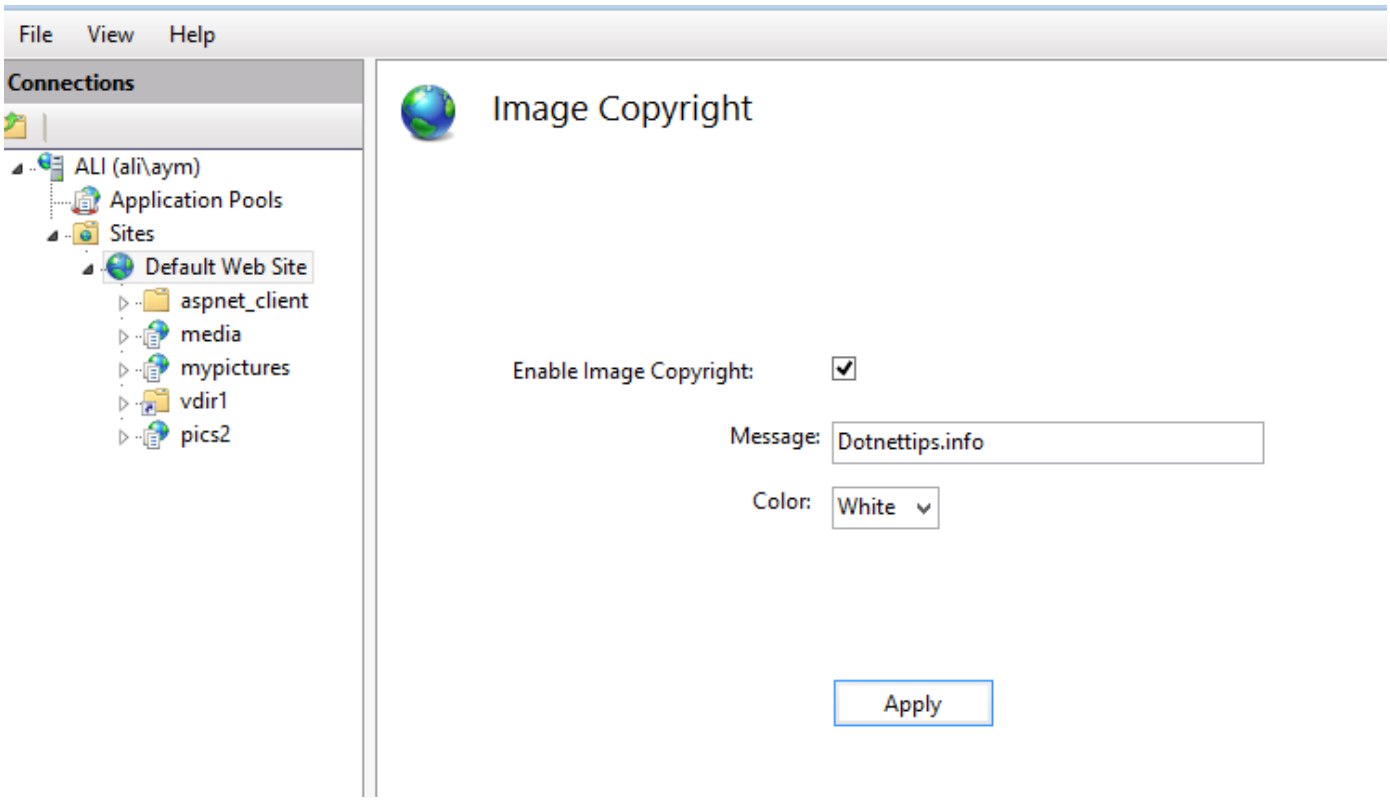
عنوان: **آشنایی با ساختار IIS قسمت نهم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۱/۱۷ ۱:۴۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, IIS, Strong Name, UI, GAC

