# 北京九鼎图业科技有限公司
# 地图引擎服务系统
# （MyGIS Server）
# 二次开发手册

北京九鼎图业科技有限公司

2011 年

如果您对本产品有任何建议和疑问，请与以下地址联系：
北京九鼎图业科技有限公司
地 址：北京市海淀区安宁庄西路 15 号怡美家园 20 号楼 302 室
邮 编：100085
传 真：（010）82758693
电 话：（010）82758693
E-mail：support@mapyeah.com

# 目录

# 1 简介

## 1.1 概述

地图引擎服务系统是北京九鼎图业科技有限公司所开发的用于地图发布的软件，该平台将地理信息采集技术和地图出版技术融为一体，将 MGIS Desktop 地图出版的优势发挥到尽至。通过它可以实现地图的快速发布、浏览、查询等，它是下一代 WEBGIS 技术的主流。

北京九鼎图业科技有限公司特别注重产品质量，地图引擎服务系统开发全面采用 ISO9000 和 CMM 软件质量管理体系，经过全面的功能和性能测试，系统具有很高稳定性和可靠性。

地图引擎服务系统作为九鼎图业 MGIS Desktop 地理信息家族重要成员，将通过提供快速地图发布、浏览、查询、和高响应速度，实现您对地图的完美展示。

我们坚信：地图引擎服务系统将给您带来无限的地图创意，提高您的数据快速发布能力，提升您的竞争能力；地图引擎服务系统与 MGIS Desktop、MGIS、MIMS 等将为您提供全面的地理信息开发与应用解决方案。

### 1.1.1 地图引擎服务系统特点

地图引擎服务系统与传统的 WEBGIS 地图服务相比较以下的优点：

- **功能强大**

A、支持地图的基本操作；
B、支持信息点的节点编辑；
C、支持版本地图及多地图对比；
D、支持标绘功能；
E、支持空间查询功能；
F、支持专题图叠加功能；
G、支持自定义地图的开发模式；
H、支持 KML、GML、GeoRSS 聚合信息。

- **兼容各种浏览器**

IE、FireFox，Google 浏览器、Opera、safari、MY IE。

- **地图高品质和高精度**

模板化的专业地图制图，出版级地图数据制作和显示品质

- **高性能、海量并发**

金字塔地图组织，快速的地图显示，实现地图拼接与地图缓存机制，AJAX 异步传输机制；支持集群和网格地图服务

- **与 GIS 平台无关的服务模式**

提供开放性的地图请求服务，通过 WebSevice（WMS、WFS）访问 GIS 平台，实现与 ArcGIS 的集成

- **多模式应用支持**

支持 WebGIS 应用、桌面 GIS 应用（轻量级客户端）、手机 GIS 的一张图应用和一套服

务

■ **API 简单、快速的应用开发模式**

开发人员无需掌握复杂 GIS 软件几分钟即可开展应用

■ **与应用无关的运营维护**

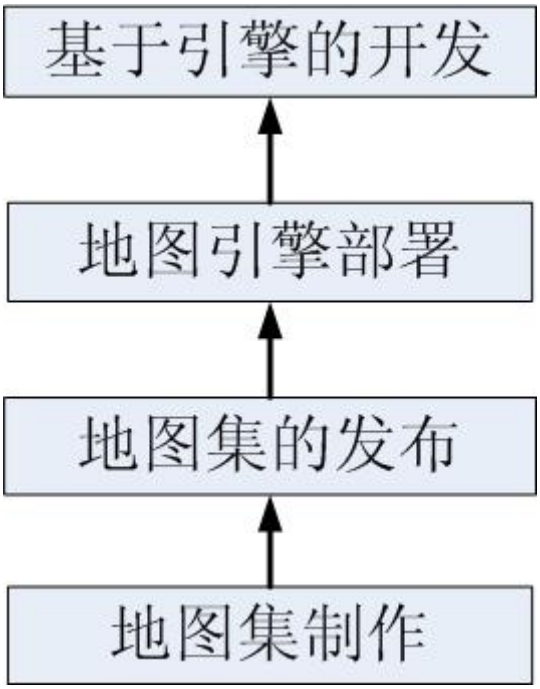
提供空间数据采集制图、空间数据管理、地图服务发布、GIS 应用支撑的完整流程，应用单位无需维护

■ **高性能，为用户减少软硬件成本**

能支持 300 并发用户，相当于 3000 个在线用户 ，可以节省软件集群软件及硬件设备

### 1.1.2 地图引擎服务系统流程图

MYGIS Server 的使用流程如下:

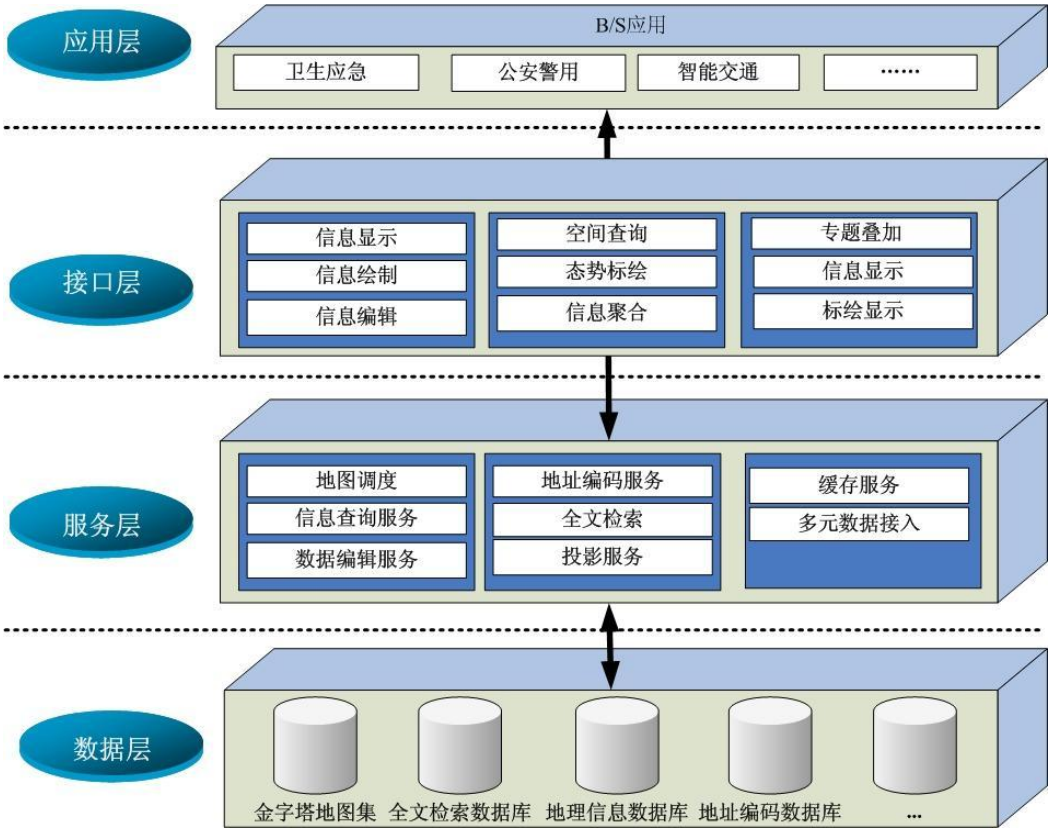



- 地图集制作：主要负责不同等级的地图配图和切割。
- 地图集的发布：把切割完成的地图存放到地图图片库中
- 地图引擎部署：对一定级别和和范围的图片进行解析，并提供该区域的地图，服务部署。
- 基于引擎的开发 :网络 GIS 客户端是采用 Javascript 技术封装、桌面 GIS 客户端采用 COM 技术封装、手机客户端采用 DLL 技

术封装

### 1.1.3 地图引擎服务系统的软件结构

系统的体系机构如下:




系统分为数据层、服务层、接口层、应用层，其中：

数据层：主要包括金字塔地图集、地理信息数据库、地址编码数据库、全文检索数据库;

主要提供用于电子地图显示、信息查询、地址编码查询等所需要的原始数据。

服务层：通过地理信息服务接口访问地理信息、空间查询，通过地址搜索服务接口访问地址的编码信息。

接口层：接口层主要提供信息显示、信息绘制、信息编辑、态势标绘、专题叠加、信息聚合、空间查询等二次开发接口，让二次开发更简单的使用系统进行开发。

应用层：提供面向接口应用的业务系统，通过业务系统可以访问系统的地图调度、空间查询等应用，方便业务部门对业务数据进行分析、决策、调度等。

应用系统采用基于 Internet/Intranet 结构、B/S 模式。服务器端采用 J2EE 的架构，在 Tomcat 应用服务器的基础上，建立各个接口服务系统，这些接口服务系统都可以通过 Java Servlet 的方式来提供，各个服务系统通过访问空间数据库及相关的应用组件来实现自己的功能。

客户端对于服务的调用可以支持多个平台，包括 J2EE 和.Net 平台，服务器端和客户端的服务调用采用 SOAP/HTTP 的方式来实现。

## 1.2 依赖和假设

使用此开发手册需要熟悉以下内容：

1) 了解 GIS 及其相关的概念
2) 熟悉 HTML、Javascript 和 XML 知识

# 2 开发接口说明

在网络 GIS 服务中，mygis.jsp 文件定义了 MMap 对象，该对象是面向对象的，它包含有构造函数、属性和方法，它主要用于实例化一个地图。

描述说明：

## 2.1 构造函数

| 构造方法 | 参数 | 描述/返回说明（没有声名返回值即没有返回值） |
|---|---|---|
| MMap(container) | container：装载地图组件的容器对象的引用<br>类型：HTML DOM 的 block Element 元素对象，例如一个 容器 | 在容器（container）创建一个实例，地图就在该容器下进行显示. |

## 2.2 地图控件操作

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| addControl(pControl) | pControl：地图控件实例对象<br>类型：MapControl、MapStandControl 或 MapSmallControl 的对象实例 | 加入地图控件。目前可用控件：MMapControl, MMapStandControl, MMapSmallControl |
| showMapControl()<br>showSmallMapControl()<br>showStandMapControl() | 无 | 显示地图操作控件 |
| hideMapControl() | 无 | 隐藏地图操作控件 |
| showCopyright() | 无 | 显示版权信息 |
| hideCopyright() | 无 | 隐藏版权信息 |
| showMapScale() | 无 | 显示比例尺信息 |
| hideMapScale() | 无 | 隐藏比例尺信息 |

| | | |
|---|---|---|
| showMapServer() | 无 | 显示“矢量地图、影像地图、矢量地图”按钮 |
| hideMapServer() | 无 | 隐藏“矢量地图、影像地图、矢量地图”按钮 |
| switchMapServer（iIndex） | iIndex：图像类型（取值 0：矢量地图，1：影像地图，2：矢量影像）<br>类型：Number | 在矢量地图、影像地图和矢量影像之间进行切换 |

## 2.3 鹰眼操作

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| addOverView(pOverView) | pQverView：鹰眼对象实例<br>类型：OverView | 在当前地图中增加一个鹰眼 |
| showOverView() | 无 | 显示鹰眼. |
| hideOverView() | 无 | 隐藏鹰眼 |
| reverseOverView() | 无 | 鹰眼显示和不显示之间来回切换 |

## 2.4 地图操作

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| zoomIn() | 无 | 放大 |
| zoomInExt() | 无 | 拉框放大 |
| zoomOut() | 无 | 缩小 |
| zoomOutExt() | 无 | 拉框缩小 |
| pan(x,y) | 1)x：x 方向上的移动增量，是像素值<br>2)y：y 方向上的移动增量，是像素值 | 平移地图<br>说明：参数为空的情况下设置当前地图操作状态为平移状态 |

| | | |
|---|---|---|
| | x 和 y 类型为：Number<br>函数中的参数可以为空 | 如果存在参数 x,y,则相应向 X,Y 轴方向移动 x 和 y 的距离增量 |
| measureLength() | 无 | 测距离 |
| measureArea() | 无 | 测面积 |
| fullExtent() | 无 | 全图显示. |
| gotoCenter() | 无 | 对中 |
| print(css,title,foottitle) | css:页面的样式<br>title：标题<br>foottitle：脚注 | 打印 |
| clear() | 无 | 清除 |
| centerAtLatLng(latLng) | latLng：指定的点对象<br>类型：Point | 地图对中到给定的点. |
| recenterOrPanToLatLng(latLng) | latLng：指定的点对象<br>类型：Point | 对中到给定的坐标，如果该点在当前视图上，则进行平移到该点为地图中心. |
| zoomTo(zoomLevel) | zoomLevel：地图等级<br>类型：Number | 缩放到给定的级别. |
| centerAtBounds (dInMinX,dInMinY,dInMaxX,dInMaxY) | 1)dInMinX：指定范围的 X 轴坐标上的最小值<br>2)dInMinY：指定范围的 Y 轴坐标上的最小值<br>3)dInMaxX：指定范围的 X 轴坐标上的最大值<br>4)dInMaxY：指定范围的 Y 轴坐标上的最大值<br>上边四参数类型均为：Number<br>说明：此方法中的参数也可以为一个 Bounds 对象 | 对指定的范围进行地图对中 |
| centerAndZoom (latLng, zoomLevel) | 1)latLng：坐标中心点<br>类型：Point<br>2)zoomLevel：指定的级别<br>类型：Number | 地图以给定的坐标为中心并缩放到给定的级别下. |
| openInfoWindow(pPoint,html) | 1)pPoint：指定的点位置<br>类型：Point | 在指定的位置显示信息 |

| | | |
|---|---|---|
| | 2)html：要显示的 html 格式的信息内容<br>类型：String | |

## 2.5 获得地图状态

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| getCenterLatLng() | 无 | 返回地图中心的坐标<br>返回值类型：MPoint |
| getBoundsLatLng() | 无 | 返回当前视窗的经纬度边框<br>返回值类型：Bounds |
| getSpanLatLng() | 无 | 返回当前视窗的经纬度跨度<br>返回值类型：MRectangle |
| getZoomLevel() | 无 | 返回地图的当前级别<br>返回值类型：Number |
| getMaxLevel() | 无 | 返回地图的最大级别，返回类型为 int |
| getLevelOfBounds (dInMinX,dInMinY,dInMaxX,dInMaxY) | 1)dInMinX：指定范围的 X 轴坐标上的最小值<br>2)dInMinY：指定范围的 Y 轴坐标上的最小值<br>3)dInMaxX：指定范围的 X 轴坐标上的最大值<br>4)dInMaxY：指定范围的 Y 轴坐标上的最大值<br>上边四参数类型均为：Number<br>说明：此方法中的参数也可以为一个 Bounds 对象 | 获取指定的范围的级别<br>返回值类型：Number |
| getDragMode() | 无 | 获取当前绘制模式<br>返回值类型：String |

## 2.6 设置地图绘制模式

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| changeDragMode(drawmode,inputform1,inputform2,callback) | 1)drawmode：操作模式（取值为：<br>"measure":测量<br>"pan":平移模式<br>“drawPoint”：获取坐标点<br>“drawCircle”：画圆<br>“drawRect”：画矩形<br>“drawPolyline”：画线<br>“drawPolygon”：画多边型）<br>类型：String<br>2)Inputform1(可选参数)：用于显示 X 坐标返回值的 html 元素的 ID<br>类型：String<br>3)Inputform2(可选参数)：用于显示 Y 坐标返回值的 html 元素的 ID<br>类型：String<br>4)callback(可选参数)：回调函数<br>类型：Object | 改变操作模式(不建议使用) |
| drawPoint(callback) | 把点坐标（x,y）返回到回调函数中,字符类型，如:”116.5,39” | callback 回调函数<br>示例：<br>function callback(point){<br>alert(pnt);<br>//116.39,40.3<br>} |
| drawCircle(callback) | 把圆坐标（x,y,radius）返回到回调函数中，字符类型，如:”116.5,39,0.002” | callback 回调函数<br>示例：<br>function callback(circle){<br>alert(circle);<br>//116.39,40.3,100<br>} |
| drawRect(callback) | 把矩形坐标（minx,miny,maxx,maxy）返回到回调函数中，字符类型，如:”116.5,39, 116.6,39.7” | callback 回调函数<br>示例：<br>function callback(rect){<br>alert(rect);<br>//116.39,40.3, 116.59,45.3 |

| | | |
|---|---|---|
| | | } |
| drawPolyline(callback) | 把折线的坐标串返回到回调函数中,字符类型 | callback 回调函数<br>示例：<br>function callback(line){<br>alert(line);<br>//116.39,40.3, 116.59,45.3 |
| drawPolygon(callback) | 把多边形的坐标串返回到回调函数中,字符类型 | callback 回调函数<br>示例：<br>function callback(polygon){<br>alert(polygon);<br>//116.39,40.3, 116.59,45.3 |
| drawArrow(callback) | 把单箭头的坐标串返回到回调函数中,字符类型 | callback 回调函数<br>示例：<br>function callback(path){<br>alert(path);<br>//116.39,40.3, 116.59,45.3 |
| drawDoubleArrow(callback) | 把多箭头的坐标串返回到回调函数中,字符类型 | callback 回调函数<br>示例：<br>function callback(path){<br>alert(path);<br>//116.39,40.3, 116.59,45.3 |
| drawActionLine (callback) | 把行动线的坐标串返回到回调函数中,字符类型 | callback 回调函数<br>示例：<br>function callback(path){<br>alert(path);<br>//116.39,40.3, 116.59,45.3 |

