# [SharePoint 2007图文开发教程(1)---简介，安装，配置及创建Web应用程序](http://www.cnblogs.com/gaoweipeng/archive/2009/10/23/1587665.html)

**简介**   Microsoft Office SharePoint Server 2007 是一个服务器功能集成套件，它提供全面的内容管理和企业搜索、加速共享业务流程并便利跨界限信息共享以更好地了解业务，从而有助于提高组织的工作效率。Office SharePoint Server 2007 通过一个集成平台而不是依靠分散的系统来支持整个企业内的所有 Intranet、Extranet 和 Web 应用程序。此外，该协作和内容管理服务器还为 IT 专业人员和开发人员提供了实现服务器管理、应用程序可扩展性和互操作性所需的平台和工具。当然，现在最新版本的SharePoint是2010，大概在今年的十一月份推出测试版，并且Visual Studio 2010中也集成了SharePoint的开发，所以学习如何使用SharePoint开发是很有必要的！有关SharePoint更详细的介绍，大家可以参看：<http://baike.baidu.com/view/260261.htm>



SharePoint 的**主要应用领域**：



简单的介绍就到这里，开始我们的SharePoint之旅。

**安装**



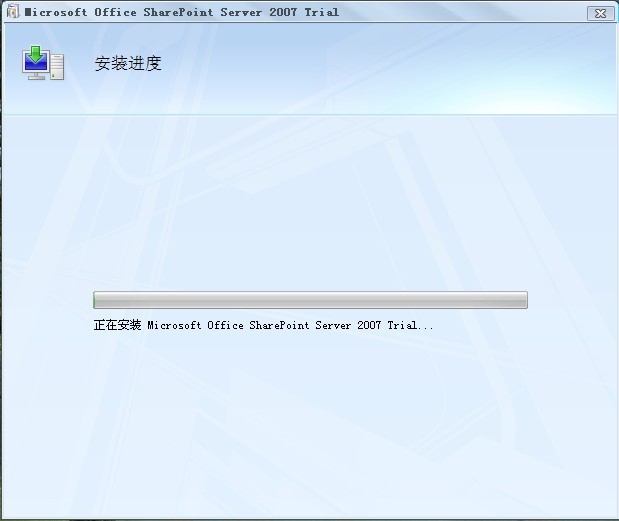
安装SharePoint可以说相当的简单，但是前提是需要一些必要的条件：

1、安装Microsoft Windows 2003。  
2、安装Microsoft SQL Server 2005。  
3、安装Microsoft Visual Studio 2005。  
4、接着是最轻易忘记的一环：安装Microsoft Windows Workflow Foundation 2.2以上的版本。  
5、最后就是安装Microsoft Office SharePoint Server 2007和Microsoft Office SharePoint Designer 2007。  
相关文件的下载：<http://www.microsoft.com/downloads/details.aspx?FamilyID=2e6e5a9c-ebf6-4f7f-8467-f4de6bd6b831&DisplayLang=zh-cn>。这里有适用版本的SharePoint Server 2007，KEY可以在页面中找到。

下载完毕后，运行下载的OfficeServerwithSP1.exe，开始安装：

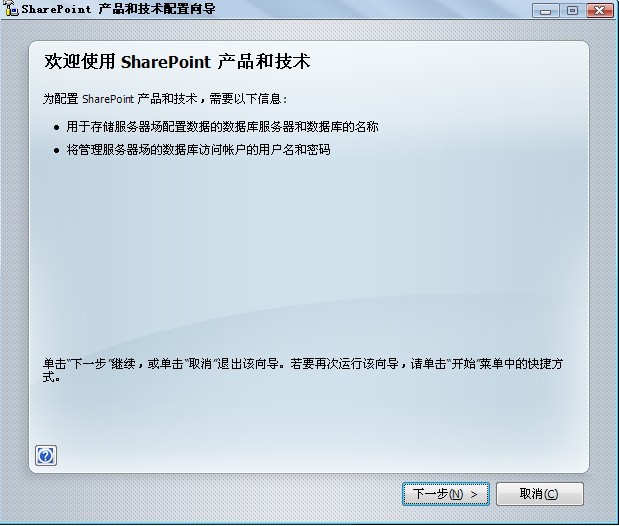


输入KEY以后，点击高级，我们可以修改安装路径等信息，接下来我们需要做的就是等待.......



