عنوان: **JQuery Plugins #1**
نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۱/۱۲/۰۷ ۱۶:۲۵
آدرس: www.dotnettips.info
برچسب‌ها: jQuery, jQuery-Tips, Plugin

جی‌کوئری به عنوان مهم‌ترین و پرکاربردترین کتابخانه جاوا اسکریپتی، حالا در اکثر سایت‌های اینترنتی استفاده می‌شود و هر روز به قابلیت‌ها و امکانات آن اضافه می‌گردد. اما بیش از خود این کتابخانه، پلاگین‌های آن است که تحول عظیمی را در طراحی وب سایت‌ها ایجاد نموده است. از انواع اسلایدها، تصاویر، منوها، Tooltip ها، نمودارها، انیمیشن، جداول و هزاران پلاگین دیگر، همه و همه کدهای جاوا اسکریپتی است که با استفاده از جی کوئری به صورت پلاگین نوشته شده است و امکان استفاده مجدد را به ما می‌دهد.

از کجا شروع کنیم

برای نوشتن پلاگین یک تابع با نام خاصیتی جدید را به jQuery.fn اضافه می‌نماییم.

```
jQuery.fn.myPlugin = function() {
//محتویات پلاگین را اینجا می‌نویسیم
};
```

اما، برای اینکه بتوانیم از میانبر $ در پلاگین استفاده نماییم و تداخلی با سایر کتابخانه‌ها نداشته باشد، از الگوی (IIFE Immediately Invoked Function Expression D) به صورت زیر استفاده می‌نماییم:

```
(function( $ ) {
  $.fn.myPlugin = function() {
  //محتویات پلاگین را اینجا می‌نویسیم
    };
})( jQuery );
```

محتوای پلاگین

حال می‌توانیم در تابع، کدهای پلاگین خود را بنویسیم. برای دسترسی به شیء پاس داده شده به پلاگین، از کلمه کلیدی this استفاده کرده و لازم نیست از $(this) استفاده نماییم. در زیر یک پلاگین ساده تهیه شده است که با رفتن ماوس بر روی یک متن، خطی زیر آن می‌کشد:

```
(function($){
    $.fn.underline= function() {
        this.hover(function(){
            $(this).css( { text-decoration : underline })
        }, function(){
            $(this).css( { text-decoration : none } )
        });
    };
})(jQuery);

$("p").underline();
```

پلاگین بالا مقدار یا شیء ایی را بر نمی‌گرداند؛ اما اگر بخواهیم مقداری را برگردانیم از return استفاده می‌نماییم:

```
(function( $ ){

  $.fn.maxHeight = function() {

    var max = 0;

    this.each(function() {
      max = Math.max( max, $(this).height() );
    });

    return max;
```

```
  };
})( jQuery );
```

```
var tallest = $('div').maxHeight(); // بیشترین ارتفاع عنصر را برمی گرداند
```

حفظ خاصیت زنجیره‌ای پلاگین ها

در مثال بالا یک مقدار عددی برگردانده شده است؛ اما برای اینکه بتوانیم بصورت زنجیر وار خروجی پلاگین را به تابع یا هر پالاگین دیگری پاس دهیم از تابع each بصورت زیر استفاده می‌نماییم:

```
(function( $ ){

  $.fn.lockDimensions = function( type ) {

    return this.each(function() {

      var $this = $(this);

      if ( !type || type == 'width' ) {
        $this.width( $this.width() );
      }

      if ( !type || type == 'height' ) {
        $this.height( $this.height() );
      }

    });

  };
})( jQuery );
```

```
$('div').lockDimensions('width').css('color', 'red');
```

در پلاگین بالا با از تابع each برای روی this و برگرداندن آن با return برای حفظ خاصیت زنجیره‌ای پلاگین استفاده می‌نماییم. در تابع each می‌بایست از $(this) برای انجام عملیات بر روی شیء پاس داده شده استفاده کنیم. بدین صورت بعد از صدا زدن پلاگین، دوباره می‌توانیم از هر پلاگین یا تابع جی کوئری دیگری بر روی خروجی استفاده نماییم.

پیش فرض‌ها و تنظیمات

در پلاگین‌های پیشرفته‌تر می‌توانیم تنظیمات پیش فرضی را برای پلاگین در نظر بگیریم و این تنظیمات را به عنوان پارامتر ورودی از کاربر دریافت نماییم. جی کوئری دارای تابعی به نام extend است که امکان گسترش و ترکیب دو شیء را امکان پذیر می‌سازد به مثال زیر توجه نمایید:

```
(function( $ ){

  $.fn.tooltip = function( options ) {

    var settings = $.extend( {
      'location'         : 'top',
      'background-color' : 'blue'
    }, options);

    return this.each(function() {

      // Tooltip plugin code here

    });

  };
})( jQuery );
```

```
$('div').tooltip({
  'location' : 'left'
});
```

در این مثال، شیء settings با دو خاصیت location و background-color تعریف شده که با شیء options که از ورودی پلاگین دریافت نموده‌ایم با استفاده از تابع extend ترکیب شده است. خاصیت‌های که تعیین نشده باشند با مقادیر پیش فرض آن‌ها تکمیل می‌گردد.
ادامه دارد...

نظرات خوانندگان

نویسنده: امیر
تاریخ: ۱۳۹۱/۱۲/۰۹ ۱۳:۵۴

مرسی عالی بود .استفاده کردم.فقط زمانی که سی اس اس میدی نباید اونطوری نوشته بشه

```
$(this).css( "text-decoration" ,"none"  )
```

نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۱/۱۲/۰۹ ۱۶:۲۳

ممنون. در جاواسکریپ هر دو صحیح است

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۲/۰۹ ۱۶:۵۳

```
$(this).css( { text-decoration : underline })
```

فکر کنم منظور ایشون («اونطوری» که نوشته) وجود dash در نام متغیر بوده.

نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۱/۱۲/۰۹ ۱۸:۲۸

منظورشون نحوه ست کردن css که هر دو روش استفاده می‌شود ولی در jquery 1.9 css به بعد کمی تغییر کرده است

عنوان: **JQuery Plugins #2**
نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۱/۱۲/۲۶ ۲۳:۵۵
آدرس: www.dotnettips.info
برچسب‌ها: jQuery, jQuery-Tips, Plugin

در قسمت اول آموزش jQuery Plugin #1 با نحوه ساخت اولیه پلاگین در جی کوئری آشنا شدید. در ادامه به موارد دیگری خواهم پرداخت.

فضای نام

در پلاگین شما، فضای نام، بخش مهمی از توسعه پلاگین می‌باشد. فضای نام در واقع تضمین می‌کند که پلاگین شما توسط دیگر پلاگین‌ها باز نویسی نشود یا با کدهای موجود در صفحه تداخل نداشته باشد. همچنین کمک می‌کند که توابع، رویدادها و داده‌های پلاگین خود را بهتر مدیر کنید.

توابع پلاگین

تحت هیچ شرایطی نباید یک پلاگین، در چندین فضای نام، به شی Jquery.fn اضافه گردند. به مثال زیر توجه نمایید:

```
(function( $ ){

  $.fn.tooltip = function( options ) {
    // این
  };
  $.fn.tooltipShow = function( ) {
    // تعریف
  };
  $.fn.tooltipHide = function( ) {
    // بد است
  };
  $.fn.tooltipUpdate = function( content ) {
    // !
  };

})( jQuery );
```

همین طور که در مثال بالا مشاهده می‌کنید، پلاگین به شکل بدی تعریف شده و هر تابع در یک فضای نام جداگانه تعریف گردیده‌است. برای این کار شما باید تمام توابع را در یک متغیر تعریف و آن را به پلاگین خود پاس دهید و توابع را با نام رشته ای صدا بزنید.

```
(function( $ ){

  var methods = {
    init : function( options ) {
      // این
    },
    show : function( ) {
      // تعریف
    },
    hide : function( ) {
      // خوب است
    },
    update : function( content ) {
      // !
    }
  };

  $.fn.tooltip = function( method ) {

    // منطق تابع را  از اینجا صدا زده ایم
    if ( methods[method] ) {
      return methods[ method ].apply( this, Array.prototype.slice.call( arguments, 1 ));
    } else if ( typeof method === 'object' || ! method ) {
      return methods.init.apply( this, arguments );
    } else {
      $.error( 'Method ' +  method + ' does not exist on jQuery.tooltip' );
    }
```

```
  };
})( jQuery );
```

توضیح : متغیر method اگر در متغیر methods توابع موجود باشد، تابع هم نام آن و در صورت داشتن پارامتر ورودی، به آن تابع پاس داده شده و برگردانده می‌شود (در واقع صدا زده می‌شود). در غیر اینصورت اگر نوع مقدار ورودی، object بود تابع init آن صدا زده می‌شود وگرنه پیام خطا ارسال می‌گردد.
برای استفاده از پلاگین بصورت زیر عمل می‌کنیم:

```
// تابع init صدا زده می‌شود
$('div').tooltip();
```

و

```
// تابع update با پارامتر صدا زده می‌شود
$('div').tooltip('update', 'This is the new tooltip content!');
```

این معماری به شما امکان کپسوله کردن توابع در پلاگین را می‌دهد.

رویداد ها

یکی از روش‌های کمتر شناخته شده انقیاد توابع در فضای نام، امکان انقیاد رویدادها است. اگر پلاگین شما یک رویداد را انقیاد نماید، این یک عمل و تمرین خوب استفاده از فضای نام می‌باشد. بدین ترتیب اگر لازم باشد که انقیاد یک رویدا را حذف نمایید، بدون تداخل با دیگر رویدادها و بدون اینکه در یک شی دیگر از این پلاگین، اختلالی ایجاد نماید می‌توان آن را حذف نمود. به مثال زیر توجه نمایید.

```
(function( $ ){

  var methods = {
     init : function( options ) {

       return this.each(function(){
         $(window).bind('resize.tooltip', methods.reposition);
       });

     },
     destroy : function( ) {

       return this.each(function(){
         $(window).unbind('.tooltip');
       })

     },
     reposition : function( ) {
       // ...
     },
     show : function( ) {
       // ...
     },
     hide : function( ) {
       // ...
     },
     update : function( content ) {
       // ...
     }
  };

  $.fn.tooltip = function( method ) {

    if ( methods[method] ) {
      return methods[method].apply( this, Array.prototype.slice.call( arguments, 1 ));
    } else if ( typeof method === 'object' || ! method ) {
      return methods.init.apply( this, arguments );
    } else {
```

```
      $.error( 'Method ' +  method + ' does not exist on jQuery.tooltip' );
    }

  };

})( jQuery );
```

این همان مثال قبل است که وقتی پلاگین با تابع Init مقدار دهی اولیه می‌شود، تابع reposition به رویداد resize پنجره در فضای نام پلاگین tooltip انقیاد می‌شود. پس از آن اگر توسعه دهنده نیاز داشت تا tooltip را از بین ببرد، با صدا زدن تابع destroy می تواند بصورت امن انقیاد ایجاد شده را حذف نماید.

```
$('#fun').tooltip();
// مدتی بعد...
$('#fun').tooltip('destroy');
```

ادامه دارد...

عنوان: **آشنایی با پلاگین TickTack برای Mask ورودی کاربر**
نویسنده: محسن موسوی
تاریخ: ۱۳۹۲/۰۲/۱۴ ۱۷:۵
آدرس: www.dotnettips.info
برچسب‌ها: jQuery, Plugin, TimePicker

همانطور که می‌دانیم پلاگین‌های جی‌کوئری، نقش مهمی را در محیط وب ایفا می‌کنند. در اینجا با یکی از این پلاگین‌ها و چگونگی استفاده از آن آشنا میشویم.

برای آشنایی با نوشتن Plugin در jQuery ، می‌توان مباحث پیشین این سایت را دنبال کرد.(JQuery Plugins #1 و JQuery Plugins #2)

jQueryTickTack Plugin :

این Plugin برای ایجاد یک TextBox برای ورود زمان توسط کاربر استفاده می‌شود. با توجه به اینکه قبلاً چند Plugin برای این کار نوشته شده است ولی هر کدام از آنها معایب و مزایای خاص خود را داشتند، برای نمونه می‌توانید به این سایت مراجعه کنید.

ویژگی‌های این Plugin عبارتند از:

1- تنظیم زمان پیش فرض

2- کنترل حداقل و حداکثر زمان وارد شده

3- تغییر ساعت و دقیقه بوسیله کلیدهای جهتی بالا و پایین

4- تغییر انتخاب ساعت و دقیقه بوسیله کلیدهای جهتی چپ و راست

5- تغییر ساعت و دقیقه بوسیله فشردن اعداد روی صفحه کلید

چگونگی استفاده از این Plugin

ابتدا کتابخانه jQuery و این پلاگین را به صفحه خود اضافه نمایید و سپس کدهای زیر را برای استفاده از این Plugin اضافه نمایید:

```
jQuery(document).ready(function () {
            $("#TextBox1").TickTack();
            $("#TextBox2").TickTack({
                 initialTime: '8:44',
                minHour: 8,
                minMinute: 0,
                maxHour: 22,
                maxMinute: 40
            });
        });
```

در ادامه به بررسی تنظیمات انجام شده در این پلاگین می‌پردازیم:
initialTime : زمان اولیه جهت نمایش به کاربر (حتما بایستی ساعت و دقیقه بوسیله ' : ' از یکدیگر جداشوند)

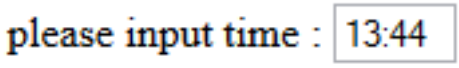
minHour : حداقل ساعت ورودی

minMinute : حداقل دقیقه ورودی

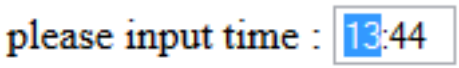
maxHour : حداکثر ساعت ورودی

maxMinute : حداکثر دقیقه ورودی

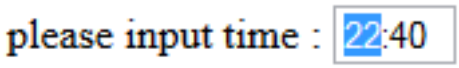
پس از انجام این تنظیمات و اجرا کردن برنامه،TextBox شما به صورت زیر نمایش داده می‌شود:



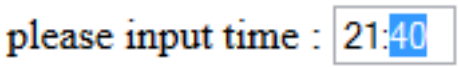
پس از انتخاب TextBox ، قسمت ساعت به صورت پیش فرض انتخاب می‌شود و کاربر باید ساعت مد نظر را وارد کند؛ در اینجا، عدد اول ساعت، مد نظر است.