## 2.7 信息叠加

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| addOverlay(overlay, bDisRemovable) | 1)overlay：要添加的覆盖对象<br>类型：MMarker、MTitle、MCircle、MPolyline、MPolygon、 | 在当前地图上加入给定的对象 |

| | | |
|---|---|---|
| | MMultiFeat、MTMLegend、MCustomOverlay 对象的实例<br>2)bDisRemovable：设置此对象否不可以删除，默认是：false，是可以删除的<br>类型：boolean | |
| removeOverlay(overlay) | overlay：要添加的覆盖对象<br>类型：MMarker、MTitle、MCircle、MPolyline、MPolygon、MMultiFeat、MTMLegend、MCustomOverlay 对象的实例 | 在地图上删除给定的对象. |
| clearOverlays() | 无 | 在地图上清除所有的对象. |
| getOverlays() | 无 | 返回信息叠加类的对象数组<br>返回值类型：[]MOverLay |
| getCurrentEditor() | 无 | 获取当前编辑的信息对象。<br>返回值类型：MOverLay |

## 2.8 信息查询、编辑

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| query(pQuery,callback) | 1)pQuery：查询对象<br>类型：QueryObject，GeoCodeObject，RouteObject 对象或 QueryObject，GeoCodeObject，RouteObject 对象的数组<br>2)callback：回调函数，用户获得查询的结果<br>类型：function<br>drawFeatureObject, drawRouteObject, | 进行查询 |

| | | |
|---|---|---|
| | drawGeoCodeObject 三个函数之一 | |
| edit(pEdit) | pEdit：编辑对象<br>类型：EditObject 或 EditObject 对象的数组 | 进行编辑，其中：pEdit 为 EditObject 对象或 EditObject 对象的数组 |
| getQueryResult(); | 无 | 获取信息查询的结果<br>返回值类型： []MLayer<br>说明：它是[]MLayer 结果集的 clone |
| setLayer(pMLyrArr) | pMLyrArr：设置地图中图层的集合<br>类型：[]MLayer | 设置地图对象中包含的图层 |
| getGeoCodeResult() | 无 | 获取地址查询的结果：<br>返回值类型：[]GeoSearchObject |
| getRouteResult () | 无 | 获取路径查询的结果：<br>返回值类型：[]RoutePathArrObj |

## 2.9 监听地图状态变化操作

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| addMapChangeListener(func) | func：当增加地图变换时调用的函数<br>类型：function | 增加地图状态变化时执行的操作，func 为函数 |
| removeMapChangeListener(func) | func：当删除地图变换时调用的函数<br>类型：function | 删除地图状态变化时执行的操作，func 为函数 |

## 2.10 版本地图的显示

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| addVersion (strVersion,pMapServer) | 1)strVersion：版本信息<br>类型：String | 增加某版本的配置信息，pMapServer 为 MMapServer 类型 |

| | | |
|---|---|---|
| | 2)pMapServer：地图设置对象<br>类型：MMapServer | |
| showVersion (strVersion) | strVersion：版本信息<br>类型：String | 显示某版本的地图 |
| getVersionInfo() | 无 | 获得版本信息，如:"2005,2006"<br>返回值类型：String |

## 2.11 专题的叠加

见《使用"专题叠加"进行开发》

## 2.12 获取指定区域的地图

| 方法 | 参数 | 描述 |
|---|---|---|
| downloadMap (minx,miny,maxx,maxy, iLevel,format, overlayurl,bTiles) | 1)minx：指定范围的 X 轴方向上的最小值;<br>2)miny：指定范围的 Y 轴方向上的最小值;<br>3)maxx：指定范围的 X 轴方向上的最大值;<br>4) maxy：指定范围的 Y 轴方向上的最大值;<br>5)iLevel：指定的级别;<br>上边五个参数类型均为：Number<br>6)format：获得图片的方式（可取的值："stream"(默认)和"html"）<br>类型：String<br>7）overlayurl：要叠加的地图地址,string 类型<br>8)bTiles：为是否采用瓦片方式,boolean 型 | 对指定的区域进行下载 |

也可以通过以下 5 种 http 请求方式获得地图

http://127.0.0.1:8888/MyGIS/mygis

协议：http

参数如下 1:

| 序号 | 参数 | 值域 | 备注 |
|---|---|---|---|
| 1. | REQUEST | GetMap | |
| 2. | BBOX | 字符型 | Minx, miny, maxx ,maxy |
| 3. | WIDTH | 整形 | 获取的图片宽 |
| 4. | HEIGHT | 整形 | 获取的图片高 |

http://127.0.0.1:8888/MyGIS/mygis?REQUEST=GetMap&BBOX=114.375,38.82421875,118.375,44.02734375&HEIGHT=300&WIDTH=300

参数如下 2:

| **序号** | **参数** | **值域** | **备注** |
|---|---|---|---|
| 1. | Service | getRectImg | |
| 2. | minx | 浮点型 | |
| 3. | miny | 浮点型 | |
| 4. | maxx | 浮点型 | |
| 5. | maxy | 浮点型 | |
| 6. | result | html\|stream | 返回的结果方式，分别为网页方式和文件流方式 |
| 7. | zoom | 整形 | 获取的级别（优先判断） |

http://127.0.0.1:8888/MyGIS/mygis?Service=getRectImg&result=html&minx=116.35167&miny=39.91972&maxx=116.38865&maxy=39.94144&zoom=3

参数如下 3:

| **序号** | **参数** | **值域** | **备注** |
|---|---|---|---|
| 1. | Service | getRectImg | |
| 2. | result | html\|stream | 返回的结果方式，分别为网页方式和文件流方式 |
| 3. | box | 字符型 | Minx, miny, maxx ,maxy |
| 4. | zoom | 整形 | 获取的级别（优先判断） |

http://127.0.0.1:8888//MyGIS/mygis?Service=getRectImg&result=html
&box=114.375,38.82421875,118.375,41.02734375&zoom=3

参数如下 4

| 序号 | 参数 | 值域 | 备注 |
|---|---|---|---|
| 1. | Service | getRectImg | |
| 2. | minx | 浮点型 | |
| 3. | miny | 浮点型 | |
| 4. | maxx | 浮点型 | |
| 5. | maxy | 浮点型 | |
| 6. | result | html\|stream | 返回的结果方式，分别为网页方式和文件流方式 |

| | | | |
|---|---|---|---|
| 7. | width | 整形 | 获取的图片宽 |
| 8. | height | 整形 | 获取的图片高 |

http://127.0.0.1:8888/MyGIS/mygis?Service=getRectImg&minx=114.375&miny=38.82421875&maxx=118.375&maxy=41.02734375&width=1024&height=562

参数如下 5:

| 序号 | 参数 | 值域 | 备注 |
|---|---|---|---|
| 1. | Service | getRectImg | |
| 2. | box | 字符型 | Minx, miny, maxx ,maxy |
| 3. | result | html\|stream | 返回的结果方式，分别为网页方式和文件流方式 |
| 4. | width | 整形 | 获取的图片宽 |
| 5. | height | 整形 | 获取的图片高 |

http://127.0.0.1:8888//MyGIS/mygis?Service=getRectImg&box=114.375,38.82421875,118.375,41.02734375&width=1024&height=562

## 2.13 通过 IMS 获得专题图

http://127.0.0.1:8888/MServer/legendServlet

协议：http

参数如下 2:

| 序号 | 参数 | 值域 | 备注 |
|---|---|---|---|
| 1. | service | 字符型 | IMS 的服务,如果为空则采用默认的服务 |
| 2. | box | 字符型 | minx, miny, maxx ,maxy |
| 3. | width | 整形 | 获取的图片宽 |
| 4. | height | 整形 | 获取的图片高 |
| 5. | layers | 字符型 | 默认为全部 |

http://127.0.0.1:8888/MyGIS/ /legendServlet?service=poi&box=114.375,38.82421875,118.375,41.02734375&width=600&height=400

## 2.14 版本信息

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| about() | 无 | 显示版本信息。<br>返回值类型：String |

# 3 应用开发样例代码

在本章我们主要是介绍采用网络 GIS 进行开发，开发前开发人员必须熟悉以下基本知识：

- 熟悉 HTML4.0 语言
- 熟悉 Javascript 脚本
- 熟悉 XML 协议

## 3.1 一个简单的例子

下面为一个利用网络 GIS 开发 web 应用的例子，下面的例子设置地图在 500x400 的地图进行显示：

```
<HTML>
<head>
<meta http-equiv="content-type" content="text/html; charset=GB2312"/>

<title>MyGIS JavaScript API Example: simple</title>

<SCRIPT type="text/javascript" src="http://127.0.0.1:8888/MyGIS/maps?">
</SCRIPT>

<script type="text/javascript">

var _MapApp;
function onLoad() {
_MapApp = new MMap(document.getElementById("map"));
_MapApp.showStandMapControl();
_MapApp.centerAndZoom(new MPoint(116.3919, 39.9419), 8);
}
</script>
<body onload="onLoad()">
<div id="map" style="width:500px;height:400px"></div>
</body>
</html>
```

效果如下:


## 3.2 使用“地图操作”进行开发

### 3.2.1 放大

```
var  _MapApp = new MMap(document.getElementById("map"));
_MapApp.zoomIn();
```

### 3.2.2 拉框放大

```
var  _MapApp = new MMap(document.getElementById("map"));
_MapApp.zoomInExt();
```

### 3.2.3 缩小

```
var  _MapApp = new MMap(document.getElementById("map"));
_MapApp.zoomOut();
```

### 3.2.4 拉框缩小

```
var  _MapApp = new MMap(document.getElementById("map"));
_MapApp.zoomOutExt();
```

### 3.2.5 对中

```
var  _MapApp = new MMap(document.getElementById("map"));
_MapApp.gotoCenter();
```

### 3.2.6 全图

```
var  _MapApp = new MMap(document.getElementById("map"));
_MapApp.fullExtent ();
```

## 3.3 使用“绘制模式”进行开发

### 3.3.1 进行平移

```
var  _MapApp = new MMap(document.getElementById("map"));
_MapApp.pan();
```

### 3.3.2 测量模式

```
var  _MapApp = new MMap(document.getElementById("map"));
_MapApp.measureLength();
```

### 3.3.3 获取点信息

```
var  _MapApp = new MMap(document.getElementById("map"));
_MapApp.drawPoint(callback);
```

### 3.3.4 设置画线

```
var  _MapApp = new MMap(document.getElementById("map"));
_MapApp.drawPolyline(callback);
```

### 3.3.5 设置画矩形

```
var  _MapApp = new MMap(document.getElementById("map"));
_MapApp.drawRect(callback);
```

### 3.3.6 设置画多边形

```
var  _MapApp = new MMap(document.getElementById("map"));
_MapApp. drawPolygon (callback);
```

### 3.3.7 设置画圆

```
var  _MapApp = new MMap(document.getElementById("map"));
_MapApp. drawCircle (callback);
```

## 3.4 显示和隐藏“地图操作控件”

你可以用 showMapControl()进行显示地图操作控件，用 hideMapControl()进行隐藏地图操作控件

### 3.4.1 显示操作控件

```
var _MapApp = new MMap(document.getElementById("map"));
_MapApp.showMapControl();
```

### 3.4.2 隐藏操作控件

```
var _MapApp = new MMap(document.getElementById("map"));
_MapApp.hideMapControl();
```

## 3.5 “鹰眼”操作

### 3.5.1 初始化鹰眼

```
var _MapApp = new MMap(document.getElementById("map"));
_MapApp.centerAndZoom(new MPoint(116.3919, 39.9419), 8);
var pOverview=new MOverView();
pOverview.setWidth(200);
pOverview.setHeight(200);
pOverview.setMinLevel(8);
pOverview.setMaxLevel(10);
_MapApp.addOverView(pOverView);
```

### 3.5.2 显示鹰眼

```
_MapApp.showOverView();
```

### 3.5.3 隐藏鹰眼

```
_MapApp.hideOverView();
```

## 3.6 使用“信息叠加接口”进行开发

### 3.6.1 增加一个图标类

为了在地图上增加一个图标，第一步是创建一个点;第二步创建 Marker 对象（用于显示点对象）；然后通过方法 addOverlay 加入即可.

```
var _MapApp = new MMap(document.getElementById("map"));
```

```
var pIcon=new MIcon();
pIcon.setImage("images/iconA.png");
pIcon.setHeight(16);
pIcon.setWidth(16);
var strMsg="msg";
var marker = new MMarker(_MapApp.getCenterLatLng(),pIcon);
marker.addListener("click",function(){marker.openInfoWindowHtml(strMsg);});// 将 click 事件加入到 Marker 的 onclick 事件当中去
_MapApp.addOverlay(marker);
```

### 3.6.2 增加一个线

为了在地图上增加一条线，第一步是创建一个线的实例;然后通过方法 addOverlay 加入即可.

```
var _MapApp = new MMap(document.getElementById("map"));
var pLine=new MPolyline("116.4033,39.96989,116.43162,39.90055","#ff0000", 3,1);
var strMsg="msg";
pLine.addListener("click",function(){pLine.openInfoWindowHtml(strMsg);}); //将 click 事件加入到 MPolyline 的 onclick 事件当中去
_MapApp.addOverlay(pLine);
```

### 3.6.3 增加一个矩形

为了在地图上增加一个矩形，第一步是创建一个矩形的实例;然后通过方法 addOverlay 加入即可.

```
var _MapApp = new MMap(document.getElementById("map"));
var pPoints="116.38481,39.95996,116.39481,39.98996";
var pRectangle=new MRectangle(pPoints,"#ff00FF", 2,0.5,"green");
var strMsg="msg";
pRectangle.addListener("click",function(){pRectangle.openInfoWindowHtml(strMsg);});
_MapApp.addOverlay(pRectangle);
```

### 3.6.4 增加一个多边形

为了在地图上增加一个圆，第一步是创建一个多边形的实例;然后通过方法 addOverlay 加入即可.

```
var _MapApp = new MMap(document.getElementById("map"));
```

```
var pPoints="116.38481,39.95996,
116.39019,39.92529,116.39019,39.94287,116.38677,39.95361,116.38335,39.95361,116.
38481,39.95996";
var pPolygon=new MPolygon(pPoints,"#ff00FF", 3,0.5,"blue");
var strMsg="msg";
pPolygon.addListener("click",function(){pPolygon.openInfoWindowHtml(strMsg);});
_MapApp.addOverlay(pPolygon);
```

### 3.6.5 增加一复杂个多边形

为了在地图上增加一个圆，第一步是创建一个多边形的实例;然后通过方法 addOverlay 加入即可.

```
var _MapApp = new MMap(document.getElementById("map"));
var pPoints="
116.29857,39.95482,116.26684,39.87816,116.36303,39.83519,116.42943,39.87474,116.
39721,39.97533,116.34008,40.00121,116.29857,39.95482;116.31273,39.93676,116.297
6,39.88256,116.35521,39.85912,116.41137,39.87816,116.37572,39.96459,116.31273,39
.93676";
var pPolygon=new MMultiFeat (pPoints,"#ff00FF", 3,0.5,"blue");
var strMsg="msg";
pPolygon.addListener("click",function(){pPolygon.openInfoWindowHtml(strMsg);});
_MapApp.addOverlay(pPolygon);
```

### 3.6.6 增加圆

为了在地图上增加一个圆，第一步是创建一个圆的实例;然后通过方法 addOverlay 加入即可.

```
var _MapApp = new MMap(document.getElementById("map"));
var pPoints="116.38481,39.95996,0.003";
var pCircle=new MCircle(pPoints,"#ff00FF", 2,0.5,"green");
var strMsg="msg";
pCircle.addListener("click",function(){pCircle.openInfoWindowHtml(strMsg);});
_MapApp.addOverlay(pCircle);
```

### 3.6.7 WMS

为了在地图上增加一个圆，第一步是创建一个圆的实例;然后通过方法 addOverlay 加入即可.

```
var _MapApp = new MMap(document.getElementById("map"));
pOverlay=new MCustomOverlay();
```

```
pOverlay.setImgPNG(true);
var strURL="http://124.205.128.35:8888/ogc/smwms?VERSION=1.1.1&REQUEST=
GetMap&LAYERS=FLUX@ccjj.1&STYLES=&CRS=&BBOX=MBOX&WIDTH=MWIDTH&
HEIGHT=MHEIGHT&FORMAT=png&BGCOLOR=0xFFFFFF&TRANSPARENT=TRUE";
pOverlay.setFormat(strURL);
_MapApp.addOverlay(pOverlay);
```

### 3.6.8 清除指定对象

```
var _MapApp = new MMap(document.getElementById("map"));
var pPoints="116.38481,39.95996,0.003";
var pCircle=new MCircle(pPoints,"#ff00FF", 2,0.5,"green");
var strMsg="msg";

_MapApp.addOverlay(pCircle);
_MapApp.removeOverlay(pCircle);
```

### 3.6.9 清除所有对象

```
var _MapApp = new MMap(document.getElementById("map"));
_MapApp.clearOverlays();
```

## 3.7 使用“专题叠加”进行开发

使用专题图叠加接口进行开发，只要按照以下步骤即可：

➢ 初始化专题图对象：

```
_CustomOverlay=new MCustomOverlay();
```

➢ 设置专题图地址：

A) 假如地图访问地址为:

http://*:8888/getMap?BBOX=486546,307556,488415,308635&WIDTH=200&HEIGHT=300&FORMAT=GIF

则需要把红色部分分别变为：MBOX,MHEIGHT,MWIDTH，形成以下的地图地址：http://*:8888/getMap?BBOX= MBOX &WIDTH=MWIDTH&HEIGHT=MHEIGHT&FORMAT=GIF

B) 在专题图格式中，把变换后的地址设置为专题图生成格式：

```
_CustomOverlay.setFormat("http://*:8888/getMap?BBOX=MBOX&WIDTH=MWIDTH&HEIGHT=MHEIGHT&FORMAT=GIF");
//其中MBOX,MHEIGHT,MWIDTH在使用过程中是实时生成的
```

