عنوان: **نمایش رکوردها به ترتیب اولویت به کمک jQuery UI sortable در ASP.NET MVC**
نویسنده: امیرحسین مرجانی
تاریخ: ۱۳۹۱/۰۹/۰۸ ۲۰:۴۷
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net MVC, jQuery, jQuery Ajax, UI, MVC

همان طور که می‌دانید کاربرد پذیری در خیلی از پروژه‌ها حرف اول رو می‌زند و کاربر دوست دارد کارهایی که انجام می‌دهد خیلی راحت و با استفاده از موس باشد.یکی از کار هایی که در اکثر پروژه‌ها نیاز است ، چیدمان ترتیب رکوردها است. ما می‌خواهیم در این پست ترتیبی اتخاذ کنیم که کاربر بتواند رکوردها را به هر ترتیبی که دوست دارد نمایش دهد.

از توضیحاتی که قبلا دادم مشخص است که این کار احتمالا در ASP.NET WebForm کار سختی نیست ولی این کار باید در MVC از ابتدا طراحی شود.

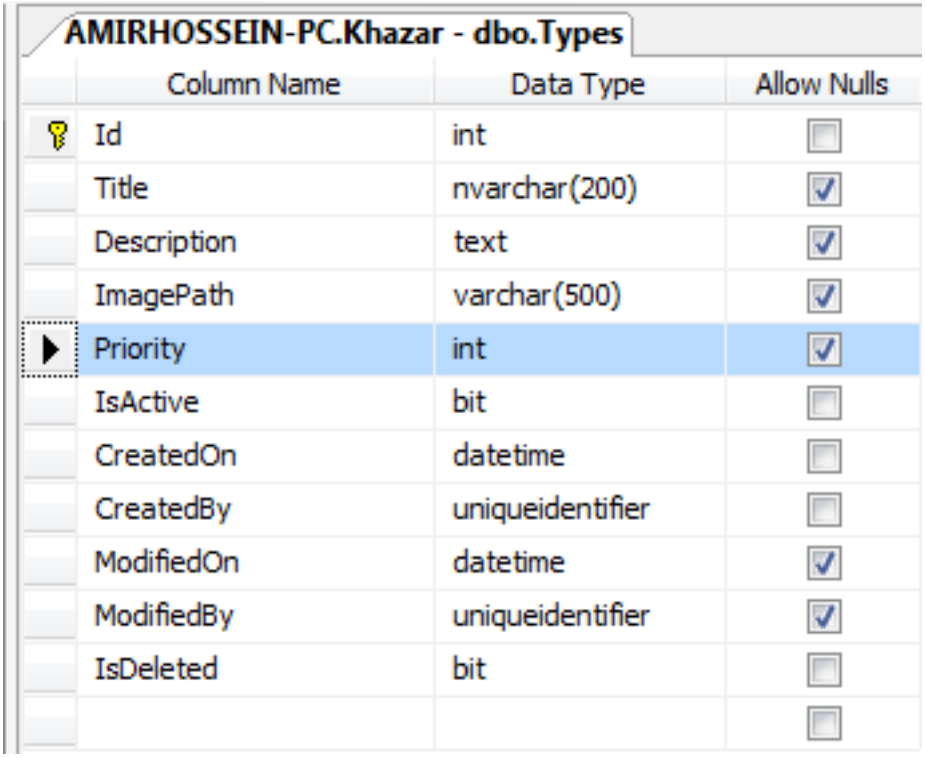
طرح سوال : یک سری رکورد از یک Table داریم که می‌خواهیم به ترتیب وارد شدن رکوردها نباشد و ترتیبی که ما می‌خواهیم نمایش داده شود.

پاسخ کوتاه : خب باید ابتدا یک فیلد (برای اولویت بندی) به Table اضافه کنیم بعد اون فیلد رو بنا به ترتیبی که دوست داریم رکوردها نمایش داده شود پر کنیم (Sort می کنیم) و در آخر هم هنگام نمایش در View رکوردها را بر اساس این فیلد نمایش می‌دهیم.

(این پست هم در ادامه پست قبلی در همان پروژه است و از همان Table ها استفاده شده است)

اضافه کردن فیلد :

ابتدا یک فیلد به Table مورد نظر اضافه می‌کنیم. من اسم این فیلد رو Priority گذاشتم. Table من چنین وضعیتی دارد.



افزودن فایل‌های jQuery UI :

در این مرحله شما نیاز دارید فایل‌های مورد نیاز برای Sort کردن رکوردها را اضافه کنید. شما می‌توانید فقط فایل‌های مربوط به Sortable را به صفحه خودتان اضافه کنید و یا مثل من فایل هایی که حاوی تمام قسمت‌های jQuery UI هست را اضافه کنید.

من برای این کار از Section استفاده کردم ، ابتدا در Head فایل Layout دو Section تعریف کردم برای CSS و JavaScript . و فایل‌های مربوط به Sort کردن را در صفحه ای که باید عمل Sort انجام بشود در این Section ها قرار دادم.

فایل Layout

```
<head>
    <meta charset="utf-8" />
    @RenderSection("meta", false)
    <title>@ViewBag.Title</title>
    <link href="@Url.Content("~/Content/~Site.css")" rel="stylesheet" type="text/css" />
    <link href="@Url.Content("~/Content/redactor/css/redactor.css")" rel="stylesheet" type="text/css"
/>
    <link href="@Url.Content("~/Content/css/bootstrap-rtl.min.css")" rel="stylesheet" type="text/css"
/>
    @RenderSection("css", false)
    <script src="@Url.Content("~/Scripts/jquery-1.8.2.min.js")" type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/modernizr-1.7.min.js")" type="text/javascript"></script>
    <script src="@Url.Content("~/Content/js/bootstrap-rtl.js")" type="text/javascript"></script>
    <script src="@Url.Content("~/Content/redactor/redactor.min.js")" type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.unobtrusive-ajax.min.js")"
type="text/javascript"></script>
    @RenderSection("js", false)

</head>
```

فایل Index.chtml در پوشه کنترلر Type

```
@model IEnumerable<KhazarCo.Models.Type>
@{
    ViewBag.Title = "Index";
    Layout = "~/Areas/Administrator/Views/Shared/_Layout.cshtml";
}
@section css
{<link href="@Url.Content("~/Content/themes/base/jquery-ui.css")" rel="stylesheet" type="text/css" />
}
@section js
{
    <script src="@Url.Content("~/Scripts/jquery-ui-1.9.0.min.js")" type="text/javascript"></script>
}
```

در آخر فایل Index.chtml به اینصورت شده است:

```
<h2>
    نوع ها</h2>
<p>
    @Html.ActionLink("ایجاد یک مورد جدید", "Create", null, new { @class = "btn btn-info" })
</p>
<table>
    <thead>
        <tr>
            <th>
                عنوان
            </th>
            <th>
                توضیحات
            </th>
            <th>
                فعال
            </th>
            <th>
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model.OrderBy(m => m.Priority))
        {
            <tr id="@item.Id">
                <td>
                    @Html.DisplayFor(modelItem => item.Title)
                </td>
                <td>
                    @(new HtmlString(item.Description))
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.IsActive)
                </td>
                <td>
                    @Html.ActionLink("ویرایش", "Edit", new { id = item.Id }, new { @class = "btnEdit
label label-warning" })
                    |
                    @Html.ActionLink("مشاهده", "Details", new { id = item.Id }, new { @class =
"btnDetails label label-info" })
                    |
                    @Html.ActionLink("حذف", "Delete", new { id = item.Id }, new { @class = "btnDelete
label label-important" })
                </td>
            </tr>
        }
    </tbody>
</table>
```

** توجه داشته باشید که من به هر tr یک id اختصاص داده ام که این مقدار id همان مقدار فیلد Id همان رکورد هست ، ما برای مرتب کردن به این Id نیاز داریم (خط 25).