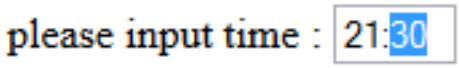
برای نمونه در اینجا عدد 2 توسط کاربر وارد می‌شود؛ پس از ورود عدد و با توجه به تنظیمات انجام شده، ساعت به صورت اتوماتیک به حداکثر مقداری که می‌تواند بپذیرد تغییر می‌کند (در این مثال چون کاربر عدد 2 را وارد کرده و در تنظیمات انجام شده حداکثر ساعت دریافتی 22 و حداکثر دقیقه 40 تعریف شده است، ساعت به صورت پیشفرض به 22:40 تغییر می‌یابد)



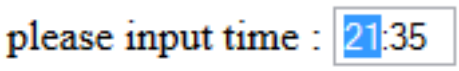
و پس از وارد کردن عدد دوم ساعت توسط کاربر مکان نما به قسمت دقیقه منتقل می‌شود که در این جا عدد اول دقیقه مد نظر است



وارد کردن عدد 3 برای دقیقه



وارد کردن عدد دوم دقیقه



پس از وارد کردن کامل دقیقه مکان نما دوباره به قسمت ساعت باز می‌گردد.

در ادامه دوستان علاقمند لطفا جهت بهبود کیفیت کار، باگ و یا مشکلات کدنویسی را اطلاع دهند.

با تشکر

نظرات خوانندگان

نویسنده: بهروز
تاریخ: ۱۳۹۲/۰۶/۲۵ ۱۶:۱۸

با سلام خدمت جناب آقای محسن موسوی

یه مشکلی داشتم تو این پلاگین که وقتی از تو کد به تکست باکس مقداری میدیم اون مقدارو نمیگیره و فقط مقدار پیشفرض خودشو نشون میده... میخواستم در صورت امکان راهنمایی بفرمایید.

با تشکر فراوان

و من الله توفیق...

نویسنده: محسن موسوی
تاریخ: ۱۳۹۲/۰۶/۲۵ ۱۷:۵۴

سلام
از طریق سرور پلاگین را صدا بزنید یا اینکه از طریق سرور یک مقدار به متغییر جاوااسکریپتی بدهید.

نویسنده: سعید حر
تاریخ: ۱۳۹۲/۰۷/۰۶ ۱۱:۳۷

سلام
در آدرس https://ticktack.codeplex.com/ هیچ فایلی برای دانلود وجود ندارد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۰۶ ۱۱:۴۵

به برگه سورس پروژه مراجعه کنید.

نویسنده: محسن موسوی
تاریخ: ۱۳۹۲/۰۷/۰۶ ۱۲:۲۹

با تشکر از آقای نصیری
هنوز بازخورد خاصی از پروژه نگرفتم.در هفته‌ی آتی احتمالا release پروژه را ارائه میدهم.

عنوان: **نوشتن افزونه برای مرورگرها: قسمت اول : کروم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۱/۰۳ ۱:۴۰
آدرس: www.dotnettips.info
گروه‌ها: Plugin, Google Chrome, Extension, Addon

افزونه چیست؟

افزونه‌ها جزء مهمترین قسمت‌های یک مرورگر توسعه پذیر به شمار می‌آیند. افزونه‌ها سعی دارند تا قابلیت هایی را به مرورگر شما اضافه کنند. افزونه‌ها از آخرین فناوری‌های html,CSS و جاوااسکریپت تا به آنجایی که مرورگر آن‌ها را پشتیبانی کند، استفاده می‌کنند.

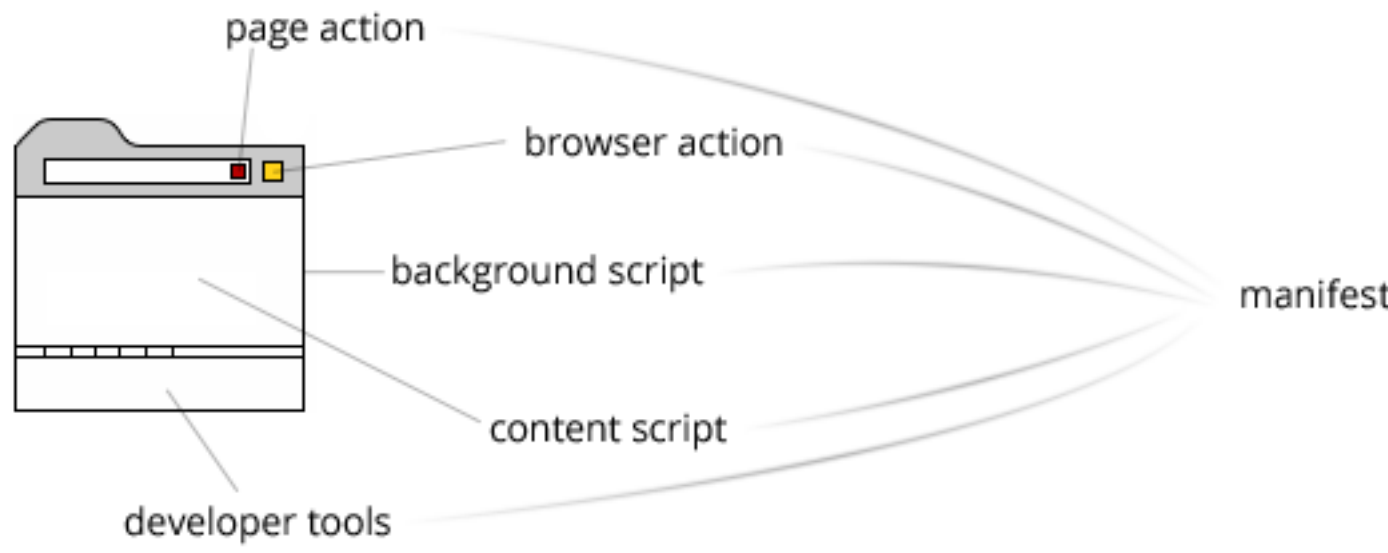
در این سری سعی خواهیم کرد برای هر مرورگر شناخته شده، یک افزونه ایجاد کنیم و ابتدا از آنجا که خودم از کروم استفاده می‌کنم، اولین افزونه را برای کروم خواهم نوشت.

این افزونه قرار است چه کاری انجام دهد؟

کاری که برای این افزونه تدارک دیده‌ام این است: موقعی‌که سایت dotnettips.info به روز شد مرا آگاه کند. این آگاه سازی را از طریق یک نوتیفیکیشن به اطلاع کاربر میرسانیم. صفحه تنظیمات این افزونه شامل گزینه‌های "آخرین مطالب"،"نظرات آخرین مطالب"،"آخرین اشتراک ها"و"آخرین نظرات اشتراک ها" خواهد بود که به طور پیش فرض تنها گزینه اول فعال خواهد بود و همچنین یک گزینه نیز برای وارد کردن یک عدد صحیح جهت اینکه به افزونه بگوییم هر چند دقیقه یکبار سایت را چک کن. چک کردن سایت هم از طریق فید RSS صورت می‌گیرد.

فایل manifest.json

این فایل برای ذخیره سازی اطلاعاتی در مورد افزونه به کار می‌رود که شامل نام افزونه، توضیح کوتاه در مورد افزونه و ورژن و ... به کار می‌رود که همه این اطلاعات در قالب یا فرمت json نوشته می‌شوند و در بالاترین حد استفاده برای تعریف اهداف افزونه و اعطای مجوز به افزونه از آن استفاده می‌کنیم. این فایل بخش‌های زیر را در یک افزونه تعریف می‌کند که به مرور با آن آشنا می‌شویم.



کد زیر را در فایل manifest.json می‌نویسیم:

```
{
  "manifest_version": 2,
```

```
  "name": "Dotnettips Updater",
  "description": "This extension keeps you updated on current activities on dotnettips.info",
  "version": "1.0",
  "icons": { "16": "icon.png",
             "48": "icon.png",
            "128": "icon.png" },

  "browser_action": {
    "default_icon": "icon.png",
    "default_popup": "popup.html"
  },
  "permissions": [
    "activeTab",
    "http://www.dotnettips.info"
  ]
}
```

اطلاعات اولیه شامل نام و توضیح و ورژن افزونه است. ورژن برنامه برای به روزآوری افزونه بسیار مهم است. موقعی که ورژن جدیدی از افزونه ارائه شود، گوگل وب استور اعلان آپدیت جدیدی را برای افزونه میکند. آیکن قسمت‌های مختلف افزونه هم با icons مشخص می‌شود که در سه اندازه باید ارائه شوند و البته اگه اندازه آن نباشد scale می‌شود. قسمت بعدی تعریف UI برنامه هست که گوگل کروم، به آن Browser Action می‌گوید. در اینجا یک آیکن و همچنین یک صفحه اختصاصی برای تنظیمات افزونه معرفی می‌کنیم. این آیکن کنار نوار آدرس نمایش داده می‌شود و صفحه popup موقعی نشان داده می‌شود که کاربر روی آن کلیک می‌کند. آیکن‌ها برای browser action در دو اندازه 19 و 38 پیکسلی هستند و در صورتی که تنها یک آیکن تعریف شود، به صورت خودکار عمل scale و تغییر اندازه صورت می‌گیرد. برای تعیین عکس برای هر اندازه می‌توانید کد را به صورت زیر بنویسید:

```
"default_icon": {                    // optional
            "19": "images/icon19.png",           // optional
            "38": "images/icon38.png"            // optional
          }
```

قسمت popup برای نمایش تنظیمات به کار می‌رود و درست کردن این صفحه همانند صفحه همیشگی html هست و خروجی آن روی پنجره popup افزونه رندر خواهد شد.

گزینه default_title نیز یکی از دیگر خصیصه‌های مهم و پرکاربرد این قسمت هست که متن tooltip می‌باشد و موقعی که که کاربر، اشاره‌گر را روی آیکن ببرد نمایش داده می‌شود و در صورتی که نوشته نشود، کروم نام افزونه را نمایش می‌دهد؛ برای همین ما هم چیزی ننوشتیم.

صفحات پس‌زمینه

اگر بخواهید برای صفحه popup کد جاوااسکریت بنویسید یا از jquery استفاده کنید، مانند هر صفحه‌ی وبی که درست می‌کنید آن را کنار فایل popup قرار داده و در popup آنها را صدا کرده و از آن‌ها استفاده کنید. ولی برای پردازش هایی که نیاز به UI وجود ندارد، می‌توان از صفحات پس زمینه استفاده کرد. در این حالت ما دو نوع صفحه داریم:

صفحات مصر یا Persistent Page

صفحات رویدادگرا یا Events Pages

اولین نوع صفحه، همواره فعال و در حال اجراست و دومی موقعی فعال می‌شود که به استفاده از آن نیاز است. گوگل توصیه می‌کند که تا جای ممکن از نوع دوم استفاده شود تا مقدار حافظه مصرفی حفظ شود و کارآیی مروگر بهبود بخشیده شود. کد زیر یک صفحه پس زمینه را از نوع رویدادگرا می‌سازد. به وضوح روشن است در صورتی که خاصیت Persistent با true مقداردهی شود، این صفحه مصرانه در تمام وقت باز بودن مرورگر، فعال خواهد بود:

```
"background": {
    "scripts": ["background.js"],
    "persistent": false
}
```

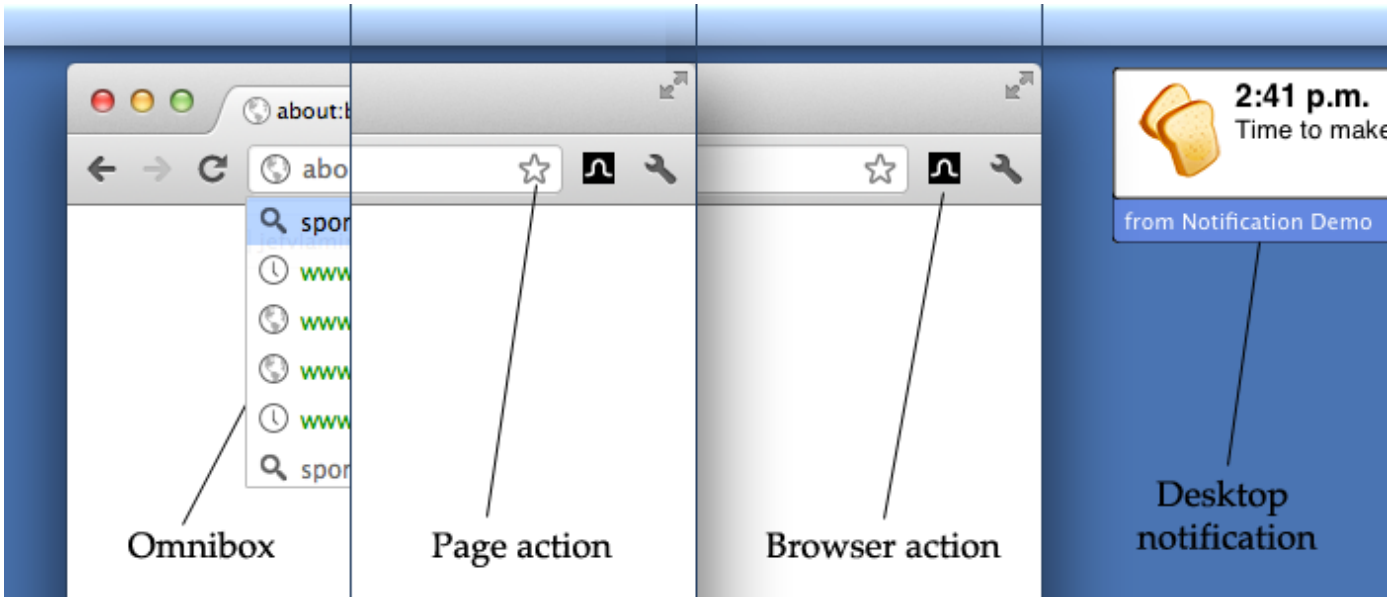
Content Script یا اسکریپت محتوا

در صورتی که بخواهید با هر صفحه‌ای که باز یا رفرش می‌شود، به DOM آن دسترسی پیدا کنید، از این خصوصیت استفاده کنید. در

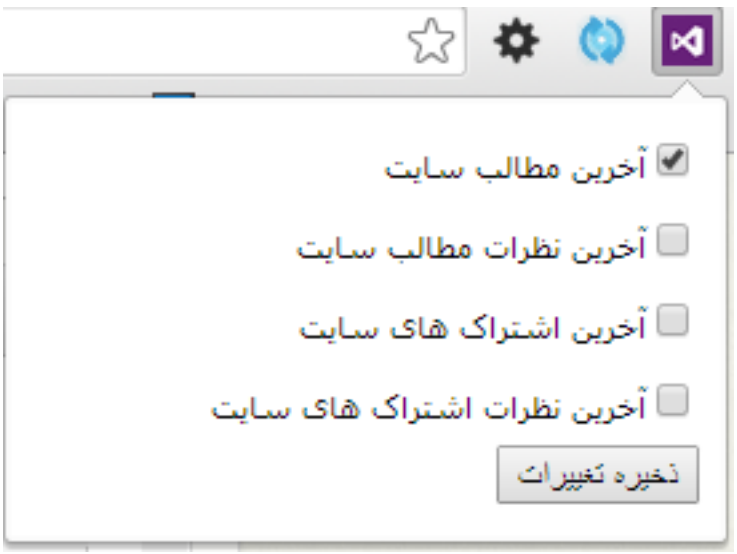
کد زیر برای پردازش اطلاعات DOM از فایل جاوااسکریپت بهره برده و در قسمت matches می‌گویید که چه صفحاتی باید از این کد استفاده کنند که در اینجا از پروتکل‌های HTTP استفاده میشود و اگر مثلا نوع FTP یا file صدا زده شود کد مورد نظر اجرا نخواهد شد. در مورد اینکه matches چگونه کار می‌کند و چگونه می‌توان آن را نوشت، از این صفحه استفاده کنید.

```
"content_scripts": [
    {
        "matches": ["http://*/*", "https://*/*"],
        "js": ["content.js"]
    }
]
```