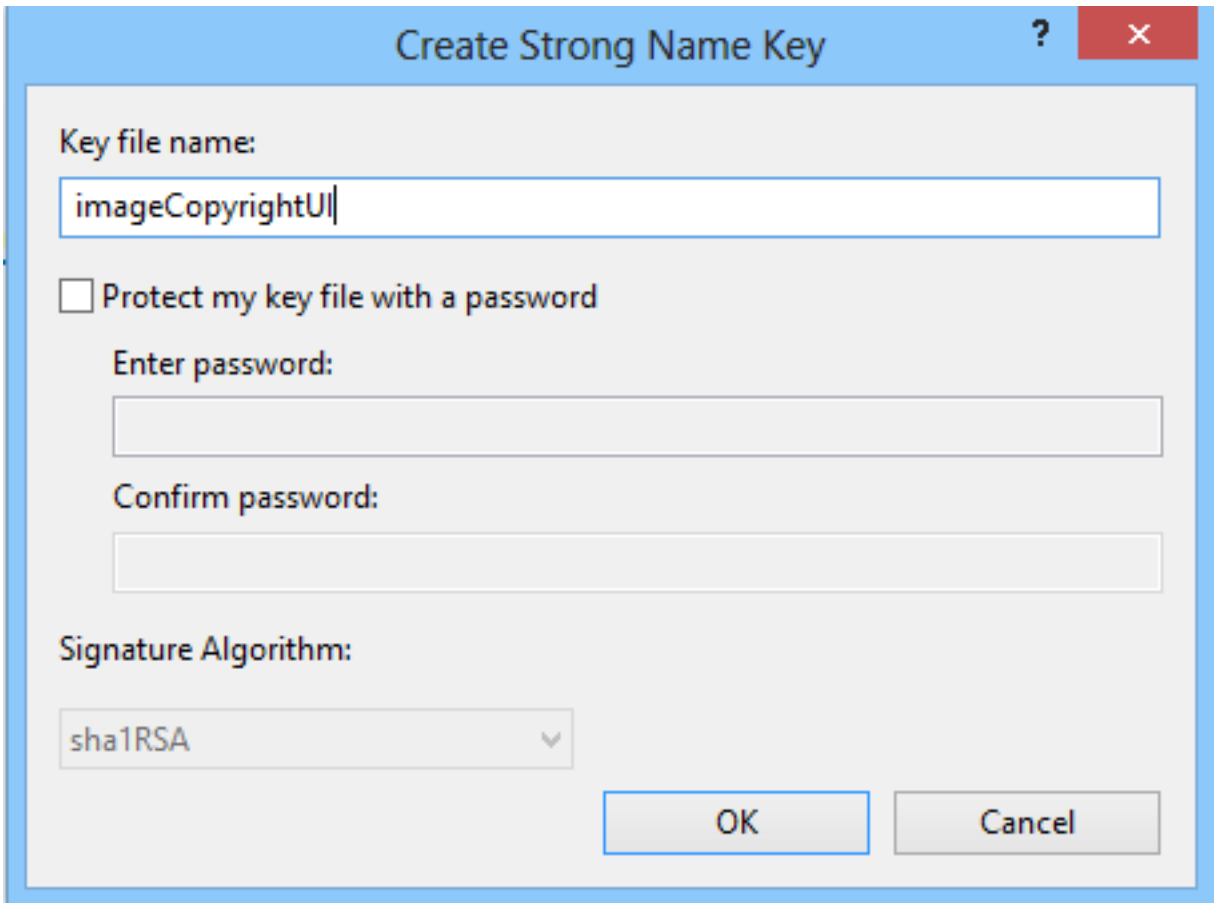
در قسمت قبلی ما یک هندلر ایجاد کردیم و درخواست‌هایی را که برای فایل jpg و به صورت GET ارسال میشد، هندل می‌کردیم و تگی را در گوشه‌ی تصویر درج و آن را در خروجی نمایش میدادیم. در این مقاله قصد داریم که کمی هندلر مورد نظر را توسعه دهیم و برای آن یک UI یا یک رابط کاربری ایجاد نماییم. برای توسعه دادن ماژولها و هندلرها ما یک dll نوشته و باید آن را در GAC که مخفف عبارت Global Assembly Cache ریجستر کنیم.



جهت اینکار یک پروژه از نوع class library ایجاد کنید. فایل class1.cs را که به طور پیش فرض ایجاد می‌شود، حذف کنید و رفرنس‌های Microsoft.Web.Management.dll و Microsoft.Web.Administration.dll را از مسیر زیر اضافه کنید:

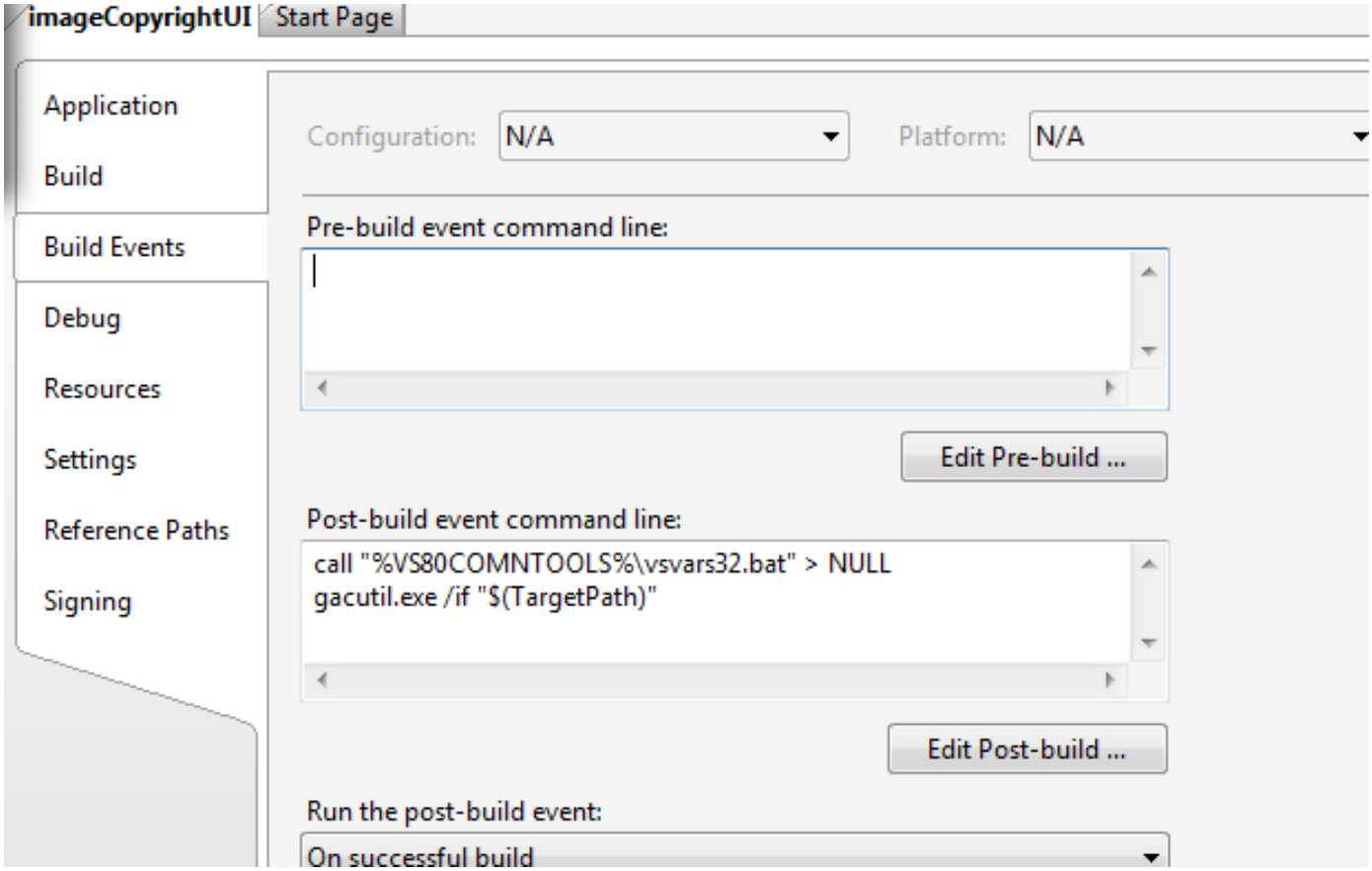
```
\Windows\system32\inetsrv
```

اولین رفرنس شامل کلاس‌هایی است که جهت ساخت ماژول‌ها برای کنسول IIS مورد نیاز است و دومی هم برای خواندن پیکربندی‌های نوشته شده مورد استفاده قرار می‌گیرد.
برای طراحی UI بر پایه winform باید رفرنس‌های System.Windows.Forms.dll و System.Web.dll را از سری اسمبلی‌های دات نت نیز اضافه کنیم و در مرحله‌ی بعدی جهت ایجاد امضاء یا strong name (sn.exe و sn64.exe) به خاطر ثبت در GAC پروژه را انتخاب و وارد Properties پروژه شوید. در تب signing گزینه sign the assembly را تیک زده و در لیست باز شده گزینه new را انتخاب نمایید و نام imageCopyrightUI را به آن نسبت داده و گزینه تعیین کلمه عبور را غیرفعال کنید و تایید و تمام. الان باید یک فایل snk مخفف strong name key ایجاد شده باشد تا بعدا با استفاده از این کلید dll ایجاد شده را در GAC ریجستر کنیم.



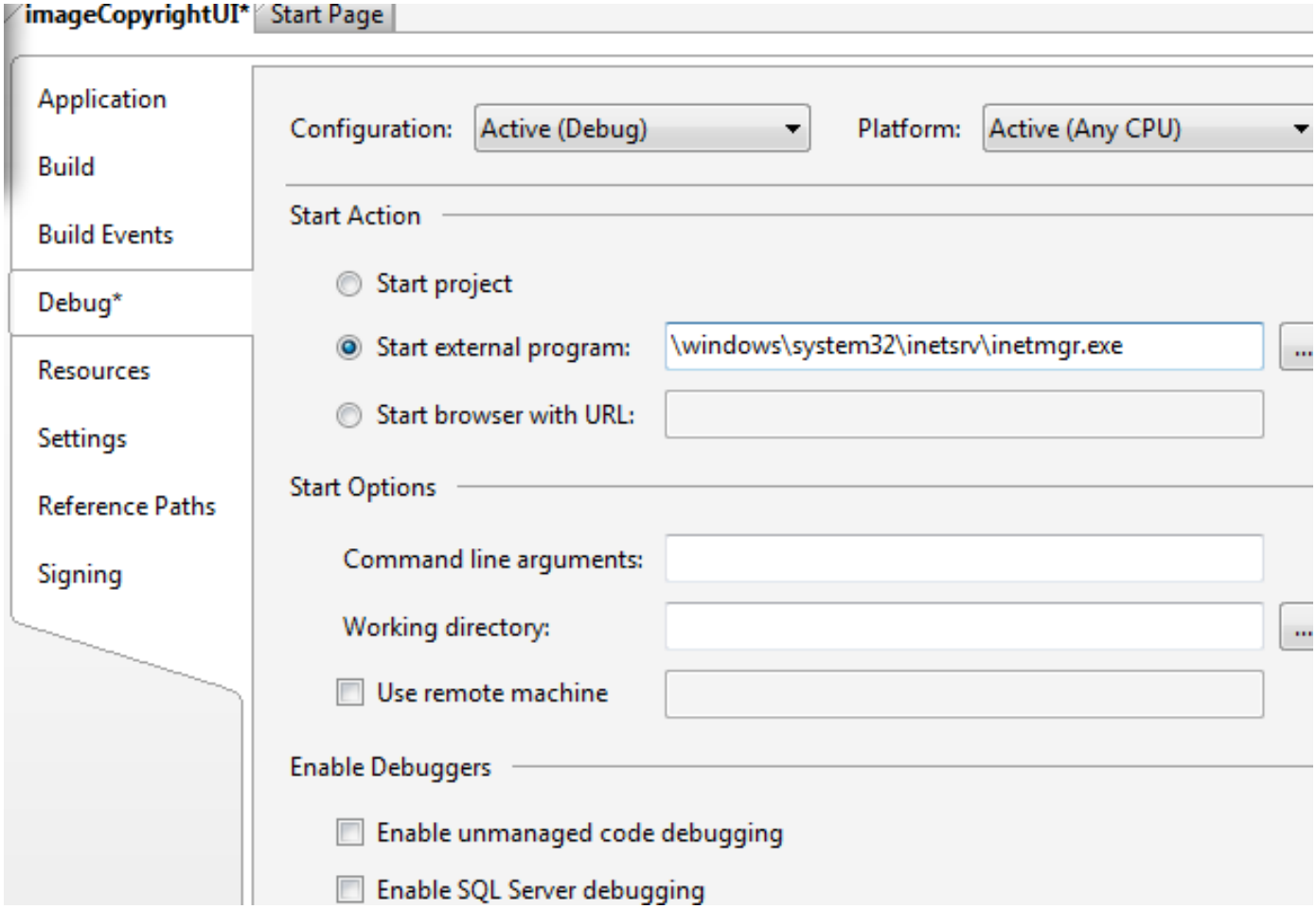
در مرحله بعدی در تب Build Events کد زیر را در بخش Post-build event command line اضافه کنید. این کد باعث می‌شود بعد از هر بار کامپایل پروژه، به طور خودکار در GAC ثبت شود:

```
call "%VS80COMNTOOLS%\vsvars32.bat" > NULL
gacutil.exe /if "$(TargetPath)"
```