افزودن کدهای کلاینت :

حالا باید کدی بنویسم که دو کار را برای ما انجام دهد : اول حالت Sort پذیری را به سطرهای Table بدهد و دوم اینکه هنگامی که ترتیب سطرهای تغییر کرد ما را با خبر کند:

```
<script type="text/javascript">
    $(function () {
        $("table tbody").sortable({
            helper: fixHelper,
            update: function (event, ui) {
                jQuery.ajax('@Url.Action("Sort", "Type", new { area = "Administrator" })', {
                    data: { s: $(this).sortable('toArray').toString() }
                });
            }
        }).disableSelection();
    });
    var fixHelper = function (e, ui) {
        ui.children().each(function () {
            $(this).width($(this).width());
        });
        return ui;
    };
</script>
```

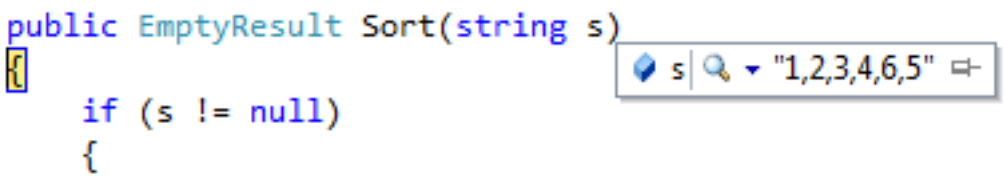
توضیح کد :

در این کد ما حالت ترتیب پذیری را به Table می دهیم و هنگامی که عمل Update در Table انجام شد تابع مربوطه اجرا می‌شود. ما در این تابع، ترتیب جدید سطرها را می‌گیریم (** به کمک مقدار Id که به هر سطر دادیم ، این مقدار Id برابر بود با Id خود رکورد در Database) و به کمک jQuery.ajax به تابع Sort از کنترلر Type در منطقه (Administrator (area ارسال می‌کنیم و در آنجا ادامه کار را انجام میدهیم.

تابع fixHelper هم به ما کمک می‌کند که هنگامی که سطرها از جای خود جدا می‌شوند ، دارای عرض یکسانی باشند و عرض آن‌ها تغییری نکند.

افزودن کد Server :

حالا باید تابع Sort که مقادیر را به آن ارسال کردیم بنویسم. من این تابع را بر اساس مقداری که از کلاینت ارسال می‌شود اینگونه طراحی کردم .




```
public EmptyResult Sort(string s)
        {
            if (s != null)
            {
                var ids = new List<int>();
                foreach (var item in s.Split(','))
```

```
                {
                    ids.Add(int.Parse(item));
                }
                int intpriority = 0;

                foreach (var item in ids)
                {
                    intpriority++;
                    db.Types.Single(m => m.Id == item).Priority = intpriority;
                }
                db.SaveChanges();
            }
            return new EmptyResult();
        }
```

در ابتدا مقادیر Id که از کلاینت به صورت String ارسال شده است را می‌گیریم و بعد به همان ترتیب ارسال در لیستی از int قرار می‌دهیم ids.

سپس به اضای هر رکورد Type مقدار اولویت را به فیلدی که برای همین مورد اضافه کردیم Priority اختصاص می‌دهیم . و در آخر هم تغییرات را ذخیره می‌کنیم. (خود کد کاملا واضح است و نیاری به توضیح بیشتر نیست)

حالا باید هنگامی که لیست Type ها نمایش داده می‌شود به ترتیب (OrderBy) فیلد Priority نمایش داده شود پس تابع Index را اینطور تغییر می‌دهیم.

```
public ViewResult Index()
        {
            return View(db.Types.Where(m => m.IsDeleted == false).OrderBy(m => m.Priority));
        }
```

این هم خروجی کار من:

نوع ها

ایجاد یک مورد جدید

عنوان	توضیحات	فعال	
مدل اهرمی	این مدل از طرح هاي جديد و نوين بهره مي برد.	☑	ویرایش | مشاهده | حذف
مدل کلاسیک	طرح هاي قديمي شيرآلات	☑	ویرایش | مشاهده | حذف
سینک شاوری	توضيحاتي در مورد سينک هاي شاوري	☑	ویرایش | مشاهده | حذف
دسته ها و پوسته ها	توضيحاتي در مورد دسته ها و پوسته ها	☑	ویرایش | مشاهده | حذف
علم های دوش	توضيحاتي در مورد علم هاي دوش	☑	ویرایش | مشاهده | حذف
سایر محصولات		☑	ویرایش | مشاهده | حذف

این عکس مربوط به است به قسمت مدیریت پروژه شیرآلات مرجان خزر .

نظرات خوانندگان

نویسنده: سعید
تاریخ: ۱۳۹۱/۰۹/۰۸ ۲۱:۱۲

« این کار احتمالا در ASP.NET WebForm کار سختی نیست »

اتفاقا کار ساده‌ای نیست و همین مراحل باید طی شود. ضمن اینکه کار با jquery ajax در آنجا به این یک دستی نیست. نیاز است در code behind فرم، متد وب سرویس مانندی به صورت استاتیک تعریف شود (که خودش سبب می‌شود تا دسترسی به اعتبار سنجی توکار مبتنی بر فرم‌ها محدود شود) یا اینکه از یک هندلر مجزا بجای یک اکشن متد کمک گرفته شود ... خلاصه خیلی داستان دارد.

نویسنده: امیرحسین مرجانی
تاریخ: ۱۳۹۱/۰۹/۰۸ ۲۱:۱۶

من خودم این کار رو در webform انجام ندادم و خیلی هم با سختی هاش آشنایی ندارم.
ولی فکر کردم شاید با update panel بشه این کار رو راحت انجام داد.
خب نظرتون راجب به راه حل من چیه؟

راه حل خوبی ارائه دادم؟ متن قابل فهم بود؟

نویسنده: سعید
تاریخ: ۱۳۹۱/۰۹/۰۸ ۲۱:۲۰

بسیار عالی.