OK，到这里SharePoint的安装就完成啦。接下来我们开始配置SharePoint。

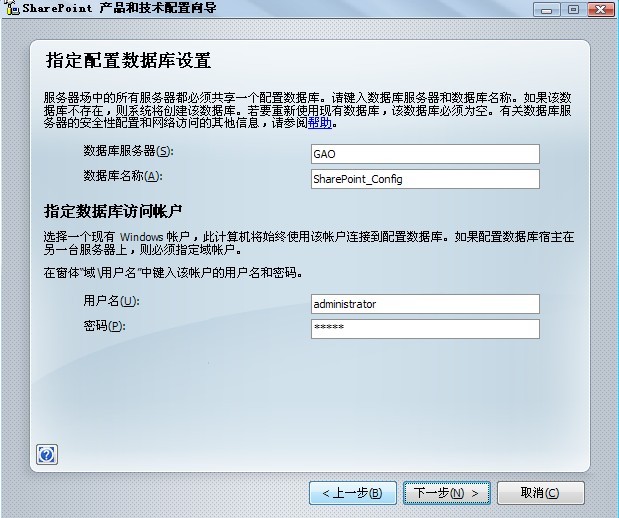
**配置**



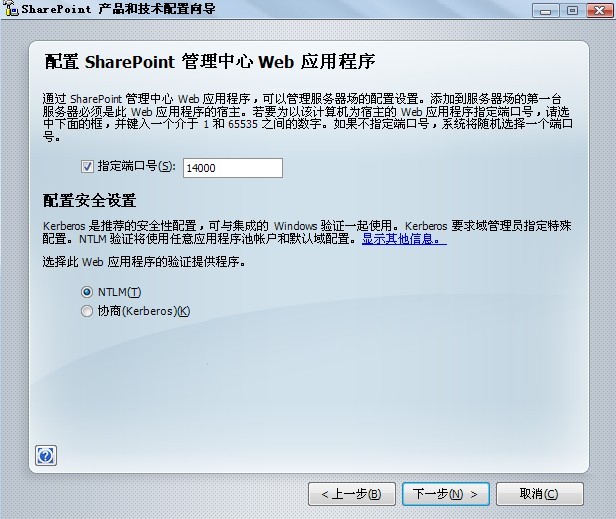
下一步后，会提示开启一些必要的服务，点击“是”就可以啦。



配置数据库的信息和身份验证：



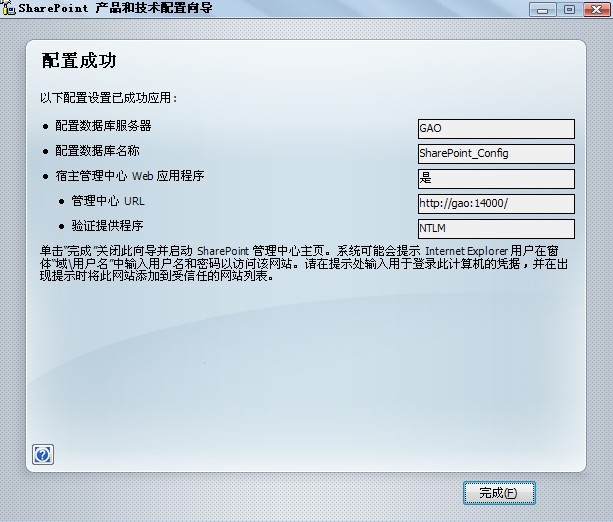
为我们的应用程序指定一个端口，大家要记住设置的端口号：



下一步后，我们需要做的还是继续等待.......



OK，看到下面的页面，说明我们的配置成功啦！



**创建Web应用程序**



接下来我们利用SharePoint创建第一个Web应用程序。配置完成后会自动跳转到一个网页：http://gao:14000/default.aspx

这里是SharePoint的管理中心，很多操作会在这里完成，以后我们会经常到这个页面：



点击操作，选择“服务器上的服务”：



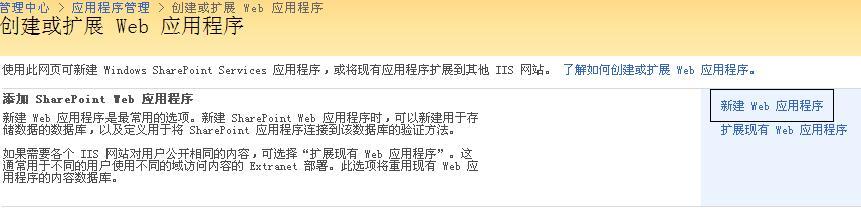
在“服务器上的服务”中我们可以更改多需要的服务项：



点击“应用程序管理”，选择“创建或扩展Web应用程序”：



在这里我们就可以新建一个Web应用程序：



对Web应用程序经行相应的配置：



点击确定后就可以耐心的等待，看到下面的页面则代表我们的应用程序创建成功啦！



小结：通过此文和大家一起了解了SharePoint的简单介绍，安装和配置，还有通过SharePoint创建Web应用程序的方法。纯基础知识，没有什么太大的难度，相信通过图片的演示大家很容易理解！

# [SharePoint 2007图文开发教程(2)---使用SharePoint创建网站](http://www.cnblogs.com/gaoweipeng/archive/2009/10/24/1588672.html)

[上文](http://www.cnblogs.com/gaoweipeng/archive/2009/10/23/1587665.html)简单介绍了有关SharePoint的安装，配置等内容的介绍。此文我们继续SharePoint之旅，通过SharePoint创建网站。

**创建网站**



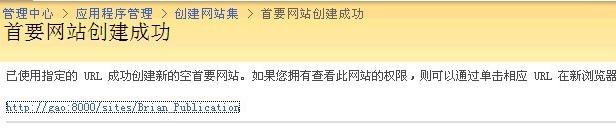
来到熟悉的应用程序管理界面，在SharePoint 网站管理中，我们就可以创建网站：



 点击“创建网站集”进入创建网站集的管理界面，SharePoint为我们提供了很多网站类型的模板，如发布网站，会议工作区....。根据自己的需求，选择合适的网站类型。这里我们以发布门户网站为例，进行网站配置：