آغاز کدنویسی (رابطهای کاربری)



اجازه دهید بقیه موارد را در حین کدنویسی تجربه کنیم و هر آنچه ماند را بعدا توضیح خواهیم داد. در اینجا من از یک صفحه با کد HTML زیر بهره برده ام که یک فرم دارد به همراه چهار چک باکس و در نهایت یک دکمه جهت ذخیره مقادیر. نام صفحه را popup.htm گذاشته ام و یک فایل popup.js هم دارم که در آن کد jquery نوشتم. قصد من این است که بتوان یک action browser به شکل زیر درست کنم:



کد html آن به شرح زیر است:

```
<html>
<head>
<meta charset="utf-8"/>

<script src="jquery.min.js"></script> <!-- Including jQuery -->
<script type="text/javascript" src="popup.js"></script>
</head>
<body style="direction:rtl;width:250px;">
<form >
<input type="checkbox" id="chkarticles" value="" checked="true">آخرین مطالب سایت</input><br/>
<input type="checkbox" id="chkarticlescomments" value="" >آخرین نظرات مطالب سایت</input><br/>
<input type="checkbox" id="chkshares" value="" >آخرین اشتراک‌های سایت</input><br/>
<input type="checkbox"  id="chksharescomments" value="" >آخرین نظرات اشتراک‌های سایت</input><br/>
<input id="btnsave" type="button" value="ذخیره تغییرات" />
    <div id="messageboard" style="color:green;"></div>
</form>

</body>
</html>
```

کد popup.js هم به شرح زیر است:

```
$(document).ready(function () {
    $("#btnsave").click(function() {
        var articles = $("#chkarticles").is(':checked');
        var articlesComments = $("#chkarticlescomments").is(':checked');
        var shares = $("#chkshares").is(':checked');
        var sharesComments = $("#chksharescomments").is(':checked');

        chrome.storage.local.set({ 'articles': articles, 'articlesComments': articlesComments,
'shares': shares, 'sharesComments': sharesComments }, function() {
            $("#messageboard").text( 'تنظیمات جدید اعمال شد');
        });
    });
});
```

در کد بالا موقعی که کاربر بر روی دکمه ذخیره، کلیک کند رویداد کلیک jquery فعال شده و مقادیر چک باکس‌ها را در متغیرهای مربوطه نگهداری می‌کند. نهایتا با استفاده از کلمه کلیدی کروم به ناحیه ذخیره سازی داده‌های کروم دست پیدا کرده و درخواست ذخیره مقادیر چک باکس را بر اساس ساختار نام و مقدار، ذخیره می‌کنیم و بعد از اعمال، توسط یک تابع callback به کاربر اعلام می‌کنیم که اطلاعات ذخیره شده است.
اولین مورد جدیدی که در بالا دیدیم، کلمه‌ی کلیدی chrome است. کروم برای توسعه دهندگانی که قصد نوشتن افزونه دارند api هایی را تدارک دیده است که میتوانید با استفاده از آنها به قسمت‌های مختلف مرورگر مثل بوک مارک یا تاریخچه فعالیت‌های مرورگر و ... دست پیدا کنید. البته برای اینکار باید در فایل manifest.json هم مجوز اینکار را درخواست نماییم. این ویژگی باید برای برنامه نویسان اندروید آشنا باشد. برای آشنایی هر چه بیشتر با مجوزها این صفحه را ببینید.
برای دریافت مجوز، کد زیر را به manifest اضافه می‌کنیم:

```
"permissions": [
    "storage"
  ]
```

مجوزی که در بالا درخواست کرده‌ایم مجوز دسترسی به ناحیه ذخیره سازی است. بعد از کلمه کلیدی chrome، کلمه‌ی local آمده است و می‌گوید که باید داده‌ها به صورت محلی و لوکال ذخیره شوند ولی اگر میخواهید داده‌ها در گوگل سینک شوند، باید به جای لوکال از کلمه کلیدی sync استفاده کنید یعنی:

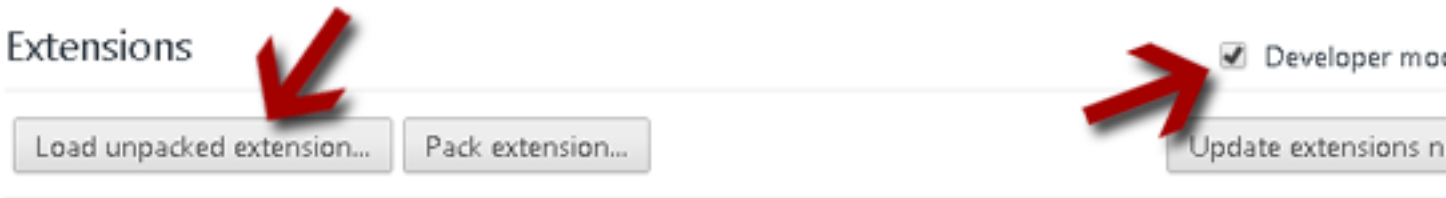
```
chrome.storage.sync.set
```

فایل manifest نهایی:

```
{
  "manifest_version": 2,

  "name": "Dotnettips Updater",
  "description": "This extension keeps you updated on current activities on dotnettips.info",
  "version": "1.0",

  "browser_action": {
    "default_icon": "icon.png",
    "default_popup": "popup.html"
  },
  "permissions": [
    "storage"
  ]
}
```

الان باید 4 فایل داشته باشید: فایل آیکن، popup.htm,popup.js و manifest.json. همه را داخل یک دایرکتوری قرار داده و در مروگر کروم به قسمت extensions بروید و گزینه Developer mode را فعال کنید تا یک تستی از کد نوشته شده بگیریم. گزینه Load Unpacked Extension را بزنید و آدرس دایرکتوری ایجاد شده را به آن بدهید.

```
chrome://extensions
```



الان باید مانند تصویر بالا یک آیکن کنار نوار آدرس یا به قول گوگل، Omni box ببینید. گزینه‌ها را تیک بزنید و روی دکمه ذخیره کلیک کنید. باید پیام مقادیر ذخیره شدند، نمایش پیدا کند. الان یک مشکل وجود دارد؛ داده‌ها ذخیره می‌شوند ولی موقعی که دوباره تنظیمات افزونه را باز کنید حالت اولیه نمایش داده میشود. پس باید تنظیمات ذخیره شده را خوانده و به آن‌ها اعمال کنیم. کد زیر را جهت دریافت مقادیر ذخیره شده می‌نویسیم. اینبار به جای استفاده از متد set از متد get استفاده می‌کنیم. به صورت آرایه، رشته نام مقادیر را درخواست می‌کنیم و در تابع callback، مقادیر به صورت آرایه برای ما برگشت داده می‌شوند.

```
chrome.storage.local.get(['articles', 'articlesComments', 'shares', 'sharesComments'], function (
items) {
        console.log(items[0]);
        $("#chkarticles").attr("checked", items["articles"]);
        $("#chkarticlescomments").attr("checked", items["articlesComments"]);
        $("#chkshares").attr("checked", items["shares"]);
        $("#chksharescomments").attr("checked", items["sharesComments"]);

    });
```

حالا برای اینکه افزونه‌ی شما متوجه تغییرات شود، به تب extensions رفته و در لیست افزونه‌ها به دنبال افزونه خود بگردید و گزینه Reload را انتخاب نمایید تا افزونه تغییرات را متوجه شود و صفحه را تست کنید.

Page Action

روش دیگر برای ارائه یک رابط کاربری، page action هست. این روش دقیقا مانند روش قبلی است، ولی جای آیکن عوض می‌شود. قبلا بیرون از نوار آدرس بود، ولی الان داخل نوار آدرس قرار می‌گیرد. جالب‌ترین نکته در این مورد این است که این آیکن در ابتدا مخفی شده است و شما تصمیم می‌گیرید که این آیکن چه موقع نمایش داده شود. مثلا آیکن RSS تنها موقعی نمایش داده

می‌شود که وب سایتی که باز شده است، دارای محتوای RSS باشد یا بوک مارک کردن یک آدرس برای همه‌ی سایت‌ها باز باشد و سایر موارد.

کد زیر نحوه‌ی تعریف یک page action را در manifest نشان می‌دهد. ما در این مثال یک page action را به طور موقت اضافه می‌کنیم و موقعی هم آن را نشان میدهیم که سایت dotnettips.info باز باشد. دلیل اینکه موقت اضافه می‌کنیم این است که باید یکی از دو گزینه رابط کاربری که تا به حال گفتیم، استفاده شود. در غیر این صورت کروم در هنگام خواندن فایل manifest در هنگام افزودن افزونه به مرورگر، پیام خطا خواهد داد و این مطلب را به شما گوشزد می‌کند. پس نمی‌توان دو گزینه را همزمان داشت و من میخواهم افزونه را در حالت browser action ارائه کنم. پس در پروژه نهایی، این مطلب page action نخواهد بود.

برای داشتن یک page action کد زیر را در manifest بنویسید.

```
"page_action": {
    "default_icon": {
        "19": "images/icon19.png",
        "38": "images/icon38.png"
    },
    "default_popup": "popup.html"
```

گزینه page action تعریف شد حالا باید کاری کنیم تا هر موقع صفحه‌ای باز می‌شود چک کند آیا سایت مورد نظر است یا خیر، اینکار را توسط صفحه‌ی پردازشی انجام می‌دهیم. پس تکه کد زیر را هم به manifest اضافه می‌کنیم:

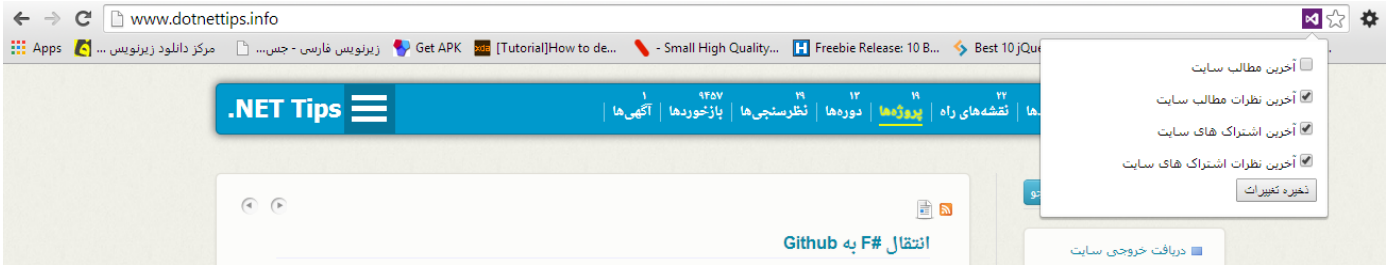
```
"background": {
    "scripts": ["page_action_validator.js"]
}
```

تا اینجا فایل جاوااسکریپت معرفی شد که کد زیر را دارد و در پس زمینه شروع به اجرا می‌کند.

```
function UrlValidation(tabId, changeInfo, tab) {
if (tab.url.indexOf('dotnettips.info') >-1) {
chrome.pageAction.show(tabId);
}
};
chrome.tabs.onUpdated.addListener(UrlValidation);
```

چون از api در این کد بهره برده‌ایم و آن هم مدیریت بر روی تب هاست، پس باید مجوز آن هم گرفته شود. کلمه "tabs" را در قسمت permissions اضافه کنید.

یک listener برای tabها ایجاد کرده‌ایم که اگر تب جدید ایجاد شد، یا تب قبلی به آدرس جدیدی تغییر پیدا کرد تابع UrlValidation را اجرا کند و در این تابع چک می‌کنیم که اگر url این تب شامل نام وب سایت می‌شود، page action روی این تب ظاهر شود. پس از انجام تغییرات، مجددا افزونه را بارگذاری می‌کنیم و تغییرات اعمال شده را می‌بینیم. سایت dotnettips را باز کنید یا صفحه را مجددا رفرش کنید تا تغییر اعمال شده را ببینید.



تغییرات موقت را حذف و کدها را به حالت قبلی یعنی browser action بر میگردانم.

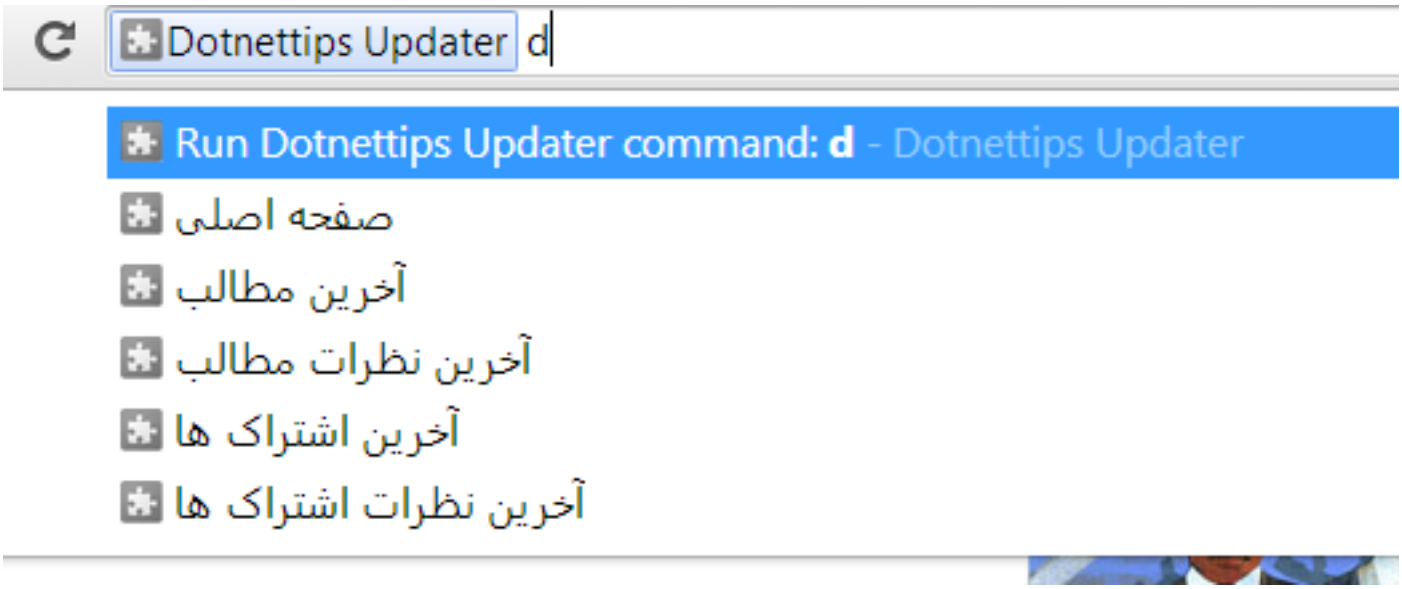
OmniBox

omnibox یک کلمه کلیدی است که در نوار آدرس مرورگر وارد می‌شود و در واقع می‌توانیم آن را نوع دیگری از رابط کاربری بنامیم. موقعی که شما کلمه کلیدی رزرو شده را وارد می‌کنید، در نوار آدرس کلماتی نشان داده میشود که کاربر میتواند یکی از آن‌ها را انتخاب کند تا عملی انجام شود. ما هم قرار است این کار را انجام دهیم. به این مثال دقت کنید:
میخواهیم موقعی که کاربرکلمه net. را تایپ می‌کند، 5 عبارت آخرین مطالب و آخرین اشتراک‌ها و آخرین نظرات مطالب و آخرین نظرات اشتراک‌ها و صفحه اصلی سایت نمایش داده شود و با انتخاب هر کدام، کاربر به سمت آن صفحه هدایت شود.
برای افزودن کلمه کلیدی در manifest خطوط زیر را اضافه کنید:

```
"omnibox": { "keyword" : ".net" }
```

با نوشتن خط بالا کلمه net. در مرورگر یک کلمه‌ی کلیدی به حساب خواهد آمد و موقعی که کاربر این کلمه را وارد کند، در سمت راست نوشته خواهد شد. در این حالت باید کلید تب را بزند تا به محیط دستوری آن برود.