فقط در مورد update panel ... تنها کاری که انجام می‌دهد کپسوله کردن ارسال مقادیر به سرور است به صورت ajax سازگار با کنترل‌های دارای view state . بیشتر از این کاری انجام نمی‌دهد. مابقی آن اگر یک سری کنترل در toolkit آن موجود باشد برای این کارها یا خیر. این toolkit هم محدود است و آنچنان به روز نمی‌شود.

نویسنده: امیرحسین مرجانی
تاریخ: ۱۳۹۱/۰۹/۰۸ ۲۱:۲۳

ممنونم از پاسختون
از اینکه سوییچ کردم روی MVC خوشحالم :)
البته یکی از دلایلی هم که این کار رو کردم همین راحتی استفاده Ajax بود.

نویسنده: امیر آشنا
تاریخ: ۱۳۹۱/۰۹/۰۸ ۲۲:۴

سلام
مطلب مفیدی بود ولی می‌تونستید بیشتر توضیح بدید.
برای ما افراد مبتدی درک بعضی قسمت‌ها کمی مشکل هست.
ولی باز هم ممنونم

نویسنده: امیرحسین مرجانی

تاریخ: ۱۳۹۱/۰۹/۰۸ ۲۲:۷

سلام
بفرمایید کدوم قسمت‌ها تا بیشتر توضیح بدم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۰۸ ۲۲:۲۱

پیشنیازهای مطلب جاری:
- در مورد RenderSection : (^)
- مقدمه‌ای بر jQuery Ajax در MVC : (^)
- db.SaveChanges : کل مباحث Entity framework سایت (^)
- EmptyResult و کلا خروجی‌های اکشن متدها: (^)
- علت استفاده از Url.Action Sort@ در حین آدرس دهی: (^)

نویسنده: امیرحسین مرجانی
تاریخ: ۱۳۹۱/۰۹/۰۸ ۲۲:۲۳

واقعا تشکر می‌کنم بابت این لیستی که ارائه کردید.
بحث تکمیل شد
به نظر شما باید بیشتر به جزییات اهمیت داده می‌شد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۰۸ ۲۲:۴۱

ضرورتی نداره. چون واقعا به اندازه لیستی که عنوان شد نیاز به پیشنیاز درک این مطالب هست و فرصت تکرار آن‌ها نیست. این مطالب جدید، یک سری مطالب تکمیلی هستند نه مطالب پایه و از صفر.

نویسنده: محسن موسوی
تاریخ: ۱۳۹۱/۰۹/۰۹ ۰:۲۴

با استفاده از Web API Controller تقریبا به همین راحتی امکانپذیره. تغییر زیادی هم نیاز نداره.

نویسنده: امیرحسین مرجانی
تاریخ: ۱۳۹۱/۰۹/۰۹ ۰:۳۷

ممنونم آقای موسوی
قابل توجه آقا سعید (احتمالا این مطلب برای شما مفیده)

نویسنده: سعید
تاریخ: ۱۳۹۱/۰۹/۰۹ ۰:۵۲

- نیاز به vs2012 داره.
- نمیشه یک فرم رو strongly typed تعریف کرد مثل مثال بالا:

```
@model IEnumerable<KhazarCo.Models.Type>
```

نویسنده: محسن موسوی
تاریخ: ۱۳۹۱/۰۹/۰۹ ۱۰:۲۱

بحث در مورد UI نبود.بحث در مورد استفاده کردن بود.
در ضمن در Web Formها نیز از امکاناتی شبیه به MVC می‌توان بهره برد.(^)

نویسنده: سعید
تاریخ: ۱۳۹۱/۰۹/۰۹ ۱۰:۴۶

فقط mvc هست که امکان استفاده از چند view engine را با هم دارد.

صفحه‌ای که لینک دادید مربوط است به web pages asp.net و نه web forms. این web pages برای کار با webmatrix طراحی شده.

نویسنده: محسن موسوی
تاریخ: ۱۳۹۱/۰۹/۰۹ ۱۱:۴

- استفاده از این قابلیت در Web Page ها توسط VS
- استفاده از این قابلیت در Web Form ها توسط VS
هر دو مورد توسط VS استفاده شده است.

نویسنده: سعید
تاریخ: ۱۳۹۱/۰۹/۰۹ ۱۱:۴۵

```
<iframe src="/twitter" frameborder="0" height="400" width="270" scrolling="no"></iframe>
```

فلسفه و کاربرد web pages متفاوت است. در مورد وب فرم‌ها هم نمی‌تونید از razor استفاده کنید چون در یک فایل نمی‌شود از دو موتور view استفاده کرد. موتور view وب فرم‌ها، نامش همین web forms view engine است و قابل تعویض هم نیست (برخلاف MVC). در مقاله‌ای که لینک دادید، داره از یک *iframe* استفاده می‌کنه برای الحاق razor.

به علاوه در مورد Web API که کلا مطلب جدیدی نیست. نسخه ساده شده WCF است.

نویسنده: محسن موسوی
تاریخ: ۱۳۹۱/۰۹/۰۹ ۱۲:۲۷

دوست عزیز
بحث ما در مورد توانایی انجام موارد فوق الذکر بود. نه اینکه حالا چون ...
به توانایی‌های MVC شکی نیست.مسئله اصلی اینه که آیا پست جاری را میتوان به راحتی با WebPage و یا WebForm انجام داد؟!
چه از طریق قابلیت‌های ASP.NET Web Form و ASP.NET Web Page راحته.
استفاده از Razor چه از طریق Jquery و یا Iframe (نظر قبلی)و یا به طور مستقل و یا روش‌های دیگر در ASP.NET Web Form و ASP.NET Web Page
و استفاده معمولی با توانایی‌های DataBind
یا بطور کامل از Razor View Engine در Web Page
در نهایت کار سختی نیست.

