展，Yahoo！、Google 等公司先后推出了自己的网络地图服务，给用户带来了极大的方便。Google 公司于 2005 年推出了 Google Map API，开发者可以使用 JavaScript 技术方便地将地图嵌入到自己的网页中。但是，现有的地图系统都是基于单一的室外环境或室内环境，而没有一种方式将室外环境的地图和室内环境的地图结合起来。此外，根据现有的图书馆系统进行图书查询，返回的图书信息只是一些文本信息的描述，用户需要根据图书的索书号，找到相应得书架，再从书架一堆书中找到图书，查找图书的过程比较耗费时间。

为了解决以上的问题，本文基于 Google Map API，将图书馆室内地图和室外地图无缝地结合起来，并在图书馆数据库中添加了图书的位置信息，将图书的查询结果在地图上显示出来，方便读者快速地找到图书。

作者简介：李慧（1985-），女，硕士研究生，计算机科学与技术. E-mail: 13426074379@139.com

1 背景和相关工作

1.1 Google Map API

2005 年 6 月，Google 推出了自己的 API 让开发者在地图上做开发，开发者只需要使用 JavaScript 脚本语言就可以轻轻松松地将 Google Map 嵌入到自己的网页中，并且和数据库结合起来，创建自己的应用程序，以发挥更大的作用。在 Google API 使用上有些法律上的限制[1]，主要有：(1)Google Maps 服务必须免费开放给一般使用者；(2)如果地图服务的浏览量每天超过 50，000 页面时，就要主动通知 Google；(3)Google 会不定期的更新 API，开发者有责任Google更新API时更新自己网页中的API;(4)开发者不能更改或者隐藏Google的logo等等。

Google Map API[2]按其功能主要可以分为两部分，一部分为地图显示功能，另一部分一部分为 API 扩展功能，地图显示功能的 API 如 GMap2、GPoint、GIcon、GLatLng 等，如果想开发自己的控件、标注和地图类型，则使用 API 扩展功能的类，这些包括：GControl、GMapPane、GMapType、GOverlay 等。

1.2 Google 地图模型

通过对 Google Map 的研究，它使用地图预先生成技术[3]，以金字塔的形式来组织地图瓦片的，这些金字塔样式的瓦片存放在服务器端，当移动或缩放地图时，只需要下载新的图片来填充新的区域，Google 地图是由多个小地图瓦片拼接而成，每一个地图瓦片都是一个大小为 256*256 的图片，可以从 Google Map Server 上获取到这些地图瓦片，获取地图瓦片的 URL 类似于 http://mt3.google.cn/vt/v=w2.119&hl=zh-CN&gl=cn&x=1726391&y=794209&z=21， x 和 y 代表瓦片的坐标， z 代表缩放级别，Google 以金字塔形式组织瓦片，也就是说每放大一个级别，之前的一个瓦片会裂变成四个瓦片。每一个级别的瓦片个数为 4^{level} 个，level 为缩放级别。地图瓦片的裂变示意图[4]，如下图所示。

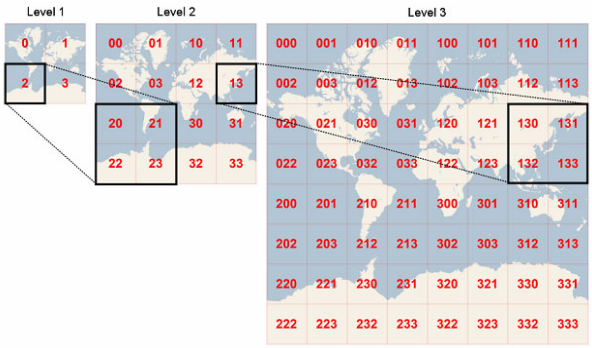

图 1 Google 地图瓦片裂变示意图

Figure1 Google Map Tiles Fission Figure

本文用到了 Google 地图中 Ground Resolution[5]概念，即地面分辨率，主要是关注一个像素（pixel）代表的地面尺寸（米），地面分辨率取决于两个参数，缩放级别 level 和纬度 latitude，缩放级别决定像素的多少，纬度决定地面距离的长短，地面分辨率的单位为：米/像素（m/pixel），计算的公式为：ground resolution=(cos(lat*pi/180)*2*pi*6378137)/(256*2^{level})(m/pixel)。