در این حین می‌توانیم همزمان با تایپ کاربر، دستوراتی را به آن نشان بدهیم. من دوست دارم موقعی که کاربر حرفی را وارد کرد، لیستی از نام صفحات نوشته شود.



برای اینکار باید کدنویسی کنیم ، پس یک فایل پس زمینه را به manifest معرفی کنید:

```
"background": {
    "scripts": ["omnibox.js"]
```

در فایل ominbox.js دستوراتی که مرتبط با omnibox است را می‌نویسیم و کد زیر را به آن اضافه می‌کنیم:

```
chrome.omnibox.onInputChanged.addListener(function(text, suggest) {
    suggest([
  {content: ".net tips Home Page", description: "صفحه اصلی"},
      {content: ".net tips Posts", description: "آخرین مطالب"},
      {content: ".net tips News", description: "آخرین نظرات مطالب"},
      {content: ".net tips Post Comments", description: "آخرین اشتراک ها"},
      {content: ".net tips News Comments", description: "آخرین نظرات اشتراک ها"}
    ]);
});
```

chrome.omnibox شامل 4 رویداد می‌شود:

onInputStarted	بعد از اینکه کاربر کلمه کلیدی را وارد کرد اجرا می‌شود
onInputChanged	بعد از وارد کردن کلمه کلیدی هربار که کاربر تغییری در ورودی نوارد آدرس می‌دهد اجرا می‌شود.
onInputEntered	کاربر ورودی خود را تایید می‌کند. مثلا بعد از وارد کردن، کلید enter را می‌فشارد
onInputCancelled	کاربر از وارد کردن ورودی منصرف شده است؛ مثلا کلید ESC را فشرده است.

با نوشتن chrome.omnibox.onInputChanged.addListener ما یک listener ساخته‌ایم تا هر بار کاربر ورودی را تغییر داد، یک تابع callback که دو آرگومان را دارد، صدا بزند. این آرگومان‌ها یکی متن ورودی‌است و دیگری آرایه‌ی suggest که شما با تغییر آرایه می‌توانید عباراتی که همزمان با تایپ به کاربر پیشنهاد می‌شود را نشان دهید. البته می‌توانید با تغییر کد کاری کنید تا بر اساس حروفی که تا به حال تایپ کرده‌اید، دستورات را نشان دهد؛ ولی من به دلیل اینکه 5 دستور بیشتر نبود و کاربر راحت باشد، چنین کاری نکردم. همچنین وقتی شما برای هر یک description تعریف کنید، به جای نام پیشنهادی، توضیح آن را نمایش می‌دهد.
حالا وقت این است که کد زیر را جهت اینکه اگر کاربر یکی از کلمات پیشنهادی را انتخاب کرد، به صفحه‌ی مورد نظر هدایت شود، اضافه کنیم:

```
chrome.omnibox.onInputEntered.addListener(function (text) {

var location="";
    switch(text)
{
case ".net tips Posts":
location="http://www.dotnettips.info/postsarchive";
break;
case ".net tips News":
location="http://www.dotnettips.info/newsarchive";
break;
case ".net tips Post Comments":
location="http://www.dotnettips.info/commentsarchive";
break;
case".net tips News Comments":
location="http://www.dotnettips.info/newsarchive/comments";
break;
default:
location="http://www.dotnettips.info/";
}

    chrome.tabs.getSelected(null, function (tab) {
        chrome.tabs.update(tab.id, { url: location });
    });
});
```

ابتدا یک listener برای روی رویداد onInputEntered قرار داده تا وقتی کاربر عبارت وارد شده را تایید کرد، اجرا شود. در مرحله بعد چک می‌کنیم که عبارت وارد شده چیست و به ازای هر عبارت مشخص شده، آدرس آن صفحه را در متغیر location قرار می‌دهیم. در نهایت با استفاده از عبارت chrome.tabs.getSelected تب انتخابی را به یک تابع callback بر میگردانیم. اولین آرگومان windowId است، برای زمانی که چند پنجره کروم باز است که می‌توانید وارد نکنید تا پنجره فعلی و تب فعلی محسوب شود. برای همین نال رد کردیم. در تابع برگشتی، شیء tab شامل اطلاعات کاملی از آن تب مانند url و id و title می‌باشد و در نهایت با استفاده از دستور chrome.tabs.update اطلاعات تب را به روز می‌کنیم. آرگومان اول id تب را میدهیم تا بداند کدام تب باید تغییر کند و آرگومان بعدی می‌توانید هر یک از ویژگی‌های تب از قبیل آدرس فعلی یا عنوان آن و ... را تغییر دهید که ما آدرس آن را تغییر داده ایم.

Context Menu

یکی دیگر از رابطهای کاربری، منوی کانتکست هست که توسط chrome.contextmenus ارائه می‌شود و به مجوز "contextmenus" نیاز دارد. فعال سازی منوی کانتکست در قسمت‌های زیر ممکن است:

```
all, page, frame, selection, link, editable, image, video, audio
```

من گزینه‌ی dotenettips.info را برای باز کردن سایت، به Contextmenus اضافه می‌کنم. کد را در فایلی به اسم contextmenus.js ایجاد می‌کنم و در قسمت background آن را معرفی می‌کنم. برای باز کردن یک تب جدید برای سایت، نیاز به chrome.tabs داریم که البته نیاز به مجوز tabs هم داریم.

محتوای فایل contextmenus.js

```
var root = chrome.contextMenus.create({
    title: 'Open .net tips',
    contexts: ['page']
}, function () {
    var Home= chrome.contextMenus.create({
        title: 'Home',
        contexts: ['page'],
        parentId: root,
        onclick: function (evt) {
            chrome.tabs.create({ url: 'http://www.dotnettips.info' })
        }
    });
var Posts = chrome.contextMenus.create({
        title: 'Posts',
        contexts: ['page'],
        parentId: root,
        onclick: function (evt) {
            chrome.tabs.create({ url: 'http://www.dotnettips.info/postsarchive/' })
        }
    });
});
```

در کد بالا یک گزینه به context menu اضافه میشود و دو زیر منو هم دارد که یکی صفحه‌ی اصلی سایت را باز میکند و دیگری هم صفحه‌ی مطالب سایت را باز می‌کند.

تا به اینجا ما قسمت ظاهری کار را آماده کرده ایم و به دلیل اینکه مطلب طولانی نشود، این مطلب را در دو قسمت ارائه خواهیم کرد. در قسمت بعدی نحوه خواندن RSS و اطلاع رسانی و دیگر موارد را بررسی خواهیم کرد.

عنوان: **نوشتن افزونه برای مرورگرها: قسمت دوم : کروم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۱/۰۴ ۱:۱۵
آدرس: www.dotnettips.info
گروه‌ها: RSS, Plugin, Google Chrome, Extension, Addon

در مقاله پیشین ما ظاهر افزونه را طراحی و یک سری از قابلیت‌های افزونه را معرفی کردیم. در این قسمت قصد داریم پردازش پس زمینه افزونه یعنی خواندن RSS و اعلام به روز آوری سایت را مورد بررسی قرار دهیم و یک سری قابلیت هایی که گوگل در اختیار ما قرار داده است.

خواندن RSS توسط APIهای گوگل

گوگل در تعدادی از زمینه‌ها و سرویس‌های خودش apiهایی را ارائه کرده است که یکی از آن ها خواندن فید است و ما از آن برای خواندن RSS یا اتم وب سایت کمک می‌گیریم. روند کار بدین صورت است که ابتدا ما بررسی می‌کنیم کاربر چه مقادیری را ثبت کرده است و افزونه قرار است چه بخش هایی از وب سایت را بررسی نماید. در این حین، صفحه پس زمینه شروع به کار کرده و در هر سیکل زمانی مشخص شده بررسی می‌کند که آخرین بار چه زمانی RSS به روز شده است. اگر از تاریخ قبلی بزرگتر باشد، پس سایت به روز شده است و تاریخ جدید را برای دفعات آینده جایگزین تاریخ قبلی کرده و یک پیام را به صورت نوتیفیکیشن جهت اعلام به روز رسانی جدید در آن بخش به کاربر نشان می‌دهد.
اجازه دهید کدها را کمی شکیل‌تر کنیم. من از فایل زیر که یک فایل جاوااسکریپتی است برای نگه داشتن مقادیر بهره می‌برم تا اگر روزی خواستم یکی از آن‌ها را تغییر دهم راحت باشم و در همه جا نیاز به تغییر مجدد نداشته نباشم. نام فایل را (const.js) به خاطر ثابت بودن آن‌ها انتخاب کرده‌ام.

```
برای ذخیره مقادیر از ساختار نام و مقدار استفاده می‌کنیم که نام‌ها را اینجا ثبت کرده ام//
var Variables={
 posts:"posts",
 postsComments:"postsComments",
 shares:"shares",
 sharesComments:"sharesComments",
}

برای ذخیره زمان آخرین تغییر سایت برای هر یک از مطالب به صورت جداگانه نیاز به یک ساختار نام و مقدار//
است که نام‌ها را در اینجا ذخیره کرده ام
var DateContainer={
 posts:"dtposts",
 postsComments:"dtpostsComments",
 shares:"dtshares",
 sharesComments:"dtsharesComments",
 interval:"interval"
}

//برای نمایش پیام‌ها به کاربر
var Messages={
SettingsSaved:"تنظیمات ذخیره شد",
SiteUpdated:"سایت به روز شد",
PostsUpdated:"مطلب ارسالی جدید به سایت اضافه شد",
CommentsUpdated:"نظری جدیدی در مورد مطالب سایت آرسال شد",
SharesUpdated:"اشتراک جدید به سایت ارسال شد",
SharesCommentsUpdated:"نظری برای اشتراک‌های سایت اضافه شد"
}
//لینک‌های فید سایت
var Links={
 postUrl:"http://www.dotnettips.info/feeds/posts",
 posts_commentsUrl:"http://www.dotnettips.info/feeds/comments",
 sharesUrl:"http://www.dotnettips.info/feed/news",
 shares_CommentsUrl:"http://www.dotnettips.info/feed/newscomments"
}
//لینک صفحات سایت
var WebLinks={
Home:"http://www.dotnettips.info",
 postUrl:"http://www.dotnettips.info/postsarchive",
 posts_commentsUrl:"http://www.dotnettips.info/commentsarchive",
 sharesUrl:"http://www.dotnettips.info/newsarchive",
 shares_CommentsUrl:"http://www.dotnettips.info/newsarchive/comments"
}
```

موقعی که اولین بار افزونه نصب می‌شود، باید مقادیر پیش فرضی وجود داشته باشند که یکی از آن‌ها مربوط به مقدار سیکل زمانی

است (هر چند وقت یکبار فید را چک کند) و دیگری ذخیره مقادیر پیش فرض رابط کاربری که قسمت پیشین درست کردیم؛ پروسه پس زمینه برای کار خود به آن‌ها نیاز دارد و بعدی هم تاریخ نصب افزونه است برای اینکه تاریخ آخرین تغییر سایت را با آن مقایسه کند که البته با اولین به روزرسانی تاریخ فید جای آن را می‌گیرد. جهت انجام اینکار یک فایل init.js ایجاد کرده‌ام که قرار است بعد از نصب افزونه، مقادیر پیش فرض بالا را ذخیره کنیم.

```
chrome.runtime.onInstalled.addListener(function(details) {
var now=String(new Date());

var params={};
params[Variables.posts]=true;
params[Variables.postsComments]=false;
params[Variables.shares]=false;
params[Variables.sharesComments]=false;

params[DateContainer.interval]=1;

params[DateContainer.posts]=now;
params[DateContainer.postsComments]=now;
params[DateContainer.shares]=now;
params[DateContainer.sharesComments]=now;

 chrome.storage.local.set(params, function() {
  if(chrome.runtime.lastError)
   {
       /* error */
       console.log(chrome.runtime.lastError.message);
       return;
   }
        });
});
```

chrome.runtime شامل رویدادهایی چون onSuspend ، onStartup ، onInstalled و ... است که مربوطه به وضعیت اجرایی افزونه میشود. آنچه ما اضافه کردیم یک listener برای زمانی است که افزونه نصب شده است و در آن مقادیر پیش فرض ذخیره می‌شوند. اگر خوب دقت کنید می‌بینید که روش دخیره سازی ما در اینجا کمی متفاوت از مقاله پیشین هست و شاید پیش خودتان بگویید که احتمالا به دلیل زیباتر شدن کد اینگونه نوشته شده است ولی مهمترین دلیل این نوع نوشتار این است که متغیرهای بین {} آنچنان فرقی با خود string نمی‌کنند یعنی کد زیر:

```
chrome.storage.local.set('mykey':myvalue,....
```