点击“确定”，稍作等待。看待下面的界面，我们的网站就创建成功啦！



点击上图中的链接，就会访问刚刚创建的发布网：



 从图中我们也可以看出，我们可以对网站的访问权限等进行相应的设置。

**修改网站内容**



 （1）修改网站导航

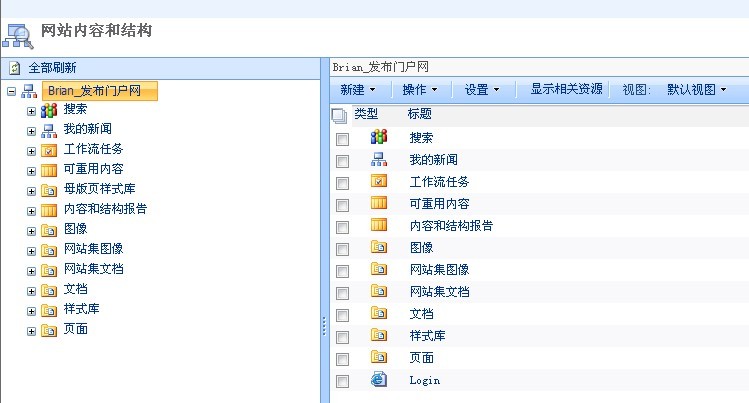
点击上图的链接，我们就可以进行相应的管理：



添加完毕后，点击确定，在首页就可以看到添加后的效果。

（2）管理网站的内容和结构

进入网站内容和结构管理界面，在这里我们可以对网站进行更多内容的修改：



这里简单举个小例子，修改网站的徽标。点击根节点就会弹出下拉框，点击常规设置：



 这里我们就可以修改网站的徽标：



小结：通过SharePoint创建网站就简单的介绍到这里，通过SharePoint可以很容易的创建自己的门户网站。并且对网站的参数进行修改，希望能对新手有帮助。

# [SharePoint 2007图文开发教程(3)---实现简单的WebPart](http://www.cnblogs.com/gaoweipeng/archive/2009/10/26/1589269.html)

在前面的文章中，我们讲解了很多基础的内容，主要包括 [安装配置](http://www.cnblogs.com/gaoweipeng/archive/2009/10/23/1587665.html)、简单的[创建网站](http://www.cnblogs.com/gaoweipeng/archive/2009/10/24/1588672.html)等。相对来说比较简单。此文实现简单的WebPart，继续我们的SharePoint之旅。

**WebPart**



WebPart大家应该不会陌生，WebPart是ASP.NET 2.0中一个非常令人激动的特性。它为创建动态的网页接口提供了一系列的可用控件，使得用户很容易地进行配置或者个性化页面。并且，用户可以象在桌面应用中一样自由地显示、隐藏或者移动WebPart组件。

如果想向SharePoint中添加WebPart，大体上分为：创建WebPart，部署WebPart DLL，在SharePoint站点中添加WebPart等步骤。下面逐一说明。

**创建WebPart**



WebPart也是服务器控件的一种，也就是说创建WebPart和创建一般的服务器控件没有什么太大的区别，我们只需要创建一个 类库应用程序，在其中实现就可以了。

using System;  
using System.Collections.Generic;  
using System.Text;  
using System.Web.UI.WebControls.WebParts;//添加WebParts命名空间  
  
namespace Brian\_HelloWebPart  
{  
    public class Hello : WebPart  
    {  
        private string \_Text = "Say Hello To Everyone !";  
  
        [WebBrowsable(true), Personalizable(true)]  
        public string Text  
        {  
            get { return \_Text; }  
            set { \_Text = value; }  
        }  
  
        protected override void Render(System.Web.UI.HtmlTextWriter writer)  
        {  
            writer.Write(\_Text);  
        }  
    }  
}

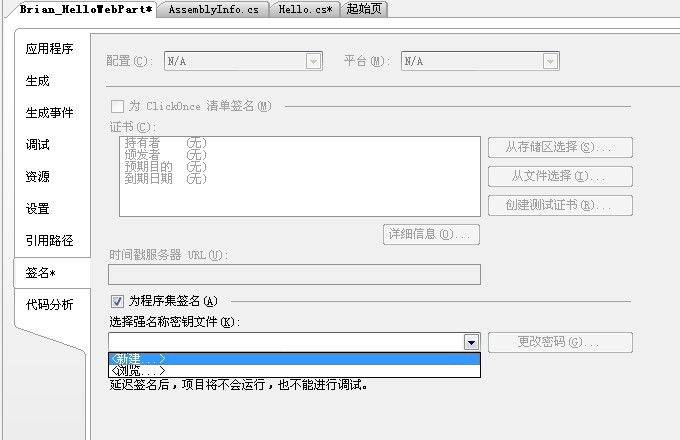
然后我们需要设定下程序集的权限，在AssemblyInfo.cs中添加命名空间：System.Security，然后加上了[assembly: AllowPartiallyTrustedCallers()]这一行代码。由于现在的程序集是部分信任程序集，所以在向SharePoint中添加的时候会有错误，所以这里需要添加[assembly: AllowPartiallyTrustedCallers()]。

这样，一个简单的WebPart就完成了。

**部署WebPart**



右键我们工程---属性---签名:

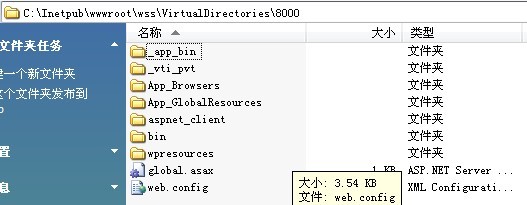


选中为程序集签名---选择强名称密钥文件：



添加密钥的目的是为了使我们的程序集变成强签名的程序集。编译我们的工程，生成DLL。

接下来将这个DLL放到SharePoint网站工程bin文件夹中。由于WebPart在页面中运行时必须是一个安全的Control，所以我们必须在Web.config中把它注册成为安全的控件。在测试Web Parts之前在Web.config中添加入口点。找到项目文件，打开Web.config：



在Web.config中添加如下代码：

<SafeControl Assembly="Brian\_HelloWebPart, Version=1.0.0.0, Culture=neutral, PublicKeyToken=c53a25ce64c488e1" Namespace="Brian\_HelloWebPart" TypeName="Hello" Safe="True" />