نکته:در صورتی که از VS2005 استفاده می‌کنید در تب Debug در قسمت Start External Program مسیر زیر را قرار بدهید.
اینکار برای تست و دیباگینگ پروژه به شما کمک خواهد کرد. این تنظیم شامل نسخه‌های اکسپرس نمی‌شود.

```
\windows\system32\inetsrv\inetmgr.exe
```



بعد از پایان اینکار پروژه را Rebuild کنید. با اینکار dll در GAC ثبت می‌شود. استفاده از سوییچ‌های if به طور همزمان در درستور gacutil به معنی این هست که اگر اولین بار است نصب می‌شود، پس با سوییچ i نصب کن. ولی اگر قبلا نصب شده است نسخه جدید را به هر صورتی هست جایگزین قبلی کن یا همان reinstall کن.

ساخت یک Module Provider

رابط‌های کاربری IIS همانند هسته و کل سیستمش، ماژولار و قابل خصوصی سازی است. رابط کاربری، مجموعه‌ای از ماژول هایی است که میتوان آن‌ها را حذف یا جایگزین کرد. تگ ورودی یا معرفی برای هر UI یک module provider است. خیلی خودمانی، تگ ماژول پروایدر به معرفی یک UI در IIS می‌پردازد. لیستی از module providerها را می‌توان در فایل زیر در تگ بخش <modules> پیدا کرد.

```
%windir%\system32\inetsrv\Administration.config
```

در اولین گام یک کلاس را به اسم imageCopyrightUIModuleProvider.cs ایجاد کرده و سپس آن‌را به کد زیر، تغییر می‌دهیم. کد زیر با استفاده از ModuleDefinition یک نام به تگ Module Provider داده و کلاس imageCopyrightUI را که بعدا تعریف می‌کنیم، به عنوان مدخل entry رابط کاربری معرفی کرده:

```
using System;
using System.Security;
using Microsoft.Web.Management.Server;

namespace IIS7Demos
{
    class imageCopyrightUIProvider : ModuleProvider
    {
        public override Type ServiceType
```

```
        {
            get { return null; }
        }

        public override ModuleDefinition GetModuleDefinition(IManagementContext context)
        {
            return new ModuleDefinition(Name, typeof(imageCopyrightUI).AssemblyQualifiedName);
        }

        public override bool SupportsScope(ManagementScope scope)
        {
            return true;
        }
    }
}
```

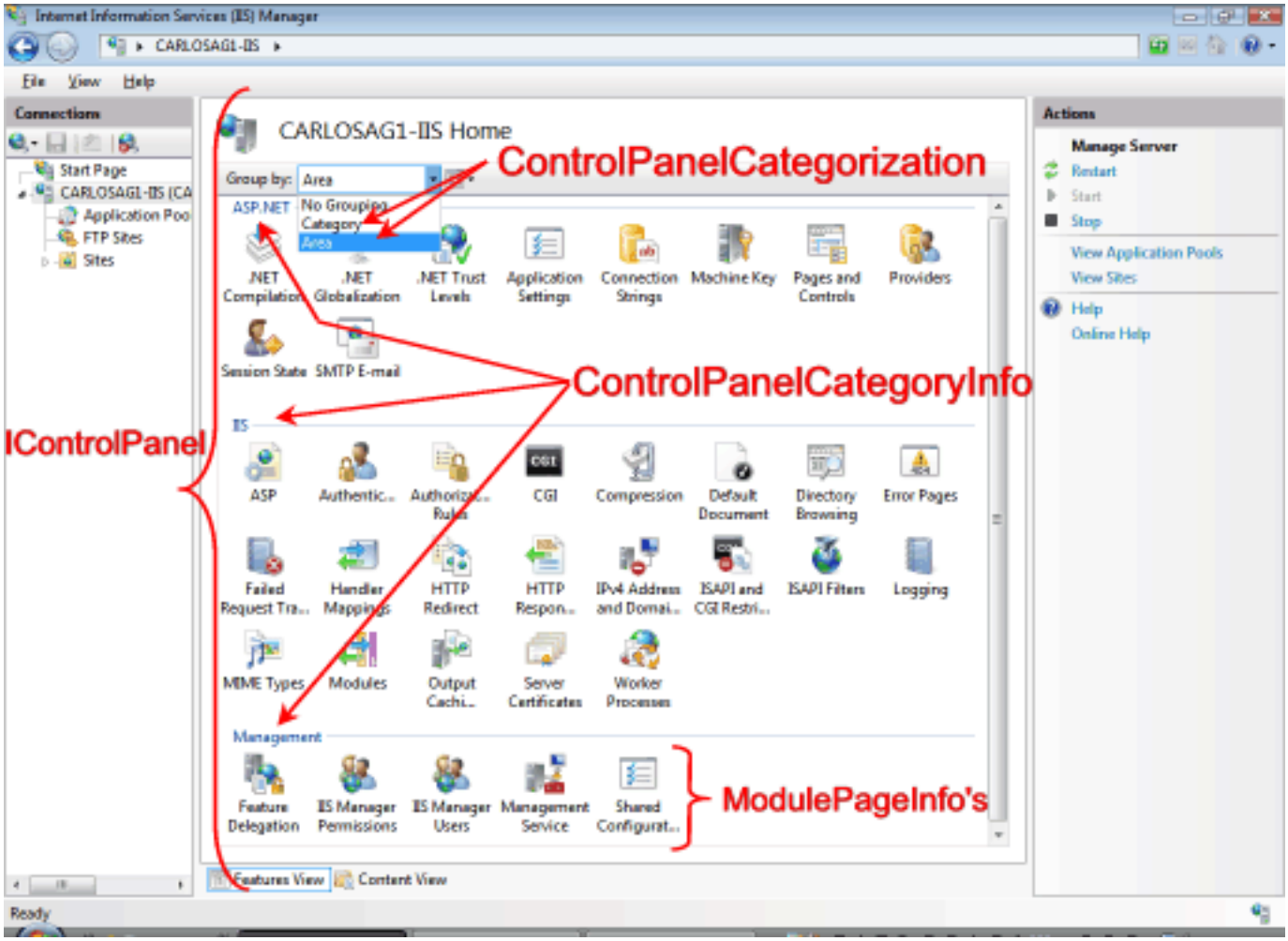
با ارث بری از کلاس module provider، سه متد بازنویسی می‌شوند که یکی از آن ها SupportsScope هست که میدان عمل پروایدر را مشخص می‌کند، مانند اینکه این پروایدر در چه میدانی باید کار کند که می‌تواند سه گزینه‌ی server,site,application باشد. در کد زیر مثلا میدان عمل application انتخاب شده است ولی در کد بالا با برگشت مستقیم true، همه‌ی میدان را جهت پشتیبانی از این پروایدر اعلام کردیم.

```
public override bool SupportsScope(ManagementScope scope)
        {
            return (scope == ManagementScope.Application) ;
        }
```