问题一：为什么需要变化格式呢？

答：由于获取地图需要传递地图的范围、图片大小，而在地图缩放、地图窗口变化时这

些参数都有所变化，设置占位符是为了把这些参数留给系统，让系统自动填充这些参数。以便地图能在缩放情况下自动计算请求地图的范围、地图宽度和高度。

问题二、为什么需要设置图片格式？

答：由于IE浏览器对于PNG和GIF格式的透明处理方式不同，系统无法从二进制图形流中获取图像的格式，因而无法处理其透明情况，所以需要指定图片格式。当然，如果接口中没有Format等参数，只要在请求中填写占位符即可，如可以在请求中追加:“&image=GIF”等

1) 打开专题图：

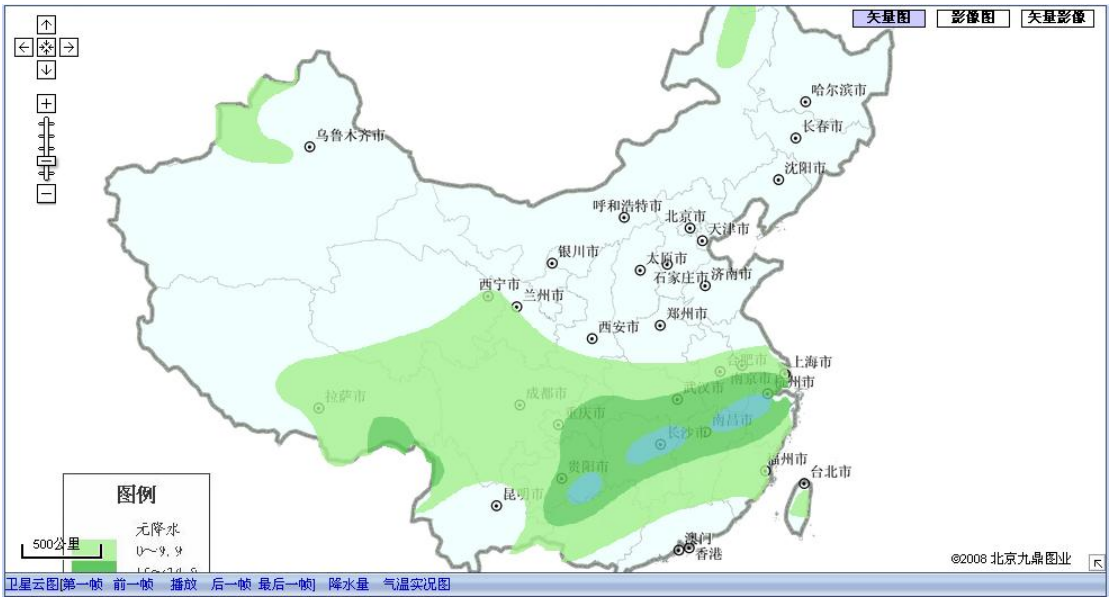
```
_CustomOverlay.open();
```

结合中心的例子如下：

```
<input        type="button"                    style="width:100%"
onclick="showLegend(this);"
 value="显示专题图"></input>
<jscript>
    Var CustomOverlay=null;
    function showLegend(pEle){
    if(pEle.value=="显示专题图"){
        pEle.value="隐藏专题图";
        _CustomOverlay=new MCustomOverlay();
        //其中MBOX,MHEIGHT,MWIDTH在使用过程中是实时生成的,它是占位符

    _CustomOverlay.setFormat("http://*:8888/service/GovEMap/wms?
            BBOX=MBOX&WIDTH=MWIDTH&HEIGHT=MHEIGHT&
            SRS=EPSG:NONE&layers=26 &FORMAT=GIF&TRANSPARENT=TRUE
            &request=getmap&ServiceName=wmstest");
        _CustomOverlay.open();
    }else{
        pEle.value="显示专题图";
        _CustomOverlay.close();
        _CustomOverlay=null;
    }
}
</script>
```

显示的专题图如下：




## 3.8 使用“信息查询接口”进行开发

使用信息查询接口可以进行属性和空间查询操作。

注意：如果 MMapService 程序和业务系统不在同一个域中，则建议下载代理程序: xmlhttpprox_1.1.jar,放到业务系统的\WEB-INF\lib 目录下，并在系统中进行注册，

如：

_MapApp.registerProx("/demo/servlet/com.map.service.XmlHttpProxy");

其中：demo 为部署的应用服务

### 3.8.1 点查询

```
var _subFields="casetype:类型;caseadd:发案地点;caseconten:案件描述";
var _table="dcxsaj2_pt";
var _imgURL="images/pnt/A.gif";
var pQuery=new QueryObject();
pQuery.queryType=1;
pQuery.tableName=_table;
pQuery.dispField="casetype";
pQuery.coords="116.41,39.92";
pQuery.addSubFields(_subFields);
pQuery.imgURL=_imgURL;

_MapApp.query(pQuery,drawFeatureObject);;
```

### 3.8.2 拉筐查询

```
var _subFields="casetype:类型;caseadd:发案地点;caseconten:案件描述";
var _table="dcxsaj2_pt";
var _imgURL="images/pnt/A.gif";
var pQuery=new QueryObject();
pQuery.queryType=2;
pQuery.tableName=_table;
pQuery.dispField="casetype";
pQuery.coords="116.39409,39.9209,116.4356,39.9458";
pQuery.addSubFields(_subFields);
pQuery.imgURL=_imgURL;

_MapApp.query(pQuery,drawFeatureObject);;
```

### 3.8.3 圆形查询

```
var _subFields="casetype:类型;caseadd:发案地点;caseconten:案件描述";
var _table="dcxsaj2_pt";
var _imgURL="images/pnt/A.gif";
var pQuery=new QueryObject();
pQuery.queryType=3;
pQuery.tableName=_table;
pQuery.dispField="casetype";
pQuery.coords="116.41655,39.93506,0.01281962947983106";
pQuery.addSubFields(_subFields);
pQuery.imgURL=_imgURL;

_MapApp.query(pQuery,drawFeatureObject);
```

### 3.8.4 多边形查询

```
var _subFields="casetype:类型;caseadd:发案地点;caseconten:案件描述";
var _table="dcxsaj2_pt";
var _imgURL="images/pnt/A.gif";
var pQuery=new QueryObject();
pQuery.queryType=4;
pQuery.tableName=_table;
pQuery.dispField="casetype";
```

```
pQuery.coords="116.40728,39.96289,116.39556,39.94482,116.41802,39.92773,116.42095,39.91113,116.42827,39.93506,116.40728,39.96289";
pQuery.addSubFields(_subFields);
pQuery.imgURL=_imgURL;

_MapApp.query(pQuery,drawFeatureObject);
```

### 3.8.5 点周边查询

```
var _subFields="casetype:类型;caseadd:发案地点;caseconten:案件描述";
var _table="dcxsaj2_pt";
var _imgURL="images/pnt/A.gif";
var pQuery=new QueryObject();
pQuery.queryType=5;
pQuery.tableName=_table;
pQuery.dispField="community";
pQuery.coords="116.406,39.927736";
pQuery.radius="1000";
pQuery.unit="meter";
pQuery.addSubFields(_subFields);
pQuery.imgURL=_imgURL;

_MapApp.query(pQuery,drawFeatureObject);
```

### 3.8.6 线周边查询

```
var _subFields="casetype:类型;caseadd:发案地点;caseconten:案件描述";
var _table="dcxsaj2_pt";
var _imgURL="images/pnt/A.gif";
var pQuery=new QueryObject();
pQuery.queryType=5;
pQuery.tableName=_table;
pQuery.dispField="casetype";
pQuery.coords="116.39941,39.92138,116.42583,39.91943";
pQuery.radius="1000";
pQuery.unit="meter";
pQuery.addSubFields(_subFields);
pQuery.imgURL=_imgURL;

_MapApp.query(pQuery,drawFeatureObject);
```

### 3.8.7 面周边查询

```
var _subFields="casetype:类型;caseadd:发案地点;caseconten:案件描述";
var _table="dcxsaj2_pt";
var _imgURL="images/pnt/A.gif";
var pQuery=new QueryObject();
pQuery.queryType=5;
pQuery.tableName=_table;
pQuery.dispField="casetype";
pQuery.coords="
116.3115,39.90436,116.36033,39.95124,116.42088,39.93561,116.42283,39.88092,116.3
4861,39.87311,116.3115,39.90436";
pQuery.radius="1000";
pQuery.unit="meter";
pQuery.addSubFields(_subFields);
pQuery.imgURL=_imgURL;

_MapApp.query(pQuery,drawFeatureObject);
```

### 3.8.8 归属地查询

```
var _subFields="casetype:类型;caseadd:发案地点;caseconten:案件描述";
var _table="dcxsaj2_PT";
var _imgURL="images/pnt/A.gif";
var pQuery=new QueryObject();
pQuery.queryType=7;
pQuery.tableName=_table;
pQuery.dispField="casetype";
pQuery.addSubFields(_subFields);
pQuery.filtertblName="pcs_PG";
pQuery.filterShape="shape";
pQuery.filterWhere="pcsName='ddd'";

_MapApp.query(pQuery,drawFeatureObject);
```

### 3.8.9 信息查询回调函数

系统内置了一个回调函数，用户在使用时可以适当进行修改

```
/**
 *Description: drawFeatureObject 信息查询回调函数
 *将信息查询返回的结果通过 getMapApp().getQueryResult()得到后，通过 drawLayers 绘制出来
 *@param null
 *@return 无
*/
function drawFeatureObject(){
        // 清除 map 中的所有绘制图形
        getMapApp().clearOverlays();
        // 获得空间信息查询返回的结果（结果为 MLayer 对象数组）
        var pLayers=getMapApp().getQueryResult();
        // 将查找到的 MLayer 对象数组绘制出来
        drawLayers(pLayers);
}

/**
 *Description:bIsContain 判断当前信息查询对象是否已经在查询新数组中
 *@param pQuerys []QueryObject 信息查询对象数组
 *@param pQuery QueryObject 信息查询对象
 *@return bool
*/
function bIsContain(pQuerys,pQuery){
        var bIs=false;
        for(var i=0;i<pQuerys.length;i++)
        {
           var pTmp=pQuerys[i];
           if(pTmp.queryType==pQuery.queryType &&        pTmp.coords==pQuery.coords
&& pTmp.radius==pQuery.radius )
           {
                bIs=true;
                break;
           }
        }
        return bIs;
}

/**
 *Description: addFeatInfo 绘制要素对象，并加入点击事件的响应函数
 *@param pObj Marker|Polyline|Polygon| MultiFeat 绘制的几何对象
```

```
 *@param strHtml String 点击时，信息显示的 html 字符串
 *@return 无
*/
function addFeatInfo(pObj, strHtml) {
        // 绘制几何对象
        getMapApp().addOverlay(pObj);
        // 添加几何对象点击事件的信息显示回调函数
        pObj.addListener("click", function () {pObj.openInfoWindowHtml(strHtml);});
}

/**
 *Description:drawLayers 绘制出 MLayer 数组中的所有要素
 *@param pLayers []MLayer 结果图层对象数组
 *@return 无
*/
function drawLayers(pLayers)
{
    try
    {
        // 防止查询对象重复绘制，将未绘制的查询对象 push 到 pDrawArea 其中，然后通过
bIsContain 函数检查
        var pDrawArea = new Array();

        // 根据查询的结果集(MLayer 对象数组 pLayers)，绘制出查询结果
        for(var i=0; i<pLayers.length; i++)
        {
                var pLayer = pLayers[i];
                if (!pLayer.bShow) continue;

                // 绘制查询的区域绘制查询区域和缓冲区区域
                switch (pLayer.queryObj.queryType)
                {
                        case 1:
                                break;
                        case 6:
                                 break;
                        default:
                                // 根据传回来的区域绘制查询区域和缓冲区区域
                                if( pLayer.queryObj.bufferAreaCoords != ""
&& !bIsContain(pDrawArea,pLayer.queryObj) )
                                {
                                    drawQueryArea(pLayer.queryObj);
                                    pDrawArea.push(pLayer.queryObj);
                                }
```

```
                    break;
            }

            // 绘制查找到的几何要素
            var pFeats = pLayer.features;
            var pFeat = null;
            var pCenterPoint = null; // 获得几何要素的中心点
            var strLine = ""; // 获得几何要素的坐标序列
            var pOverLay = null; // 用于绘制的对象
            for(var j=0; j<pFeats.length; j++)
            {
                pFeat = pFeats[j];
                pCenterPoint = pFeat.point;
                strLine = pFeat.linestr;

                switch (pLayer.type)
                {
                    case "nil" :
                            break;
                    case "point" :
                    case "multipoint" :
                            // 点和多点对象需要实例化 Icon 对象来标注出位置
                            var pIcon = new Icon();
                            pIcon.image = pLayer.imgURL;
                            pIcon.height = pLayer.imgHeight;
                            pIcon.width = pLayer.imgWidth;
                            pIcon.leftOffset = pLayer.leftOffset;
                            pIcon.topOffset = pLayer.topOffset;

                            // 点和多点对象显示时的标题设置
                            var pTitle = null;
                            if(pLayer.bIsLabel)
                            pTitle=new Title( Trim(pFeat.dispname), 12, 7 );

                            var pPointList = strLine.split(",");
                            if (pPointList.length>0)
                            {
                                for (var m=0; m<pPointList.length; m=m+2 )
                                {
                                    // 绘制出点或多点要素并显示其查询信息窗口
                                    var pPoint =new Point( pPointList[m],
pPointList[m+1] );
                                    pOverLay = new Marker( pPoint, pIcon, pTitle );
                                    addFeatInfo( pOverLay, pFeat.toHTML() );
```

```
                                }
                            }
                            pPointList = null;
                            break;
                        case "polyline" :
                        case "multipolyline" :
                            var pPointList = strLine.split(";");
                            if (pPointList.length>0)
                             {
                                for (var m=0; m<pPointList.length; m++ )
                                {
                                    // 绘制出多义线要素并显示其查询信息窗口
                                    pOverLay = new Polyline( pPointList[m], "#ff0000",
3 );
                                    addFeatInfo( pOverLay, pFeat.toHTML() );
                                }
                             }
                             pPointList = null;
                             break;
                        case "polygon" :
                        case "multipolygon" :
                            // 绘制出多边形要素并显示其查询信息窗口
                            pOverLay = new MultiFeat(strLine,"#ff00FF", 3,0.5,"blue");
                            addFeatInfo(pOverLay, pFeat.toHTML() );
                            break;
                        default : break;

                }
                pFeat = null;
                pCenterPoint = null;
                pOverLay = null;
                pLayer = null;
            }
        }
        catch (e)
        {
            alert("drawLayers:" + e.message);
        }
}

/**
 *Description:drawQueryArea 根据查询对象绘制出查询区域或缓冲区域（例如线周边查询的线周边区域以及线本身）
 *@param pQuery QueryObject 空间信息查询对象
```

```
 *@return 无
*/
function drawQueryArea(pQuery)
{
    var pQueryFilledColor = "green";
    var pQueryBorderColor = "#ff00FF";
    var pOpacity = 0.3;
    var pBOUNDS = null;
    var pOverlay = null;
    var strCoods = pQuery.bufferAreaCoords;
    try
    {
      switch(pQuery.queryType)
      {
        case 1:
            return;
        case 6:
            return;
        default:
            if(strCoods != "")
            {
                // 绘制查询区域
                var pOverlay= new MultiFeat( strCoods, pQueryBorderColor, 1, pOpacity,
pQueryFilledColor );
                if(pOverlay != null)
                {
                    getMapApp().addOverlay(pOverlay);
                }

                // 根据输入的查询坐标序列判断是线周边还是面周边，画出查询对象
                if (pQuery.queryType == 5)
                {
                    var pQueryCoords = pQuery.coords.replace(/\|/g,";")
                    var pAllPoints = pQueryCoords.split(",");
                    if(pAllPoints.length>2)
                    {
                        var pPointList = pQueryCoords.split(";");
                        for (var j=0; j<pPointList.length; j++)
                        {
                            var pPoints=pPointList[j].split(",");
                            if( pPoints[0]!=pPoints[pPoints.length-2] ||
pPoints[1]!=pPoints[pPoints.length-1] )
                            {
                                // 绘出线周边查询所对应线的形状
```

```
                                        pOverLay = new Polyline( pPointList[j], "#ff0000",
3 );
                                        if( pOverlay != null )
                                        {
                                        getMapApp().addOverlay(pOverLay);
                                        }
                                        }else
                                        {
                                                // 绘出面周边查询所对应的面的形状
                                                pOverlay = new
MultiFeat( pQueryCoords,"#ff00FF", 2,0.3,"blue");
                                                if( pOverlay != null )
                                                {
                                                        getMapApp().addOverlay(pOverlay);
                                                }
                                                break;
                                        }
                                }
                                pPointList = null;
                        }
                        pAllPoints = null;
                }
                }

                // 将查找的区域居中
                if (pOverlay != null)
                {
                pBOUNDS = pOverlay.getBOUNDS();
                        if(pBOUNDS != null)
                        {
                                pBOUNDS.scale(1.1); // BOUNDS 的周边扩大 1.1
                                getMapApp().centerAtBOUNDS(pBOUNDS);
                        }
                }
                break;
        }
    }
    catch (e)
    {
        alert("drawQueryArea:" + e.message);
    }
}
```