可能会有人问上面的PublicKeyToken怎么获得，这里简单说一下：运行VS2005的命令提示符，执行下面的命令：



要注意需要进入dll文件所在的路径。到此，我们的WebPart就部署完毕了！

**在SharePoint站点中添加WebPart**



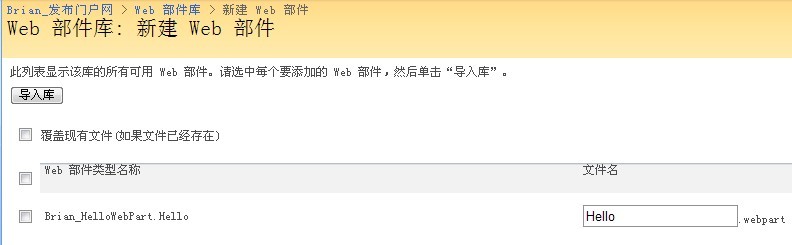
进入我们[上文](http://www.cnblogs.com/gaoweipeng/archive/2009/10/24/1588672.html)创建的网站，进入网站设置：



点击web控件，进入添加WebPart界面，点击“新建”：



进入Web部件库中我们就可以看到要添加的HelloWebPart，选中后点击“导入库”：



然后，网站控制---编辑网页：



回到首页，点击页面--添加Web部件--浏览：



选中Hello，点击“添加”：



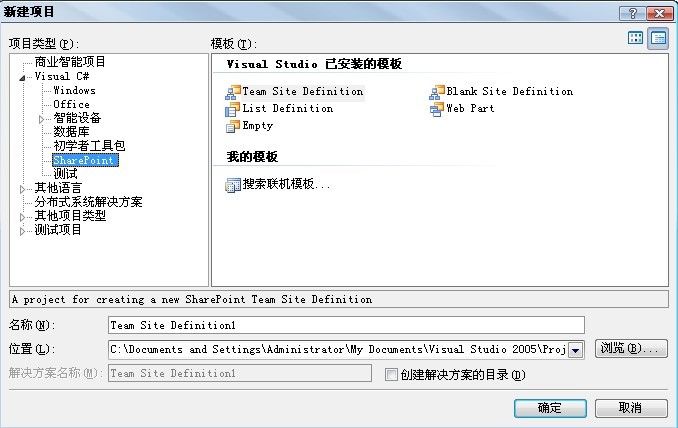
Ok，看到了吧，我们简单的WebPart就添加成功啦！



**使用WSS开发模板**



这里补充一点，我们可以使用开发模板来完成我们的开发，[模板下载](http://www.oceanstudio.net/Lists/List2/DispForm.aspx?ID=24&Source=http%3A%2F%2Fwww.oceanstudio.net%2Fdefault.aspx)。安装完模板后，再次运行VS2005，就会看到模板：



使用模板，我们可以更方便的实现我们的开发，比如模板会自动为我们创建密钥文件等，大家可以试试。

小结：至此，简单的WebPart的实现就基本完成了，相信通过此文，大家可以很容易的掌握WebPart的基本开发。望对新手有帮助。

# [SharePoint 2007图文开发教程(4)---实现高级的WebPart](http://www.cnblogs.com/gaoweipeng/archive/2009/10/27/1589488.html)

前面的文章中，我们编写了一个非常简单的WebPart，只是在网页中输出一段字符串。在实际的应用中没有什么作用。此文和大家分享如何开发高级的WebPart，为控件添加事件，实现有复合控件的WebPart等，使我们的WebPart更佳实用。

如果大家有开发服务器控件的基础，此文对你来说没有什么难度。想了解相关内容，参看我的[《ASP.NET的服务器控件开发》系列](http://www.cnblogs.com/gaoweipeng/archive/2009/06/13/1501833.html)。废话不多说，直接进入今天的主题。

**Demo 1**



第一个例子也很简单，点击一个按钮，显示当前系统时间。如何创建WebPart就不细说了，不清楚的朋友可以参看[《SharePoint 2007图文开发教程(3)---实现简单的WebPart》](http://www.cnblogs.com/gaoweipeng/archive/2009/10/26/1589269.html)。

直接看代码：

using System;  
using System.Runtime.InteropServices;  
using System.Web.UI;  
using System.Web.UI.WebControls.WebParts;  
using System.Web.UI.WebControls;  
using System.Web.UI.HtmlControls;  
using System.Xml.Serialization;  
  
using Microsoft.SharePoint;  
using Microsoft.SharePoint.WebControls;  
using Microsoft.SharePoint.WebPartPages;  
  
namespace GetNow  
{  
    [Guid("0708695c-869c-4e8f-a6e9-cfcf148f404c")]  
    public class GetNow : System.Web.UI.WebControls.WebParts.WebPart, INamingContainer  
    {  
        private string \_text;  
        private HtmlTableCell \_tablecell = new HtmlTableCell();  
        private Label lb = new Label();  
        private Button btn = new Button();  
  
        [WebBrowsable(true), Personalizable(true)]  
        public string Text  
        {  
            get { return \_text; }  
            set { \_text = value; }  
        }  
        public GetNow()  
        {  
            this.btn.Click += new EventHandler(btn\_Click);  
            this.ExportMode = WebPartExportMode.All;  
        }  
        /// <summary>  
        /// 重写OnLoad  
        /// </summary>  
        protected override void OnLoad(EventArgs e)  
        {  
            base.OnLoad(e);  
  
            if (this.Page.IsPostBack == true)  
            {  
                this.AddControls();  
            }  
        }  
        /// <summary>  
        /// 重写CreateChildControls  
        /// </summary>  
        protected override void CreateChildControls()  
        {  
            this.Controls.Add(new LiteralControl("<table>" + "\n"));  
            this.Controls.Add(new LiteralControl("    <tr>" + "\n"));  
            this.Controls.Add(this.\_tablecell);  
            this.Controls.Add(new LiteralControl("    </tr>" + "\n"));  
            this.Controls.Add(new LiteralControl("</table>" + "\n"));  
  
            if (this.Page.IsPostBack == false)  
            {  
                this.AddControls();  
            }  
        }  
        /// <summary>  
        /// click处理函数  
        /// </summary>  
        void btn\_Click(object sender, EventArgs e)  
        {  
            this.\_text = DateTime.Now.ToString();  
            this.\_tablecell.InnerHtml = "";  
            this.AddControls();  
        }  
        /// <summary>  
        /// 重写Render  
        /// </summary>  
        protected override void Render(HtmlTextWriter writer)  
        {  
            base.Render(writer);  
        }  
  
        private void AddControls()  
        {  
            this.lb.ID = this.ID + "Label";  
            this.lb.Text = this.\_text;  
            this.\_tablecell.Controls.Add(this.lb);  
            this.\_tablecell.Controls.Add(new LiteralControl("<br>"));  
            this.btn.ID = this.ID + "Button";  
            this.btn.Text = "获取当前时间:";  
            this.\_tablecell.Controls.Add(this.btn);  
        }  
    }  
}