حالا که پروایدر (معرف رابط کاربری به IIS) تامین شده، نیاز است قلب کار یعنی ماژول معرفی گردد. اصلی‌ترین متدی که باید از اینترفیس ماژول پیاده سازی شود متد initialize است. این متد جایی است که تمام عملیات در آن رخ می‌دهد. در کلاس زیر imageCopyrightUI ما به معرفی مدخل entry رابط کاربری می‌پردازیم. در سازنده‌های این متد، پارامترهای نام، صفحه رابط کاربری وتوضیحی در مورد آن است. تصویر کوچک و بزرگ جهت آیکن سازی (در صورت عدم تعریف آیکن، چرخ دنده نمایش داده می‌شود) و توصیف‌های بلندتر را نیز شامل می‌شود.

```
internal class imageCopyrightUI : Module
    {
        protected override void Initialize(IServiceProvider serviceProvider, ModuleInfo moduleInfo)
        {
            base.Initialize(serviceProvider, moduleInfo);
            IControlPanel controlPanel = (IControlPanel)GetService(typeof(IControlPanel));
            ModulePageInfo modulePageInfo = new ModulePageInfo(this, typeof(imageCopyrightUIPage),
"Image Copyright", "Image Copyright",Resource1.Visual_Studio_2012,Resource1.Visual_Studio_2012);
            controlPanel.RegisterPage(modulePageInfo);
        }
    }
```

شیء ControlPanel مکانی است که قرار است آیکن ماژول نمایش داده شود. شکل زیر به خوبی نام همه قسمت‌ها را بر اساس نام کلاس و اینترفیس آن‌ها دسته بندی کرده است:



پس با تعریف این کلاس جدید ما روی صفحه‌ی کنترل پنل IIS، یک آیکن ساخته و صفحه‌ی رابط کاربری را به نام imageCopyrightUIPage، در آن ریجستر می‌کنیم. این کلاس را پایینتر شرح داده‌ایم. ولی قبل از آن اجازه بدهید تا انواع کلاس هایی را که برای ساخت صفحه کاربرد دارند، بررسی نماییم. در این مثال ما با استفاده از پایه‌ای‌ترین کلاس، ساده‌ترین نوع صفحه ممکن را خواهیم ساخت. 4 کلاس برای ساخت یک صفحه وجود دارند که بسته به سناریوی کاری، شما یکی را انتخاب می‌کنید.

ModulePage	شامل اساسی‌ترین متدها و سورس‌ها شده و هیچگونه رابط کاری ویژه‌ای را در اختیار شما قرار نمی‌دهد. تنها یک صفحه‌ی خام به شما می‌دهد که می‌توانید از آن استفاده کرده یا حتی با ارث بری از آن، کلاس‌های جدیدتری را برای ساخت صفحات مختلف و ویژه‌تر بسازید. در حال حاضر که هیچ کدام از ویژگی‌های IIS فعلی از این کلاس برای ساخت رابط کاربری استفاده نکرده‌اند.
ModuleDialogPage	یک صفحه شبیه به دیالوگ را ایجاد می‌کند و شامل دکمه‌های Apply و Cancel میشود به همراه یک سری متدهای اضافی‌تر که اجازه‌ی override کردن آنها را دارید. همچنین یک سری از کارهایی چون refresh و از این دست عملیات خودکار را نیز انجام میدهد. از نمونه رابطهایی که از این صفحات استفاده می‌کنند میتوان machine key و management service را اسم برد.