## 3.9 使用“信息编辑接口”进行开发

信息编辑接口可以对指定的表进行增加、修改和删除等操作。

### 3.9.1 增加

```
var _table="dcxsaj2_pt";
var pEdit=new EditObject("add","POINT",_table);
pEdit.addField("mapcaseid","00000001");
pEdit.addField("shape","116.4002,39.9");
pEdit.addField("caseconten","2006-04-12 鼎钧大厦 5001 室");

_MapApp.edit(pEdit,editCallBack);
```

### 3.9.2 删除

```
var _table="dcxsaj2_pt";
var pEdit=new EditObject("del","POINT",_table);
pEdit.where="mapcaseid='00000001'";

_MapApp.edit(pEdit,editCallBack);
```

### 3.9.3 修改

```
var _table="dcxsaj2_pt";
var pEdit=new EditObject("update","POINT",_table);
pEdit.addField("mapcaseid","00000002");
pEdit.addField("shape","116.4902,39.9");
pEdit.addField("caseconten","2006-04-12 鼎钧大厦 5001 室");
pEdit.where="mapcaseid='00000001'";

_MapApp.edit(pEdit,editCallBack);
```

### 3.9.4 信息编辑回调函数

```
/**
 *Description: editCallBack 信息编辑回调函数
```

```
 *@param bIsOK 编辑更新状态
 *@return 无
*/
function editCallBack(bIsOK)
{
        alert("更新状态:"+bIsOK);
}
```

## 3.10 使用“路径分析进行”进行开发

路径分析可以进行点(起始点坐标序列)到点(终止点坐标序列)的查询。

### 3.10.1 点到点的查询

```
        var pRouteObject = new RouteObject("118.73486,32.05078;118.73486,32.05077",
"118.80078,32.02636;118.80078,32.02637", "3");
        _MapApp.query(pRouteObject,drawRouteObject);
```

### 3.10.2 路径分析回调函数

```
/**
 *Description: drawRouteObject 路径查询回调函数
 *@param null
 *@return 无
*/
function drawRouteObject()
{
        // 获得路径查询结果集（结果为集路线集合对象数组：[]RoutePathArrObj）
        var uRoutePathResult = getMapApp().getRouteResult();
        // 清除地图中的已有的绘图对象
        getMapApp().clearOverlays();
        // 绘出查询结果点集合对象
        drawRoutePath( uRoutePathResult );
}

/**
 *Description:drawRoutePath 绘制出所有查到的路径以及查询起始点、终止点、起始匹配
点、终止匹配点
 *@param vRoutePathResult []RoutePathArrObj 路线集合对象数组
 *@return 无
*/
function drawRoutePath( vRoutePathResult )
```

```
{
        for (var i=0; i<vRoutePathResult.length; i++)
        {
            // 绘制出所有的路线
            var uRoutePathObjArr =vRoutePathResult [i];
            for (var j=0; j<uRoutePathObjArr.mRoutePathObjArr.length; j++)
            {
                var uRoutePathObj = uRoutePathObjArr.mRoutePathObjArr[j];
                for (var n=0; n<uRoutePathObj.mRoadObjArr.length; n++)
                {
                    var uRoadObj = uRoutePathObj.mRoadObjArr[n];
                    var uLine = new Polyline(uRoadObj.mCoords,"#ff0000", 2);
                    getMapApp().addOverlay(uLine);
                    uLine = null;
                    uRoadObj = null;
                }
                uRoutePathObj = null;
        }

            // 将起始点和起始匹配点相连，终止点和终止匹配点相连
            var uStrPoint = uRoutePathObjArr.mStartPoint.x + "," +
uRoutePathObjArr.mStartPoint.y + "," + uRoutePathObjArr.mStartMatchPoint.x + "," +
uRoutePathObjArr.mStartMatchPoint.y;
            var uJoinLine = new Polyline(uStrPoint,"#66FF00", 2);
            getMapApp().addOverlay(uJoinLine);
            uStrPoint = uRoutePathObjArr.mStopPoint.x + "," +
uRoutePathObjArr.mStopPoint.y + "," + uRoutePathObjArr.mStopMatchPoint.x + "," +
uRoutePathObjArr.mStopMatchPoint.y;
            uJoinLine = new Polyline(uStrPoint,"#66FF00", 2);
            getMapApp().addOverlay(uJoinLine);

            // 标出查询起始点和终止点以及起始匹配点和终止匹配点
            var uIcon = new Icon();
            uIcon.height = 98;
            uIcon.width = 98;
            uIcon.image = "WebRoot/images/start.gif";
            var uMarker = new Marker( uRoutePathObjArr.mStartPoint, uIcon);
            getMapApp().addOverlay(uMarker);

            uIcon.image = "WebRoot/images/end.gif";
            uMarker = new Marker(uRoutePathObjArr.mStopPoint, uIcon);
            getMapApp().addOverlay(uMarker);

            uIcon.image = "WebRoot/images/MatchStart.gif";
```

```
            uMarker = new Marker(uRoutePathObjArr.mStartMatchPoint, uIcon);
            getMapApp().addOverlay(uMarker);

            uIcon.image = "WebRoot/images/MatchEnd.gif";
            uMarker = new Marker(uRoutePathObjArr.mStopMatchPoint, uIcon);
            getMapApp().addOverlay(uMarker);
        }
}
```

## 3.11 使用“地址编码”进行开发

地址编码实现地名的查询。

### 3.11.1 点到点的查询

```
        var pAddress = document.all("vAddress").value;
        var pAddressClass = document.all("vAddressClass").value;
        var pGeoCode = new GeoCodeObject("0",pAddress, pAddressClass); //0 为正向查询，即通过地名查询坐标
      _MapApp.query(pGeoCode,drawGeoCodeObject);
```

### 3.11.2 地址查询回调函数

```
/**
 *Description: drawGeoCodeObject 地址编码查询回调函数
 *@param null
 *@return 无
*/
function drawGeoCodeObject ()
{
        // 获得地址编码查询结果集（结果集为地理编码查询对象数组：[]GeoSearchObject）
        var uGeocodeResult = getMapApp().getGeoCodeResult();
        // 清除地图中已有的绘图对象
        getMapApp().clearOverlays();
        // 绘出查询结果点集合中所有对象
        drawGeoCode(uGeocodeResult);
}

/**
 *Description: drawGeoCode 查询结果绘制函数，绘制出查询结果集合中_geoCodes 所有点状地址 Marker
```

```
 *@param pGeoCodes []GeoSearchObject 查找到的 GeoSearchObject 对象数组
*@return 无
*/
function drawGeoCode(pGeoCodes)
{
        // 设置查询时状态栏信息
        window.status = "绘制地址编码....";

        // 设置绘制查询结果点的显示样式
        var pIcon = new Icon();
        pIcon.image = "images/tack.gif";
        pIcon.height = 38;
        pIcon.width = 28;

        // 用 Marker 对象绘制出结果集合
        for(var iIndex = 0; iIndex < pGeoCodes.length; iIndex++)
        {
                var pGeoCodeObj = pGeoCodes[iIndex];
                var pMarker = new Marker( pGeoCodeObj.mPoint, pIcon );

                // 将点击事件所响应的地址加入到相应的 Marker 对象中
                addClickListener(pMarker, pGeoCodeObj.mAddress);

                // 清除对象变量
                pMarker = null;
                pGeoCodeObj = null;
        }
        pIcon = null;
        pGeoCodes = null;
}

/**
 *Description: addClickListener 将点击事件所响应的地址加入到相应的 Marker 对象中
 *@param vMarker Marker 一个 Marker 对象
 *@param vAddress String 加入到消息栏中的地址
 *@reutrn 无
*/
function addClickListener(vMarker,vAddress)
{
        vMarker.addListener("click",function(){vMarker.openInfoWindowHtml(vAddress);});
        getMapApp().addOverlay(vMarker);
}
```

## 3.12 监听地图状态变化操作

```
function queryMap(){
 //进行查询的操作
}
_MapApp. addMapChangeListener (queryMap);
```

## 3.13 多图对比

```
pMultiMaps=new MultiMaps();
_Map1=pMultiMaps.openMap("map1","2003");
_Map2=pMultiMaps.openMap("map2","2005");
_Map3=pMultiMaps.openMap("map3","2003");
_Map41=pMultiMaps.openMap("map4","2005");
```

显示界面图 3 如下：

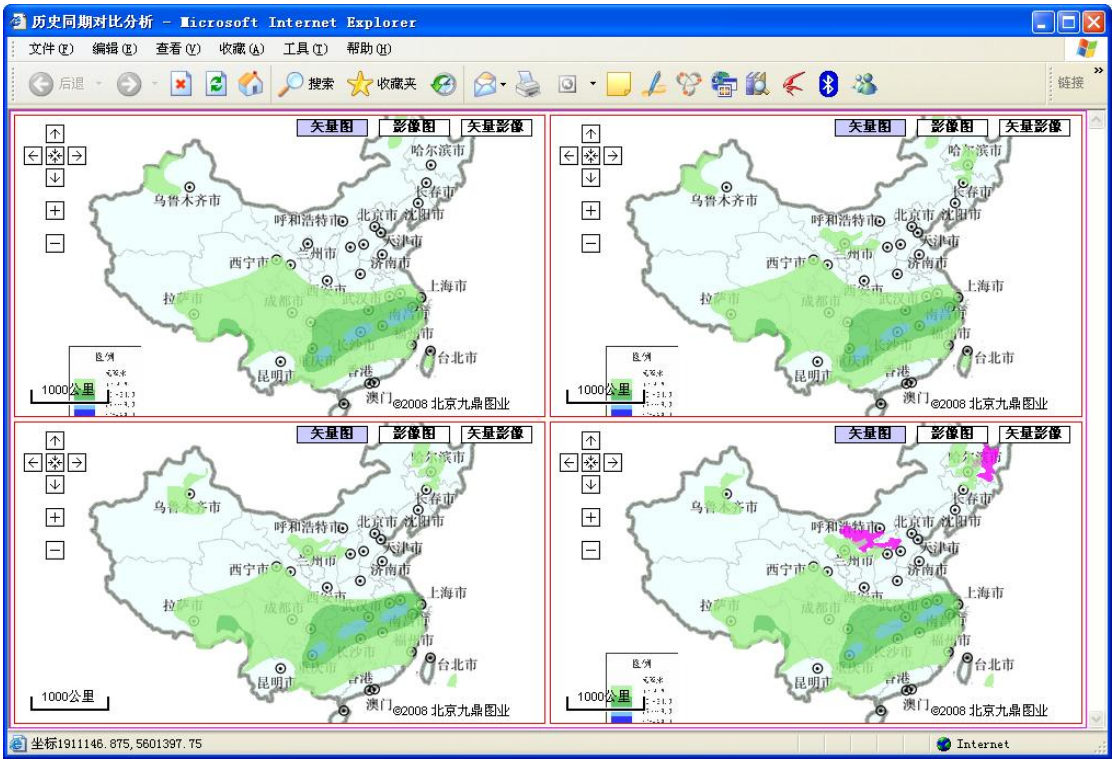



图 3 历史同期对比分析

# 4 附录 1 ——API 参考

## 4.1 MMap

地图控件

1) 构造函数

| 构造方法 | 参数 | 描述/返回说明（没有声名返回值即没有返回值） |
|---|---|---|
| MMap(container) | container：装载地图组件的容器<br>类型：Element 元素，例如一个 容器 | 在容器（container）创建一个实例，地图就在该容器下进行显示. |

2) 静态函数

| 构造方法 | 参数 | 描述/返回说明（没有声名返回值即没有返回值） |
|---|---|---|
| getMapTypes () | | 返回 MMapType 的数组 |
| addMapType(pMapType) | pMapType 为 MMapType 类型 | 无 |
| setCopyright(strMsg) | strMsg：字符型 | |
| setMapLevel (minLevel,maxLevel,initLevel) | | 设置地图显示级别范围 |
| setFullExtent(ele) | 数组 | 如对象 [70.12646,16.29784,138.75146,59.54784] |

3) 地图控件操作

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| addControl(pControl) | pControl：地图控件实例对象<br>类型：MapControl、MapStandControl 或 | 加入地图控件。目前可用控件：MMapControl, MMapStandControl, MMapSmallControl |

| | | |
|---|---|---|
| | MapSmallControl 的对象实例 | |
| showMapControl(strStyle)<br>showSmallMapControl(strStyle)<br>showStandMapControl(strStyle) | strStyle:为样式，字符类型 | 显示地图操作控件 |
| hideMapControl() | 无 | 隐藏地图操作控件 |
| showCopyright() | 无 | 显示版权信息 |
| hideCopyright() | 无 | 隐藏版权信息 |
| showMapScale() | 无 | 显示比例尺信息 |
| hideMapScale() | 无 | 隐藏比例尺信息 |
| showMapServer(strStyle) | strStyle:为样式，字符类型 | 显示“矢量地图、影像地图、矢量地图”按钮 |
| hideMapServer() | 无 | 隐藏“矢量地图、影像地图、矢量地图”按钮 |
| switchMapServer（iIndex） | iIndex：图像类型（取值 0：矢量地图，1：影像地图，2：矢量影像）<br>类型：Number | 在矢量地图、影像地图和矢量影像之间进行切换 |
| getMapName（） | 字符型 | 获取当前服务的名称，如：default,vect,sate |
| showMapCenter() | 无 | 显示地图中心点 |
| hideMapCenter() | 无 | 隐藏地图中心点 |
| hideMaskMap（） | 无 | 隐藏蒙版 |

**4) 鹰眼操作**

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| addOverView(pOverView) | pQverView：鹰眼对象实例<br>类型：MOverView | 在当前地图中增加一个鹰眼 |
| showOverView() | 无 | 显示鹰眼. |
| hideOverView() | 无 | 隐藏鹰眼 |
| reverseOverView() | | 鹰眼显示和不显示之间来回切换 |

| | | |
|---|---|---|
| | 无 | |

5) 地图操作

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| zoomIn() | 无 | 放大 |
| zoomInExt() | 无 | 拉框放大 |
| zoomOut() | 无 | 缩小 |
| zoomOutExt() | 无 | 拉框缩小 |
| pan(x,y) | 1)x：x 方向上的移动增量<br>2)y：y 方向上的移动增量<br>x 和 y 类型为：Number<br>函数中的参数可以为空 | 平移地图<br>说明：参数为空的情况下设置状态为平移状态<br>如果存在参数 x,y,则相应向 X,Y 轴方向移动 x 和 y 的距离增量 |
| measureLength() | 无 | 测距离 |
| measureArea() | 无 | 测面积 |
| fullExtent() | 无 | 全图显示. |
| gotoCenter() | 无 | 对中 |
| print() | 无 | 打印 |
| clear() | 无 | 清除 |
| centerAtLatLng(latLng) | latLng：指定的点对象<br>类型：Point | 地图对中到给定的点. |
| recenterOrPanToLatLng(latLng) | latLng：指定的点对象<br>类型：Point | 对中到给定的坐标，如果该点在当前视图上，则进行平移到该点为地图中心. |
| zoomTo(zoomLevel) | zoomLevel：地图等级<br>类型：Number | 缩放到给定的级别. |
| centerAtBounds(dInMinX,dInMinY,dInMaxX,dInMaxY) | 1)dInMinX：指定范围的 X 轴坐标上的最小值<br>2)dInMinY：指定范围的 Y 轴坐标上的最小值<br>3)dInMaxX：指定范围的 X 轴坐标上的最大值<br>4)dInMaxY：指定范围的 Y | 对指定的范围进行地图对中 |

| | | |
|---|---|---|
| | 轴坐标上的最大值<br>上边四参数类型均为：Number | |
| centerAndZoom (latLng, zoomLevel) | 1)latLng：坐标中心点<br>类型：Point<br>2)zoomLevel：指定的级别<br>类型：Number | 地图以给定的坐标为中心并缩放到给定的级别下. |
| openInfoWindow(pPoint,html) | 1)pPoint：指定的点位置<br>类型：Point<br>2)html：要显示的 html 格式的信息内容<br>类型：String | 在指定的位置显示信息 |

**6) 获得地图状态**

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| getCenterLatLng() | 无 | 返回地图中心的坐标<br>返回值类型：MPoint |
| getBoundsLatLng() | 无 | 返回当前视窗的经纬度边框<br>返回值类型：Bounds |
| getSpanLatLng() | 无 | 返回当前视窗的经纬度跨度<br>返回值类型：MRectangle |
| getZoomLevel() | 无 | 返回地图的当前级别<br>返回值类型：Number |
| getMaxLevel() | 无 | 返回地图的最大级别，返回类型为 int |
| getLevelOfBounds (dInMinX,dInMinY,dInMaxX,dInMaxY) | 1)dInMinX：指定范围的 X 轴坐标上的最小值<br>2)dInMinY：指定范围的 Y 轴坐标上的最小值<br>3)dInMaxX：指定范围的 X 轴坐标上的最大值<br>4)dInMaxY：指定范围的 Y 轴坐标上的最大值<br>上边四参数类型均为：Number<br>说明：此方法中的参数也可以为一个 Bounds 对象 | 获取指定的范围的级别<br>返回值类型：Number |

| | | |
|---|---|---|
| getDragMode() | 无 | 获取当前绘制模式<br>返回值类型：String |
| convert2Wpoint（x,y） | X:地理 X 方向坐标<br>Y:地理 Y 方向坐标<br>返回：屏幕坐标（Point 类型） | 从地理坐标转换成屏幕坐标 |
| convert2LonLat（x,y） | X:屏幕 X 方向坐标<br>Y:屏幕 Y 方向坐标<br>返回：地理坐标（Point 类型） | 从屏幕坐标转换成地理坐标 |
| getMouseLngLat() | 范围类型为 MPoint 类型 | 获取鼠标在地图上的坐标。 |