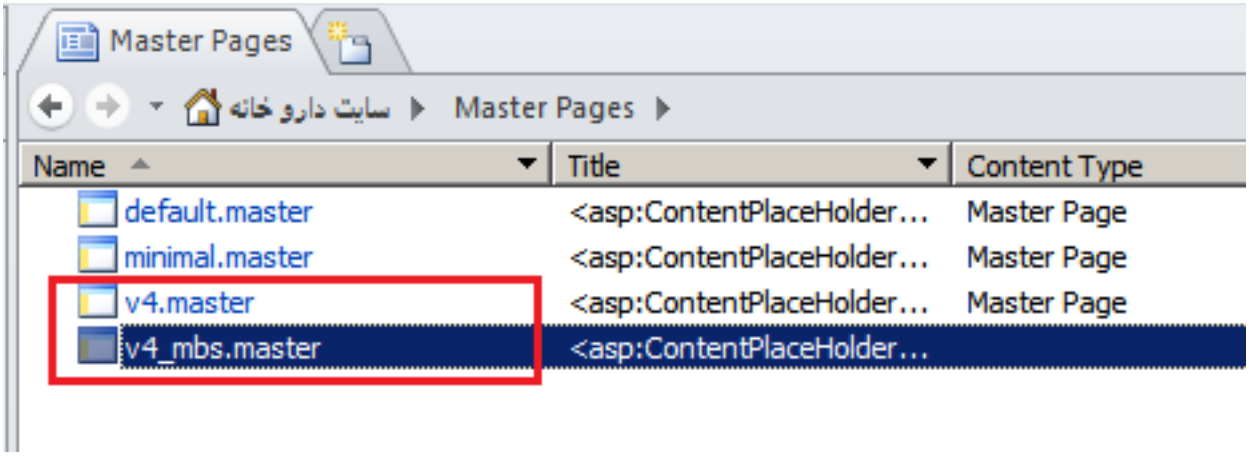
نویسنده: سعید
تاریخ: ۱۳۹۱/۰۹/۰۹ ۱۲:۴۳

razor بی‌جهت اینجا طرح شد. من در مورد strongly typed view سطری رو نوشتم. این نوع viewها مستقل از نوع view

engine هستند. در mvc حتی با موتور web forms هم میشه یک چنین viewهای تحت نظر کامپایلری رو با پشتیبانی کامل از intelisense داشت.

عنوان: **عدم نمایش Ribbon برای کاربران ناشناس (Anonymous Users) در شیرپوینت**
نویسنده: محمد باقر سیف اللهی
تاریخ: ۱۳۹۱/۱۱/۱۹ ۱۰:۲۵
آدرس: www.dotnettips.info
برچسب‌ها: SharePoint, Ribbon, Tricks, UI

یکی از نیازهای مشتریان هنگام استفاده از سایت‌های تحت شیرپوینت ، عدم نمایش نوار مدیریتی بالای صفحه یا همان Ribbon برای کاربران ناشناس است .
شاید بتوان گفت که مزیت این راهکار نسبت به دیگر راه کارها ، تصحیح نمایش Scroll Bar مرورگر است که ممکن است در برخی روش‌ها با مشکل مواجه شود. دلیل این امر هم این است که شیرپوینت برای افزودن Ribbon به بالای صفحه Vertical Scroll Bar را از صفحه حذف می‌کند و سپس Scroll Bar سفارشی خود را به صفحه طوری اضافه می‌کند تا نوار Ribbon همیشه در بالاترین نقطه از صفحه بماند
همچنین در این روش از بارگذاری نوار هنگام بالا آمدن سایت نیز البته برای کاربران ناشناس جلوگیری می‌شود
و اما روش :
سایت خود را با SharePoint Designer باز کرده و از Master Page یک کپی تهیه کنید و آن را به عنوان پیش فرض سایت تعیین کنید



سپس تگ زیر را در صفحه بیابید :

```
<div id="s4-ribbonrow">
```

و تگ زیر را به ابتدای آن (قبل از تگ) اضافه کنید :

```
<Sharepoint:SPSecurityTrimmedControl runat="server" Permissions="AddDelPrivateWebParts">
```

مانند تصویر زیر :

```
<SharePoint:DelegateControl runat="server" ControlId="GlobalNavigation"/>
<!-- Start -->
<SharePoint:SPSecurityTrimmedControl runat="server" Permissions="AddDelPrivateWebParts">

<div id="s4-ribbonrow" class="s4-pr s4-ribbonrowhidetitle">
    <div id="s4-ribboncont">
        <SharePoint:SPRibbon
            runat="server"
            PlaceholderElementId="RibbonContainer"
            CssFile="">
            <SharePoint:SPRibbonPeripheralContent
```

و تگ پایانی آن :

```
            <ContentTemplate>
                <WebPartPages:WebPartAdder ID="WebPartAdder" runat="server" />
            </ContentTemplate>
            <Triggers>
                <asp:PostBackTrigger ControlID="WebPartAdder" />
            </Triggers>
        </asp:UpdatePanel>
    </div>
</div>
<!-- END -->


</SharePoint:SPSecurityTrimmedControl>

<div id="s4-workspace">
```

نکته مهم در استفاده از این تگ ، ویژگی Permissions آن است که باید با دقت و بسته به نیاز شما تعریف شود :

برخی از این موارد عبارتند از :

.EmptyMask – Has no permissions on the Web site. Not available through the user interface

.ViewListItems – View items in lists, documents in document libraries, and view Web discussion comments

.AddListItems – Add items to lists, add documents to document libraries, and add Web discussion comments

EditListItems – Edit items in lists, edit documents in document libraries, edit Web discussion comments in documents, and customize Web Part Pages in document libraries.

DeleteListItems – Delete items from a list, documents from a document library, and Web discussion comments in documents.

.ApproveItems – Approve a minor version of a list item or document

.OpenItems – View the source of documents with server-side file handlers

.ViewVersions – View past versions of a list item or document

.DeleteVersions – Delete past versions of a list item or document

.CancelCheckout – Discard or check in a document which is checked out to another user

.ManagePersonalViews – Create, change, and delete personal views of lists

.ManageLists – Create and delete lists, add or remove columns in a list, and add or remove public views of a list

.ViewFormPages – View forms, views, and application pages, and enumerate lists

.Open – Allow users to open a Web site, list, or folder to access items inside that container

.ViewPages – View pages in a Web site

AddAndCustomizePages – Add, change, or delete HTML pages or Web Part Pages, and edit the Web site using a .SharePoint Foundation–compatible editor

.ApplyThemeAndBorder – Apply a theme or borders to the entire Web site

.ApplyStyleSheets – Apply a style sheet (.css file) to the Web site

.ViewUsageData – View reports on Web site usage

.CreateSSCSite – Create a Web site using Self-Service Site Creation

.ManageSubwebs – Create subsites such as team sites, Meeting Workspace sites, and Document Workspace sites

.CreateGroups – Create a group of users that can be used anywhere within the site collection

ManagePermissions – Create and change permission levels on the Web site and assign permissions to users and .groups

BrowseDirectories – Enumerate files and folders in a Web site using Microsoft Office SharePoint Designer 2007 .and WebDAV interfaces

.BrowseUserInfo – View information about users of the Web site

.AddDelPrivateWebParts – Add or remove personal Web Parts on a Web Part Page

.UpdatePersonalWebParts – Update Web Parts to display personalized information

ManageWeb – Grant the ability to perform all administration tasks for the Web site as well as manage content. Activate, deactivate, or edit properties of Web site scoped Features through the object model or through the user interface (UI). When granted on the root Web site of a site collection, activate, deactivate, or edit properties of site collection scoped Features through the object model. To browse to the Site Collection Features page and activate or deactivate site collection scoped Features through the UI, you must be a site collection administrator.