按照上文的部署方法，将GetNow的WebPart添加到SharePoint网站中：测试一下，获取时间成功！



**Demo 2**



第二个例子我们来实现WebPart的联动控制，也就是说通过一个WebPart来控制另外一个WebPart。还是直接看代码：

using System;  
using System.Runtime.InteropServices;  
using System.Web.UI;  
using System.Web.UI.HtmlControls;  
using System.Web.UI.WebControls;  
using System.Web.UI.WebControls.WebParts;  
using System.Xml.Serialization;  
using System.Collections;  
using System.Drawing;  
  
using Microsoft.SharePoint;  
using Microsoft.SharePoint.WebControls;  
using Microsoft.SharePoint.WebPartPages;  
using GetNow;  
  
namespace SetColor  
{  
    [Guid("803617a2-9dc9-4eb8-b05c-3e064bc16642")]  
    public class SetColor : System.Web.UI.WebControls.WebParts.WebPart, IPostBackEventHandler  
    {  
        private IList \_colorlist = new ArrayList();  
        private HtmlTableCell \_TableCell = new HtmlTableCell();  
  
        public SetColor()  
        {  
            this.GetItems();  
  
            this.ExportMode = WebPartExportMode.All;  
        }  
  
        private void GetItems()  
        {  
            this.\_colorlist.Add("红色");  
            this.\_colorlist.Add("黄色");  
            this.\_colorlist.Add("蓝色");  
            this.\_colorlist.Add("绿色");  
            this.\_colorlist.Add("紫色");  
        }  
  
        private void AddRows()  
        {  
            for (int i = 0; i < \_colorlist.Count; i++)  
            {  
                this.\_TableCell.Controls.Add(new LiteralControl("<table>" + "\n"));  
                this.\_TableCell.Controls.Add(new LiteralControl("    <tr>" + "\n"));  
                this.\_TableCell.Controls.Add(new LiteralControl("        <td style=\"cursor:hand\" onclick=\"" + this.Page.ClientScript.GetPostBackEventReference(this, this.\_colorlist[i].ToString()) + "\">" + this.\_colorlist[i].ToString() + "</td>" + "\n"));  
                this.\_TableCell.Controls.Add(new LiteralControl("    </tr>" + "\n"));  
                this.\_TableCell.Controls.Add(new LiteralControl("</table>" + "\n"));  
            }  
        }  
  
        protected override void CreateChildControls()  
        {  
            this.Controls.Add(new LiteralControl("<table>" + "\n"));  
            this.Controls.Add(new LiteralControl("    <tr>" + "\n"));  
            this.Controls.Add(this.\_TableCell);  
            this.Controls.Add(new LiteralControl("    </tr>" + "\n"));  
            this.Controls.Add(new LiteralControl("</table>" + "\n"));  
  
            if (this.Page.IsPostBack == false)  
            {  
                this.AddRows();  
            }  
        }  
  
        protected override void OnLoad(EventArgs e)  
        {  
            base.OnLoad(e);  
  
            if (this.Page.IsPostBack == true)  
            {  
                this.AddRows();  
            }  
        }  
  
        protected override void Render(HtmlTextWriter writer)  
        {  
            base.Render(writer);  
        }  
  
        public void RaisePostBackEvent(string eventArgument)  
        {  
  
            for (int i = 0; i < WebPartManager.WebParts.Count; i++)  
            {  
  
                if (WebPartManager.WebParts[i].Title == "GetNow")  
                {  
                    GetNow.GetNow instance = (GetNow.GetNow)WebPartManager.WebParts[i];  
                    switch (eventArgument)  
                    {  
                        case "红色":  
                            instance.BackColor = Color.Red;  
                            break;  
                        case "黄色":  
                            instance.BackColor = Color.Yellow;  
                            break;  
                        case "蓝色":  
                            instance.BackColor = Color.Blue;  
                            break;  
                        case "绿色":  
                            instance.BackColor = Color.Green;  
                            break;  
                        case "紫色":  
                            instance.BackColor = Color.Purple;  
                            break;  
                        default:  
                            break;  
                    }  
                }  
  
            }  
        }  
    }  
}

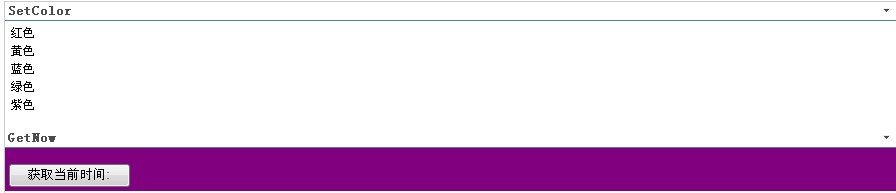
代码不是很难，创建一个WebPart，向其中添加一个List，提供用户几种颜色的选择情况，然后通过用户选择的情况改变我们上面GetNow这个WebPart的背景颜色。

按照前面的方法加到SharePoint网站中：

选择红色：



选择紫色：



 ok。我们想要的效果已经实现啦！

小结：上面只是自己抛砖引玉的小例子，希望对新手有帮助。对于WebPart的开发部分就介绍这些吧，望支持。有什么不得当的地方还望大家指教。根据自己的业务需要，大家可以开发出更实用的WebPart。还有建议大家多看看服务器控件开发的文章，会有帮助!