7) 设置地图绘制模式

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| changeDragMode(drawmode,inputform1,inputform2,callback) | 1)drawmode：操作模式（取值为：<br>"measure":测量<br>"pan":平移模式<br>“drawPoint”：获取坐标点<br>“drawCircle”：画圆<br>“drawRect”：画矩形<br>“drawPolyline”：画线<br>“drawPolygon”：画多边型）<br>类型：String<br>2)Inputform1(可选参数)：用于显示 X 坐标返回值的 html 元素的 ID<br>类型：String<br>3)Inputform2(可选参数)：用于显示 Y 坐标返回值的 html 元素的 ID<br>类型：String<br>4)callback(可选参数)：回调函数<br>类型：function | 改变操作模式 |
| clearDrawer() | | 清除绘制的对象 |
| drawPoint(callback) | 把点坐标（x,y）返回到回调函数中，字符类型，如:”116.5,39” | |
| drawCircle(callback) | 把圆坐标（x,y,radius）返回到回调函数中，字符类型，如:”116.5,39,0.002” | |
| drawRect(callback) | 把矩形坐标 | |

| | | |
|---|---|---|
| | （minx,miny,maxx,maxy）返回到回调函数中，字符类型，如:”116.5,39, 116.6,39.7” | |
| drawPolyline(callback) | | |
| drawPolygon(callback) | | |

**8) 信息叠加**

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| addOverlay(overlay, bDisRemovable) | 1)overlay：要添加的覆盖对象<br>类型：MMarker、MTitle、MCircle、MPolyline、MPolygon 或 MMultiFeat、MCustomOverlay 对象的实例<br>2)bDisRemovable：设置此对象否不可以删除，默认是：false，是可以删除的<br>类型：Boolean | 在当前地图上加入给定的对象 |
| removeOverlay(overlay) | overlay：要添加的覆盖对象<br>类型：类型：MMarker、MTitle、MCircle、MPolyline、MPolygon 或 MMultiFeat、MCustomOverlay 对象的实例 | 在地图上删除给定的对象. |
| clearOverlays() | 无 | 在地图上清除所有的对象. |
| getOverlays() | 无 | 返回信息叠加类的对象数组<br>返回值类型：[]MOverLay |
| getCurrentEditor() | 无 | 获取当前编辑的信息对象。<br>返回值类型：MOverLay |

**9) 信息查询、编辑**

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|

| | | |
|---|---|---|
| query(pQuery,callback) | 1)pQuery：查询对象<br>类型：QueryObject，GeoCodeObject，RouteObject 对象或 QueryObject，GeoCodeObject，RouteObject 对象的数组<br>2)callback：回调函数，用户获得查询的结果<br>类型：function drawFeatureObject, drawRouteObject, drawGeoCodeObject 三个函数之一 | 进行查询 |
| edit(pEdit) | pEdit：编辑对象<br>类型：EditObject 或 EditObject 对象的数组 | 进行编辑，其中：pEdit 为 EditObject 对象或 EditObject 对象的数组 |
| getQueryResult(); | 无 | 获取信息查询的结果<br>返回值类型：[]MLayer<br>说明：它是[]MLayer 结果集的 clone |
| setMLayer(pMLyrArr) | pMLyrArr：设置地图中图层的集合<br>类型：[]MLayer | 设置地图对象中包含的图层 |
| getGeoCodeResult() | 无 | 获取地址查询的结果：<br>返回值类型：[]GeoSearchObject |
| getRouteResult () | 无 | 获取路径查询的结果：<br>返回值类型：[]RoutePathArrObj |

**10) 监听地图状态变化操作**

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| addMapChangeListener(func) | func：当增加地图变换时调用的函数<br>类型：function | 增加地图状态变化时执行的操作，func 为函数 |
| removeMapChangeListener(func) | func：当删除地图变换时调用的函数<br>类型：function | 删除地图状态变化时执行的操作，func 为函数 |
| enableDblClick | | 设置鼠标双击事件 |

| | | |
|---|---|---|
| disableDblClick | | 取消鼠标双击事件 |
| addListener（action,func） | 1)action：添加事件的类型（取值可以为：”click”：点击；”dblclick”：双击；”mouseover”：鼠标在上边移动；contextmenu：右键信息）<br>类型：String<br>2)func：事件响应的函数<br>类型：function | |
| removeListener(action,func) | | |

**11) 版本地图的显示**

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| addVersion (strVersion,pMapServer) | 1)strVersion：版本信息<br>类型：String<br>2)pMapServer：地图设置对象<br>类型：MMapServer | 增加某版本的配置信息，pMapServer 为 MMapServer 类型 |
| showVersion (strVersion) | strVersion：版本信息<br>类型：String | 显示某版本的地图 |
| getVersionInfo() | 无 | 获得版本信息，如:”2005,2006”<br>返回值类型：String |

**12) 坐标跨度转换**

| 方法 | 参数 | 描述 |
|---|---|---|
| getMeter(pPoint,degree) | 1)pPoint：经纬度为参数的点状坐标<br>类型：Point<br>2)degree：跨度，经纬度下的跨度<br>类型：Number | 获得经纬度坐标下的跨度，例如给定跨度为 0.01 度，函数将根据给定点在地球上的位置返回一个跨度值（单位为米）<br>返回值类型：Number |
| getDegree (pPoint,meter) | 1)pPoint：大地坐标为参数的点状坐标<br>类型：Point<br>2)meter：跨度，大 | 获得大地坐标下的跨度，例如给定跨度为 1000 米，函数将根据给定点在地球上的位置返回一个跨度值（单 |

| | | |
|---|---|---|
| | 地坐标系下的跨度<br>类型：Number | 位为度）<br>返回值类型：Number |

13) 增加 Pop 菜单

| 方法 | 参数 | 描述 |
|---|---|---|
| showMenu（e,munus） | 1)e：为鼠标事件<br>类型：Point<br>2) munus：为菜单(MMenuObject) 数组,null 为分隔 | |
| | | |

## 4.2 MFlatProject

平面直角坐标投影

14) 构造函数

| 构造方法 | 参数 | 描述/返回说明（没有声名返回值即没有返回值） |
|---|---|---|
| MFlatProject () | | |

15) 动态函数

| 构造方法 | 参数 | 描述/返回说明（没有声名返回值即没有返回值） |
|---|---|---|
| fromLatLngToPixel (lat,lon,iLevel) | Lat:纬度<br>Lon:经度<br>iLevel:级别 | .返回屏幕的像素 |
| fromPixelToLatLng (x,y,iLevel) | X:行<br>Y:列<br>iLevel:级别 | 返回行、列、级别对应的像素 |
| getBounds (x,y,zoom) | X:列<br>Y:行<br>Zoom:级别 | 获取外包罗框 |

## 4.3 MBeijing54Project

北京 54 坐标投影

16) 构造函数

| 构造方法 | 参数 | 描述/返回说明（没有声名返回值即没有返回值） |
|---|---|---|

| | | |
|---|---|---|
| MBeijing54Project () | | |

17) 动态函数

| 构造方法 | 参数 | 描述/返回说明（没有声名返回值即没有返回值） |
|---|---|---|
| fromLatLngToPixel (lat,lon,iLevel) | Lat:纬度<br>Lon:经度<br>iLevel:级别 | .返回屏幕的像素 |
| fromPixelToLatLng (x,y,iLevel) | X:行<br>Y:列<br>iLevel:级别 | 返回行、列、级别对应的像素 |
| getBounds (x,y,zoom) | X:列<br>Y:行<br>Zoom:级别 | 获取外包罗框 |
| setMercator（bIsMercator） | 设置是否用经纬度在前端进行显示，系统默认为false. | |
| setOrginPoint（pOrginPoint） | 系统内置的 54 坐标原点为(105,0) | |
| setMapSpanScale（dSpanScale） | dSpanScale：设置变化参数，系统内置值为: 111194 北京市信息资源管理中心的地图需要设置为: 114699 | |

## 4.4 MMercatorProject

墨卡托投影

18) 构造函数

| 构造方法 | 参数 | 描述/返回说明（没有声名返回值即没有返回值） |
|---|---|---|
| MMercatorProject () | | |

19) 动态函数

| 构造方法 | 参数 | 描述/返回说明（没有声名返回值即没有返回值） |
|---|---|---|
| fromLatLngToPixel (lat,lon,iLevel) | Lat:纬度<br>Lon:经度<br>iLevel:级别 | .返回屏幕的像素 |

| | | |
|---|---|---|
| fromPixelToLatLng (x,y,iLevel) | X:行<br>Y:列<br>iLevel:级别 | 返回行、列、级别对应的像素 |
| getBounds (x,y,zoom) | X:列<br>Y:行<br>Zoom:级别 | 获取外包罗框 |

## 4.5 MTileLayer

自定义单元格层。

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MTileLayer(pCopyrights,fromlevel,tolevel, pTileLayerOptions) | pCopyrights: Copyrights 类型<br>fromlevel:最小级别，整形<br>tolevel:最大级别， 整形<br>pTileLayerOptions:选项 | . |

2) 方法

| 方法 | 参数 | 描述 |
|---|---|---|
| getTileUrl (x,y, zoom) | X：行<br>Y：列<br>Zoom：级别 | 根据行列级别返回地址<br>返回值：String |
| getBOUNDS （ x,y, zoom） | X：行<br>Y：列<br>Zoom：级别 | 返回该单元格的 BOUNDS |
| setOverlay（bOK） | bOK:设置是否为叠加层 | 该图层是叠加的图层 |
| setBlankMBR （pBounds,bIsTrans） | pBounds 为 MBounds 类型，如 new MBounds(73.375,17.8125,135.75,54.9375), bIsTrans 为 boolean 型，是否为透明的图片 | 设置范围外为空白图片 |
| setBlankImg（strURL） | strURL 为图片的 url 地址 | 设置空白图片 |
| | | |

## 4.6 MMapType

自定义地图类型。

1) 构造函数

| 构造函数 | 参数 | 描述 |
| --- | --- | --- |
| MMapType (layers, projection, title, opts) | layers: TileLayer 的数组<br>projection:投影类型。可以为空<br>title:地图的名词<br>opts:选项 | . |

2) 方法

| 方法 | 参数 | 描述 |
| --- | --- | --- |
| getName () | 无 | 获取地图名词<br>返回值：String |
| setTitle(title)<br>getTitle(); | title 为字符串 | 设置显示的名字<br>返回地图名字 |
| getTileLayers() | 无 | 获取地图的图层，类型为 TileLayer 的数组 |
| setCopyright(strCopyright) | 无 | 设置版本信息 |
| setClickFunc(func) | func 为函数 | |

## 4.7 MPoint

一个点代表一个二维的坐标，由 x 和 y 坐标表示。

1) 构造函数

| 构造函数 | 参数 | 描述 |
| --- | --- | --- |
| MPoint(x, y) | 1)x：X 坐标<br>类型：Number<br>2)y：Y 坐标<br>类型：Number | 用给定经纬坐标创建一个对象实例. |

2) 方法

| 方法 | 参数 | 描述 |
| --- | --- | --- |
| **getLng()** | | **获取经度** |
| **getLat()** | | **获取纬度** |
| **setLng(dLng)** | | **设置经度** |
| **setLat(dLat)** | | **设置纬度** |
| approxEquals(pPoint) | pPoint：待比较的点<br>类型：Point | 判断两点的坐标值是否近似相等（精度为 10E-6） |

| | | |
|---|---|---|
| | | 返回值：Boolean |
| equals（pPoint） | pPoint：待比较的点<br>类型：Point | 判断两点是否相等<br>返回值：Boolean |

## 4.8 MRect

矩形框对象

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MRect(width, height) | 1)width：矩形的宽度<br>类型：Number<br>2)height：矩形的高度<br>类型：Number | 用宽和高创建一个对象实例. |

3) 方法

| 方法 | 参数 | 描述 |
|---|---|---|
| approxEquals(pRect) | pRect：待比较的矩形<br>类型：Rect | 判断两个矩形的顶点是否近似相等<br>返回值：Boolean |
| equals（pRect） | pRect：待比较的矩形<br>类型：Rect | 判断连个矩形是否相等<br>返回值：Boolean |
| getWidth() | | 获取宽度 |
| getHeight() | | 获取纬度 |

## 4.9 MBounds

BOUNDS 代表一个外包络框的边界.

1) 构造函数

| 构造函数 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| MBounds (minX,minY,maxX,maxY) | 1)minX：X 方向上的最小值<br>2)minY：Y 方向上的最小值<br>3)maxX：X 方向上的最大值<br>4)maxY：Y 方向上的 | 用给最小最大坐标创建一个对象实例. |

| | | |
|---|---|---|
| | 最大值<br>以上四个参数的类型均为：Number | |

2) 静态方法

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| Intersection(pBOUNDS1,pBOUNDS2) | 此方法为 Bounds 类的静态方法<br>pBOUNDS1：一个 Bounds 对象<br>类型：Bounds<br>pBounds2：另一个 Bounds 对象<br>类型：Bounds | 返回指定的 2 个 BOUNDS 对象的相交部分的 BOUNDS<br>返回值类型：Bounds |
| union(pBounds1,pBounds2) | Static 方法<br>pBounds1：一个 Bounds 对象<br>类型：Bounds<br>pBounds2：另一个 Bounds 对象<br>类型：Bounds | 返回指定的 2 个 Bounds 对象的并集部分的 Bounds<br>返回值类型：Bounds |
| getCenterPoint() | 无 | 获取其中心点<br>返回值类型为：Point |

3) 方法

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| **getMinX()** | | **获取最小 X 坐标** |
| **getMinY()** | | **获取最小 Y 坐标** |
| **getMaxX()** | | **获取最大 X 坐标** |
| **getMaxY()** | | **获取最大 Y 坐标** |
| getCenterPoint () | 无 | 获取 BOUNDS 的中心点 |

| | | |
|---|---|---|
| | | 返回值类型：Point |
| getSpanX() | 无 | 获取 X 方向的跨度<br>返回值类型：Number |
| getSpanY() | 无 | 获取 Y 方向的跨度<br>返回值类型：Number |
| scale（e） | e：为包络框放大的倍数<br>类型：Number | BOUNDS 中心扩大其边框 |
| containsBounds(pBOUNDS) | pBOUNDS：以一个外包络框对象<br>类型：Bounds | 是否包含 pBOUNDS<br>返回值类型：Boolean |
| containsPoint(pPoint) | pPoint：一个点对象<br>类型：Point | 是否包含点<br>返回值类型：Boolean |
| extend (pExtStyle ) | pExtStyle：扩展类型<br>类型：pBOUNDS \| pPoint | 拓展边界，如果参数为 Point，则包络框扩展到包含此点为止；如果参数为 BOUNDS，者包络框扩展到包含此包络框为孩子 |

## 4.10 MOverLay

该对象为叠加信息类的积累

1) 方法

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| show() | 无 | 显示 |
| hide() | 无 | 隐藏 |
| setShowLevel(minLevel,maxLevel) | minLevel:最小级别<br>maxLevel:最大级别<br>注意：当前级别的计算以最小的级别以 0 为基数 | 在指定的级别中进行显示 |
| setZIndex(iIndex) | iIndex：图层的顺序<br>类型：Number | 设置其在图上的显示顺序，相当于设置其叠盖顺序。 |
| getZIndex() | 无 | 获得其在图上的显示顺序<br>返回值类型：Number |
| flash() | 无 | 闪烁，出现和不出现之间交 |

| | | |
|---|---|---|
| | | 替 3 次 |
| onclick | 无 | 触发 onclick 事件 |
| addDispStatus (iStartS,iEndS,iStatus) | 1)iStartS：开始周期<br>类型：Number<br>2)iEndS：结束周期<br>类型：Number<br>3)iStatus：状态(可以为以下 3 个值：'1'(显示)；'2'(隐藏)；'3'(闪烁))<br>类型：String | 增加显示的状态 |
| startMove（func） | func：回调函数<br>类型：function | 设定图元可以移动，并设定 Func 为回调函数，用于图元移动时响应此函数。<br>例如：可以调研该函数获取其坐标：_CurentOverLay.toString（） |
| setExtendStatus (iStartS,iEndS,dSScale,dEScale) | 1)iStartS：开始周期<br>2)iEndS：结束周期<br>3)dSScale：开始比例<br>4)dEScale：结束比例<br>类型：Number | 设置生长状态 |
| setPath (iStartS,iEndS,strPoints) | 1)iStartS：开始周期<br>返回类型：Number<br>2)iEndS：结束周期<br>类型：Number<br>3)strPoints：轨迹。如:"116.3,39.4,116.5,39.4"<br>类型：String | 设置路径 |
| showStatus (iSeq) | Iseq：周期<br>类型：Number | 显示某周期的状态 |
| scale(dScale) | dScale：缩放比例<br>类型：Number | 缩放 |
| play(bIsCenter) | bIsCenter：是否实时对中,默认为:false<br>类型：Boolean | 开始推演, |
| stop() | 无 | 停止推演 |
| enableEdit（func，bIsShowPop） | Func:为回调函数 | 可以编辑图形 |
| disableEdit() | 无 | 不可以编辑图形 |

| | | |
|---|---|---|
| getOpacity() | 无 | 获取透明度<br>返回值类型：Number |
| setOpacity(arg) | arg：透明度<br>类型：Number | 设置透明度 |
| getPoint() | 无 | 获得信息显示点的位置。如果没有通过下面方法设定，则信息显示框为默认的中点位置，如线的长度中间，面的外包络框的而中点，点即为本身 |
| setPoint(pPoint) | pPoint：设置信息显示框所在的位置<br>类型：Point | 设置信息显示框的定位点 |
| rotate(angle,container) | angle：旋转角<br>类型：Number<br>Container：为 html 的 Element 对象，如果为空，默认为本身的容器对象<br>类型:Element | 旋转叠加对象 |
| toJSON() | 生成 JSON 的字符串 | |