ModulePage	شامل اساسی‌ترین متدها و سورس‌ها شده و هیچگونه رابط کاری ویژه‌ای را در اختیار شما قرار نمی‌دهد. تنها یک صفحه‌ی خام به شما می‌دهد که می‌توانید از آن استفاده کرده یا حتی با ارث بری از آن، کلاس‌های جدیدتری را برای ساخت صفحات مختلف و ویژه‌تر بسازید. در حال حاضر که هیچ کدام از ویژگی‌های IIS فعلی از این کلاس برای ساخت رابط کاربری استفاده نکرده‌اند.
ModulePropertiesPage	این صفحه یک رابط کاربری را شبیه پنجره property که در ویژوال استادیو وجود دارد، در دسترس شما قرار میدهد. تمام عناصر آن در یک حالت گرید grid لیست می‌شوند. از نمونه‌های موجود میتوان به CGI,ASP.Net Compilation اشاره کرد.
ModuleListPage	این کلاس برای مواقعی کاربرد دارد که شما قرار است لیستی از آیتم‌ها را نشان دهید. در این صفحه شما یک ListView دارید که میتوانید عملیات جست و جو، گروه بندی و نحوه‌ی نمایش لیست را روی آن اعمال کنید.

در این مثال ما از اولین کلاس نامبرده که پایه‌ی همه کلاس هاست استفاده می‌کنیم. کد زیر را در کلاسی به اسم imageCopyrightUIPage می‌نویسیم:

```
public sealed class imageCopyrightUIPage : ModulePage
    {
        public string message;
        public bool featureenabled;
        public string color;

        ComboBox _colCombo = new ComboBox();
        TextBox _msgTB = new TextBox();
        CheckBox _enabledCB = new CheckBox();

        public imageCopyrightUIPage()
        {
            this.Initialize();
        }


        void Initialize()
        {
            Label crlabel = new Label();
            crlabel.Left = 50;
            crlabel.Top = 100;
            crlabel.AutoSize = true;
            crlabel.Text = "Enable Image Copyright:";
            _enabledCB.Text = "";
            _enabledCB.Left = 200;
            _enabledCB.Top = 100;
            _enabledCB.AutoSize = true;

            Label msglabel = new Label();
            msglabel.Left = 150;
            msglabel.Top = 130;
            msglabel.AutoSize = true;
            msglabel.Text = "Message:";
            _msgTB.Left = 200;
            _msgTB.Top = 130;
            _msgTB.Width = 200;
            _msgTB.Height = 50;

            Label collabel = new Label();
            collabel.Left = 160;
            collabel.Top = 160;
            collabel.AutoSize = true;
            collabel.Text = "Color:";
            _colCombo.Left = 200;
```

```
            _colCombo.Top = 160;
            _colCombo.Width = 50;
            _colCombo.Height = 90;
            _colCombo.Items.Add((object)"Yellow");
            _colCombo.Items.Add((object)"Blue");
            _colCombo.Items.Add((object)"Red");
            _colCombo.Items.Add((object)"White");

            Button apply = new Button();
            apply.Text = "Apply";
            apply.Click += new EventHandler(this.applyClick);
            apply.Left = 200;
            apply.AutoSize = true;
            apply.Top = 250;

            Controls.Add(crlabel);
            Controls.Add(_enabledCB);
            Controls.Add(collabel);
            Controls.Add(_colCombo);
            Controls.Add(msglabel);
            Controls.Add(_msgTB);
            Controls.Add(apply);
        }

        public void ReadConfig()
        {
            try
            {
                ServerManager mgr;
                ConfigurationSection section;
                mgr = new ServerManager();
                Configuration config =
                mgr.GetWebConfiguration(
                       Connection.ConfigurationPath.SiteName,
                       Connection.ConfigurationPath.ApplicationPath +
                       Connection.ConfigurationPath.FolderPath);

                section = config.GetSection("system.webServer/imageCopyright");
                color = (string)section.GetAttribute("color").Value;
                message = (string)section.GetAttribute("message").Value;
                featureenabled = (bool)section.GetAttribute("enabled").Value;

            }

            catch
            { }

        }

        void UpdateUI()
        {
            _enabledCB.Checked = featureenabled;
            int n = _colCombo.FindString(color, 0);
            _colCombo.SelectedIndex = n;
            _msgTB.Text = message;
        }


        protected override void OnActivated(bool initialActivation)
        {
            base.OnActivated(initialActivation);
            if (initialActivation)
            {
                ReadConfig();
                UpdateUI();
            }
        }



        private void applyClick(Object sender, EventArgs e)
        {
            try
            {
                UpdateVariables();
                ServerManager mgr;
                ConfigurationSection section;
                mgr = new ServerManager();
                Configuration config =
                mgr.GetWebConfiguration
                (
```

```
                        Connection.ConfigurationPath.SiteName,
                        Connection.ConfigurationPath.ApplicationPath +
                        Connection.ConfigurationPath.FolderPath
                );

                section = config.GetSection("system.webServer/imageCopyright");
                section.GetAttribute("color").Value = (object)color;
                section.GetAttribute("message").Value = (object)message;
                section.GetAttribute("enabled").Value = (object)featureenabled;

                mgr.CommitChanges();

            }

            catch
            { }

        }

        public void UpdateVariables()
        {
            featureenabled = _enabledCB.Checked;
            color = _colCombo.Text;
            message = _msgTB.Text;
        }
    }
```

اولین چیزی که در کلاس بالا صدا زده می‌شود، سازنده‌ی کلاس هست که ما در آن یک تابع تعریف کردیم به اسم **initialize** که به آماده سازی اینترفیس یا رابط کاربری می‌پردازد و کنترل‌ها را روی صفحه می‌چیند. این سه کنترل، یکی *Combox* برای تعیین رنگ، یک *Checkbox* برای فعال بودن ماژول و دیگری هم یک *textbox* جهت نوشتن متن است. مابقی هم که سه *label* برای نامگذاری اشیاست. بعد از اینکه کنترل‌ها روی صفحه درج شدند، لازم است که تنظیمات پیش فرض یا قبلی روی کنترل‌ها نمایش یابند که اینکار را به وسیله تابع **readConfig** انجام می‌دهیم و تنظیمات خوانده شده را در متغیرهای عمومی قرار داده و با استفاده از تابع **UpdateUI** این اطلاعات را روی کنترل‌ها ست می‌کنیم و به این ترتیب *UI* به روز می‌شود. این دو تابع را به ترتیب پشت سر هم در یک متد به اسم **OnActivated** که *override* کرده‌ایم صدا میزنیم. در واقع این متد یک جورایی همانند رویداد *Load* می‌باشد؛ اگر *true* برگرداند اولین فعال سازی رابط کاربری بعد از باز شدن IIS است و در غیر این صورت *false* بر میگرداند.