با کد زیر برابر است:

```
chrome.storage.local.set(mykey:myvalue,...
```

پس اگر مقداری را داخل متغیر بگذاریم آن مقدار حساب نمی‌شود؛ بلکه کلید نام متغیر خواهد شد.
برای معرفی این دو فایل const.js و init.js به manifest.json می‌توانید به صورت زیر عمل کنید:

```
"background": {
    "scripts": ["const.js","init.js"]
}
```

در این حالت خود اکستنشن در زمان نصب یک فایل html درست کرده و این دو فایل js را در آن صدا میزند که البته خود ما هم می‌توانیم اینکار را مستقیما انجام دهیم. مزیت اینکه ما خودمان مسقیما این کار را انجام دهیم این است که در صورتی که فایل‌های js ما زیاد شوند، فایل manifest.jason زیادی شلوغ شده و شکل زشتی پیدا می‌کند و بهتر است این فایل را تا آنجا که می‌توانیم خلاصه نگه داریم. البته روش بالا برای دو یا سه تا فایل js بسیار خوب است ولی اگر به فرض بشود 10 تا یا بیشتر بهتر است یک فایل جداگانه شود و من به همین علت فایل background.htm را درست کرده و به صورت زیر تعریف کرده‌ام:
نکته:نمی توان در تعریف بک گراند هم فایل اسکریپت معرفی کرد و هم فایل html

```
"background": {
    "page": "background.htm"
}
```

background.htm

```
<html>
  <head>
  <script type="text/javascript" src="const.js"></script>
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript" src="init.js"></script>
<script type="text/javascript" src="omnibox.js"></script>
<script type="text/javascript" src="rssreader.js"></script>
<script type="text/javascript" src="contextmenus.js"></script>
  </head>
  <body>
  </body>
</html>
```

لینک‌های بالا به ترتیب معرفی ثابت‌ها، لینک api گوگل که بعدا بررسی می‌شود، فایل init.js برای ذخیره مقادیر پیش فرض، فایل ominibox که در مقاله پیشین در مورد آن صحبت کردیم و فایل rssreader.js که جهت خواندن rss در پایینتر در موردش بحث می‌کنیم و فایل contextmenus که این را هم در مطلب پیشین توضیح دادیم.
جهت خواندن فید سایت ما از Google API استفاده می‌کنیم؛ اینکار دو دلیل دارد:
کدنویسی راحت‌تر و خلاصه‌تر برای خواندن RSS
استفاده اجباری از یک پروکسی به خاطر Content Security Policy و حتی CORS

قبل از اینکه manifst به ورژن 2 برسد ما اجازه داشتیم کدهای جاوااسکریپت به صورت inline در فایل‌های html بنویسیم و یا اینکه از منابع و آدرس‌های خارجی استفاده کنیم برای مثال یک فایل jquery بر روی وب سایت jquery ؛ ولی از ورژن 2 به بعد، گوگل سیاست امنیت محتوا Content Security Policy را که سورس و سند اصلی آن در اینجا قرار دارد، به سیستم Extension خود افزود تا از حملاتی قبیل XSS و یا تغییر منبع راه دور به عنوان یک malware جلوگیری کند. پس ما از این به بعد نه اجازه داشتیم inline بنویسیم و نه اجازه داشتیم فایل jquery را از روی سرورهای سایت سازنده صدا بزنیم. پس برای حل این مشکل، ابتدا مثل همیشه یک فایل js را در فایل html معرفی می‌کردیم و برای حل مشکل دوم باید منابع را به صورت محلی استفاده می‌کردیم؛ یعنی فایل jquery را داخل دایرکتوری extension قرار می‌دادیم.
برای حل مشکل مشکل صدا زدن فایل‌های راه دور ما از Relaxing the Default Policy استفاده می‌کنیم که به ما یک لیست سفید ارائه می‌کند و در این لیست سفید دو نکته‌ی مهم به چشم میخورد که یکی از آن این است که استفاده از آدرس هایی با پروتکل Https و آدرس لوکال local host/127.0.0.1 بلا مانع است و از آنجا که api گوگل یک آدرس Https است، می‌توانیم به راحتی از API آن استفاده کنیم. فقط نیاز است تا خط زیر را به manifest.json اضافه کنیم تا این استثناء را برای ما در نظر بگیرد.

```
"content_security_policy": "script-src 'self' https://*.google.com; object-src 'self'"
```

در اینجا استفاده از هر نوع subdomain در سایت گوگل بلامانع اعلام می‌شود.
بنابراین آدرس زیر به background.htm اضافه می‌شود:

```
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
```

استفاده از این Api در rssreader.js

فایل rssreader.js را به background.htm اضافه می‌کنیم و در آن کد زیر را می‌نویسیم:

```
google.load("feeds", "1");
google.setOnLoadCallback(alarmManager);
```

آدرسی که ما از گوگل درخواست کردیم فقط مختص خواندن فید نیست؛ تمامی apiهای جاوااسکریپتی در آن قرار دارند و ما تنها نیاز داریم قسمتی از آن لود شود. پس اولین خط از دستور بالا بارگذاری بخش مورد نیاز ما را به عهده دارد. در مورد این دستور این صفحه را مشاهده کنید.
در خط دوم ما تابع خودمان را به آن معرفی می‌کنیم تا وقتی که گوگل لودش تمام شد این تابع را اجرا کند تا قبل از لود ما از توابع

آن استفاده نکنیم و خطای undefined دریافت نکنیم. تابعی که ما از آن خواستیم اجرا کند alarmManager نام دارد و قرار است یک آلارم و یک سیکل زمانی را ایجاد کرده و در هر دوره، فید را بخواند. کد تابع مدنظر به شرح زیر است:

```
function alarmManager()
{
chrome.storage.local.get(DateContainer.interval,function ( items) {
period_time==items[DateContainer.interval];
chrome.alarms.create('RssInterval', {periodInMinutes: period_time});
});


chrome.alarms.onAlarm.addListener(function (alarm) {
console.log(alarm);
    if (alarm.name == 'RssInterval') {

var boolposts,boolpostsComments,boolshares,boolsharesComments;
chrome.storage.local.get([Variables.posts,Variables.postsComments,Variables.shares,Variables.sharesComm
ents],function ( items) {
boolposts=items[Variables.posts];
boolpostsComments=items[Variables.postsComments];
boolshares=items[Variables.shares];
boolsharesComments=items[Variables.sharesComments];


chrome.storage.local.get([DateContainer.posts,DateContainer.postsComments,DateContainer.shares,DateCont
ainer.sharesComments],function ( items) {

var Vposts=new Date(items[DateContainer.posts]);
var VpostsComments=new Date(items[DateContainer.postsComments]);
var Vshares=new Date(items[DateContainer.shares]);
var VsharesComments=new Date(items[DateContainer.sharesComments]);

if(boolposts){var result=RssReader(Links.postUrl,Vposts,DateContainer.posts,Messages.PostsUpdated);}
if(boolpostsComments){var
result=RssReader(Links.posts_commentsUrl,VpostsComments,DateContainer.postsComments,Messages.CommentsUp
dated); }
if(boolshares){var
result=RssReader(Links.sharesUrl,Vshares,DateContainer.shares,Messages.SharesUpdated);}
if(boolsharesComments){var
result=RssReader(Links.shares_CommentsUrl,VsharesComments,DateContainer.sharesComments,Messages.SharesC
ommentsUpdated);}

});
});
    }
});
}
```

خطوط اول تابع alarmManager وظیفه‌ی خواندن مقدار interval را که در init.js ذخیره کرده‌ایم، دارند که به طور پیش فرض 10 ذخیره شده است تا تایمر یا آلارم خود را بر اساس آن بسازیم. در خط chrome.alarms.create یک آلارم با نام rssinterval می‌سازد و قرار است هر 10 دقیقه وظایفی که بر دوشش گذاشته می‌شود را اجرا کند **(استفاده از api جهت دسترسی به آلارم نیاز به مجوز "alarms" دارد)** . وظایفش از طریق یک listener که بر روی رویداد chrome.alarms.onAlarm گذاشته شده است مشخص می‌شود. در خط بعدی مشخص می‌شود که این رویداد به خاطر چه آلارمی صدا زده شده است. البته از آنجا که ما یک آلارم داریم، نیاز چندانی به این کد نیست. ولی اگر پروژه شما حداقل دو آلارم داشته باشد نیاز است مشخص شود که کدام آلارم باعث صدا زدن این رویداد شده است. در مرحله بعد مشخص می‌کنیم که کاربر قصد بررسی چه قسمت‌هایی از سایت را داشته است و در تابع callback آن هم تاریخ آخرین تغییرات هر بخش را می‌خوانیم و در متغیری نگه داری می‌کنیم. هر کدام را جداگانه چک کرده و تابع RssReader را برای هر کدام صدا می‌زنیم. این تابع 4 پارامتر دارد:

آدرس فیدی که قرار است از روی آن بخواند

آخرین به روزسانی که از سایت داشته متعلق به چه تاریخی است.

نام کلید ذخیره سازی تاریخ آخرین تغییر سایت که اگر بررسی شد و مشخص شد سایت به روز شده است، تاریخ جدید را روی آن ذخیره کنیم.

در صورتی که سایت به روز شده باشد نیاز است پیامی را برای کاربر نمایش دهیم که این پیام را در اینجا قرار می‌دهیم.

کد تابع rssreader

```
function RssReader(URL,lastupdate,datecontainer,Message) {
            var feed = new google.feeds.Feed(URL);
            feed.setResultFormat(google.feeds.Feed.XML_FORMAT);
                    feed.load(function (result) {
if(result!=null)
{
var strRssUpdate = result.xmlDocument.firstChild.firstChild.childNodes[5].textContent;
var RssUpdate=new Date(strRssUpdate);

if(RssUpdate>lastupdate)
{
SaveDateAndShowMessage(datecontainer,strRssUpdate,Message)
}
}
});
}
```

در خط اول فید توسط گوگل خوانده میشود، در خط بعدی ما به گوگل میگوییم که فید خوانده شده را چگونه به ما تحویل دهد که ما قالب xml را خواسته ایم و در خط بعدی اطلاعات را در متغیری به اسم result قرار میدهد که در یک تابع برگشتی آن را در اختیار ما میگذارد. از آن جا که ما قرار است تگ **lastBuildDate** را بخوانیم که پنجمین تگ اولین گره در اولین گره به حساب می‌آید، خط زیر این دسترسی را برای ما فراهم می‌کند و چون تگ ما در یک مکان ثابت است با همین تکه کد، دسترسی مستقیمی به آن داریم:

```
var strRssUpdate = result.xmlDocument.firstChild.firstChild.childNodes[5].textContent;
```

مرحله بعد تاریخ را که در قالب رشته‌ای است، تبدیل به تاریخ کرده و با lastupdate یعنی آخرین تغییر قبلی مقایسه می‌کنیم و اگر تاریخ برگرفته از فید بزرگتر بود، یعنی سایت به روز شده است و تابع SaveDateAndShowMessage را صدا می‌زنیم که وظیفه ذخیره سازی تاریخ جدید و ایجاد notification را به عهده دارد و سه پارامتر کلید ذخیره سازی و مقدار آن و پیام را به آن پاس می‌کنیم.

کد تابع SaveDateAndShowMesage

```
function SaveDateAndShowMessage(DateField,DateValue,Message)
{
var params={
}
params[DateField]=DateValue;

chrome.storage.local.set( params,function(){

var options={
  type: "basic",
   title: Messages.SiteUpdated,
   message: Message,
   iconUrl: "icon.png"
}
chrome.notifications.create("",options,function(){
chrome.notifications.onClicked.addListener(function(){
chrome.tabs.create({'url': WebLinks.Home}, function(tab) {
});
});
});
});
}
```

خطوط اول مربوط به ذخیره تاریخ است و دومین نکته نحوه‌ی ساخت نوتیفکیشن است. اجرای یک notification نیاز به مجوز "notifications " دارد که مجوز آن در manifest به شرح زیر است:

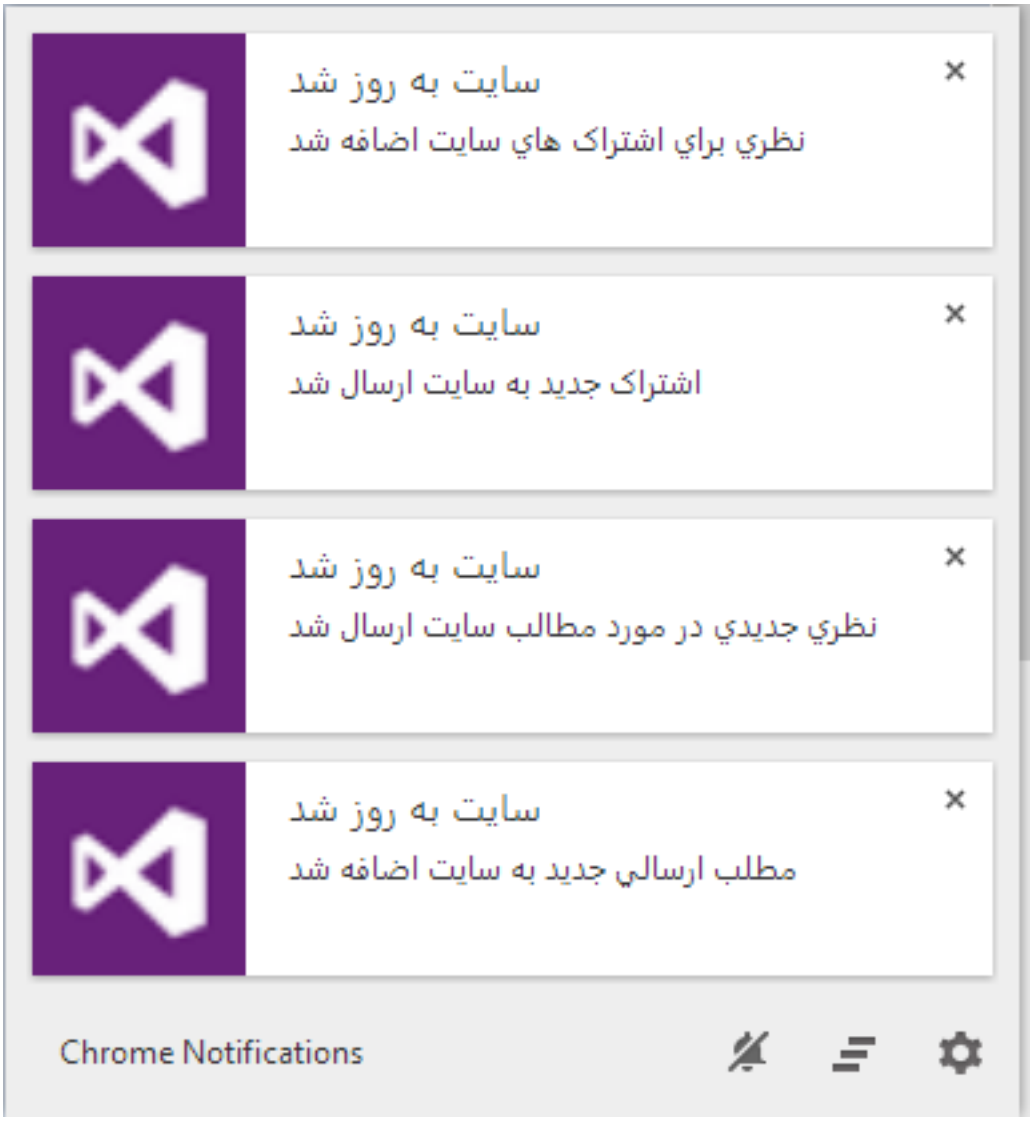
```
"permissions": [
    "storage",
     "tabs",
 "alarms",
 "notifications"
  ]
```