以下为已经实现的图元：

| 动态特性 | 点状图元 | 线状图元 | 面图元 | 文字图元 |
|---|---|---|---|---|
| 出现 | √ | √ | √ | √ |
| 隐藏 | √ | √ | √ | √ |
| 生长 | √ | √ | √ | |
| 按轨迹移动 | √ | | | |
| 闪烁 | √ | √ | √ | √ |
| | | | | |

## 4.11 MIcon

MsIcon 是图标对象。

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MIcon(name) | name（可选参数）：图标的名称、<br>类型：String | 创建一个图标对象 |

2) 方法

| 属性 | 描述 |
| --- | --- |
| setImage | 图片名称<br>类型：String |
| setWidth | 图片宽度<br>类型：Number |
| setHeight | 图片高度<br>类型：Number |
| setLeftOffset | 图片 X 方向偏移量<br>类型：Number |
| setTopOffset | 图片 Y 方向偏移量<br>类型：Number |

## 4.12 MTitle

Title 是标题类对象。父类为 iOverLay，继承了 iOverLay 的方法

1) 构造函数

| 构造函数 | 参数 | 描述 |
| --- | --- | --- |
| MTitle(name,fontSize,pos,font,color,bgColor,borderColor,borderSize) | 1)name：标题名称<br>类型：Stiring<br>2)fontSize：标题的字体<br>类型：String<br>3)pos：方位（0-7：分别代表 8 个方向）<br>类型：Number<br>4)font：字体类型<br>类型：Number<br>5)color：标题字体的颜色<br>类型：String<br>6)bgColor：背景颜色<br>类型：String<br>7)borderColor：边框颜色<br>类型：String<br>8)borderSize：边框宽度 | 创建一个标题类对象 |

| | | |
|---|---|---|
| | 类型：String | |

2) 属性（不用）

| 属性 | 描述 |
|---|---|
| name | 标题名称如:"北京"<br>类型：String |
| fontSize | 字体大小:如:14<br>类型：Number |
| pos | 位置:0~7<br>类型：Number |
| font | 字体,如:"宋体"<br>类型：String |
| color | 颜色，如："#0000ff"或“red”<br>类型：String |
| bgColor | 背景颜色<br>类型：String |
| bIsTransparent | 设置是否透明，默认为 false，不透明<br>类型：Boolean |

3) 方法

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| setPoint(pPoint) | pPoint：标题所在的位置<br>类型：Point | 设置图标显示位置 |
| getPoint() | 无 | 获取标题的位置<br>返回值类型：Point |

## 4.13 MMarker

Marker 是在地图上显示单个带有图标的对象. 父类为 iOverLay，继承了 iOverLay 的方法

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MMarker(point, icon) | 1)point：图标叠加对象所在的位置<br>类型：Point<br>2)icon：用于显示的图标对象 | 创建一个图标叠加对象 |

| | | |
|---|---|---|
| | 类型：Icon | |

2) 方法

| 方法 | 参数 | 描述/返回值说明(没有声名返回值即没有返回值) |
|---|---|---|
| openInfoWindowHtml(strHtml) | strHtml：显示信息框时显示的 html 内容<br>类型：String | 显示信息筐 |
| addListener(action,func) | 1)action：添加事件的类型（取值可以为："click"：点击；"dblclick"：双击；"mouseover"：鼠标在上边移动）<br>类型：String<br>2)func：事件响应的函数<br>类型：function | 加入响应事件 |
| getZIndex() | 无 | 获得其显示顺序<br>返回值类型：Number |
| setZIndex(iIndex) | iIndex：显示顺序<br>类型：Number | 设置其在图上的显示顺序，相当于设置其叠盖顺序。 |
| showTilte() | 无 | 显示标题 |
| hideTitle() | 无 | 隐藏标题 |
| setPoint(pPoint) | pPoint：设置图标叠加对象的位置<br>类型：Point | 设置图标叠加对象的位置 |
| getPoint() | 无 | 获取图标叠加对象的位置<br>返回值类型：Point |

## 4.14 MPolyline

线是由点的系列组成，父类为:iOverLay

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MPolyline(points, color, weight, opacity,arrow) | 1)points：线对象的坐标序列，可以是 String 类型或 Point 的数组类型<br>类型：[]Point 或 String<br>2)color：线的颜色（支持格式是如 "#0000ff" 或 | 构造一个线对象 |

| | | |
|---|---|---|
| | “red”).<br>类型：String<br>3)weight：线的宽度<br>类型：Number<br>4)opacity（可选）：线的透明度<br>类型：Number（0~1 之间）<br>5)arrow（可选）：线的方向（取值为：0（无方向（默认））；1（为正方向）；-1（为负方向））<br>类型：Number | |

2) 方法

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| openInfoWindowHtml(strHtml) | strHtml：显示信息框时显示的 html 内容<br>类型：String | 显示信息筐 |
| addListener（action,func） | 1)action：添加事件的类型（取值可以为：”click”：点击；”dblclick”：双击；”mouseover”：鼠标在上边移动）<br>类型：String<br>2)func：事件响应的函数<br>类型：function | 加入响应事件 |
| getZIndex() | 无 | 获取当前的图层序列 |
| setZIndex(iIndex) | iIndex：图层的顺序<br>类型：Number | 设置图层系列 |
| getLength() | 无 | 获取其长度（米）<br>返回值类型：Number |
| getBounds () | 无 | 获取其边框<br>返回值类型：Bounds |
| getColor() | 无 | 获得线的颜色<br>返回值类型：String |
| setColor(color) | color：线的颜色，如："#0000ff"或“red”<br>类型：String | 设置线的颜色 |
| getOpacity | | |
| getWidth() | 无 | 获得线宽<br>返回值类型：Number |

| | | |
|---|---|---|
| setWidth(width) | width：线宽<br>类型：Number | 设置线宽度 |
| getLineStyle() | 无 | 获得线型<br>返回值类型：Number |
| setLineStyle(arg) | arg：线的类型（可以取值："none","dash","dashdot","dot", "longdash","longdashdot","shortdash","shortdashdot","shortdashdotdot",,"longdashdotdot","shortdot"）<br>类型：String | 设置线型 |
| getPoints() | 无 | 获取点的坐标<br>返回值类型：[]Point |
| setPoints(pPoints) | pPoints：线对象的坐标序列，可以是String 类型或 Point 的数组类型<br>类型：[]Point 或 String | 设置线的点坐标序列 |
| setStartarrow（style） | Style 为以下值:<br>"None","Block","Classic","Diamond","Oval","Open" | 设置开始箭头的样式 |
| getStartarrow() | 返回值:<br>"None","Block","Classic","Diamond","Oval","Open" | 返回开始箭头的样式 |
| setEndarrow（style） | Style 为以下值:<br>"None","Block","Classic","Diamond","Oval","Open" | 设置结束箭头的样式 |
| getEndarrow() | 返回值:<br>"None","Block","Classic","Diamond","Oval","Open" | 返回结束箭头的样式 |
| setStartarrowWidth（width） | width 为:"Narrow","Medium","Wide" | 设置开始箭头的大小 |
| getStartarrowWidth（） | 返回值 为:<br>"Narrow","Medium","Wide" | 获得开始箭头的大小 |
| setEndarrowWidth（width） | width 为:"Narrow","Medium","Wide" | 设置结束箭头的大小 |
| getEndarrowWidth（） | 返回值 为:<br>"Narrow","Medium","Wide" | 获得结束箭头的大小 |

## 4.15 MPolygon

多边形对象（简单多边形）是首尾相连的点的系列组成，其父类为 Polyline，继承了 Polyline 的方法

1) 构造函数

| 函数 | 参数 | 描述 |
|---|---|---|
| MPolygon(points, color, weight, opacity,fillcolor) | 1)points：多边形对象的坐标序列，可以是 String 类型或 Point 的数组类型，但坐标的首尾必须相连，即起始坐标必须和终止坐标一样<br>类型：[]Point 或 String<br>2)color：边界的颜色（支持格式是如"#0000ff"或“red”).<br>类型：String<br>3)weight：边界的宽度<br>类型：Number<br>4)opacity：透明度<br>类型：Number（0~1 之间）<br>5)fillcolor：填充的颜色（支持格式是如"#0000ff" 或“red”).<br>类型：String | 构造一个多边形对象 |

2) 方法

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| openInfoWindowHtml(strHtml) | strHtml：显示信息框时显示的 html 内容<br>类型：String | 显示信息筐 |
| addListener(action,func) | 1)action：添加事件的类型（取值可以为：”click”：点击；”dblclick”：双击；”mouseover”：鼠标在上边移动）<br>类型：String<br>2)func：事件响应的函数<br>类型：function | 加入事件响应函数 |

| | | |
|---|---|---|
| getZIndex() | 无 | 获取当前的图层序列 |
| setZIndex(iIndex) | iIndex：图层的顺序<br>类型：Number | 设置图层系列 |
| getArea() | 无 | 获取其面积（平方米）<br>返回值类型：Number |
| getLength() | 无 | 获取其长度（米）<br>返回值类型：Number |
| getBounds () | 无 | 获取其边框<br>返回值类型：Bounds |
| getFillColor() | 无 | 获得填充颜色<br>返回值类型：String |
| setFillColor(color) | color：填充的颜色。如："#0000ff" 或 “red”<br>类型：String | 设置填充颜色 |
| getFillOpacity() | 无 | 获得填充透明度<br>返回值类型：Number |
| setFillOpacity(fillOpa) | fillOpa：透明度(0~1之间)<br>类型：Number | 设置填充透明度 |
| getFillPattern() | | |
| setFillPattern(strPanttern) | 填充模式，字符型。 | 枚举值为：<br>"bricks","checkerboard","circles", "crosshatch","crosshatch30", "crosshatch45","fishscales", "gray0","gray10","gray100", "gray15","gray20","gray25", "gray30","gray35","gray40", "gray45","gray5","gray50", "gray55","gray60","gray65", "gray70","gray75","gray80", "gray85","gray90","gray95", "hexagons","horizontal", "horizontalsaw","hs_bdiagonal", "hs_cross","hs_diagcross", "hs_fdiagonal","hs_horizontal", "hs_vertical","left30","left45", "leftshingle","octagons","right30", |

| | | |
|---|---|---|
| | | "right45","rightshingle", "smallfishscales","vertical", "verticalbricks","verticalleftshingle", "verticalrightshingle","verticalsaw" |
| setGradientColor(color) | 无 | 设置颜色填充过渡色 |
| getGradientColor() | 填充的颜色。如："#0000ff"或“red” 类型：String | 获得颜色填充过渡色 |

## 4.16 MCircle

圆是由中心点和半径组成，父类为 Polygon，继承了 Polygon 的方法

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MCircle(points, color, weight, opacity,fillcolor) | 1)points：圆对象的坐标序列（由经度、纬度、半径组成坐标序列，中间用”,”隔开，如”x1,y1,r”）<br>类型：String<br>2)color：边界的颜色（支持格式是如"#0000ff"或“red”).<br>类型：String<br>3)weight：边界的宽度<br>类型：Number<br>4)opacity：透明度<br>类型：Number（0~1 之间）<br>5)fillcolor：填充的颜色（支持格式是如"#0000ff"或“red”).<br>类型：String | 构造一个圆形对象 |

2) 方法

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| openInfoWindowHtml(strHtml) | strHtml：显示信息框时显示的 html 内容<br>类型：String | 显示信息筐 |
| addListener（action,fuct） | 1)action：添加事件的类型（取值可以为：”click”：点击；”dblclick”：双 | 添加事件响应函数 |

| | | |
|---|---|---|
| | 击；”mouseover”：鼠标在上边移动）<br>类型：String<br>2)func：事件响应的函数<br>类型：function | |
| getZIndex() | 无 | 获取当前的图层序列 |
| setZIndex(iIndex) | iIndex：图层的顺序<br>类型：Number | 设置图层系列 |
| getCenter() | 无 | 获取圆心<br>返回值类型：Point |
| getRadius() | 无 | 获取半径<br>返回值类型：Number |
| getArea() | 无 | 获取其面积（平方米）<br>返回值类型：Number |
| getLength() | 无 | 获取设置的半径（单位和当前地图的单位一致）<br>返回值类型：Number |
| getRadiusLength（） | 无 | 获取其长度（米）<br>返回值类型：Number |
| getBounds () | 无 | 获取其边框<br>返回值类型：Bounds |
| getFillColor() | 无 | 获得填充颜色<br>返回值类型：String |
| setFillColor(color) | color：填充的颜色。如："#0000ff"或“red”<br>类型：String | 设置填充颜色 |
| getFillOpacity() | 无 | 获得填充透明度<br>返回值类型：Number |
| setFillOpacity(fillOpa) | fillOpa：透明度(0~1 之间)<br>类型：Number | 设置填充透明度 |
| setFillPattern(strPanttern) | 填充模式，字符型。 | 枚举值为："bricks","checkerboard","circles", "crosshatch","crosshatch30", "crosshatch45","fishscales", "gray0","gray10","gray100", "gray15","gray20","gray25", "gray30","gray35","gray40", |

| | | |
|---|---|---|
| | | "gray45","gray5","gray50", "gray55","gray60","gray65", "gray70","gray75","gray80", "gray85","gray90","gray95", "hexagons","horizontal", "horizontalsaw","hs_bdiagonal", "hs_cross","hs_diagcross", "hs_fdiagonal","hs_horizontal", "hs_vertical","left30","left45", "leftshingle","octagons","right30", "right45","rightshingle", "smallfishscales","vertical", "verticalbricks","verticalleftshingle", "verticalrightshingle","verticalsaw" |

## 4.17 MRectangle

矩形有对角线的两点组成，父类为 Polygon，继承了 Polygon 的方法

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MRectangle(points, color, weight, opacity,fillcolor) | 1)points：矩形对象的坐标序列（形如"x1,y1,x2,y2"）<br>类型：String<br>2)color：边界的颜色（支持格式是如"#0000ff"或“red”).<br>类型：String<br>3)weight：边界的宽度<br>类型：Number<br>4)opacity：透明度<br>类型：Number（0~1 之间）<br>5)fillcolor：填充的颜色（支持格式是如"#0000ff"或“red”).<br>类型：String | 构造一个矩形对象. |

2) 方法

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| openInfoWindowHtml(htmlStr) | strHtml：显示信息框时显示的 html 内容<br>类型：String | 显示信息筐 |
| addListener（action,func） | 1)action：添加事件的类型（取值可以为：”click”：点击；”dblclick”：双击；”mouseover”：鼠标在上边移动）<br>类型：String<br>2)func：事件响应的函数<br>类型：function | 添加事件响应函数 |
| getZIndex() | 无 | 获取当前的图层序列 |
| setZIndex(iIndex) | iIndex：图层的顺序<br>类型：Number | 设置图层系列 |
| getArea() | 无 | 获取其面积（平方米）<br>返回值类型：Number |
| getLength() | 无 | 获取其长度（米）<br>返回值类型：Number |
| getBOUNDS() | 无 | 获取其边框<br>返回值类型：Bounds |
| getFillColor() | 无 | 获得填充颜色<br>返回值类型：String |
| setFillColor(color) | color：填充的颜色。如："#0000ff"或“red” <br>类型：String | 设置填充颜色 |
| getColor | | |
| setColor<br><br>getFillOpacity() | 无 | 获得填充透明度<br>返回值类型：Number |
| setFillOpacity(fillOpa) | fillOpa：透明度(0~1 之间)<br>类型：Number | 设置填充透明度 |
| setFillPattern(strPanttern) | 填充模式，字符型。 | 枚举值为："bricks","checkerboard","circles", |

| | | |
|---|---|---|
| | | "crosshatch","crosshatch30", "crosshatch45","fishscales", "gray0","gray10","gray100", "gray15","gray20","gray25", "gray30","gray35","gray40", "gray45","gray5","gray50", "gray55","gray60","gray65", "gray70","gray75","gray80", "gray85","gray90","gray95", "hexagons","horizontal", "horizontalsaw","hs_bdiagonal", "hs_cross","hs_diagcross", "hs_fdiagonal","hs_horizontal", "hs_vertical","left30","left45", "leftshingle","octagons","right30", "right45","rightshingle", "smallfishscales","vertical", "verticalbricks","verticalleftshingle", "verticalrightshingle","verticalsaw" |

## 4.18 MMultiFeat

复杂多边形，父类为 iOverLay，继承了 iOverLay 的方法

1) 构造函数

| 函数 | 参数 | 描述 |
|---|---|---|
| MMultiFeat (strPath, color, weight, opacity,fillcolor) | 1)strPath：代表多边形坐标序列，坐标之间用”,”隔开，内部有孔的坐标序列中间用”;”隔开，分离多边形坐标序列间用”\|”隔开<br>类型：String<br>2)color：边界的颜色（支持格式是如"#0000ff"或“red”).<br>类型：String<br>3)weight：边界的宽度<br>类型：Number | 构造复杂多边形对象 |

| | | |
|---|---|---|
| | 4)opacity：透明度<br>类型：Number（0~1 之间）<br>5)fillcolor：填充的颜色（支持格式是如"#0000ff"或“red”).<br>类型：String | |

## 4.19 MCustomOverlay

专题叠加类.（V6.2 版本以上提供）

2) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| CustomOverlay () | 无 | 创建专题图对象 |