2 系统总体设计

系统的架构为 B/S 结构，客户端完全通过 web 浏览器实现，在客户端一般没有应用程序，主要事务逻辑都是在服务器端实现，应用起来方便简单，系统按层次一共可分为三层：用户表示层、业务逻辑层以及数据存储层。

用户表示层基于浏览器，是客户端的用户界面，为用户提供丰富美观交互界面，并实现用户和整个系统的交互，使用 Html 和 JavaScript 技术显示图书查询界面和图书查询结果显示界面，浏览器与 web 服务器端的交互通过 http 协议，并且当拖拽地图或对地图进行放大缩小操作时，使用 Ajax 技术从 web 服务器获取地图瓦片，而不用重新刷新界面。

业务逻辑层，是整个系统层次的关键所在，它负责解析来自表示层的用户请求，如果需要还负责调用数据库的信息，并将处理的结果返回给表示层呈现给用户。业务逻辑层包括 Google Map Server 以及用 tomcat 搭建的 web Server，Google Map Server 主要提供室外地图的瓦片，web server 主要提供二种功能：图书查询功能和地图管理功能。本文基于 Google Map API，扩展了 Google Map 的缩放级别，当在 0-18 级别时，显示的是室外的地图，在 19-22 级别上，将图书馆室内的地图显示出来，并且级别越大，地图会越来越清晰。

数据存储层主要包括二种形式的数据，一种是地图瓦片信息，每个瓦片都是 256*256 大小的图片，地图瓦片信息保存在文件目录下；第二种信息为图书信息，本文扩展了现有图书馆的数据库，在原有数据库的基础上添加了图书的位置信息，使用 Sybase 数据库存储图书信息。

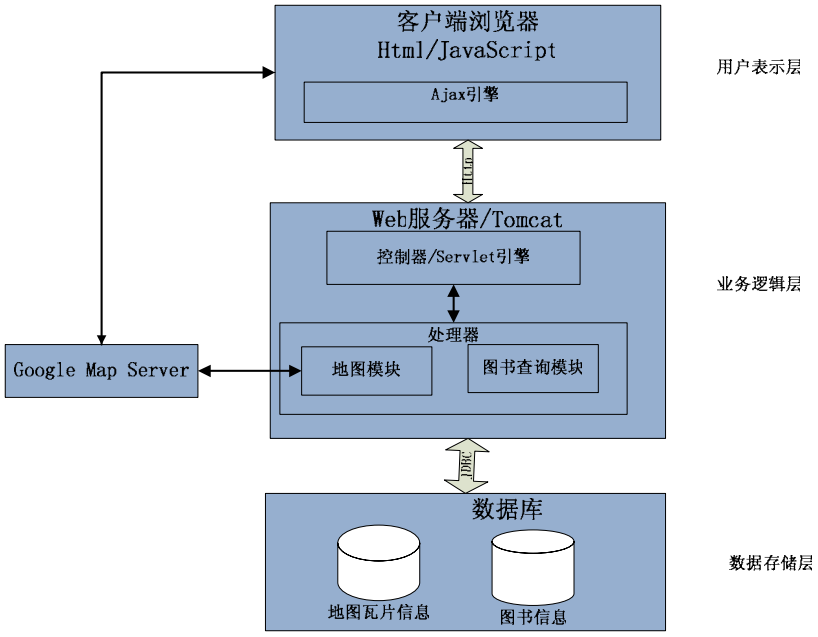

图 2 系统总体框架

Figure 2 System Architecture

3 数据库设计

3.1 地图瓦片信息

与 Google 室外地图一样，图书馆室内的地图由数个 256*256 的地图瓦片拼接而成，室内地图瓦片是基于图书馆基建图 CAD 图制作的，制作步骤如下：

1. 计算 19-22 级别覆盖图书馆范围的瓦片坐标范围，不同级别上覆盖图书馆范围的瓦片个数是不同的。根据瓦片坐标构造 URL，向 Google 服务器请求并下载瓦片，将 19-22 级别的地图瓦片拼接起来。
2. 根据 ground resolution 公式计算北京邮电大学图书馆所在纬度（39.96178611889）的 19-22 级别的地面分辨率。
3. 根据图书馆 CAD 图，得到图书馆的长宽尺寸，再根据地面分辨率的大小，计算 19-22 级别上图书馆所占的像素大小，按此像素值，在 CDA 图基础上，获得 19-22 级别的图书馆室内的地图图片。
4. 将不同级别图书馆图片粘贴到相应级别拼接好的图片上，再将图片切割成 256×256 大小的瓦片，同样以瓦片坐标和缩放级别值为瓦片进行命名，例如某一个瓦片的命名为：< x=198554&y=431603&z=19.png >，其中 x, y, z 分别代表瓦片的坐标以及缩放级别。将地图瓦片存放在文件目录，文件目录的组织结构为 images_tiles/zoom_level/tile_picture，按照这种方式组织文件目录，可以方便程序快速地找到所请求瓦片的位置。