# [SharePoint 2007图文开发教程(5)---体验Event Handler](http://www.cnblogs.com/gaoweipeng/archive/2009/10/29/1590266.html)

前面两篇文章介绍了有关WebPart的相关内容，此文继续SharePoint之旅，体验下Event Handler的相关内容。

**Event Handler**



Event Handler是SharePoint中与文档库或列表相关联的事件处理程序，当我们对项目进行某些操作时，Event Handler就会截获这个事件，然后相应的进行操作。功能上比较像数据库中的触发器。比如:删除某个文件时向管理员发送一个邮件，添加一个列表时进行其他操作等等。  
和2.0相比，WSS3.0对Event Handler提供了更强大的支持，如：列表，文档库，内容类型。并且支持同步和异步的操作等。

为SharePoint提供一个Event Handler并不是很费力，主要包括：开启Event Handler支持，编写Event Handler程序，向SharePoint中部署Event Handler。接下来逐一说明一下。

**开启Event Handler支持**



默认情况下是不开启Event Handler支持的，需要我们手动开启：进入应用程序管理---进入Web应用程序常规设置。



开启事件处理程序：



这样就完成了对Event Handler开启的支持。

**编写Event Handler程序**



SharePoint的Event Handler其实是一个类，这个类主要继承WSS中的**SPWebEventReceiver**，**SPEmailEventReceiver**，**SPListEventReceiver**，**SPItemEventReceiver，IListEventSink**等类，并且重写相应的方法来完成我们的需求。

这里做个简单的例子，当我们删除一个Items时，向日志文件中添加一条记录。

直接看代码：

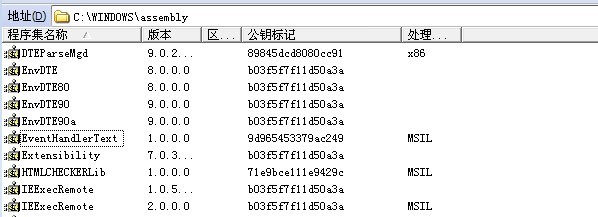
using System;  
using System.Collections.Generic;  
using System.Text;  
using System.IO;  
  
using Microsoft.SharePoint;  
  
namespace EventHandlerText  
{  
    public class ItemLogEvent : SPItemEventReceiver  
    {  
        /// <summary>  
        /// 重写ItemDeleting,向日志文件写数据  
        /// </summary>  
        public override void ItemDeleting(SPItemEventProperties properties)  
        {  
            string path = @"c:\Log.txt";  
  
            string text = "删除Item" + ":" + DateTime.Now.ToString();  
  
            StreamWriter writer = new StreamWriter(path);  
  
            writer.Write(text);  
  
            writer.Close();  
        }  
  
    }  
}  
是不是很简单，接下来我们按照为WebPart添加密钥文件的方法，为我们的Event Handler添加密钥文件，添加的目的我在部署的时候再说。这样，简单的Event Handler就完成了。

**向SharePoint中部署Event Handler**



和WebPart有所不同的是，Event Handler的dll需要放到GAC（Global Assembly Cache）中，而不能放在SharePoint网站的bin文件夹中，所以我们生成的dll必须进行强签名，这也就是上面为什么添加密钥文件的目的。

GAC的系统路径为：C:\WINDOWS\assembly，直接将生成的Event Handler dll拖入到这个路径中即可。



先来看下SharePoint中的情况：



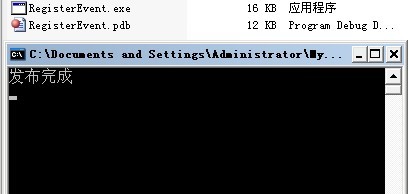
在我们先前创建的网站中，我创建了一个文档库的List，并且上传了一个word文件，我们的Event Handler就是要添加到这里。

接下来就是将这个Event Handler部署到SharePoint中。可以通过代码的形式，也可以通过Feature的形式。这里介绍下代码的形式。

创建一个批处理程序（当然win程序也可以），添加如下代码：

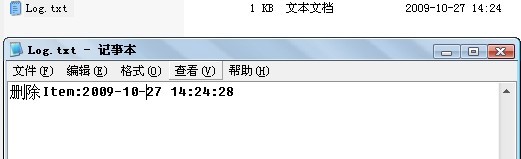
using System;  
using System.Collections.Generic;  
using System.Text;  
  
using Microsoft.SharePoint;  
  
namespace RegisterEvent  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            try  
            {  
                SPSite site = new SPSite("http://gao:8000/sites/Brian\_Publication/");//指定网站  
  
                SPWeb web = site.OpenWeb();  
  
                SPList list = web.Lists["文档库"];//指定List  
  
                list.EventReceivers.Add(SPEventReceiverType.ItemDeleting, "EventHandlerText, Version=1.0.0.0, Culture=neutral, PublicKeyToken=9d965453379ac249", "EventHandlerText.ItemLogEvent");//添加Event Handler  
  
                Console.WriteLine("发布完成");  
  
                Console.ReadLine();  
            }  
            catch (Exception ex)  
            {  
                Console.WriteLine(ex.Message);  
                Console.ReadLine();  
            }  
             
        }  
    }  
}

代码很容易，想必大家一看就能明白，找到工程生成的可执行文件，运行：



 这样就代表我们的Event Handler发布成功啦！至此，Event Handler的部署工作也就完成啦！

最后我们来测试下，进入文档库，删除那个“LINQ中文教程”word文件。到C:\下看下Log.txt：



OK，我们的Event Handler执行了，实现了我们想要的效果。

小结：有关Event Handler的内容就介绍到这里，相信大家认真的看一边就可以很轻松的掌握，当然有关Event Handler的内容还不止这些，需要在平时的应用中多多积累。