UseClientIntegration – Use features that launch client applications; otherwise, users must work on documents locally and upload changes.

UseRemoteAPIs – Use SOAP, WebDAV, or Microsoft Office SharePoint Designer 2007 interfaces to access the Web site.

ManageAlerts – Manage alerts for all users of the Web site.

CreateAlerts – Create e-mail alerts.

EditMyUserInfo – Allows a user to change his or her user information, such as adding a picture.

EnumeratePermissions – Enumerate permissions on the Web site, list, folder, document, or list item.

FullMask – Has all permissions on the Web site. Not available through the user interface.

حال خارج از تگ‌های SPSecurityTrimmedControl در ابتدا یا انتها ، باید تگ login را مانند زیر به آن اضافه کرد .

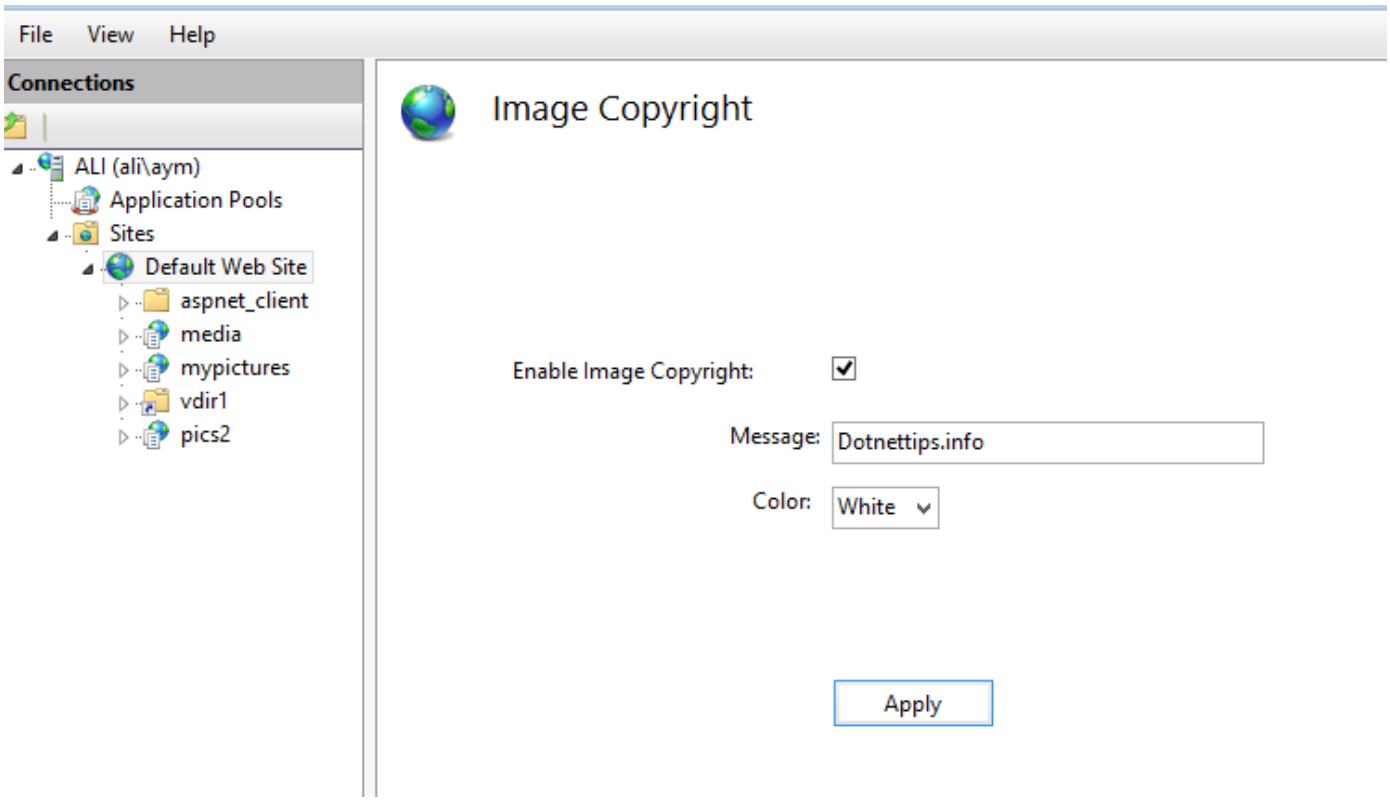
```
<!-- END -->

</SharePoint:SPSecurityTrimmedControl>
<asp:LoginView id="LoginView" runat="server">
 <AnonymousTemplate>
 <style type="text/css">
 body { overflow-y: scroll !important; overflow-x: hidden; }
 body #s4-workspace { overflow-x: hidden; overflow-y: auto !important; }
 </style>
 <!--[if lte IE 7]>
 <style type="text/css">
 html { overflow: auto !important; overflow-x: hidden; }
 body { overflow: auto !important; }
 </style>
 <![endif]-->
 </AnonymousTemplate>
</asp:LoginView>
```

و تمام :

