文章编号:1674-2869(2012)06-0074-05

空气质量实时监测系统的内存泄漏

刘黎志[1,2],刘 君[2]
(1.武汉工程大学智能机器人湖北省重点实验室,湖北 武汉 430074;
2.武汉工程大学计算机科学与工程学院,湖北 武汉 430074)

摘 要:空气质量实时监测系统要求每5 min从数据库中取出监测数据,并将数据动态显示在Google地图标记的信息窗口中.直接使用封装谷歌地图的控件及在脚本中定义局部谷歌地图实例对象,都会发生内存泄漏,原因是脚本语言没有提供回收对象实例的方法,所有局部对象实例的回收都由系统内置的垃圾回收机制自动完成.而垃圾回收机制何时运行,完全由操作系统决定,程序无法控制.提出将地图对象及相关的标记、信息窗体对象全局化定义,使得相关对象只定义及实例化一次,避免实例对象的重复定义,解决了系统内存的泄漏问题.

关键词:内存泄漏;谷歌地图;异步脚本及标记语言;实时监测
中图分类号:TP311 **文献标识码**:A **doi**:10.3969/j.issn.1674-2869.2012.06.019

1 问题描述

近几年来,随着经济的快速发展,特别是房地产业和汽车产业的发展,空气污染日益严重的问题引起了环保部门的重视.国家环保部主要针对空气中的PM10(可吸入颗粒物)、SO_2、NO_2、CO、O_3等几种主要污染物进行实时监测,并要求地方环保部门每隔5 min上报一次各个自动化监测点的实时监测数据.空气质量实时监测系统要求将每5 min从数据库中读取一次数据,并将监测数据动态展示在Google地图上.空气质量实时监测系统的系统结构图如图1所示.

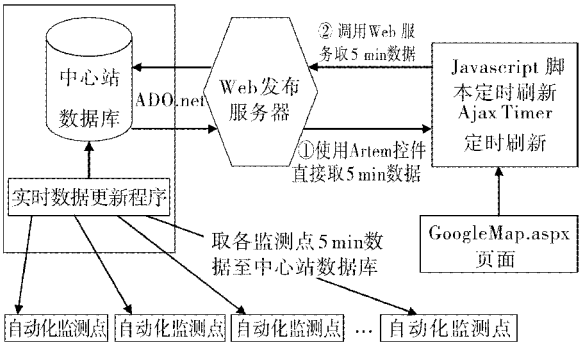

图1 空气质量实时监测系统结构图

Fig. 1 Air quality real-time monitoring system structure chart

空气质量实时监测系统的工作流程为:实时数据更新程序定时将各个自动化监测站点的数据更新到中心站的数据库服务器中;GooleMap.aspx页面使用Microsoft AJAX框架实现页面的异步刷新机制,Timer控件每300 s自动回发页面到Web发布服务器取5 min的监测数据在Google地图的标记信息窗口显示,实践中有两种方案.方案1使用著名的开源代码网站http://www.codeplex.com中的免费开源控件GoogleMap Control for ASP.NET[1],控件的标记名称为Artem.Artem控件封装Google Map API,使得开发人员可以直接在aspx页的Code Behind代码中针对该控件公开的编程接口编程,从而控制Google地图的使用逻辑.方案2直接使用Google Map API[2-3],仍然使用AJAX的异步机制,通过调用Web服务取中心站数据库中的5 min数据更新Google地图中的标记信息窗口.两种方案都不同程度的存在内存泄漏[4-5]问题.首先分析两种方案产生内存泄漏的原因,然后针对这些原因提出解决方案,从而解决内存泄漏问题.

2 内存泄漏问题分析

2.1 方案1内存泄漏问题分析

方案1直接使用封装的GoogleMap控件,并通

收稿日期:2011-09-26
基金项目:武汉工程大学智能机器人湖北省重点实验室开放基金(HBIR201006)
作者简介:刘黎志(1973-),男,湖北武汉人,副教授,硕士.研究方向:基于网络的计算机应用、商业智能、数据仓库及数据挖掘.

过对 Artem 控件公开的属性、方法、事件编程,从而控制 Google 地图中标记的信息窗口显示 5 min 的实时监测数据. 实时监测数据显示效果如图 2 所示.