در صورتی که کاربر مقادیر را تغییر دهد و روی گزینه *apply* کلیک کند تابع applyClick اجرا شده و ابتدا به تابع UpdateVariables ارجاع داده میشود که در آن مقادیر خوانده شده و در متغیرهای *Global* قرار می‌گیرند و سپس با استفاده از دو شیء از نوع *serverManger* و *ConfigSection* جایگذاری یا ذخیره می‌شوند.
استفاده از دو کلاس Servermanager و Configsection در دو قسمت خواندن و نوشتن مقادیر به کار رفته‌اند. کلاس servermanager به ما اجازه دسترسی به تنظیمات IIS و قابلیت‌های آن را میدهد. در تابع ReadConfig مسیر وب سایتی را که در لیست IIS انتخاب شده است، دریافت کرده و به وب کانفیگ آن وب سایت رجوع نموده و تگ imageCopyright آن را که در تگ system.webserver قرار گرفته است، میخواند (در صورتی که این تگ در آن وب کانفیگ موجود نباشد، خواندن و سپس ذخیره مجدد آن روی تگ داخل فایل applicationHost.config اتفاق میفتد که نتیجتا برای همه‌ی وب سایت هایی که این تگ را ندارند یا مقدارهای پیش فرض آن را تغییر نداده‌اند رخ میدهد) عملیات نوشتن هم مشابه خواندن است. تنها باید خط زیر را در آخر برای اعمال تغییرات نوشت؛ مثل EF با گزینه Context.SaveChanges:

```
mgr.CommitChanges();
```

وقت آن است که رابط کاربری را به IIS اضافه کنیم: پروژه را Rebuild کنید. بعد از آن با خطوطی که قبلا در Post-Build Command نوشتیم باید dll ما در GAC ریجستر شود. برای همین آدرس زیر را در cmd تایپ کنید:

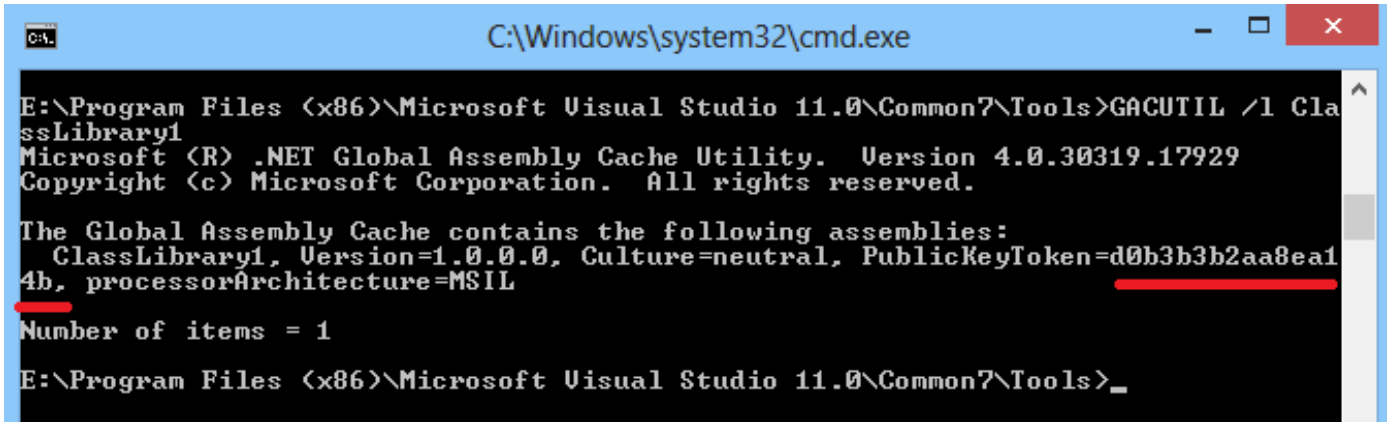
```
%vs110comntools%\vsvars32.bat
```

عبارت اول که مسیر ویژوال استودیوی شماست و عدد 110 یعنی نسخه‌ی 11. هر نسخه‌ای را که استفاده میکنید، یک صفر جلویش بگذارید و جایگزین عدد بالا کنید. مثلا نسخه 8 می‌شود 80 و فایل بچ بالا هم دستورات visual studio را برای شما آزاد می‌کند.