موفق باشید

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

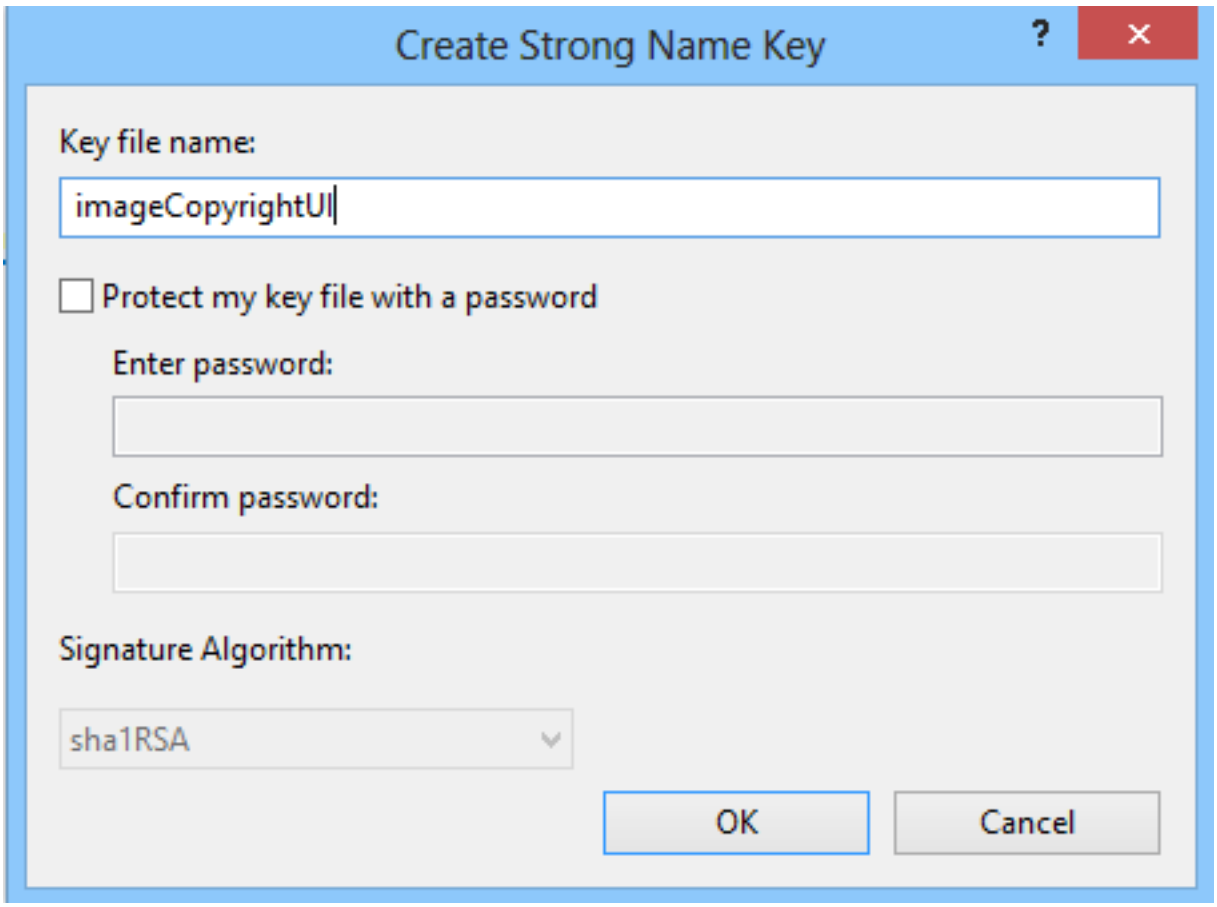
در قسمت قبلی ما یک هندلر ایجاد کردیم و درخواست‌هایی را که برای فایل jpg و به صورت GET ارسال میشد، هندل می‌کردیم و تگی را در گوشه‌ی تصویر درج و آن را در خروجی نمایش میدادیم. در این مقاله قصد داریم که کمی هندلر مورد نظر را توسعه دهیم و برای آن یک UI یا یک رابط کاربری ایجاد نماییم. برای توسعه دادن ماژولها 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:

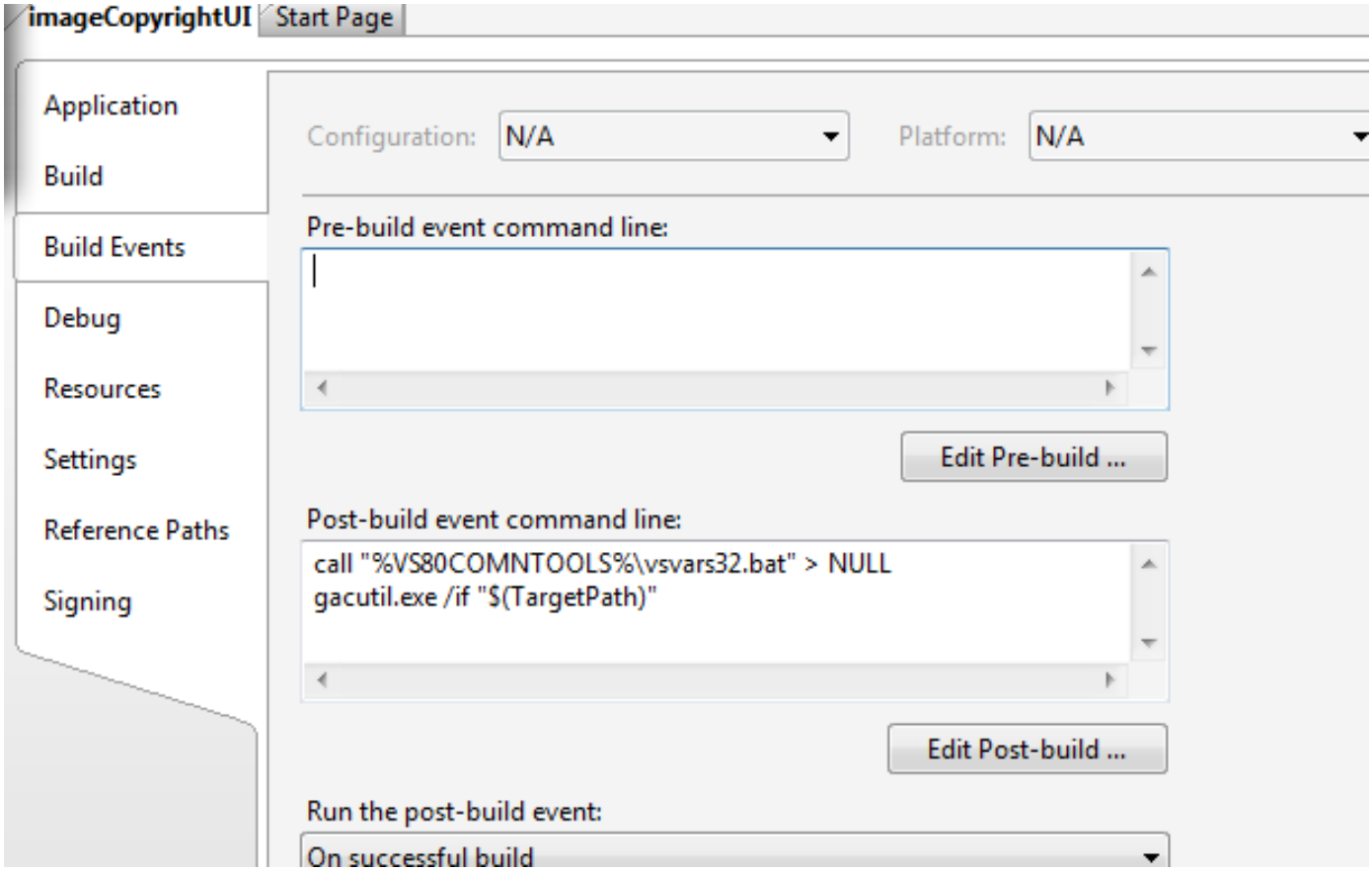
```
\Windows\system32\inetsrv
```

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.