در خطوط بالا سایر مجوزهایی که در طول این دوره به کار اضافه شده است را هم می‌بینید.
برای ساخت نوتیفکیشن از کد chrome.notifications.create استفاده می‌کنیم که پارامتر اول آن کد یکتا یا همان ID جهت ساخت نوتیفیکیشن هست که میتوان خالی گذاشت و دومی تنظیمات ساخت آن است؛ از قبیل عنوان و آیکن و ... که در بالا به اسم options معرفی کرده ایم و در آگومان دوم آن را معرفی کرده ایم و آرگومان سوم هم یک تابع callback است که نوشتن آن اجباری است. options شامل عنوان، پیام، آیکن و نوع notification می‌باشد که در اینجا basic انتخاب کرده‌ایم. برای دسترسی به دیگر خصوصیت‌های options به اینجا و برای داشتن notificationهای زیباتر به عنوان rich notification به اینجا مراجعه کنید. برای اینکه این امکان باشد که کاربر با کلیک روی notification به سایت هدایت شود باید در تابع callback مربوط به notifications.create این کد اضافه گردد که در صورت کلیک یک تب جدید با آدرس سایت ساخته شود:

```
chrome.notifications.create("",options,function(){
chrome.notifications.onClicked.addListener(function(){
chrome.tabs.create({'url': WebLinks.Home}, function(tab) {
});});
});
```

نکته مهم: پیشتر معرفی آیکن به صورت بالا کفایت میکرد ولی بعد از این باگ کد زیر هم باید جداگانه به manifest اضافه شود:

```
"web_accessible_resources": [
    "icon.png"
  ]
```



خوب؛ کار افزونه تمام شده است ولی اجازه دهید این بار امکانات افزونه را بسط دهیم:
من می‌خواهم برای افزونه نیز قسمت تنظیمات داشته باشم. برای دسترسی به options میتوان از قسمت مدیریت افزونه‌ها در مرورگر یا حتی با راست کلیک روی آیکن browser action عمل کرد. در اصل این قسمت برای تنظیمات افزونه است ولی ما به خاطر آموزش و هم اینکه افزونه ما UI خاصی نداشت تنظیمات را از طریق browser action پیاده سازی کردیم و گرنه در صورتی که افزونه شما شامل UI خاصی مثلا نمایش فید مطالب باشد، بهترین مکان تنظیمات، options است. برای تعریف options در manifest.json به روش زیر اقدام کنید:

```
"options_page": "popup.html"
```

همان صفحه popup را در این بخش نشان میدهم و اینبار یک کار اضافه‌تر دیگر که نیاز به آموزش ندارد اضافه کردن input با Type=number است که برای تغییر interval به کار می‌رود و نحوه ذخیره و بازیابی آن را در طول دوره یاد گرفته اید.

جایزگزینی صفحات یا Override Pages
بعضی صفحات مانند بوک مارک و تاریخچه فعالیت‌ها History و همینطور newtab را می‌توانید جایگزین کنید. البته یک اکستنشن میتواند فقط یکی از صفحات را جایگزین کند. برای تعیین جایگزین در manifest اینگونه عمل می‌کنیم:

```
"chrome_url_overrides": {
    "newtab": "newtab.htm"
  }
```

ایجاد یک تب اختصاصی در Developer Tools
تکه کدی که باید manifest اضافه شود:

```
"devtools_page": "devtools.htm"
```

شاید فکر کنید کد بالا الان شامل مباحث ui و ... می‌شود و بعد به مرورگر اعمال خواهد شد؛ در صورتی که اینگونه نیست و نیاز دارد چند خط کدی نوشته شود. ولی مسئله اینست که کد بالا تنها صفحات html را پشتیبانی می‌کند و مستقیما نمی‌تواند فایل js را بخواند. پس صفحه بالا را ساخته و کد زیر را داخلش می‌گذاریم:

```
<script src="devtools.js"></script>
```

فایل devtools.js هم شامل کد زیر می‌شود:

```
chrome.devtools.panels.create(
    "Dotnettips Updater Tools",
    "icon.png",
    "devtoolsui.htm",
    function(panel) {
    }
);
```

خط chrome.devtools.panels.create یک پنل یا همان تب را ساخته و در پارامترهای بالا به ترتیب عنوان، آیکن و صفحه‌ای که باید در آن رندر شود را دریافت می‌کند و پس از ایجاد یک callback اجرا می‌شود. اطلاعات بیشتر

APIها

برای دیدن لیست کاملی از APIها می‌توانید به مستندات آن رجوع کنید و این مورد را به یاد داشته باشید که ممکن است بعضی apiها در بعضی موارد پاسخ ندهند. به عنوان مثال در content scripts نمی‌توانید به chrome.devtools.panels دسترسی داشته باشید یا اینکه در DeveloperTools دسترسی به DOM میسر نیست. پس این مورد را به خاطر داشته باشید. همچنین بعضی apiها از نسخه‌ی خاصی به بعد اضافه شده‌اند مثلا همین مثال قبلی devtools از نسخه 18 به بعد اضافه شده است و به این معنی است با خیال راحت می‌توانید از آن استفاده کنید. یا آلارم‌ها از نسخه 22 به بعد اضافه شده‌اند. البته خوشبختانه امروزه با دسترسی آسانتر به اینترنت و آپدیت خودکار مرورگرها این مشکلات دیگر آن چنان رخ نمی‌دهند.

Messaging

همانطور که در بالا اشاره شد شما نمی‌توانید بعضی از apiها را در بعضی جاها استفاده کنید. برای حل این مشکل می‌توان از messaging استفاده کرد که دو نوع تبادلات پیغامی داریم:

One-Time Requests یا درخواست‌های تک مرتبه‌ای

Long-Lived Connections یا اتصالات بلند مدت یا مصر

درخواست‌های تک مرتبه ای

این درخواست‌ها همانطور که از نامش پیداست تنها یک مرتبه رخ می‌دهد؛ درخواست را ارسال کرده و منتظر پاسخ می‌ماند. به عنوان مثال به کد زیر که در content script است دقت کنید:

```
window.addEventListener("load", function() {
    chrome.extension.sendMessage({
        type: "dom-loaded",
        data: {
            myProperty   : "value"
        }
    });
}, true);
```

کد بالا یک ارسال کننده پیام است. موقعی که سایتی باز می‌شود، یک کلید با مقدارش را ارسال می‌کند و کد زیر در background گوش می‌ایستد تا اگر درخواستی آمد آن را دریافت کند:

```
chrome.extension.onMessage.addListener(function(request, sender, sendResponse) {
    switch(request.type) {
        case "dom-loaded":
            alert(request.data.myProperty   );
        break;
    }
    return true;
});
```

اتصالات بلند مدت یا مصر

اگر نیاز به یک کانال ارتباطی مصر و همیشگی دارید کدها را به شکل زیر تغییر دهید *contentscripts*

```
var port = chrome.runtime.connect({name: "my-channel"});
port.postMessage({myProperty: "value"});
port.onMessage.addListener(function(msg) {
    // do some stuff here
});
```

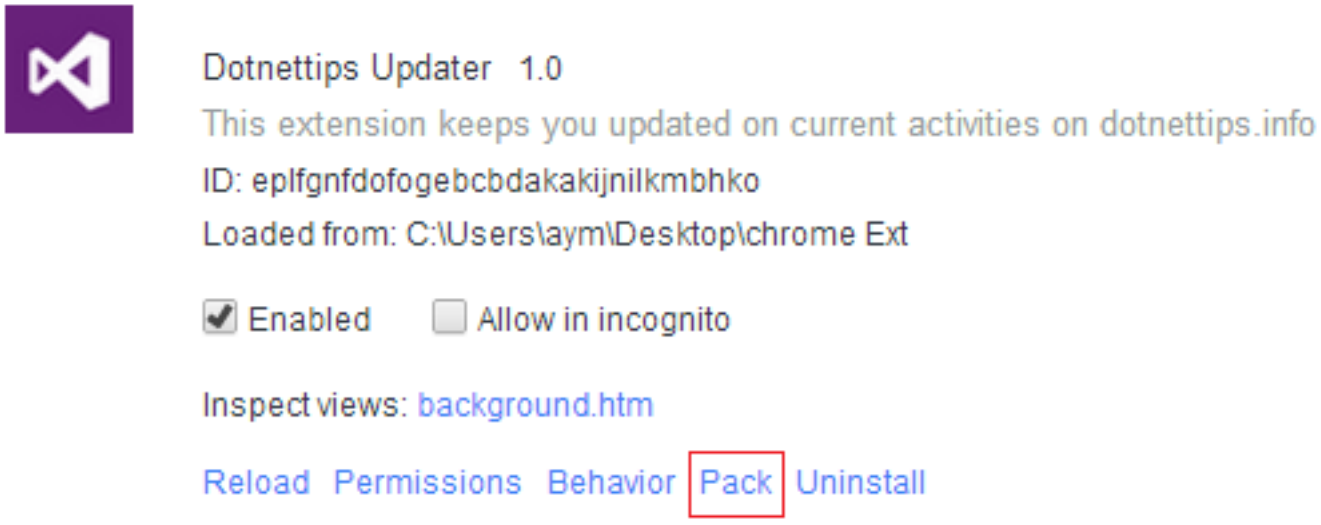
background

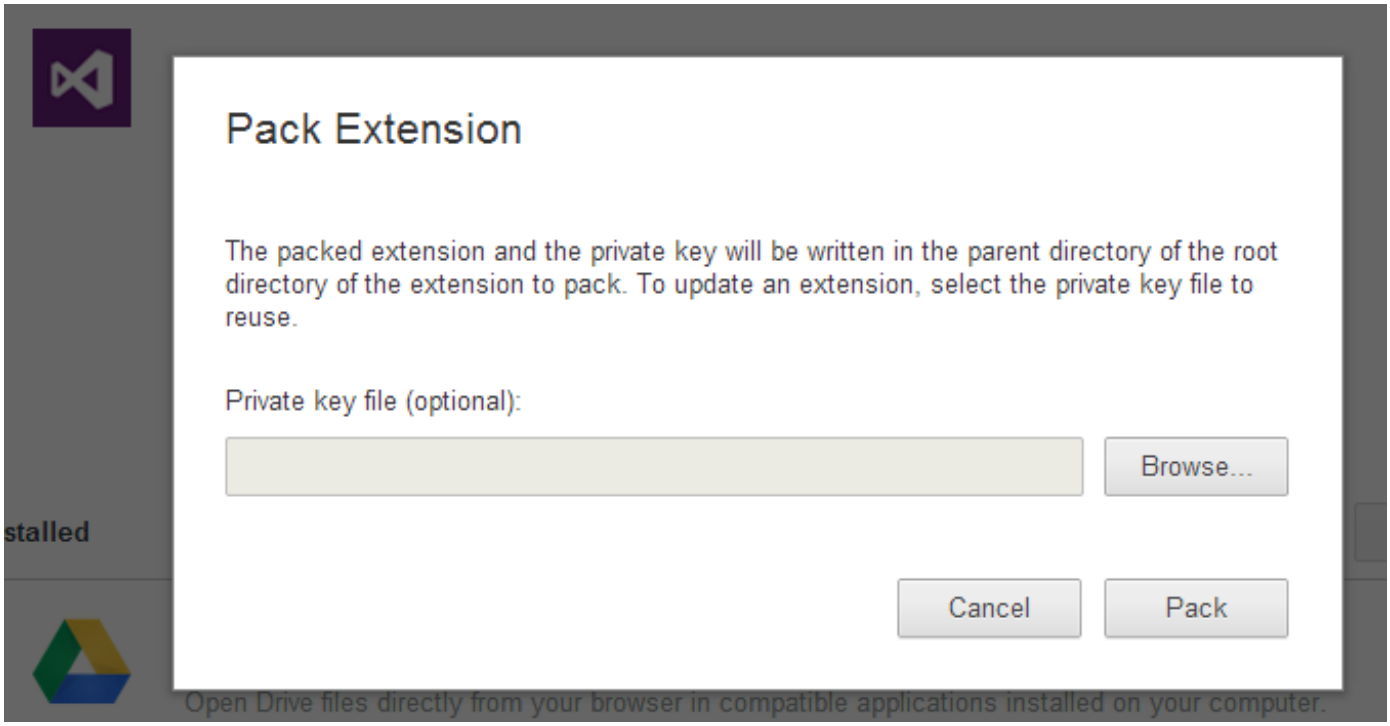
```
chrome.runtime.onConnect.addListener(function(port) {
    if(port.name == "my-channel"){
        port.onMessage.addListener(function(msg) {
            // do some stuff here
        });
    }
});
```

نمونه کد نمونه کدهایی که در سایت گوگل موجود هست می‌توانند کمک بسیاری خوبی باشند ولی اینگونه که پیداست اکثر مثال‌ها مربوط به نسخه‌ی یک manifest است که دیگر توسط مرورگرها پشتیبانی نمی‌شوند و مشکلاتی چون اسکریپت inline و CSP که در بالا اشاره کردیم را دارند و گوگل کدها را به روز نکرده است.