1) 方法

| 方法 | 参数 | 描述 |
|---|---|---|
| setFormat(strFormat) | 字符型 | 地图 url 的格式，如:<br>/wms?BBOX=MBOX&WIDTH=MWIDTH&HEIGHT=MHEIGHT&…<br>其中，红色部分在程序内部自动把这些参数替换形式如下：<br>MBOX:116.0,39,117.0,40<br>MWIDTH:600<br>MHEIGHT:400 等<br>类型：String |
| getFormat() | 字符型 | 获取地图格式 |
| open(pMMap) | pMMap(可选)：地图对象<br>类型：MMap | 在指定的 MMap 对象中打开专题图，默认为当前地图 |
| close() | 无 | 关闭专题图显示 |
| getContainer() | 无 | 获取绘制容器<br>返回值类型：html 中的 Element 元素 |
| setLoadingFunc(callback) | callback：回调函数<br>类型：function | 设置装载过程中的回调函数 |
| setCompleteFunc(calback) | callback：回调函数<br>类型：function | 设置装载完成后的回调函数 |
| setRefreshTime(ms) | ms：刷新时间间隔 | 设置专题图自动刷新间隔，Ms:为毫秒数 |

| | | |
|---|---|---|
| | 类型：Number | |
| setTile(bIsTrue) | bIsTrue:boolean 类型。True:为采用多个单元格显示 | 设置专题图是否采用多单元格方式显示。 |
| setImgPNG | bIsTrue:boolean 类型。True:为采用 png 图片格式 | 方便透明显示 |

## 4.20 MGroundOverlay

在指定的外包箩筐的内显示图片

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MGroundOverlay (url, Bounds) | url:为图像的地址，string 类型或 string 的数组 | 创建专题图对象 |

2) 方法

| 方法 | 参数 | 描述 |
|---|---|---|
| initialize () | pMMap(可选)：地图对象<br>类型：MMap | 在指定的 MMap 对象中打开专题图，默认为当前地图 |
| remove () | 无 | 关闭专题图显示 |
| getBounds() | | |
| getUrl() | | |
| setInterval（iTime） | 整形，时间为微秒 | 设计时间间隔 |
| Play() | 无 | 循环播放 |
| stop() | 无 | 停止播放 |
| showFirst() | | 显示第一帧 |
| showLast() | | 显示最后一帧 |
| showNext() | | 显示下一帧 |
| showPre | | 显示前一帧 |

| | | |
|---|---|---|
| setImgPNG | bIsTrue:boolean 类型。True:为采用 png 图片格式 | 方便透明显示 |
| | | |
| setLoadingFunc(callback) | callback：回调函数 类型：function | 设置装载过程中的回调函数 |
| setCompleteFunc(calback) | callback：回调函数 类型：function | 设置装载完成后的回调函数 |

## 4.21 MTileLayerOverlay

把一个图层作为叠加对象

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MTileLayerOverlay (pTileLayer) | pTileLayer:为 TileLayer 对象 | 创建自定义图层 |

2) 方法

| 方法 | 参数 | 描述 |
|---|---|---|
| initialize () | pMMap(可选)：地图对象<br>类型：MMap | 在指定的 MMap 对象中打开专题图，默认为当前地图 |
| remove () | 无 | 关闭专题图显示 |
| setRefreshTime（itime） | Itime：为微妙 | 设置多长时间更新一次 |

## 4.22 MAction

箭头类，父类为 Polygon，继承了 Polygon 的方法把一个图层作为叠加对象

3) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MAction (strPath,color,weight,opacity,fillcolor,gradientColor) | strPath:点串,4 个控制点为一个箭头，8 个控制点为两个箭头，12 个控制点为三个箭头<br><br>color:线的颜色 | |

| | | |
|---|---|---|
| | weight:边界线的宽度<br>opacity：透明度<br>fillcolor：填充颜色<br>gradientColor：过渡颜色 | |

3) 方法

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| openInfoWindowHtml(htmlStr) | strHtml：显示信息框时显示的 html 内容<br>类型：String | 显示信息筐 |
| addListener（action,func） | 1)action：添加事件的类型（取值可以为：”click”：点击；”dblclick”：双击；”mouseover”：鼠标在上边移动）<br>类型：String<br>2)func：事件响应的函数<br>类型：function | 添加事件响应函数 |
| | | |
| setInterval（iTime） | 整形，时间为微秒 | 设计时间间隔 |
| setType（itype） | itype：<br>1：行动符合<br>2：单箭头<br>3：多箭头 | 设置时序的长度 |
| getType() | 整形，itype：<br>1：行动符合<br>2：单箭头<br>3：多箭头 | 获取当前的对象类型 |
| setGradientColor(color) | 无 | 设置颜色填充过渡色 |
| getGradientColor() | 填充的颜色。如："#0000ff"或“red”<br>类型：String | 获得颜色填充过渡色 |
| setRepeat(b) | b:boolean | 设置是否重复推演 |

| | | |
|---|---|---|
| getFillColor() | 无 | 获得填充颜色<br>返回值类型：String |
| setFillColor(color) | color：填充的颜色。如："#0000ff"或“red”<br>类型：String | 设置填充颜色 |
| getFillOpacity() | 无 | 获得填充透明度<br>返回值类型：Number |
| setFillOpacity(fillOpa) | fillOpa：透明度(0~1 之间)<br>类型：Number | 设置填充透明度 |
| setFillPattern(strPanttern) | 填充模式，字符型。 | 枚举值为："bricks","checkerboard","circles", "crosshatch","crosshatch30", "crosshatch45","fishscales", "gray0","gray10","gray100", "gray15","gray20","gray25", "gray30","gray35","gray40", "gray45","gray5","gray50", "gray55","gray60","gray65", "gray70","gray75","gray80", "gray85","gray90","gray95", "hexagons","horizontal", "horizontalsaw","hs_bdiagonal", "hs_cross","hs_diagcross", "hs_fdiagonal","hs_horizontal", "hs_vertical","left30","left45", "leftshingle","octagons","right30", "right45","rightshingle", "smallfishscales","vertical", "verticalbricks","verticalleftshingle", "verticalrightshingle","verticalsaw" |
| getColor() | 无 | 获得线的颜色<br>返回值类型：String |
| setColor(color) | color：线的颜色，如："#0000ff"或“red”<br>类型：String | 设置线的颜色 |

| | | |
|---|---|---|
| getWidth() | 无 | 获得线宽<br>返回值类型：Number |
| setWidth(width) | width：线宽<br>类型：Number | 设置线宽度 |
| getLineStyle() | 无 | 获得线型<br>返回值类型：Number |
| setLineStyle(arg) | arg：线的类型（可以取值："none","dash","dashdot","dot",<br>"longdash","longdashdot","shortdash","shortdashdot","shortdashdotdot",,"longdashdotdot","shortdot"）<br>类型：String | 设置线型 |
| getPoints() | 无 | 获取点的坐标<br>返回值类型：[]Point |
| setPoints(pPoints) | pPoints：线对象的坐标序列，可以是 String 类型或 Point 的数组类型<br>类型：[]Point 或 String | 设置线的点坐标序列 |

## 4.23 MGeoXml

MGeoXml 对象将可公开访问的网络服务器上储存的 XML 文件（如 KML 文件）中的地理内容添加到地图上。它实现 MiOverlay 界面，因此可使用 MMap.addOverlay() 方法添加到地图中。

**1) 构造函数**

| 构造函数 | 描述 |
|---|---|
| MGeoXml(urlOfXml, | 创建表示该 XML 文件的 MIOverlay。当 |

| | |
|---|---|
| callback) | MGeoXml 对象完成 XML 文件的装载时会调用可选的回调函数。 |

2) 方法

| 方法 | 返回值 | 描述 |
|---|---|---|
| getDefaultCenter() | MPoint | 以 lat/lng 返回默认视窗的中心点。只有在装载完文件后才可调用此函数。 |
| getDefaultBounds() | Bounds | 返回默认视窗的边界框。只有在装载完文件后才可调用此函数。 |
| hasLoaded() | Boolean | 检查 XML 文件是否已完成装载，如果完成则返回 true。如果尚未装载完 XML 文件，则此方法返回 false。 |
| hide() | none | 如果叠加层当前可见并且叠加层的 supportsHide() 方法返回 true |
| isHidden() | Boolean | 如果 MGeoXml 对象因被 MGeoXml.hide() 方法作了更改而当前不可见，则返回 true。否则，返回 false。 |
| loadedCorrectly() | Boolean | 检查 XML 文件是否已正确装载，如果是则返回 true。如果 XML 文件装载错误，则此方法返回 false。如果尚未装载完 XML 文件，则此方法的返回值未定义。 |
| show() | none | 如果 MGeoXml 对象创建的子叠加层当前不可见，则显示它们。 |
| supportsHide() | Boolean | 总是返回 true。 |

## 4.24 MOverView

鹰眼类.

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MOverView() | 无 | 创建一个鹰眼对象 |

2) 属性(不用)

| 属性 | 描述 |
|---|---|
| width | 鹰眼的宽,默认为:150<br>类型：Number |
| height | 鹰眼的高,默认为:150<br>类型：Number |
| minLevel | 鹰眼显示的最小级别，默认为:8<br>类型：Number |
| maxLevel | 鹰眼显示的最大级别，默认为:10<br>类型：Number |

3) 方法

| 属性 | 描述 |
|---|---|
| setWidth(iWidth) | 鹰眼的宽,默认为:150<br>类型：Number |
| setHeight(iHeight) | 鹰眼的高,默认为:150<br>类型：Number |
| SetMinLevel | 鹰眼显示的最小级别，默认为:8<br>类型：Number |
| setMaxLevel | 鹰眼显示的最大级别，默认为:10<br>类型：Number |

## 4.25 MQueryObject

信息查询类.

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MQueryObject() | 无 | 创建一个查询对象 |

2) 属性（不用）

| 属性 | 描述 |
| --- | --- |
| queryType | 查询类别，可取以下几个值：<br>1：点查询；2：拉筐查询;3：圆形查询;4:多变形查询;5:周边查询;6:模糊查询(默认)；7:属地模糊查询<br>类型：Number |
| tableName | 要查询的表<br>类型：String |
| layerName | 图层名称<br>类型：String |
| dispField | 要显示的字段名<br>类型：String |
| coordsType | 查询的坐标序列类型：” point”或”polyline”或”polygon”或”circle”等等<br>类型：String |
| coords | 查询对象的坐标序列<br>类型：String |
| radius | 缓冲周边的查询半径：适用周边查询和点查询。<br>类型：Number |
| unit | 查询的单位，默认为“degree”（可以是”meter”和”degree”）<br>类型：String |
| where | 查询条件，如:"name like '%社区%'"<br>类型：String |
| featurelimit | 要返回的记录数，默认为 10<br>类型：Number |
| beginrecord | 要查询的开始记录，默认为 0<br>类型：Number |
| filtertblName | 进行属地模糊查询的时候，属地对应图层表名【针对属地模糊查询】<br>类型：String |
| filtershape | 进行属地模糊查询的时候，属地对应图层表中对应的 shape 字段名【针对属地模糊查询】<br>类型：String |
| filterwhere | 进行属地模糊查询的时候，属地对应图层表中对应的查询条件【针对属地模糊查询】<br>类型：String |
| serviceSource | 设定空间查询服务地址，如果不设定即为默认 MMapService 的地址【V7.0.1 以上版本】<br>类型：String |

**3) 方法**

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| toxml() | 无 | 生成查询所请求的 XML<br>返回类型：String |
| addSubFields(strFields) | strFields：设置查询和返回的字段和显示名，如："字段名 1:显示名 1;字段名 2:显示名 2"<br>类型：String | 设置查询和返回的字段值。 |

## 4.26 MLayer

图层结果类.

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MLayer() | 无 | 创建图层结果 |

2) 属性（不用）

| 属性 | 描述 |
|---|---|
| tableName | 表名称<br>类型：String |
| layerName | 图层名称<br>类型：String |
| Bounds | 图层结果对象的外包络框<br>类型：Bounds（扩展用） |
| queryObj | 对应的查询对象<br>类型：QueryObject |
| editObj | 对应的编辑对象<br>类型：EditObject |
| features | 结果对象中的要素集合<br>类型：[]FeatureObject（FeatureObject 对象数组） |
| imgURL | 结果对象显示时的图片<br>类型：String |
| isVisable | 未知<br>类型：Boolean（扩展用） |
| minLevel | 未知<br>类型：Number（扩展用） |

| | |
|---|---|
| maxLevel | 未知<br>类型：Number（扩展用） |
| maxRecord | 总共的记录数<br>类型：Mumber |
| bShow | 当前层是否显示<br>类型：Boolean |

3) 方法

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| getFieldsSize() | 无 | 获取字段的长度<br>返回值类型：Number |

## 4.27 MFeatureObject

要素结果类.

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MFeatureObject() | 无 | 构造一个要素对象 |

2) 属性

| 属性 | 描述 |
|---|---|
| dispname | 要显示的字段名<br>类型：String |
| type | 要显示的类型：如”point”,”polyline”;”polygon”<br>类型：String |
| point | 中心点坐标<br>类型：Point |
| linestr | 要素的坐标系列<br>类型：String |

3) 方法

| 方法 | 参数 | 描述/返回值说明（没有声名返回值即没有返回值） |
|---|---|---|
| getField(index) | Index：第几个字段<br>类型：Number | 获取字段名称<br>返回值类型：String |
| getFieldValue(iField) | iField：字段名或字段的序号 | 获取对应字段的值 |

| | | |
|---|---|---|
| | 类型：Number\|String | |
| toHTML() | 无 | 将 FeatureObject 对象的字段和属性对应生生成 Table 的 HTML 字符串<br>返回值类型：String |
| getLength() | 无 | 获取其长度，单位为米。对于线、面才有<br>返回值类型：Number |
| getBounds () | 无 | 获取其外边框<br>返回值类型：Bounds |

## 4.28 MEditObject

编辑类.

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| EditObject(strOp,strgtype,table) | 1)strOp：编辑操作;<br>类型：String<br>2)strgtype：空间图层类型;<br>类型：String<br>3)table：空间图层名<br>类型：String | 创建一个编辑对象. |

2) 属性

| 属性 | 描述 |
|---|---|
| otype | 编辑操作：（"add" \| "del" \| "update"）<br>类型：String |
| gtype | 编辑的图层（"POINT" \| "MULTIPOINT " \| "POLYLINE" \| "MULTIPOLYLINE" \| "POLYGON" \| "MULTIPOLYGON" \| "NIL"）<br>类型：String |
| where | 表示查询的条件,主要用于删除和更新，如:"mapcaseid='0000001'"<br>类型：String |

3) 方法

| 方法 | 参数 | 描述 |
|---|---|---|
| toxml() | 无 | 编辑时生成请求的 XML<br>返回值类型：String |

| | | |
|---|---|---|
| addField(strField,strValue) | 1)strField：字段名<br>类型：String<br>2)strValue：<br>类型：String | 用于增加字段和字段的值 |

## 4.29 MGeoCodeObject

地址编码查询类.

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| GeoCodeObject(vMode,vAddress, vAddressClass) | 1)vMode ：查询方式。”0”（正向查询，即通过地址查询坐标）,”1”（反向查询，暂且不支持）；<br>类型：String<br>2）vAddress：要查找的地址.<br>类型：String<br>3)VaddressClass ：要查找地址的分类（例如要查找的地名为：电话局，地址分类可以填：海淀区，以此缩小查询范围）<br>类型：String | 构造一个地址编码查询对象. |

## 4.30 MGeoSearchObject

地理编码查询结果类.

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MGeoSearchObject() | 无 | 构造一个地址编码查询结果 |

2) 属性

| 属性 | 描述 |
|---|---|
| mID | 标识（用于唯一标识）<br>类型：String |

| | |
|---|---|
| mPoint | 坐标序列串<br>类型：String |
| mAddress | 地址串<br>类型：String |
| mAddressClass | 地址分类<br>类型：String |
| mSimilarity | 字面相似度<br>类型：Number |

## 4.31 MRouteObject

路径规划查询类.

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MRouteObject( vStartCoordsList, vStopCoordsList, vSearchType) | 1)vStartCoordsList ：查询起始点坐标序列<br>类型：String<br>2)vStopCoordsList ：查询终止点坐标序列<br>类型：String<br>3)vSearchType ：查询方式（3：表示距离最短原则；2：时间最短原则；1：代价最低原则。暂且只能选 3，即只能选择距离最短原则）<br>类型：String | 构造一个路径规划查询对象. |

## 4.32 MRoutePathArrObj

路线结果对象集合类（即起始点和终止点相同的路线的集合）.