图 2　实时监测数据显示

Fig. 2　Real – time monitoring data presentation

GoogleMap. aspx 页面内容为:

…… < asp: ScriptManager ID = " ScriptManager1 " runat = " server " > </asp: ScriptManager >

< asp: UpdatePanel ID = " UpdatePanel1 " runat = " server " >

< asp: Timer runat = " server " id = " tmUpdate " ontick = " tmUpdate_Tick " Interval = " 300000 " > </asp: Timer >

< artem: GoogleMap ID = " gmMain " runat = " server " Width = " 810px " Height = " 500px " Key = " Google Map Key " Latitude = " 30. 58 " Longitude = " 114. 31 " Zoom = " 11 " ZoomPanType = " Large3D " EnableDragging = " true " EnableContinuousZoom = " true " EnableScrollWheelZoom = " false " >

< Markers > </Markers >

</artem: GoogleMap >

</asp: UpdatePanel > ……

GoogleMap. aspx. cs 后台代码内容为:

protected void tmUpdate_Tick (object sender, EventArgs e)

{ gmMain. Markers. Clear( );//清空以前添加的标记

foreach (t_SStation ts in 从数据库中取得站点列表)

{ GoogleMarker gk = new GoogleMarker(站点经度,站点纬度); //实例化站点标记

gk. Title = ts. SStationName; //设置站点标题

gk. Text = “信息窗口中的显示内容,为从数据库中取得的 5 min 监测数据值”;

gmMain. Markers. Add(gk) ;}}

方案 1 设计页面每 5 min 回发一次,在后台代码中的 tmUpdate_Tick 事件中从中心站数据库中取得各个监测站点 5 min 的监测值,使用最新的 5 min值更新地图标记的信息窗口. 此方案从功能实现的角度看,是最简单的实现方法,但该方案内存泄漏严重,几乎每次回发,Windows 任务管理器显示承载 Google 地图的 IE 浏览器的 iexplore. exe 进程的内存使用增加 10 MB 左右. 若以系统可用内存为2 GB计算,2 048 MB/(10 MB Gbyte * (60/5)) =17 h 后,系统内存耗尽. 分析原因认为,该控件的设计者在每次页面刷新时并没有释放掉前一次的 GoogleMap 实例,而是又重新生成了一个实例,故产生内存泄漏.

2.2　方案 2 的内存泄漏问题分析

由于方案 1 中的 Artem 控件是封装发布的,从编程的角度出发,已经无法解决其在定时刷新数据时产生的内存泄漏问题. 故只能直接采用 Google Map API,用 Javascript 脚本语言编程实现. 具体实现描述如下.

* GoogleMap. aspx 页面内容为:

…… < asp: ScriptManager ID = " ScriptManager1" runat = "server" >

//定义 Web 服务调用地址

< Services > < asp: ServiceReference Path = " ~/MapService/MapDataWS. asmx "/ > </ Services >

//定义脚本语言调用地址

< Scripts > < asp: ScriptReference Path = " ~/Scripts/MapDataWS. js" / > </Scripts > </asp: ScriptManager >

< div id = " map _ canvas" style = " width: 810px; height: 520px;" > ……//定义 Google 地图显示层

* MapDataWS. asmx Web 服务内容为:

public List < Station > GetStations( )

{从数据库从取得监测站点的基本信息,包括名称、经度、纬度等,以集合形式返回; }

public List < StationCurData > GetCurData( )

{从数据库中取每个站点 5 min 的实时监测值,以集合形式返回,与 GetStation( )返回的集合一一对应; }

* MapDataWS. js 脚本内容为:

var mdsProxy; //定义全局 Web 服务代理类

var map; //定义全局 Google 地图实例

var sBasicInfo; //定义全局站点基本信息集合

function pageLoad(sender, args) {

if (! args. get_isPartialLoad( )) { mdsProxy = new MapDataWS( ); // 初始化 Web 服务代理类

var latlng = new google. maps. LatLng(30. 58, 114. 31); //定义 Google 地图

var myOptions = { zoom: 9, center: latlng, mapTypeId: google. maps. MapTypeId. ROADMAP };

map = new google. maps. Map($ get('map _ canvas'), myOptions);

mdsProxy. GetStations( ); //获取站点基本信息集合

window. setInterval(mdsProxy. GetCurData( ), 300000); //设置定时刷新,每 5 min 取一次监测值} }

function SucceededCallback (result, userContext, methodName) {

switch(methodName)

{case ("GetStations"): { sBasicInfo = result; break; } //获取返回的站点集合

case ("GetCurData"): //获取 5 min 实时监测数据,并生成与之对应的图标和信息窗口

{ for (i in result) { //定义站点的经纬度位置对象

var Latlng = new google. maps. LatLng (sBasicInfo [i]. SLatitude, sBasicInfo [i]. SLongitude);

//根据站点基本信息定义地图标记

var mark = new google. maps. Marker ({ position: Latlng, map: map, title: sBasicInfo [i]. SStationName });

//用取得的 5 min 监测数据定义信息窗口显示内容

var contentString = result[i]. SStationName + "PM10: " + result[i]. PM10. toString( ) + …….