表 1 北京邮电大学图书馆在 19-22 级别所占地图瓦片的坐标范围及地面分辨率

Tab.1 the ranges of indoor map tiles coordinates and the ground resolution value

级别	瓦片横坐标范围	瓦片纵坐标范围	瓦片个数	地面分辨率
19	431602 - 431603	198554 - 198556	6	0.2288518713921921
20	863204 - 863207	397108 - 397112	20	0.11442593569609605
21	1726408 - 1726414	794217 - 794224	56	0.057212967848048024
22	3452817-3452826	1588435-1588448	140	0.028606483924024012

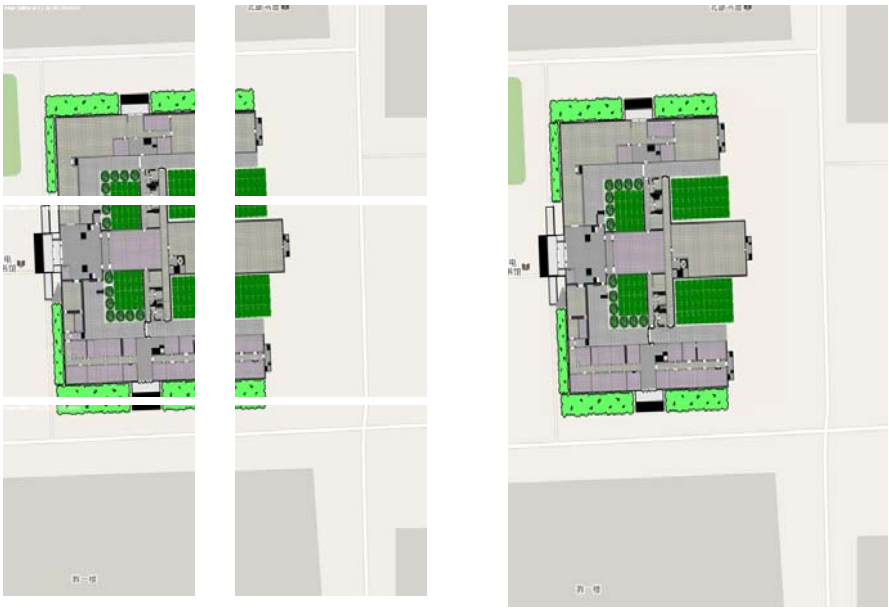

图 3 19 级别图书馆室内地图瓦片分割及拼接效果图

Figure 3 Library Indoor map tiles in 19 level

3.2 图书信息

图书信息数据库中的图书信息来源于北京邮电大学图书馆馆藏图书信息，并在现有图书

馆数据库的基础上，增加了图书的位置信息。图书馆数据库中一共包含了 5 个实体，实体间的关系如下图所示。

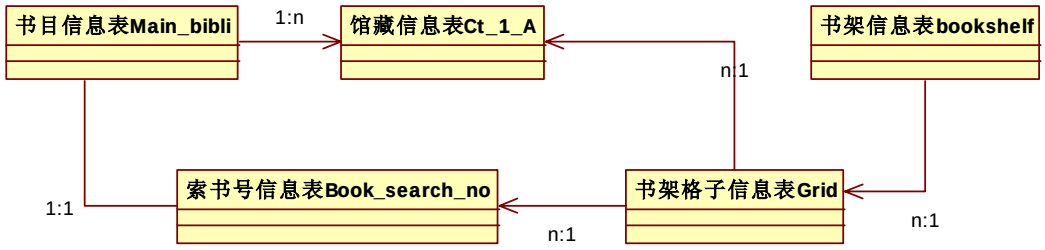

图 4 图书信息数据库实体关系图

Figure 4 The Relationship of Book Tables in Database

书目信息表 main_bibli、馆藏信息表 ct_1_A、索书号信息 book_search_no 来自于北京邮电大学图书馆信息数据库，书架信息表 bookshelf 和书架格子信息表为自定义的表，主要存放图书的位置。一个书架分两侧，每一侧由很多个格子组成，每个格子的位置可以用在书架的行（层）号和列号唯一标识，一本图书的位置可以精确到所在的格子，构建 bookshelf（书架）表和 grid（书架格子）表来描述图书的位置。