在路径查询中涉及到的对象结构如下：

——RoutePathArrObj 路线结果集合对象（包含多条路线对象）
——RoutePathObj 路线对象（一条规划的路线，包含多条路）
——RoadObj 路的对象（一条路的对象包含多条子路）
——SubRoadObj 子路对象

1) 构造函数

| 构造函数 | 参数 | 描述 |
| --- | --- | --- |
| MRoutePathArrObj() | 无 | 构造一个路线结果对象集合 |

2) 属性

| 属性 | 描述 |
| --- | --- |
| mStartPoint | 起始点<br>类型：Point |
| mStopPoint | 终止点<br>类型：Point |
| mStartMatchPoint | 与起始点匹配的点（即与起始点最近的匹配点）<br>类型：Point |
| mStopMatchPoint | 与终止点匹配的点（即与终止点匹配的匹配点）<br>类型：Point |
| mSearchType | 查询方式（3：表示距离最短原则；2：时间最短原则；1：代价最低原则。暂且只能选 3，即只能选择距离最短原则）<br>类型：String |
| mRoutePathObjArr | 路线对象的集合<br>类型：[]RoutePathObj |

## 4.33 MRoutePathObj

路线对象类

1) 构造函数

| 构造函数 | 参数 | 描述 |
| --- | --- | --- |
| MRoutePathObj() | 无 | 构造一个路线对象 |

2) 属性

| 属性 | 描述 |
| --- | --- |
| mTime | 花费的时间（扩展用，"1 小时"）<br>类型：String |
| mLength | 路线的长度<br>类型：String |

| | |
|---|---|
| mCost | 路线和花费（扩展用）<br>类型：Number |

## 4.34 MRoadObj

路对象类

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MRoadObj() | 无 | 构造一段路对象（指在路线对象中所需要走的一段路，不是整条现实中的路，一般为部分路段） |

2) 属性

| 属性 | 描述 |
|---|---|
| mRoadName | 路名<br>类型：String |
| mLength | 这段路的长度（"7.764 公里"）<br>类型：String |
| mCoords | 这段路的坐标序列（x1,y1,x2,y2,.....）<br>类型：String |
| mSubRoadObjArr | 子路对象的集合<br>类型：[]SubRoadObj |

## 4.35 MSubRoadObj

子路对象类

1) 构造函数

| 构造函数 | 参数 | 描述 |
|---|---|---|
| SubRoadObj () | 无 | 构造一段路子路对象 |

2) 属性

| 属性 | 描述 |
|---|---|
| mSubRoadName | 子路名<br>类型：String |

| | |
|---|---|
| mSubRoadID | 子路的标识<br>类型：String |
| mLength | 子路的长度<br>类型：String |
| mSubCoords | 子路的最表序列（x1,y1,x2,y2,.....）<br>类型：String |

## 4.36 MMapServer

地图设置

**构造函数**

| 构造函数 | 参数 | 描述 |
|---|---|---|
| MMapServer (pVectMap,pSateMap, pVectSate) | 1)pVectMap：矢量地图对象服务的地址<br>2)pSateMap：影像地图对象服务的地址<br>3)pVectSate：矢量影像对象服务的地址<br>以上 3 个参数类型：String 或[]String<br>如：["http://172.26.15.21:8088/MServer", "http://172.26.15.21:8088/MServer"] | 构造地图设置对象 |

## 4.37 MMapControl

常用地图控件

1) 方法

| 方法 | 参数 | 描述 |
|---|---|---|
| getContainer() | 无 | 获取该对象的容器，以便设置其位置、样式<br>返回值类型：html 中的 Element 对象 |

## 4.38 MMapStandControl

标准地图空间

1) 方法

| 方法 | 参数 | 描述 |
|---|---|---|

| | | |
|---|---|---|
| getContainer() | 无 | 获取该对象的容器，以便设置其位置、样式<br>返回值类型：html 中的 Element 对象 |

## 4.39 MMapSmallControl

小控件地图

1) 方法

| 方法 | 参数 | 描述 |
|---|---|---|
| getContainer() | 无 | 获取该对象的容器，以便设置其位置、样式<br>返回值类型：html 的 Element 对象 |

## 4.40 MMultiMaps

历史对比对象，当其中一个地图移动是，另外的地图也处于该级别同时一点进行对中

1) 方法

| 方法 | 参数 | 描述 |
|---|---|---|
| openMap(map,version) | 1) map：地图象的 id 或容器对象<br>类型：String 或 html 中 Element 对象容器<br>2)version：版本信息<br>类型：String | 显示一个地图对象，同时返回该对象<br>返回值类型：MMap |

## 4.41 MMenuObject

20) 构造函数

| 构造方法 | 参数 | 描述/返回说明（没有声名返回值即没有返回值） |
|---|---|---|
| MMenuObject (name,func) | name：菜单名字<br>func：函数，可以是字符类型或 function | 构建菜单 |

## 4.42 MLog

调式信息

2) 静态方法

| 方法 | 参数 | 描述 |
| --- | --- | --- |
| debug (strInfo) | strInfo:为调试信息 | 设置调试信息 |
| write(strInfo) | strInfo:为调试信息 | 设置调试信息 |
| setDisplay(bShow) | bShow 为 booean | 设置是否显示 |

## 4.43 公共函数

1) **MCompatIE()**

作用：判断当前浏览器是否支持

返回值:bolean

输入参数:无

输入参数: pPoint，类型为 Point

2) **MCancelEventBubble (e)**

作用：阻止事件冒泡

返回值:无

输入参数:e 为事件

3) **MLoadJS（id,jsurl,callback）**

Id：string 类型

Jsurl：js 的地址

Callback：回调函数

4) **MShowMapInfo (strInfo,strTitle,strStyle)**

strInfo:为要显示的信息

strTitle：为主题

strStyle：为显示的位置

**5) MGISfromJSON (strJSON)**

从 JSON 字符串中生成对象，如：MMarker、MTitle、MPolyline、MPolygon、MCircle、MRectangle、MAction 等。

# 5 附录 2 ——常见开发问题（FAQ）

## 5.1 如何引用脚本的问题

为了能使得地图引擎能很好的升级和开发，建议引用脚本是在页面中采用以下方式进行引用:

<SCRIPT language="javascript" src="http://Ip:port/MyGIS/maps"></SCRIPT>

## 5.2 如何自定义地图

在地图矢量化前进行地图的自定定义：

```
function    initMap(){
      addMapType();
      _bMapProx=true;
      _MapApp = new MMap(document.getElementById("map"));
      _MapApp.showStandMapControl();
      _MapApp.showMapServer();
}
function addMapType(){
      var MAP_TYPES=MMap.getMapTypes();
   MAP_TYPES.clear();
   var pProj=new MFlatProject();
      var pTilelayer1=new MTileLayer(null,4,14,null);
      pTilelayer1.prj=pProj;
      pTilelayer1.getTileUrl=function(x,y,zoom){
      var pMBR=this.prj.getBounds(x,y,zoom);
      var url =null;
url="/testMyGIS/mygis?Service=getImage&mapname=default&x="+x+"&y="+y+"&Zoom="+zo
om+"&V="+_Ver;
url="../imsServlet?Service=getImage&BBOX="+pMBR.toString()+"&WIDTH=256&HEIGHT=2
56&Layers=3,1&ServiceName=china";
            return    url;
      }
      var pMapType=new MMapType([pTilelayer1],pProj,"矢量图",null);
      pMapType.setName("sate");
      MAP_TYPES.push(pMapType);
}
```

## 5.3 前台设置配置参数

```
//设置版权信息
MMap.setCopyright("九鼎图业提供服务");
//设置显示级别
MMap.setMapLevel(8,14);
//设置外全图显示范围
MMap.setFullExtent([70.12646,16.29784,138.75146,59.54784]);
```

## 5.4 关于专题图的叠加问题

使用专题图叠加接口进行开发，只要按照以下步骤即可：

1) 初始化专题图对象：

```
_CustomOverlay=new MCustomOverlay();
```

2) 设置专题图地址：

A) 假如地图访问地址为：http://*:8888/getMap?BBOX=486546,307556,488415,308635 &WIDTH=200&HEIGHT=300&FORMAT=GIF

则需要把红色部分分别变为：MBOX,MHEIGHT,MWIDTH，形成以下的地图地址：

http://*:8888/getMap?BBOX= MBOX &WIDTH=MWIDTH&HEIGHT=MHEIGHT&FORMAT=GIF

B) 在专题图格式中，把变换后的地址设置为专题图生成格式：

```
_CustomOverlay.format="http://*:8888/getMap?BBOX=MBOX&WIDTH=MWIDTH&HEIGHT=MHEIGHT&FORMAT=GIF";
//其中MBOX,MHEIGHT,MWIDTH在使用过程中是实时生成的
```

问题一：为什么需要变化格式呢？

答：由于获取地图需要传递地图的范围、图片大小，而在地图缩放、地图窗口变化时这些参数都有所变化，设置占位符是为了把这些参数留给系统，让系统自动填充这些参数。以便地图能在缩放情况下自动计算请求地图的范围、地图宽度和高度。

问题二、为什么需要设置图片格式？

答：由于IE浏览器对于PNG和GIF格式的透明处理方式不同，系统无法从二进制图形流中获取图像的格式，因而无法处理其透明情况，所以需要指定图片格式。当然，如果接口中没有Format等参数，只要在请求中填写占位符即可，如可以在请求中追加:“&image=GIF”等

3) 打开专题图：

```
_CustomOverlay.open();
```

## 5.5 关于专题图的叠加的回调问题

CustomOverlay 类目前增加两个回调

| 方法 | 描述 |
| --- | --- |

| | |
|---|---|
| open(pMMap) | 在指定的 MMap 对象中打开专题图，默认为当前地图 |
| close() | 关闭专题图显示 |
| getContainer() | 获取绘制容器 |
| setLoadingFunc(calback) | 设置装载过程中的回调函数 |
| setCompleteFunc(calback) | 设置装载完成后的回调函数 |
| setRefreshTime(ms) | 设置专题图自动刷新间隔，Ms:为毫秒 |

```
_CustomOverlay=new MCustomOverlay();

_CustomOverlay.setFormat(http://127.0.0.1:8888/testMServer/test/WMS.gif?BBOX=MBOX&WIDTH=MWIDTH&HEIGHT=MHEIGHT&SRS=EPSG:NONE&layers=13,3,11,16,6,9&version=1.0.0&service=WMS&FORMAT=PNG&TRANSPARENT=TRUE&request=getmap&ServiceName=yj)
_CustomOverlay.open();
_CustomOverlay.setLoadingFunc(function(){window.status='loading....'});
_CustomOverlay.setCompleteFunc(function(){window.status='loading....Complete'});
```

## 5.6 关于专题图的叠加自动刷新问题

CustomOverlay 类目前增加自动刷新的功能：

| 方法 | 描述 |
|---|---|
| open(pMMap) | 在指定的 MMap 对象中打开专题图，默认为当前地图 |
| close() | 关闭专题图显示 |
| getContainer() | 获取绘制容器 |
| setLoadingFunc(calback) | 设置装载过程中的回调函数 |
| setCompleteFunc(calback) | 设置装载完成后的回调函数 |
| setRefreshTime(ms) | 设置专题图自动刷新间隔，Ms:为毫秒 |

```
_CustomOverlay=new MCustomOverlay();
_CustomOverlay.setFormat(http://127.0.0.1:8888/testMServer/test/WMS.gif?BBOX=MBOX&WIDTH=MWIDTH&HEIGHT=MHEIGHT&SRS=EPSG:NONE&layers=13,3,11,16,6,9&version=1.0.0&service=WMS&FORMAT=PNG&TRANSPARENT=TRUE&request=getmap&ServiceName=yj);
_CustomOverlay.open();
```

_CustomOverlay.setRefreshTime(2000);

## 5.7 关于地图全屏和原屏进行切换的问题

### 问题代码

```
<HTML >
  <head>
        <meta http-equiv="content-type" content="text/html; charset=GB2312"/>
        <title>九鼎图业地图图片引擎接口例子</title>
<SCRIPT type="text/javascript" src="/MyGIS/mygis ">
</SCRIPT>

  </head>
  <body>
<table width="100%" height="100%" border="0" cellpadding="0" cellspacing="0">
     <tr id="topTr">
        <td height="20">
              </td>
     </tr>
     <tr>
        <td height="15">
                                    <table width="100%" height="100%" border="0"
cellpadding="0" cellspacing="0" background="images/map_bg.gif">
                                       <tr>
                                          <td width="30"> </td>
                                          <td width="40" align="center"><input type='button'
value="测距" onClick="_MapApp.changeDragMode('measure');"/></td>
                                          <td width="40" align="center"><input type='button'
value="测面" onClick="_MapApp.measureArea();"/></td>
                                          <td> </td>
                                          <td width="50" align="center"><a id="tgr5" name="tgr5"
href="#" class="L9" onclick="fullScreen(this);">全屏</a></td>
                                       </tr>
                                    </table>
                                 </td>
           </tr>
           <tr>
                 <td>
                                    <table width="100%" height="100%" border="0"
cellpadding="0" cellspacing="0" bgcolor="#FFFFFF">
                                       <tr>
```

```
                                        <td style="display:none;border-left: 1px solid
#00B2FF; border-bottom: 1px solid #00B2FF;" id="leftWin" width="250">

                                        </td>
                                    <td id="mapTd" style="border-left: 1px solid #00B2FF;
border-bottom: 1px solid #00B2FF; border-right: 1px solid #00B2FF">
                                        <div id="map"
style="width:100%;height:100%" ></div>

                                    </td>
                                </tr>
                            </table>
        </td>
            </tr>
        </table>


    </body>
    <script type="text/javascript">

var _MapApp;

window.onload=function() {

        _MapApp = new MMap(document.getElementById("map"));
        _MapApp.showMapControl();

}


    function fullScreen(pEle){
        var topTr = document.getElementById("topTr");
        if(!isFull){
            isFull = true;
            pEle.innerHTML = "还原";
            topTr.style.display = "none";

        } else{
            isFull = false;
            pEle. innerHTML = "全屏";
            topTr.style.display = "";
        }
    };
```

```
</script>
</html>
```

修正后代码

```
function fullScreen(pEle){
    var topTr = document.getElementById("topTr");
    if(!isFull){
        isFull = true;
        pEle.innerText = "还原";
        topTr.style.display = "none";

    }else{
        isFull = false;
        pEle.innerText = "全屏";
         topTr.style.display = "";
    }
}
```

## 5.8 如何编辑几何对象并返回坐标

1. 按照以下方式即可编辑对象：
   var pLine=new MPolyline(strPoint,"#ff0000", 5,0.7,1);
   pLine.setColor("blue");
   pLine.setDashStyle("dash");
   var strMsg="长度:"+pLine.getLength()+",面积:"+pLine.getArea();
   pLine.addListener("click",function(){pLine.openInfoWindowHtml(strMsg);});
   _MapApp.addOverlay(pLine);
   pLine.enableEdit();
2. 获取当前编辑对象:
   var pLine=_MapApp.getCurrentEditor()
3. 获取坐标:
   pLine.getPoints();

   或者:
   pLine.enableEdit(function (){
       alert(this. getPoints());
   },true)

## 5.9 关于线性化的编程

根据这个线图元算跟起始点米数的坐标串:

```
pFeat.getSpoints(300);
```

## 5.10 关于打印问题

打印函数为: _MapApp.print();

在常规没有参数情况下，打印是有页脚和页眉的。

如:


如果你想定制打印的方式可以

_MapApp.print("/MyGIS/css/print.css","页眉","页脚");

进行，其中 print.css 是样式表，内容如下:

```
//页眉的样式
.printtitle{
    background-color :white ;
    border : 0px ;
    font-size : 40px ;
    font-family :隶书 ;
    width : 100% ;
    height:40px;
    text-align : center ;
    }
```

```
//页脚的样式
.printbottom{
    background-color : white ;
    border : 0px ;
    font-size : 30px ;
    font-family : 隶书 ;
    width : 100% ;
    height:30px;
    text-align : right ;

}
//地图的窗口的样式
#MMaps_Container{
    border:2px solid black;
}
```

如: _MapApp.print("/MyGIS /css/print.css");

## 5.11 关于线的宽度的设置问题

### 对于线、面的线宽度设置

如何设置线的宽度？如周边 200 米？
设置方式 1：设置 20px 象素
pPolylin.setWidth("20px")
设置方式 2：设置距离，200 米
pPolylin.setWidth("200meter");

### 对于圆的半径设置

```
var pPoints="116.38481,39.95996,0.03";
var pCircle=new MCircle(pPoints,"#ff00FF", 2,0.5,"green");
_MapApp.addOverlay(pCircle);
pCircle.setRadius("3000meter");
//单位支持：degree/meter
```


## 5.12 如何阻止事件冒泡？

需要采用以下语句：
MCancelEventBubble(e);
return false;

例子：
以下有：contextmenu 的两个事件

```
<HTML><HEAD><TITLE class=title>显示 Marker 对象</TITLE>
<SCRIPT language=javascript src="http://192.168.1.108:8888/MyGIS/maps?">
</SCRIPT>
<SCRIPT id=www type=text/javascript reload="true">
_MapApp=null;
function onLoad() {
    /*
    if(typeof MMap =="undefined"){
        window.setTimeout("onLoad()",10);
        return;
    }*/
  //地图构造函数；
    _MapApp = new MMap(document.body);
    _MapApp.addOverView(new MOverView());
    _MapApp.showStandMapControl();
    _MapApp.showMapServer();
    _MapApp.showCopyright();

    _MapApp.addListener("contextmenu",function(e){
        pMenu=[];
        pMenu.push(new MMenuObject("该点对中",function(){
            _MapApp.centerAtMouse();
            }));
```

```
            pMenu.push(null);
            pMenu.push(new MMenuObject("放大",function(){
                _MapApp.zoomIn()
                }));
            pMenu.push(new MMenuObject("缩小",function(){
                _MapApp.zoomOut()
                }));
            _MapApp.showMenu(e,pMenu);
            //取消冒泡
            return false;
            });

        window.setTimeout("showMarker()",1000);
}
function showMarker(){
        var pIcon=new MIcon();
          pIcon.setImage("../examples/images/vehicle_active.gif");
          pIcon.setHeight(32);
          pIcon.setWidth(32);
          pIcon.setTopOffset(0);
          pIcon.setLeftOffset(0);
          if(typeof iPos =="undefined" || iPos==null)iPos=7;
          var marker = new MMarker(_MapApp.getCenterLatLng(),pIcon,new MTitle("警车
001",12,iPos,"宋体",null,null,"red","2"));
          _MapApp.addOverlay(marker,false);
marker.addListener("click",function(){marker.openInfoWindowHtml('<div dir="ltr">警车
001</div>');});
marker.addListener("contextmenu",function(e){
marker.openInfoWindowHtml('<div dir="ltr">警车 001</div>');
MCancelEventBubble(e);
return false;
});



}
</SCRIPT>
</HEAD>
<BODY onload="onLoad()">
</BODY></HTML>
```