# [SharePoint 2007图文开发教程(6)---实现Search Services](http://www.cnblogs.com/gaoweipeng/archive/2009/10/30/1591283.html)

此文和大家分享如何在SharePoint中实现Search的功能。

SharePoint中为我们提供了Search的功能，和Event Handler的实现比较类似，还是需要启动支持，定制服务，运行服务几部分内容，下面逐一说下。

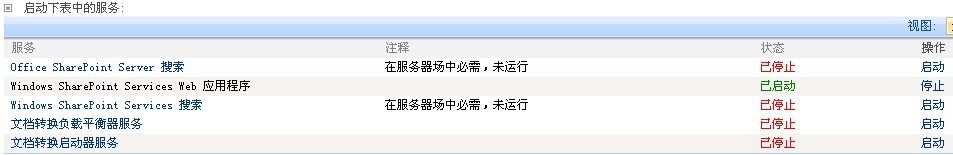
**启动Search服务**



 我们进入SharePoint的管理中心，进入操作界面，选择“服务器上的服务”：



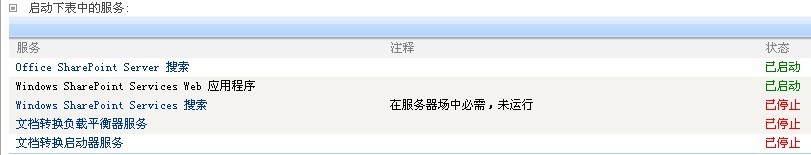
进入到服务器上的服务界面后，就会看到当前SharePoint所启动的服务：



可以看到默认情况下Office SharePoint Server搜索是停止状态的，所以需要我们手动的开启搜索服务，点击“启动”后，配置服务器上Office SharePoint Server 搜索服务设置：



点击“开始”后，就可以开启Office SharePoint Server搜索，这里要**注意**的是，用户名需要写成：**域名+用户名**的形式。否则启动服务时就会出错。开启成功后会自动返回服务器上的服务界面：

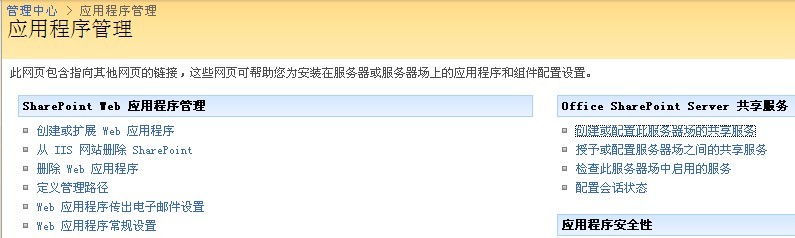


可以看到，我们的Office SharePoint Server搜索开启成功。

**创建共享提供服务程序**



从新回到应用程序管理界面，在Office SharePoint Server 共享服务中，选择“创建或配置此服务器的共享服务”：



进入后进行填写：



点击确定，稍作等待：

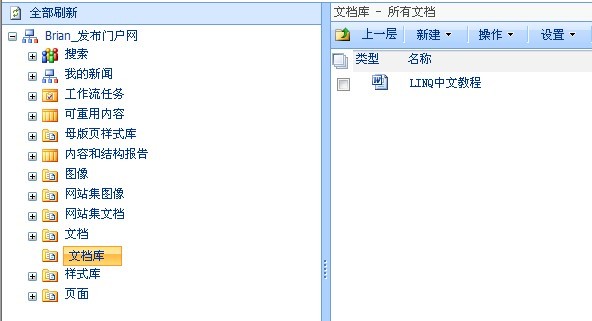


这样，我们的共享服务提供程序就创建成功啦！

**运行爬网**



这一部分有点像蜘蛛的功能，在我们指定的站点内实现抓取资源，并放到数据库中，为了能看到明显的效果，我提前在我前面创建的网站中上传了一个Word文件：



然后就是对SharePoint中爬网的设置，进入管理中心，可以看到上面创建的共享服务程序：



点击进入：



选择“搜索设置”：



进入“内容源和爬网计划”，这里就可以指定我们要爬行的数据源：



这里只有本地的SharePoint网站，点击“开始所有爬网”，就可以进行数据的采集。数据采集的时候会稍微慢些，内存和CPU的使用量都不小，由于SharePoint是服务器端得产品，这点也难免。

OK，我们回到网站进行测试一下，在搜索框中输入检索内容：



简单的查询：



效果还不错，至此，SharePoint中提供的Search服务我们就基本上完成啦~

**小结**：SharePoint为我们提供了方便的数据检索的服务，至于内部的实现手段这里没有做深究，但是从运行的结果上看，关键词的索引，中文的分词，高亮显示等实现的都还是比较OK的，很值得学习，有机会定要好好研究下内部实现的方法！话说微软现在有了Bing，在SharePoint中融入Bing的API也是不错的选择。

# [SharePoint 2007图文开发教程(7)---Workflow初体验](http://www.cnblogs.com/gaoweipeng/archive/2009/11/05/1596417.html)

继续我们的SharePoint之旅，体验Workflow的应用，由于以前也很少弄WF，望有什么问题WF的高手多多指教。

在[SharePoint 2007图文开发教程(1)---简介，安装，配置及创建Web应用程序](http://www.cnblogs.com/gaoweipeng/archive/2009/10/23/1587665.html)中，我介绍了SharePoint所需要安装的软件，里面提到了需要安装Microsoft Windows Workflow Foundation 2.2以上的版本，目的也就是为对WF的支持。