دیباگ کردن و پک کردن فایل‌ها برای تبدیل به فایل افزونه Debugging and packing

برای دیباگ کردن کدها می‌توان از دو نمونه console.log و alert برای گرفتن خروجی استفاده کرد و همچنین ابزار Chrome Apps & Extensions Developer Tool هم نسبتا امکانات خوبی دارد که البته میتوان از آن برای پک کردن اکستنشن نهایی هم استفاده کرد. برای پک کردن روی گزینه pack کلیک کرده و در کادر باز شده گزینه‌ی pack را بزنید. برای شما دو نوع فایل را در مسیر والد دایرکتوری extension نوشته شده درست خواهد کرد که یکی پسوند crx دارد که می‌شود همان فایل نهایی افزونه و دیگری هم پسوند pem دارد که یک کلید اختصاصی است و باید برای آپدیت‌های آینده افزونه آن را نگاه دارید. در صورتی که افزونه را تغییر دادید و خواستید آن را به روز رسانی کنید موقعی که اولین گزینه pack را می‌زنید و صفحه باز می‌شود قبل از اینکه دومین گزینه pack را بزنید، از شما می‌خواهد اگر دارید عملیات به روز رسانی را انجام می‌دهید، کلید اختصاصی آن را وارد نمایید و بعد از آن گزینه pack را بزنید:





آپلود نهایی کار در Google web store

برای آپلود نهایی کار به google web store که در آن تمامی برنامه‌ها و افزونه‌های کروم قرار دارند بروید. سمت راست آیکن تنظیمات را بزنید و گزینه developer dashboard را انتخاب کنید تا صفحه‌ی آپلود کار برای شما باز شود. دایرکتوری محتویات اکستنشن را zip کرده و آپلود نمایید. توجه داشته باشید که محتویات و سورس خود را باید آپلود کنید نه فایل crx را. بعد از آپلود موفقیت آمیز، صفحه‌ای ظاهر می‌شود که از شما آیکن افزونه را در اندازه 128 پیکسل میخواهد بعلاوه توضیحاتی در مورد افزونه، قیمت گذاری که به طور پیش فرض به صورت رایگان تنظیم شده است، لینک وب سایت مرتبط، لینک محل پرسش و پاسخ برای افزونه، اگر لینک یوتیوبی در مورد افزونه دارید، یک شات تصویری از افزونه و همینطور چند تصویر برای اسلایدشو سازی که در همان صفحه استاندارد آن‌ها را توضیح می‌دهد و در نهایت گزینه‌ی جالب‌تر هم اینکه اکستنشن شما برای چه مناطقی تهیه شده است که متاسفانه ایران را ندیدم که می‌توان همه موارد را انتخاب کرد. به خصوص در مورد ایران که آی پی‌ها هم صحیح نیست، انتخاب ایران چنان تاثیری ندارد و در نهایت گزینه‌ی publish را می‌زنید که متاسفانه بعد از این صفحه درخواست می‌کند برای اولین بار باید 5 دلار آمریکا پرداخت شود که برای بسیاری از ما این گزینه ممکن نیست.

سورس پروژه را می‌توانید از اینجا ببینید و خود افزونه را از اینجا دریافت کنید.

عنوان: **افزونه نویسی برای مرورگرها : فایرفاکس : قسمت اول**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۱/۰۵ ۲۲:۵۰
آدرس: www.dotnettips.info
گروه‌ها: Firefox, Plugin, Extension, Addon

در دو مقاله پیشین ^ ، ^ به بررسی نوشتن افزونه در مرورگر کروم پرداختیم و اینبار قصد داریم همان پروژه را برای فایرفاکس پیاده کنیم. پس در مورد کدهای تکراری توضیحی داده نخواهد شد و برای فهم آن می‌توانید به دو مقاله قبلی رجوع کنید. همه‌ی ما فایرفاکس را به خوبی می‌شناسیم. اولین باری که این مرورگر آمد سرو صدای زیادی به پا کرد و بازار وسیعی از مرورگرها را که در چنگ IE بود، به دست آورد . این سر و صدا بیشتر به خاطر امنیت و کارآیی بالای این مرورگر، استفاده از آخرین فناوری‌های تحت وب و دوست داشتنی برای طراحان وب بود. همچنین یکی دیگر از مهمترین ویژگی‌های آن، امکان سفارشی سازی آن با افزونه‌ها extensions یا addon بود که این ویژگی در طول این سال‌ها تغییرات زیادی به خود دیده است. در مورد افزونه نویسی برای فایرفاکس در سطح نت مطالب زیادی وجود دارند که همین پیشرفت‌های اخیر در مورد افزونه‌ها باعث شده خیلی از این مطالب به روز نباشند. اگر در مقاله پیشین فکر می‌کنید که کروم چقدر در نوشتن افزونه جذابیت دارد و امکانات خوبی را در اختیار شما می‌گذارد، الان دیگر وقت آن است که نظر خودتان را عوض کنید و فایرفاکس را نه تنها یک سرو گردن بلکه بیشتر از این حرف‌ها بالاتر بدانید.
شرکت موزیالا برای قدرتمندی و راحتی کار طراحان یک sdk طراحی کرده است است و شما با استفاده از کدهای موجود در این sdk قادرید کارهای زیادی را انجام دهید. برای نصب این sdk باید پیش نیازهایی بر روی سیستم شما نصب باشد:
نصب پایتون 2.5 یا 2.6 یا 2.7 که فعلا در سایت آن، نسخه‌ی 2.7 در دسترس هست. توجه داشته باشید که هنوز برای نسخه‌ی 3 پایتون پشتیبانی صورت نگرفته است.

آخرین نسخه‌ی sdk را هم می‌توانید از این آدرس به صورت zip و یا از این آدرس به صورت tar دانلود کنید و در صورتیکه دوست دارید به سورس آن دسترسی داشته باشید یا اینکه از سورس‌های مشارکت شده یا غیر رسمی استفاده کنید، از این صفحه آن را دریافت کنید.

بعد از دانلود sdk به شاخه‌ی bin رفته و فایل activate.bat را اجرا کنید. موقعی که فایل activate اجرا شود، باید چنین چیزی دیده شود:

```
(C:\Users\aym\Downloads\addon-sdk-1.17) C:\Users\aym\Downloads\addon-sdk-1.17\bin>
```

برای سیستم‌های عامل Linux,FreeBSD,OS X دستورات زیر را وارد کنید:
اگر یک کاربر پوسته‌ی bash هستید کلمه زیر را در کنسول برای اجرای activate بزنید:

```
source bin/activate
```

اگر کاربر پوسته‌ی بش نیستید:

```
bash bin/activate
```

نهایتا باید کنسول به شکل زیر در آید یا شبیه آن:

```
(addon-sdk)~/mozilla/addon-sdk >
```

بعد از اینکه به کنسول آن وارد شدید، کلمه cfx را در آن تایپ کنید تا راهنمای دستورات و سوییچ‌های آن‌ها نمایش داده شوند. از این ابزار میتوان برای راه اندازی فایرفاکس و اجرای افزونه بر روی آن، پکیج کردن افزونه، دیدن مستندات و آزمون‌های واحد استفاده کرد.

آغاز به کار

برای شروع، فایل‌های زیادی باید ساخته شوند، ولی نگران نباشید cfx این کار را برای شما خواهد کرد. دستورات زیر را جهت

ساخت یک پروژه خالی اجرا کنید:

```
mkdir fxaddon
cd fxaddon
cfx init
```

یک پوشه را در مسیری که کنسول بالا اشاره میکرد، ساختم و وارد آن شدم و با دستور cfx init دستور ساخت یک پروژه‌ی خالی را دادم و باید بعد از این دستور، یک خروجی مشابه زیر نشان بدهد:

```
* lib directory created
* data directory created
* test directory created
* doc directory created
* README.md written
* package.json written
* test/test-main.js written
* lib/main.js written
* doc/main.md written
Your sample add-on is now ready for testing:
try "cfx test" and then "cfx run". Have fun!"
```

در این پوشه یک فایل به اسم package.json هم وجود دارد که اطلاعات زیر داخلش هست:

```
{
  "name": "fxaddon",
  "title": "fxaddon",
  "id": "jid1-QfyqpNby9lTlcQ",
  "description": "a basic add-on",
  "author": "",
  "license": "MPL 2.0",
  "version": "0.1"
}
```

این اطلاعات شامل نام و عنوان افزونه، توضیحی کوتاه در مورد آن، نویسنده‌ی افزونه، ورژن افزونه و ... است. این فایل دقیقا معادل manifest.json در کروم است. در افزونه نویسی‌های قدیم این فایل install.rdf نام داشت و بر پایه‌ی فرمت rdf بود. ولی در حال حاضر با تغییرات زیادی که افزونه نویسی در فایرفاکس کرده‌است، الان این فایل بر پایه یا فرمت json است. اطلاعات package را به شرح زیر تغییر می‌دهیم:

```
{
  "name": "dotnettips",
  "title": ".net Tips Updater",
  "id": "jid1-QfyqpNby9lTlcQ",
  "description": "This extension keeps you updated on current activities on dotnettips.info",
  "author": "yeganehaym@gmail.com",
  "license": "MPL 2.0",
  "version": "0.1"
}
```

رابطهای کاربری Action Button و Toggle Button فایل main.js را در دایرکتوری lib باز کنید:
موقعی که در کروم افزونه می‌نوشتیم امکانی به اسم browser action داشتیم که در اینجا با نام action button شناخته می‌شود. در اینجا باید کدها را require کرد، همان کاری در خیلی از زبان‌ها مثلا مثل سی برای صدا زدن سرآیندها می‌کنید. مثلا برای action button اینگونه است:

```
var button= require('sdk/ui/button/action');
```

نحوه‌ی استفاده هم بدین صورت است:

```
buttons.ActionButton({...});
```

که در بین {} خصوصیات دکمه‌ی مورد نظر نوشته می‌شود. ولی من بیشتر دوست دارم از شیء دیگری استفاده کنم. به همین جهت ما از یک مدل دیگر button که به اسم toggle button شناخته می‌شود، استفاده می‌کنیم. از آن جا که این button دارای دو حالت انتخاب (حالت فشرده شده) و غیر انتخاب (معمولی و آماده فشرده شدن توسط کلیک کاربر) است، بهترین انتخاب هست.

کد زیر یک toggle button را برای فایرفاکس می‌سازد که با کلیک بر روی آن، صفحه‌ی popup.htm به عنوان یک پنل روی آن رندر می‌شود:

```
var tgbutton = require('sdk/ui/button/toggle');
var panels = require("sdk/panel");
var self = require("sdk/self");

var button = tgbutton.ToggleButton({
  id: "updaterui",
  label: ".Net Updater",
  icon: {
    "16": "./icon-16.png",
    "32": "./icon-32.png",
    "64": "./icon-64.png"
  },
  onChange: handleChange
});

var panel = panels.Panel({
  contentURL: self.data.url("./popup.html"),
  onHide: handleHide
});

function handleChange(state) {
  if (state.checked) {
    panel.show({
      position: button
    });
  }
}

function handleHide() {
  button.state('window', {checked: false});
}
```

در سه خط اول، فایل‌هایی را که نیاز است Required شوند، می‌نویسیم و در یک متغیر ذخیره می‌کنیم. اگر در متغیر نریزیم مجبور هستیم همیشه هر کدی را به جای نوشتن عبارت زیر:

```
tgbutton.ToggleButton
```

به صورت زیر بنویسیم:

```
require('sdk/ui/button/toggle').ToggleButton
```

که اصلا کار جالبی نیست. اگر مسیرهای نوشته شده را از مبدا فایل zip که اکسترکت کرده‌اید، در دایرکتوری sdk در شاخه lib بررسی کنید، با دیگر موجودیت‌های sdk آشنا خواهید شد.

در خط بعدی به تعریف یک شیء از نوع toggle button به اسم button میپردازیم و خصوصیاتی که به این دکمه داده ایم، مانند یک کد شناسایی، یک برچسب که به عنوان tooltip نمایش داده خواهد شد و آیکن‌هایی در اندازه‌های مختلف که در هرجایی کاربر آن دکمه را قرار داد، در اندازه‌ی مناسب باشد و نهایتا به تعریف یک رویداد می‌پردازیم. تابع handlechange زمانی صدا زده می‌شود که در وضعیت دکمه‌ی ایجاد شده تغییری حاصل شود. در خط بعدی شیء panel را به صورت global میسازیم. شیء self دسترسی ما را به اجزا یا فایل‌های افزونه خودمان فراهم می‌کند که در اینجا دسترسی ما به فایل html در شاخه‌ی data میسر شده است و مقدار مورد نظر را در contentURL قرار می‌دهد. نهایتا هم برای رویداد onhide تابعی را در نظر می‌گیریم تا موقعی که پنجره بسته شد بتوانیم وضعیت toggle button را به حالت قبلی بازگردانیم و حالت فشرده نباشد. چرا که این دکمه تنها با کلیک ماوس به حالت فشرده و حالت معمولی سوییچ میکند. پس اگر کاربر با کلیک بر روی صفحه‌ی مرورگر پنجره را ببندد، دکمه در همان وضعیت فشرده باقی می‌ماند.

همانطور که گفتیم تابع handlechnage موقعی رخ میدهد که در وضعیت دکمه، تغییری رخ دهد و نمیدانیم که این وضعیت فشرده شدن دکمه هست یا از حالت فشرده خارج شده است. پس با استفاده از ویژگی checked بررسی میکنم که آیا دکمه‌ای فشرده شده یا خیر؛ اگر برابر true بود یعنی کاربر روی دکمه، کلیک کرده و دکمه به حالت فشرده رفته، پس ما هم پنل را به آن نشان می‌دهیم و خصوصیات دلخواهی را برای مشخص کردن وضعیت پنل نمایشی به آن پاس می‌کنیم. خصوصیت یا پارامترهای زیادی را می‌توان در حین ساخت پنل برای آن ارسال کرد. با استفاده از خصوصیت position محل نمایش پنجره را مشخص می‌کنیم. در صورتی که ذکر نشود پنجره در وسط مرورگر ظاهر خواهد شد.

تابع onhide زمانی رخ میدهد که به هر دلیلی پنجره بسته شده باشد که در بالا یک نمونه‌ی آن را عرض کردیم. ولی اتفاقی که می‌افتد، وضعیت تابع را با متد state تغییر می‌دهیم و خصوصیت checked آن را false می‌کنیم. بجای پارامتر اولی، دو گزینه را میتوان نوشت؛ یکی window و دیگری tab است. اگر شما گزینه tab را جایگزین کنید، اگر در یک تب دکمه به حالت فشرده برود و به تب دیگر بروید و باعث بسته شدن پنجره بشوید، دکمه تنها در تبی که فعال است به حالت قبلی باز می‌گردد و تب اولی همچنان حالت خود را حفظ خواهد کرد پس می‌نویسیم window تا این عمل در کل پنجره اعمال شود.