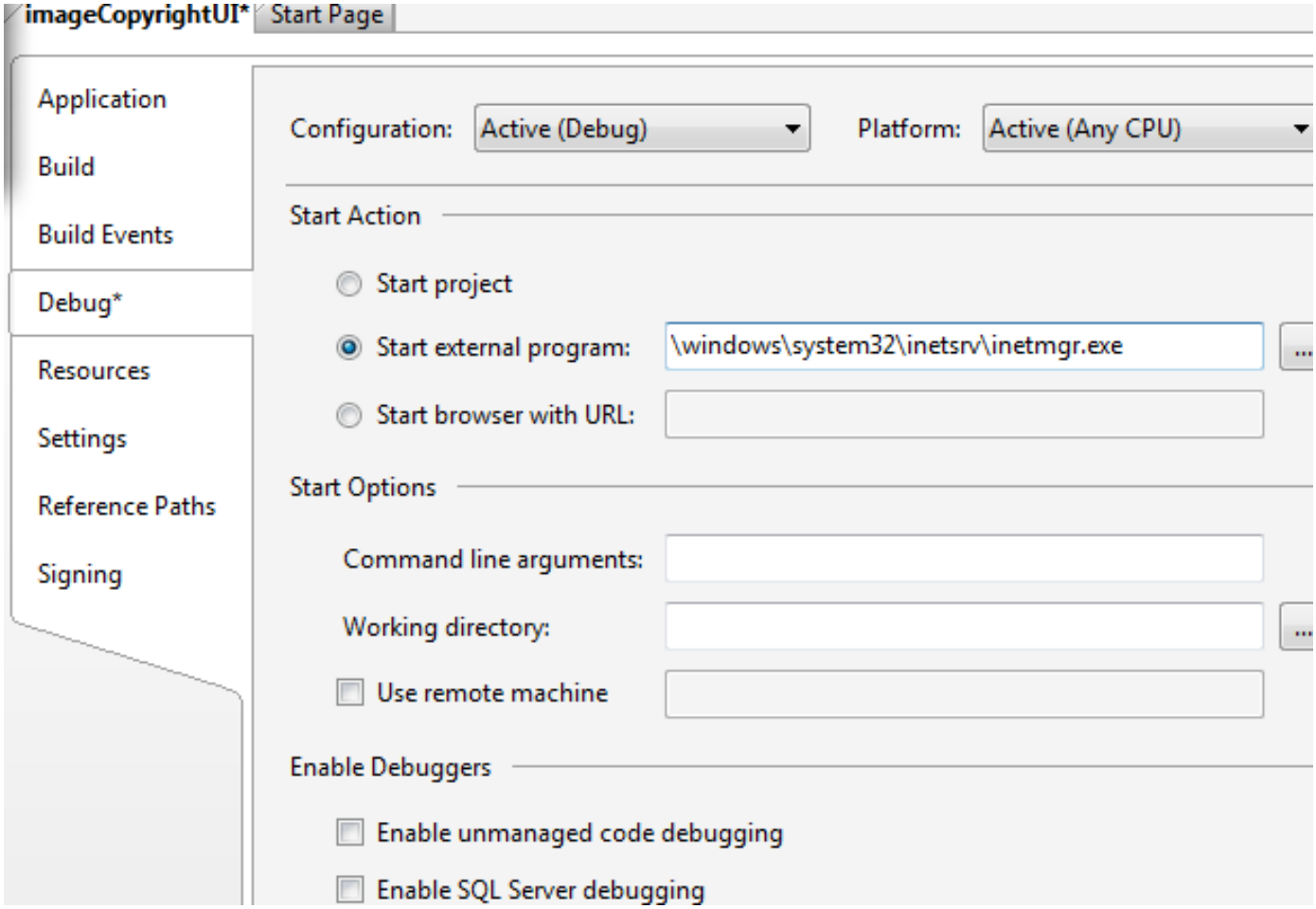
در مرحله بعدی در تب Build Events کد زیر را در بخش Post-build event command line اضافه کنید. این کد باعث می‌شود بعد از هر بار کامپایل پروژه، به طور خودکار در GAC ثبت شود:

```
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;
        }
    }
}
```

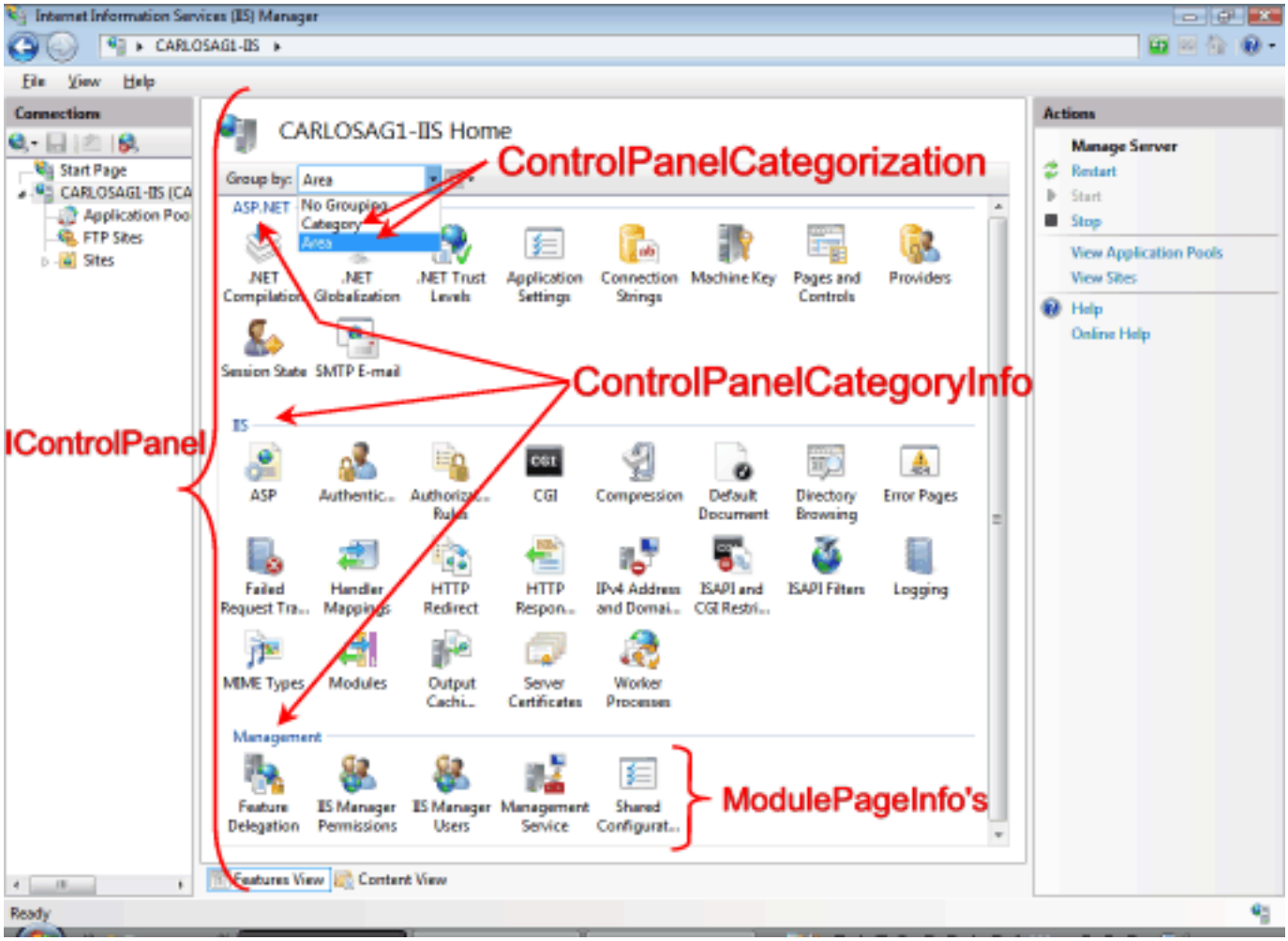
با ارث بری از کلاس module provider، سه متد بازنویسی می‌شوند که یکی از آن ها SupportsScope هست که میدان عمل پروایدر را مشخص می‌کند، مانند اینکه این پروایدر در چه میدانی باید کار کند که می‌تواند سه گزینه‌ی server,site,application باشد. در کد زیر مثلا میدان عمل application انتخاب شده است ولی در کد بالا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