表 2 书架信息表

Tab.2 Bookshelf Table

属性名	变量类型	可否为空	是否主键	描述
id	Char(8)	Not null	是	书架标识
floorz	int	Not null	否	书架所在楼层
cordx	float	Not null	否	书架的经度值
cordy	float	Not null	否	书架纬度值

表 3 书架格子信息表(grid)

Tab.3 Grid Table

属性名	变量类型	可否为空	是否主键	描述
id	Int	Not null	是	书架格子标识
sn1	Varchar(24)	Not null	否	格子第一本书的索书号
Sn2	float	Not null	否	格子最后一本书的索书号
rowx	int	Not null	否	格子在书架的行（层）号
coly	int	Not null	否	格子在书架的列号
shelf	Char(8)	Not null	否	格子所在书架号，表的外键
side	Char(2)	Not null	否	格子在书架的侧位号

4 系统设计与实现

系统主要实现了图书查询功能、地图功能。图书馆查询包含出版社查询、书名查询、作者查询、图书条码查询等查询方式，地图功能包括地图显示和地图瓦片获取。地图显示功能通过调用地图放大缩小模块、地图类型重写模块以及地图控件重写模块来完成。

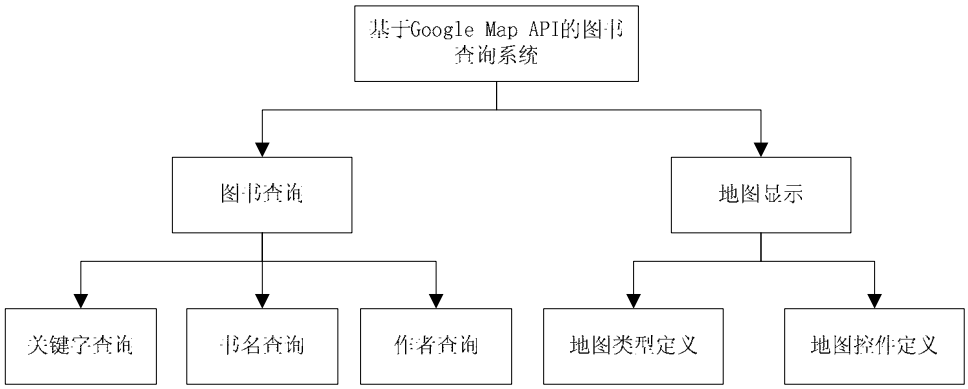

图 5 系统功能图

Figure 5 System Functions

4.1 图书查询

由于图书查询系统在现阶段已经发展地比较成熟，本文借鉴了现有的图书馆查询系统使用的技术，因此对此部分简单介绍。本文采用 MVC 模式实现图书查询功能[6]，MVC 模式也就是模型-视图-控制器模式，在 MVC 模式中，实现 web 应用系统中的各部分需要不同的类型的技巧和工具，视图负责产生返回客户端 HTML 页面，它是系统的视图组件，JSP、JavaScript 技术常用在视图的构建中，模型是最终满足客户端需要的程序代码，JavaBeans, EJB, JDBC 等技术用于模型的构造中，控制器的首要功能是整合业务逻辑元素，然后委托页面构建组件，生成客户端的响应页面。

从逻辑上来说，服务器端包括控制器，处理器和数据库三方面，控制器用来接收用户的请求，并将请求信息传递给处理器，处理器根据数据库信息将结果计算出来，并将计算结果返回给控制器，控制器对结果数据进行组织，返回给客户端。

在图书查询功能中，控制器获取客户端请求中图书查询的关键词，图书查询功能支持多种查询方式，比如出版社查询、书名查询以及作者查询等，并支持多种匹配方式，比如前向匹配，模糊匹配和精确匹配，控制器从请求中提取出查询条件和关键词，将之传递给处理器。处理器接收到查询条件和关键词后，使用 JDBC 技术访问数据库，处理器从数据库中按条件查询出图书信息，返回给控制器，控制器将信息进行组织，根据图书的位置将图书信息标注地图上，并将标有图书信息的地图返回给客户端。