//定义信息窗口,及与地图标记相关联的 click 事件

var infowindow = new google. maps. InfoWindow({ content: contentStr });

google. maps. event. addListener(mark, 'click', function ( ) { infowindow. open(map, mark); });

break; } }

方案 2 使用 Google Map API 直接在脚本中定义地图、地图标记及与地图标记对应的信息窗口,保证了地图实例对象只在页面初始化时定义一次,这就避免了方案 1 中每次页面刷新就重复定义一次地图实例的问题. 但经过仔细观察,仍然存在内存泄漏问题,只是没有方案 1 那么明显,大约每回发两次,Windows 任务管理器显示承载 Google 地图的 IE 浏览器的 iexplore. exe 进程的内存使用增加 1 MB 左右. 以系统可用内存为 2 GB 计算, 2 048 MB/(0. 5 MB * (60/5)) = 341 h 后,系统内存耗尽,还是不能满足不间断实时监测的需要. 经分析得出,虽然地图实例没有在每次页面刷新时生成,但仍然生成地图标记对象、信息窗口对象,且前次生成的地图标记及信息窗口对象均没有被 IE 浏览器释放,故产生内存泄漏.

3 内存泄漏问题的解决

分析方案 1 及方案 2 产生内存泄漏的原因,得出的结论是 IE 浏览器不会自动的回收由 Javascript 脚本语言定义的 Google 地图、标记、信息窗口实例对象,故不论采用何种方式定时刷新页面,只要存在实例对象的重复定义,就会产生内存泄漏. 由于不知道 IE 浏览器如何回收自定义对象实例的机制,故要避免内存泄漏,解决的方案只有避免对象的重复定义,具体的方法描述如下:

即在方案 2 的 MapDataWS. js 脚本全局定义部分,增加 var infowindowArray = [ ] 及 var markerArray = [ ]全局数组定义,分别表示信息窗口数组及地图标记数组,用于保存和各个监测站点对应的地图标记实例对象及信息窗口对象. 改进的更新算法为:

```
function SucceededCallback ( result, userContext,methodName) {
switch(methodName)
{ …… case ("GetCurData"):
{ if (infowindowArray. length = = 0) {//保证只生成一次标记实例、信息窗口实例
for (var i in sBasicInfo) {
var Latlng = new google. maps. LatLng ( sBasicInfo [ i ]. SLatitude, sBasicInfo [ i ]. SLongitude);
var mark = new google. maps. Marker ({ position: Latlng, map: map, title: sBasicInfo [ i ]. SStationName });
//保存标记实例到 markerArray 数组
markerArray. push(mark);
var infowindow = new google. maps. InfoWindow({ content: MakeInfoWindowContent(i, result) });
//保存信息窗口实例到 infowindowArray 数组
infowindowArray. push(infowindow);
google. maps. event. addListener(mark, 'click', function () { infowindow. open(map,mark); });
}}
//页面定时读取各个监测站点 5 min 值,更新对应的信息窗口内容,但不再重复生成信息窗口实例
else {for (var i in result) {infowindowArray [ i ]. setContent ( MakeInfoWindowContent ( i, result)); }}break;
} ……
```

其中 MakeInfoWindowContent 函数根据监测站点 5 min 值,生成信息窗口展示的信息,具体实现不再描述. 由于地图、标记、信息窗口对象实例均只生成一次,系统在 IE 浏览器中运行时,Windows 任务管理器显示承载 Google 地图的 IE 浏览器的 iexplore. exe 进程的内存不再增加,内存泄漏问题得到解决. 各方案的 iexplore. exe 进程的内存泄漏对比如图 3 所示,随着时间的增加,没有内存使用增长则表示没有内存泄漏.