**Workflow**



[工作流（Workflow）](http://baike.baidu.com/view/880657.htm?fr=ala0)就是“业务过程的部分或整体在计算机应用环境下的自动化”，它主要解决的是“使在多个参与者之间按照某种预定义的规则传递文档、信息或任务的过程自动进行，从而实现某个预期的业务目标，或者促使此目标的实现”。简单地说，工作流就是一系列相互衔接、自动进行的业务活动或任务。一个工作流包括一组任务（或活动）及它们的相互顺序关系，还包括流程及任务（或活动）的启动和终止条件，以及对每个任务（或活动）的描述。

**在SharePoint中实现Workflow**



实现工作流可以有多种方式，SharePoint中为我们提供了一些**内置的Workflow**，我们也可以利用**SharePoint Desigener来实现定制的工作流**，但是这两种方法都会有一定的局限性，往往在我们的开发中不能满足我们的需求，但是我们还有第三种方法，那就是**利用Vistual Studio开发工作流**，然后添加到SharePoint中。

简单的看下SharePoint中提供的Workflow模板，我们可以向文档等中添加工作流模板：



接下来重点说说利用SharePoint Desigener创建工作流，借鉴了网上的一些例子，虽然不是很成熟，但是可以说明问题了：

首先做一些准备条件：

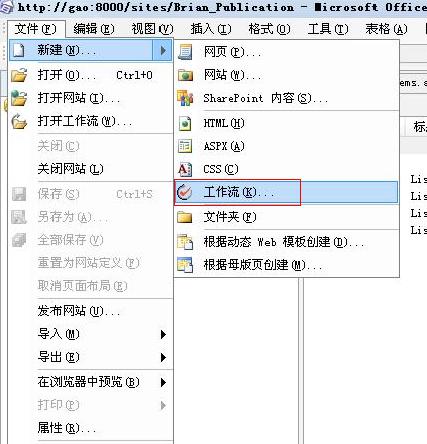
我在我的网站中创建一个“请假申请”的列表，并在设置中“创建栏”：



创建栏：



点击确定，我们得准备工作就完成了。下面我们利用SharePoint Desigener创建工作流，打开网站：新建---工作流



下面就是对工作流的设置：

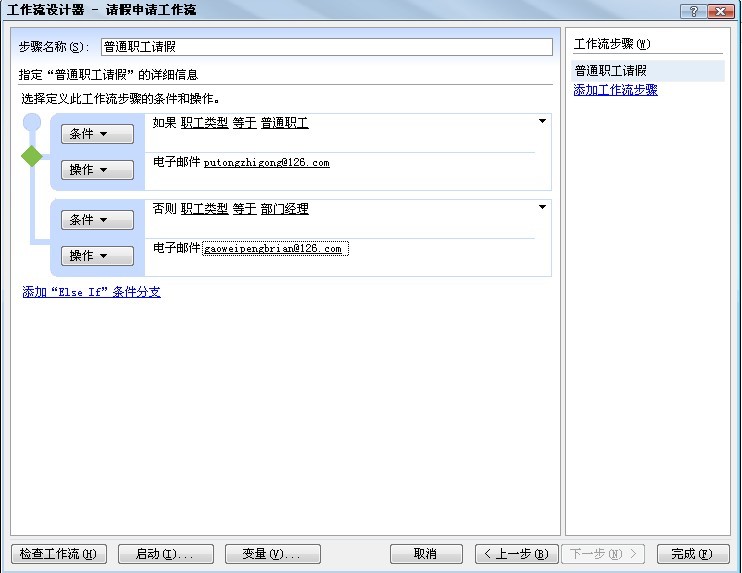


为工作流起一个名字，然后将这个工作流添加到SharePoint中我们前面创建好的那个请假申请列表。这里我选择的是手动启动工作流，当然也可以选择自动开启。

点击下一步：



在这里我们对条件和操作经行设置，也就是说对工作流程的描述，填写完毕后点击“完成”。



这里设想的工作流程就是：如果有普通员工请假，给部门经理发邮件，如果有部门经理请假，就向总经理发邮件。点击完成后就会将工作流添加到SharePoint中。

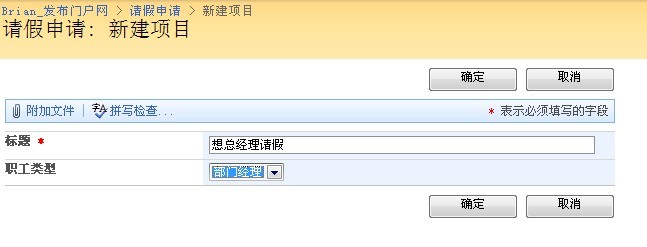


OK，这样我们就成功的在SharePoint中添加了我们自己设计的一个工作流。

**测试工作流**



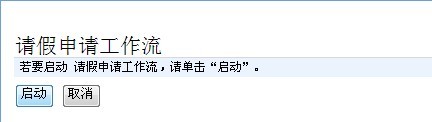
最后要做的就是测试下我们的工作流，回到SharePoint中的请假申请列表，新建一个项目：



由于创建工作流时是设置的手动开启工作流，所以这里需要我们自己开下，选择下图的“工作流”：



点击开启，这样这个工作流就执行了。



于是乎我高高兴兴的跑到自己邮箱里看有没有发邮件，遗憾的是没有！为什么呢？回来看了下工作流的最后执行状态：



原来是由于SharePoint电子邮件设置不正确，在网上找了下解决方案：<http://office.microsoft.com/zh-cn/sharepointdesigner/HA102379122052.aspx#44>

这里还包括了许多其他问题的解决方案，大家可以参考下。

小结：在SharePoint中实现Workflow就简单说这么多，如何利用Vistual Studio开发工作Workflow就不说了，水平有限，就不献丑了，希望通过此文，能够对学习SharePoint的朋友们有帮助。