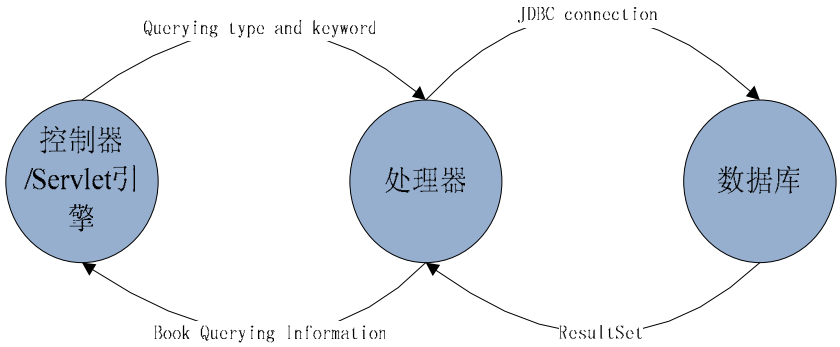

图 6 控制器、处理器、数据库信息交互图

Figure 6 Data Exchanges among Controller, Processor and Database

4.2 地图显示

地图显示功能主要是将图书馆室内地图和室外地图无缝地融合起来，用户可以拖拽，放

大缩小地图，当放大到一定级别时，显示出室内地图。确切地说，当 0-18 级别时，显示室外的地图，当 19-22 级别时，则呈现出图书馆室内地图。地图显示功能包含二个子模块：定义地图类型模块和定义地图控件模块。

地图类型定义主要功能是重新定义新的地图类型，将图书馆室内地图和室外地图无缝结合起来，使用 JavaScript 调用 Google Map API，重写 GMapType[7]类，代码在客户端浏览器内运行。

GMapType 类的构造函数为 GMapType(layers:GTileLayer[], projection:GProjection, name:String, opts?:GMapTypeOptions)，一共有 3 个参数，第一个为 GTileLayer 的对象数组，通过脚本来生成，第二个参数是投影 Gprojection 的对象，一般情况下采用 Google 地图的默认投影即可，本系统采用 G_NORMAL_MAP 的地图投影，第三个参数是用户定义的新地图类型的名称。定义新的地图类型的代码如下：

```
var copyright = new GCopyrightCollection("");
copyright.addCopyright(Cditu);
//扩展地图缩放级别到22级别
var dituTileLayer = new GTileLayer(copyright, 0, 22);
dituTileLayer.getTileUrl = function(tile, zoomlevel) {
//获取瓦片坐标
var x1 = tile.x;
var y1 = tile.y;
return  "getTile.jsp?x="+x1+"&y="+y1+"&z="+zoomlevel;
}
var ditu = new GMapType([dituTileLayer],
G_NORMAL_MAP.getproection(),
"Ditu",
{ shortName: "ditu", alt: "layer from ditu.google.com" }
);
```

地图控件定义主要重写了地图放大缩小及移动的控件，提高了用户体验，同重写地图类型，重写地图空间通过子类化 GControl[7]类来实现的，以重写放大控件为例，首先定义 GZoomInCoontrol 类继承 GControl 类，实现其中的 initialize ()和 getDefaultPositon()方法，核心代码如下：

```
function GZoomInControl() {}
GZoomInControl.prototype = new GControl();
GZoomInControl.prototype.initialize = function(map) {
var container = document.createElement("div");
var img_zoomin = document.createElement("img");
img_zoomin.id ="zoomin1";
img_zoomin.src = "images/zoomin2.png";
img_zoomin.onclick = function(){ map.zoomIn();}
container.appendChild(img_zoomin);
map.getContainer().appendChild(container);
return container;
}
GZoomInControl.prototype.getDefaultPosition = function(){
return new GControlPosition(G_ANCHOR_TOP_RIGHT, new GSize(20, 7));
}
```