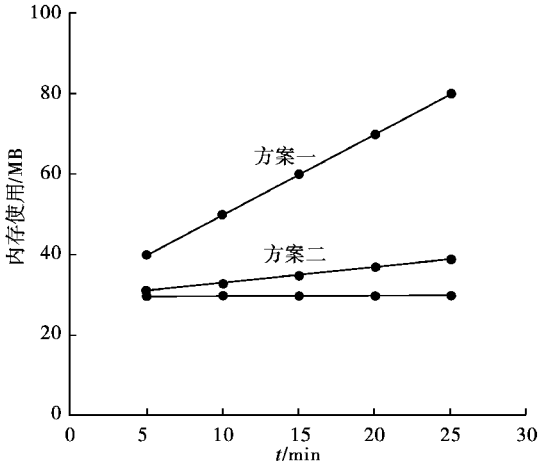

图 3　内存使用对比图

Fig. 3　Memory using comparison chart

4　结　语

利用地理位置信息来展示与之相关数据信息变化的技术在 Internet 网上已经得到广泛应用,特别是通过定时异步刷新数据信息,用户可以直接通过 Web 浏览器实时观察数据变化,地理信息和数据展示就更加的生动形象. IE 浏览器不会主动释放由 Javascript 脚本语言声明的对象实例,从而导致内存泄漏. 如果这个问题不解决,数据的实时监控就无法实现连续不间断运行. 通过对空气质量实时监测系统中内存泄漏的问题的分析,找到导致系统内存泄漏的原因,并最终解决内存泄漏问题,希望对从事相关工作的同仁有所借鉴.

参考文献:

[1] Velio Ivanov. Google Map Control for ASP. NET [EB/OL]. http://googlemap. codeplex. com. 2011 - 02 - 16.

[2] 黄瑶. C/C + + 程序内存泄漏检测和分析技术的研究与实现[D]. 北京:北京航空航天大学软件学院. 2004.

[3] 胡燕,龚育昌. 一种混合式内存泄漏静态检测方法. 小型微型计算机系统[J]. 2008,29(10): 1935 - 1939.

[4] Google Code. Google Maps JavaScript API V3 [EB/OL]. http://code. google. com/intl/zh - CN/zh - CN/apis/maps/documentation/javascript/. 2010 - 06 - 15.

[5] Regehr J, Cooprider N, Archer W. Efficient type and memory safety for tiny embedded systems [C]//In Proceedings of the PLOS 2006 Workshop on Linguistic Support for Modern Operating Systems. California: San Jose, 2006: 64 - 68.

Memory leak in air quality real – time monitoring system

LIU Li-zhi[1,2], *LIU Jun*[2]
(Hubei Province Key Laboratory of Intelligent Robot, Wuhan Institute of Technology, Wuhan 430074, China;
School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan 430074, China)

Abstract: Air quality monitoring system gets data from database every 5 minutes and displays data in the information window of Google map marker dynamically. A memory leak occurs directly using map controls and defining local Google map object instances in Javascript, because Javascript doesn't provide the method to recycle object instances. All local object instances are recycled by garbage collection mechanism within operation system. But when and how the garbage collection mechanism to run is decided entirely by operation system, programmer can't control. We propose that map objects and tags, information windows are defined only once as global object to avoid duplicate definition of instance objects, so as to solve the memory leak problems.
Key words: memory leak; goggle map; asynchronous javascript and XML; real – time monitoring
本文编辑:陈小平

☆

(上接第64页)

Study on O_2 plasma surface modification of aramid fiber Ⅲ

JI Jia-you, SU Qi, CHEN Zhuo, HUANG Zhi-xiong
(School of Materials Science and Engineering, Wuhan University of Technology, Wuhan 430070, China)

Abstract: In order to improve the interfacial bonding properties of the aramid fibers reinforced composites, O_2 plasma treatment was used to improve modification of aramid fiber, and the epoxy resin and aramid fibers were praepared in this paper. Scanning electron microscope (SEM), X – ray photoelectron spectroscope (XPS), Dynamic contact angle (DCA) were used to characterize the change of fiber surface, tensile performance and bending performance of aramid fiber were tested before and after plasma treatment. The result shows that O/C of aramid fiber is increased, the mean surface of fiber is sinificantly enlarged, and the contact angle with water is lowered after plasma treatment, the bending intensity is increased by 30%.
Key words: plasma treatment; interfacial performance; aramid fiber; composites
本文编辑:龚晓宁