سپس دستور زیر را وارد کنید:

```
GACUTIL /l ClassLibrary1
```

کلمه classLibrary1 نام پروژه‌ی ما بود که در GAC ریجستر شده است. با سوییچ l تمامی اطلاعات اسمبلی‌هایی که در GAC ریجستر شده‌اند، نمایش می‌یابند. ولی اگر اسم آن اسمبلی را جلویش بنویسید، فقط اطلاعات آن اسمبلی نمایش میابد. با اجرای خط فوق میتوانیم کلید عمومی public key اسمبلی خود را بدانیم که در شکل زیر مشخص شده است:



پس اگر کلید را دریافت کرده‌اید، خط زیر را به فایل administration.config در تگ <ModuleProviders> اضافه کنید:

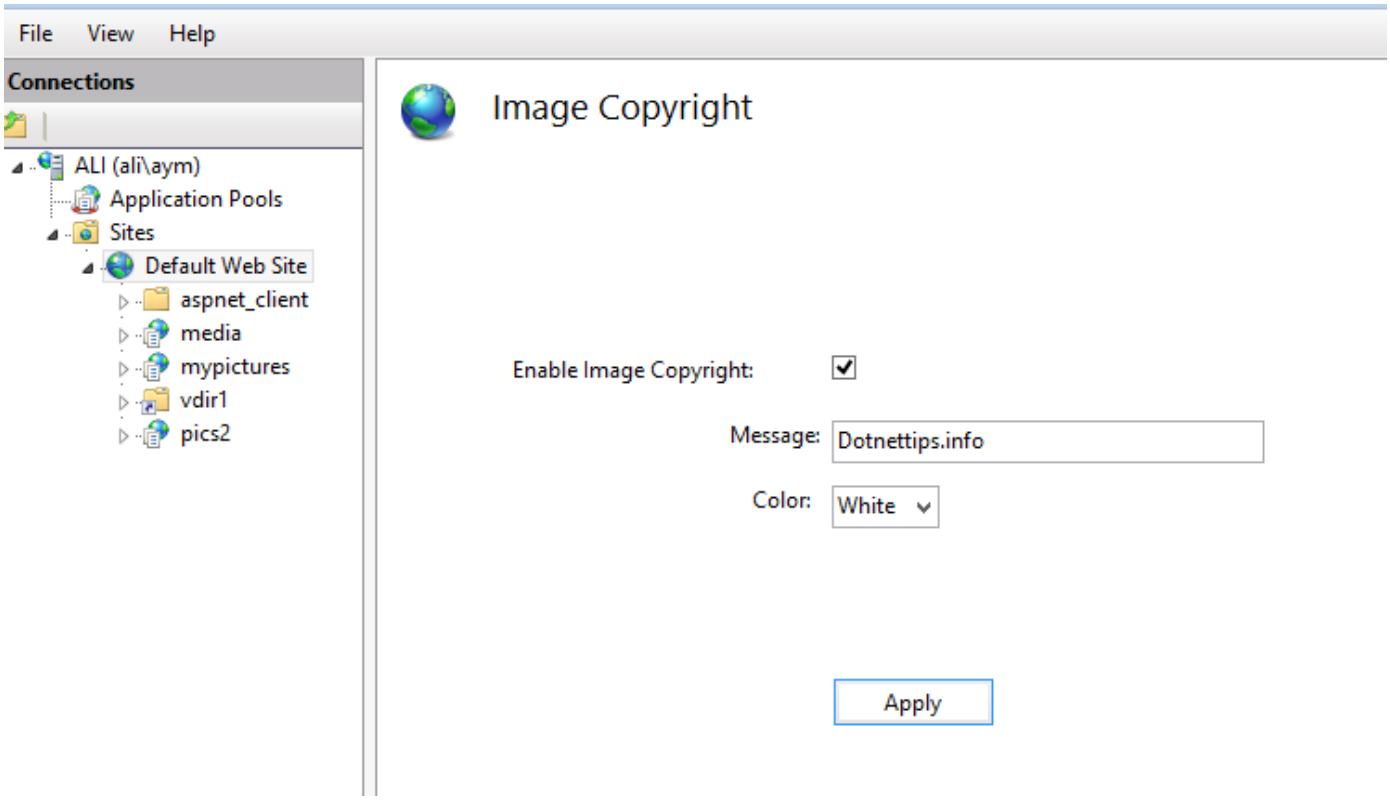
```
<add name="imageCopyrightUI" type="ClassLibrary1.imageCopyrightUIProvider,   ClassLibrary1,
Version=1.0.0.0, Culture=neutral, PublicKeyToken=d0b3b3b2aa8ea14b"/>
```

عبارت ClassLibrary1.imageCopyrightUIProvider به کلاس imageCopyrightUIProvider اشاره می‌کند که در این کلاس UI معرفی می‌شود. مابقی عبارت هم کاملا مشخص است و در لینک‌های بالا در مورد Strong name توضیح داده شده اند.

فایل administration.config در مسیر زیر قرار دارد:

```
%windir%\system32\inetsrv\config\administration.config
```

حالا تنها کاری که نیاز است، باز کردن IIS است. به بخش وب سایت‌ها رفته و اپلیکیشنی که قبلا با نام mypictures را ایجاد کرده بودیم، انتخاب کنید. در سمت راست، آخر لیست، بخش others باید ماژول ما دیده شود. بازش کنید و تنظمیات آن را تغییر دهید و حالا یک تصویر را از اپلیکیشن mypictures، روی مرورگر درخواست کنید تا تغییرات را روی تگ مشاهده کنید:



حالا دیگر باید ماژول نویسی برای IIS را فراگرفته باشیم. این ماژول‌ها می‌توانند از یک مورد ساده تا یک کلاس مهم و امنیتی باشند که روی سرور شما برای همه یا بعضی از وب سایت‌ها در حال اجرا هستند و در صورت لزوم و اجازه شما، برنامه نویس‌ها میتوانند مثل همه‌ی تگ‌های موجود در وب کانفیگ سایتی را که مینویسند، تگ ماژول شما و تنظیمات آن را با استفاده از attribute یا خصوصیت‌های تعریف شده، بر اساس سلایق و نیازهایشان تغییر دهند و روی سرور شما آپلود کنند. الان شما یک سرور خصوصی سازی شده دارید.

از آنجا که این مقاله طولانی شده است، باقی موارد ویرایشی روی این UI را در مقاله بعدی بررسی خواهیم کرد.