地图显示时，使用新定义的地图类型定义 GMap 对象，并利用 addOverlay()方法将自定义的地图控件添加到地图上，地图显示的核心代码如下：

```
//使用新定义的地图类型初始化GMap2对象
map_f1 = new GMap2(document.getElementById('map'),{mapTypes:[ditu]});
map_f1.enableDoubleClickZoom();
map_f1.setCenter(new GLatLng(this.lat,this.lng),this.zoom);
//添加自定义的地图控件（上下左右拖拽控件）
map_f1.addControl(new GDragUpControl());
map_f1.addControl(new GDragDownControl());
map_f1.addControl(new GDragLeftControl());
map_f1.addControl(new GDragRightControl());
//添加放大缩小控件
map_f1.addControl(new GZoomInControl());
map_f1.addControl(new GZoomOutControl());
```

4.3 系统实现效果

图书查询界面，图书查询支持的搜索类型有按书名、按出版社、作者查询和索书号查询等操作，并支持前向匹配、模糊匹配和精确匹配等匹配方式，图书查询的界面如下所示：

图 7 图书查询界面

Figure 7 Book Searching Interface

用户点击查询按钮后，将查询的关键字传输到服务器端，服务器进行图书信息查询后，从数据库中提取出图书的信息，并将图书信息添加到地图上，返回给客户端。服务器将图书查询的结果显示在地图上，图书信息包括标题、作者、索书号、出版商、以及出版时间，图书的位置信息包括所在的书架号，以及所在书架格子的行号、列号和经纬度信息。

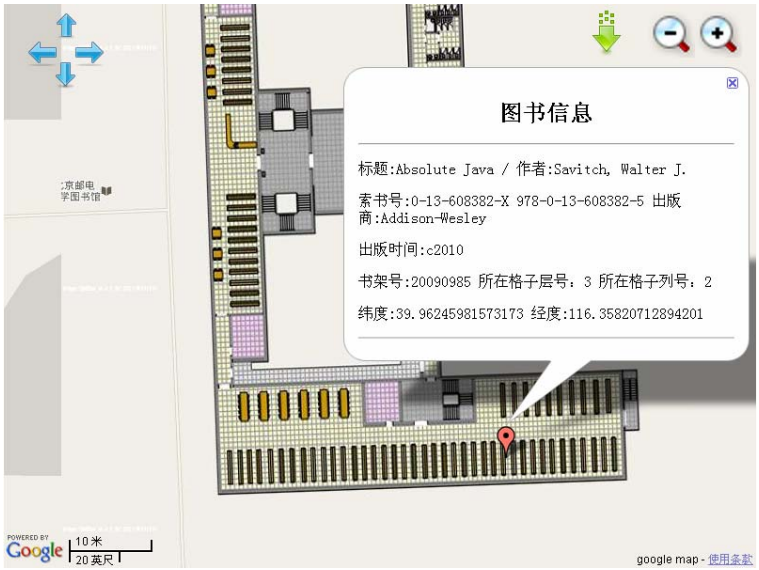

图 8 图书查询结果显示界面

Figure 8 Book Searching Results Interface

5 结论

本文实现了一种基于 Google Map API 的图书查询系统，与现有的图书查询系统相比，本系统在数据库中添加了图书的位置信息，并且基于 Google Map API 将图书馆室内地图和室外地图无缝结合，在地图上显示图书的信息，极大地节省了查找图书的时间，帮助用户快速地找到图书。

[参考文献] (References)

[1] 孙晓茹，赵军。Google Maps API 在 WEBGIS 中的应用[J]。微计算机信息，2006,22（19）：224～226.
[2] Cong Fu, Yan Wang, Yanqing Xu, Qingquan Li, The Logistics Network System Based on the Google Maps API[C]. Logistics System and Intelligent Management, 2010 International Conference. Vol.3, 2010, pages:1486~1489.
[3] Tan Qingquan , Qiao Yongjun, Wang Zhanying, Liu qun. Implementation on Google Maps-style WebGIS based on ARCGIS[A], Advanced Computer Control(ICACC),2010 2nd International Conference, Volume,5,page(s),60~64.
[4] Google 地图坐标系统[OL]. http://chuo.me/2008/10/google-map-cordinate-system/.
[5] Google Maps 地图投影全解析[OL].
http://tech.ddvip.com/2009-04/1240807328116908_4.html
[6] 顾春霞. 基于 web 的图书管理系统建模与实现[D].苏州：苏州大学, 2006, 23~30.
[7] Developer’s Guide of Google Maps API[OL].
http://code.google.com/intl/zh-CN/apis/maps/documentation/index.html