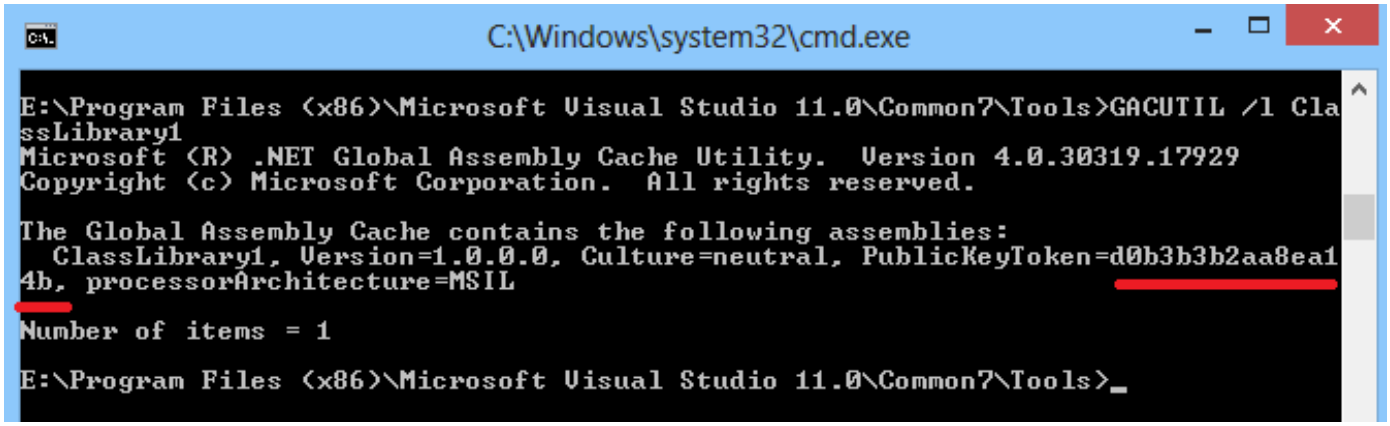
```
%vs110comntools%\vsvars32.bat
```

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:

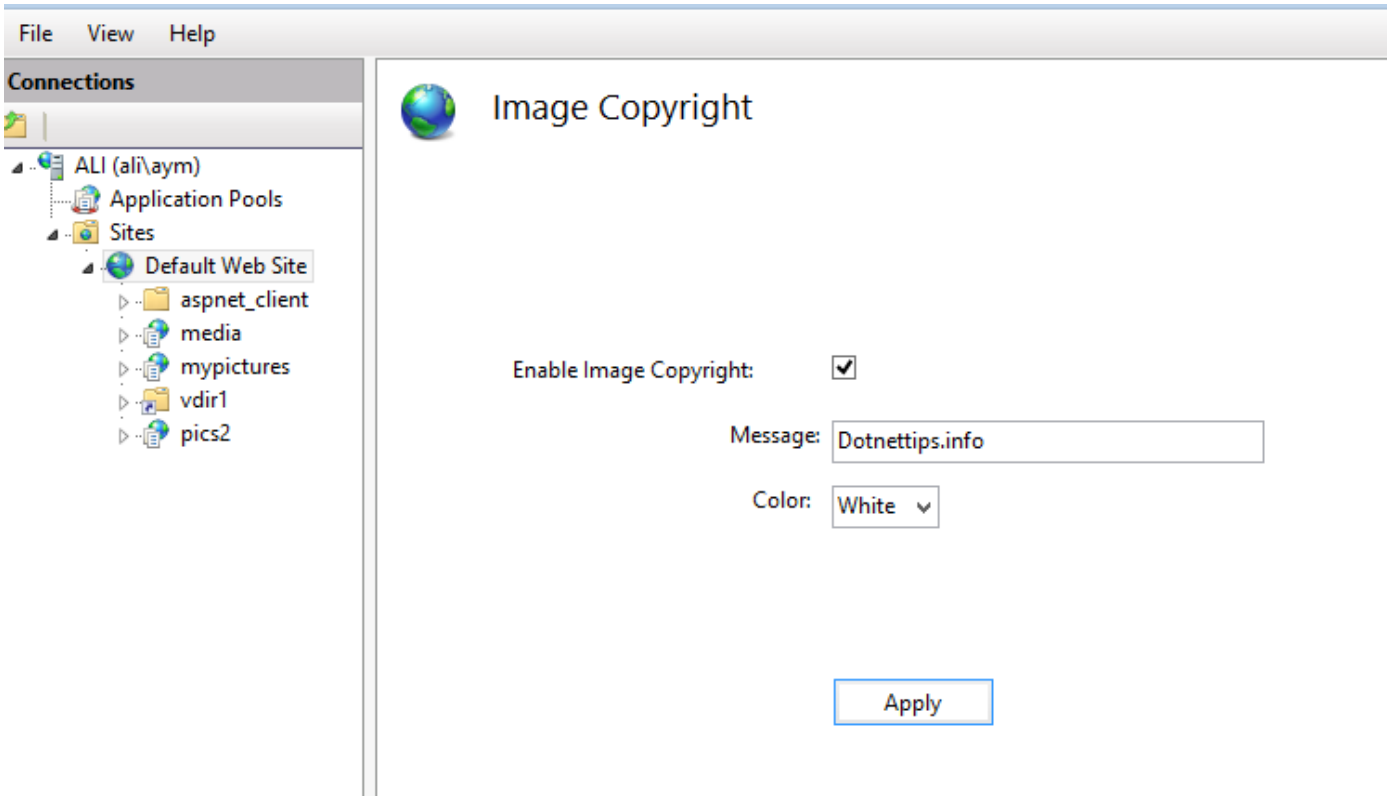
```
<add name="imageCopyrightUI" type="ClassLibrary1.imageCopyrightUIProvider,   ClassLibrary1,
Version=1.0.0.0, Culture=neutral, PublicKeyToken=d0b3b3b2aa8ea14b"/>
```

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.