Context Menus

برای ساخت منوی کانتکست از کد زیر استفاده می‌کنیم:

```
var contextMenu = require("sdk/context-menu");

var home = contextMenu.Item({
  label: "صفحه اصلی",
  data: "http://www.dotnettips.info/"
});
var postsarchive = contextMenu.Item({
  label: "مطالب سایت",
  data: "http://www.dotnettips.info/postsarchive"
});

var menuItem = contextMenu.Menu({
  label: "Open .Net Tips",
  context: contextMenu.PageContext(),
   items: [home, postsarchive],
  image: self.data.url("icon-16.png"),
  contentScript: 'self.on("click", function (node, data) {' +
                 '  window.location.href = data;' +
                 '});'
});
```

این منو هم مثل کروم دو زیر منو دارد که یکی برای باز کردن صفحه‌ی اصلی و دیگری برای باز کردن صفحه‌ی مطالب است. هر کدام یک برچسب برای نمایش متن دارند و یکی هم دیتا که برای نگهداری آدرس است. در خط بعدی منوی پدر یا والد ساخته می‌شود که با خصوصیت items، زیر منوهایش را اضافه می‌کنیم و با خصوصیت image، تصویری را در پوشه‌ی دیتا به آن معرفی می‌کنیم که اندازه‌ی آن 16 پیکسل است و دومی هم خصوصیت context است که مشخص می‌کند این گزینه در چه مواردی بر روی context menu نمایش داده شود. الان روی همه چیزی نمایش داده می‌شود. اگر گزینه، SelectionContext باشد، موقعی که متنی انتخاب شده باشد، نمایش می‌یابد. اگر SelectorContext باشد، خود شما مشخص می‌کنید بر روی چه مواردی نمایش یابد؛ مثلا عکس یا تگ p یا هر چیز دیگری، کد زیر باعث می‌شود فقط روی عکس نمایش یابد:

```
SelectorContext("img")
```

کد زیر هم روی عکس و هم روی لینکی که href داشته باشد:

```
SelectorContext("img,a[href]")
```

موارد دیگری هم وجود دارند که میتوانید مطالب بیشتری را در مورد آن‌ها در اینجا مطالعه کنید. آخرین خصوصیت باقی مانده، content script است که می‌توانید با استفاده از جاوااسکریپت برای آن کد بنویسید. موقعی که برای آن رویداد کلیک رخ داد، مشخص شود تابعی را صدا میزند با دو آرگومان؛ گره ای که انتخاب شده و داده‌ای که به همراه دارد که آدرس سایت است و آن را در نوار آدرس درج می‌کند.

آن منوهایی که با متد item ایجاد شده‌اند منوهایی هستند که با کلیک کاربر اجرا می‌شوند؛ ولی والدی که با متد menu ایجاد شده است، برای منویی است که زیر منو دارد و خودش لزومی به اجرای کد ندارد. پس اگر منویی میسازید که زیرمنو ندارد و خودش قرار است کاری را انجام دهد، به صورت همان item بنویسید که پایین‌تر نمونه‌ی آن را خواهید دید.

الان مشکلی که ایجاد می‌شود این است که موقعی که سایت را باز می‌کند، در همان تبی رخ می‌دهد که فعال است و اگر کاربر بر روی صفحه‌ی خاصی باشد، آن صفحه به سمت سایت مقصد رفته و سایت فعلی از دست میرود. روش صحیح‌تر اینست که تبی جدید بار شود و آدرس مقصد در آن نمایش یابد. پس باید از روشی استفاده کنیم که رویداد کلیک توسط کد خود افزونه مدیریت شود، تا با استفاده از شیء tab، یک تب جدید با آدرسی جدید ایجاد کنیم. پس کد را با کمی تغییر می‌نویسیم:

```
var tabs = require("sdk/tabs");
var menuItem = contextMenu.Menu({
  label: "Open .Net Tips",
  context: contextMenu.PageContext(),
   items: [home, postsarchive],
  image: self.data.url("icon-16.png"),
  contentScript: 'self.on("click", function (node, data) {' +
                 '  self.postMessage(data);' +
                 '});',
 onMessage: function (data) {
     tabs.open(data);
  }
});
```

با استفاده از postmessage، هر پارامتری را که بخواهیم ارسال می‌کنیم و بعد با استفاده از رویداد onMessage، داده‌ها را خوانده و کد خود را روی آن‌ها اجرا می‌کنیم.

بگذارید کد زیر را هم جهت سرچ مطالب بر روی سایت پیاده کنیم:

```
var Url="http://www.dotnettips.info/search?term=";
var searchMenu = contextMenu.Item({
  label: "search for",
  context: [contextMenu.PredicateContext(checkText),contextMenu.SelectionContext()],
    image: self.data.url("icon-16.png"),
  contentScript: 'self.on("click", function () {' +
  '  var text = window.getSelection().toString();' +
                 '  if (text.length > 20)' +
                 '   text = text.substr(0, 20);' +
                 '  self.postMessage(text);'+
                '})',
onMessage: function (data) {
```

```
    tabs.open(Url+data);
  }

});

function checkText(data) {

       if(data.selectionText === null)
           return false;

       console.log('selectionText: ' + data.selectionText);

       //handle showing or hiding of menu items based on the text content.
       menuItemToggle(data.selectionText);

       return true;
};

function menuItemToggle(text){
var searchText="جست و جو برای ";
    searchMenu.label=searchText+text;

};
```

در ساخت این منو، ما از ContextSelection استفاده کرده‌ایم. بدین معنی که موقعی که چیزی روی صفحه انتخاب شد، این منو ظاهر شود و گزینه‌ی دیگری که در کنارش هست، گزینه contextMenu.PredicateContext وظیفه دارد تابعی که به عنوان آرگومان به آن دادیم را موقعی که منو کانتکست ایجاد شد، صدا بزند و اینگونه میتوانیم بر حسب اطلاعات کانتکست، منوی خود را ویرایش کنیم. مثلا من دوست دارم موقعی که متنی انتخاب می‌شود و راست کلیک می‌کنم گزینه‌ی "جست و جو برای..." نمایش داده شود و به جای ... کلمه‌ی انتخاب شده نمایش یابد. به شکل زیر دقت کنید. این چیزی است که ما قرار است ایجاد کنیم:

در کل موقع ایجاد منو تابع checkText اجرا شده و متن انتخابی را خوانده به عنوان یک آرگومان برای تابع menuItemToggle ارسال می‌کند و به رشته "جست و جو برای" می‌چسباند. در خود پارامترهای آیتم اصلی، گزینه content scrip، با استفاده از جاوااسکریپت، متن انتخاب شده را دریافت کرده و با استفاده از متد postmessage برای تابع onMessage ارسال کرده و با ساخت یک تب و چسباندن عبارت به آدرس جست و جو سایت، کاربر را به صفحه مورد نظر هدایت کرده و عمل جست و جو در سایت انجام می‌گیرد.

در قسمت آینده موارد بیشتری را در مورد افزونه نویسی در فایرفاکس بررسی خواهیم کرد و افزونه را تکمیل خواهیم کرد

عنوان: **طراحی افزونه پذیر با ASP.NET MVC 4.x/5.x - قسمت اول**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۷ ۱۳:۱۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Build Events, Dependency Injection, Plugin, StructureMap

در طی چند قسمت، نحوه‌ی طراحی یک سیستم افزونه پذیر را با ASP.NET MVC بررسی خواهیم کرد. عناوین مواردی که در این سری پیاده سازی خواهند شد به ترتیب ذیل هستند:
1- چگونه Areaهای استاندارد را تبدیل به یک افزونه‌ی مجزا و منتقل شده‌ی به یک اسمبلی دیگر کنیم.
2- چگونه ساختار پایه‌ای را جهت تامین نیازهای هر افزونه جهت تزریق وابستگی‌ها تا ثبت مسیریابی‌ها و امثال آن تدارک ببینیم.
3- چگونه فایل‌های JS ، CSS و همچنین تصاویر ثابت هر افزونه را داخل اسمبلی آن قرار دهیم تا دیگر نیازی به ارائه‌ی مجزای آن‌ها نباشد.
4- چگونه Entity Framework Code-First را با این طراحی یکپارچه کرده و از آن جهت یافتن خودکار مدل‌ها و موجودیت‌های خاص هر افزونه استفاده کنیم؛ به همراه مباحث Migrations خودکار و همچنین پیاده سازی الگوی واحد کار.

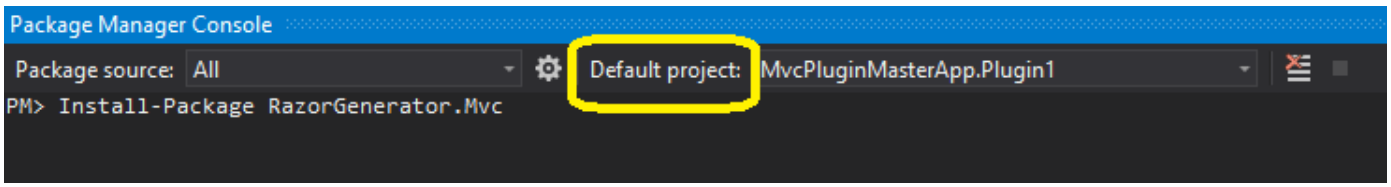
در مطلب جاری، موارد اول و دوم بررسی خواهند شد. پیشنیازهای آن مطالب ذیل هستند:
الف) منظور از یک Area چیست؟
ب) توزیع پروژه‌های ASP.NET MVC بدون ارائه فایل‌های View آن
ج) آشنایی با تزریق وابستگی‌ها در ASP.NET MVC و همچنین اصول طراحی یک سیستم افزونه پذیر به کمک StructureMap
د) آشنایی با رخدادهای Build

تبدیل یک Area به یک افزونه‌ی مستقل

روش‌های زیادی برای خارج کردن Areaهای استاندارد ASP.NET MVC از یک پروژه و قرار دادن آن‌ها در اسمبلی‌های دیگر وجود دارند؛ اما در حال حاضر تنها روشی که نگهداری می‌شود و همچنین اعضای آن همان اعضای تیم نیوگت و ASP.NET MVC هستند، همان روش استفاده از Razor Generator است.
بنابراین ساختار ابتدایی پروژه‌ی افزونه پذیر ما به صورت ذیل خواهد بود:
1) ابتدا افزونه‌ی Razor Generator را نصب کنید.
2) سپس یک پروژه‌ی معمولی ASP.NET MVC را آغاز کنید. در این سری نام MvcPluginMasterApp برای آن در نظر گرفته شده‌است.
3) در ادامه یک پروژه‌ی معمولی دیگر ASP.NET MVC را نیز به پروژه‌ی جاری اضافه کنید. برای مثال نام آن در اینجا MvcPluginMasterApp.Plugin1 تنظیم شده‌است.
4) به پروژه‌ی MvcPluginMasterApp.Plugin1 یک Area جدید و معمولی را به نام NewsArea اضافه کنید.
5) از پروژه‌ی افزونه، تمام پوشه‌های غیر Area را حذف کنید. پوشه‌های Controllers و Models و Views حذف خواهند شد. همچنین فایل global.asax آن‌را نیز حذف کنید. هر افزونه، کنترلرها و Viewهای خود را از طریق Area مرتبط دریافت می‌کند و در این حالت دیگر نیازی به پوشه‌های Controllers و Models و Views واقع شده در ریشه‌ی اصلی پروژه‌ی افزونه نیست.
6) در ادامه کنسول پاور شل نیوگت را باز کرده و دستور ذیل را صادر کنید:

```
PM> Install-Package RazorGenerator.Mvc
```

این دستور را باید یکبار بر روی پروژه‌ی اصلی و یکبار بر روی پروژه‌ی افزونه، اجرا کنید.

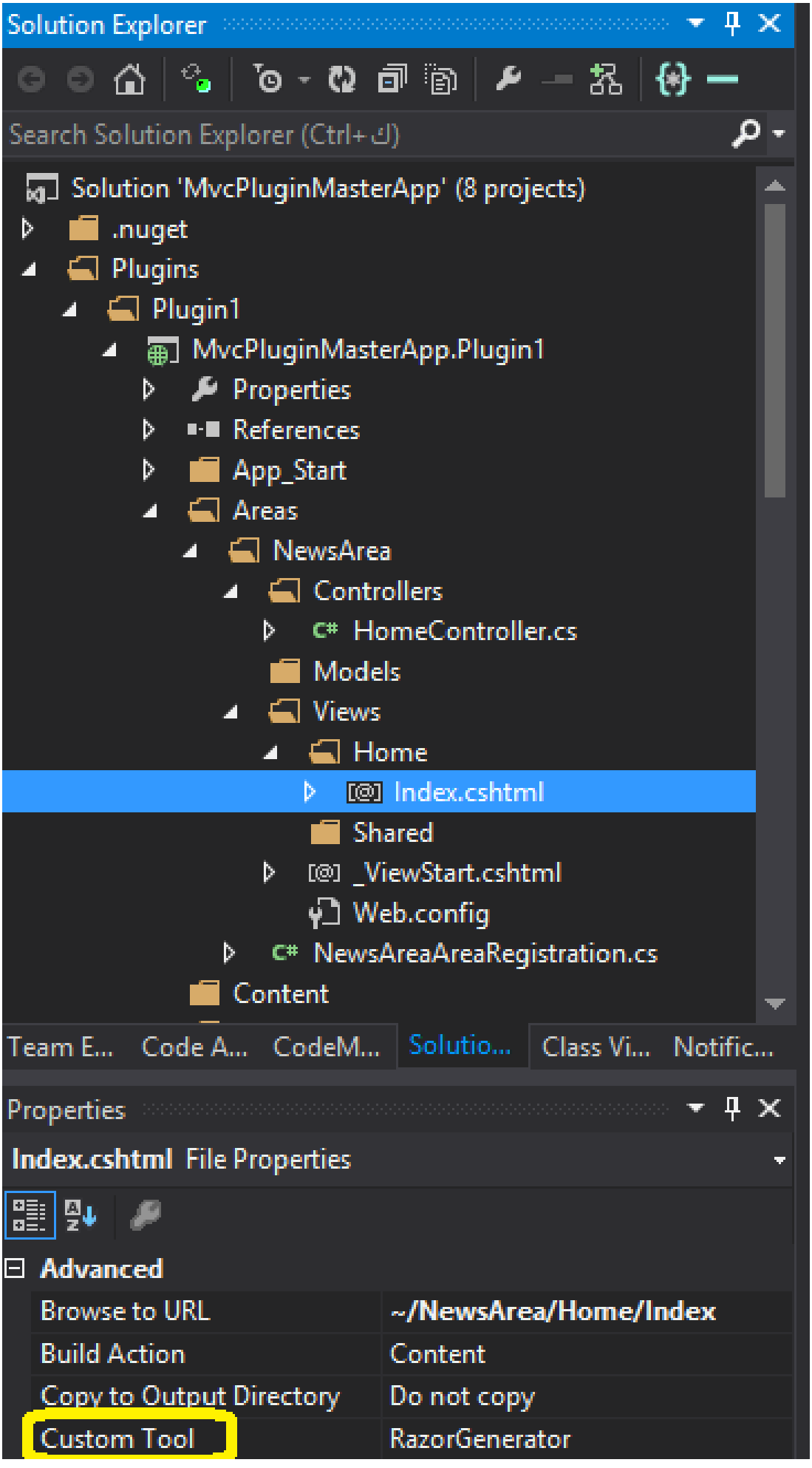


همانطور که در تصویر نیز مشخص شده‌است، برای اجرای دستور نصب RazorGenerator.Mvc نیاز است هربار پروژه‌ی پیش فرض را تغییر دهید.

7) اکنون پس از نصب RazorGenerator.Mvc، نوبت به اجرای آن بر روی هر دو پروژه‌ی اصلی و افزونه است:

```
PM> Enable-RazorGenerator
```

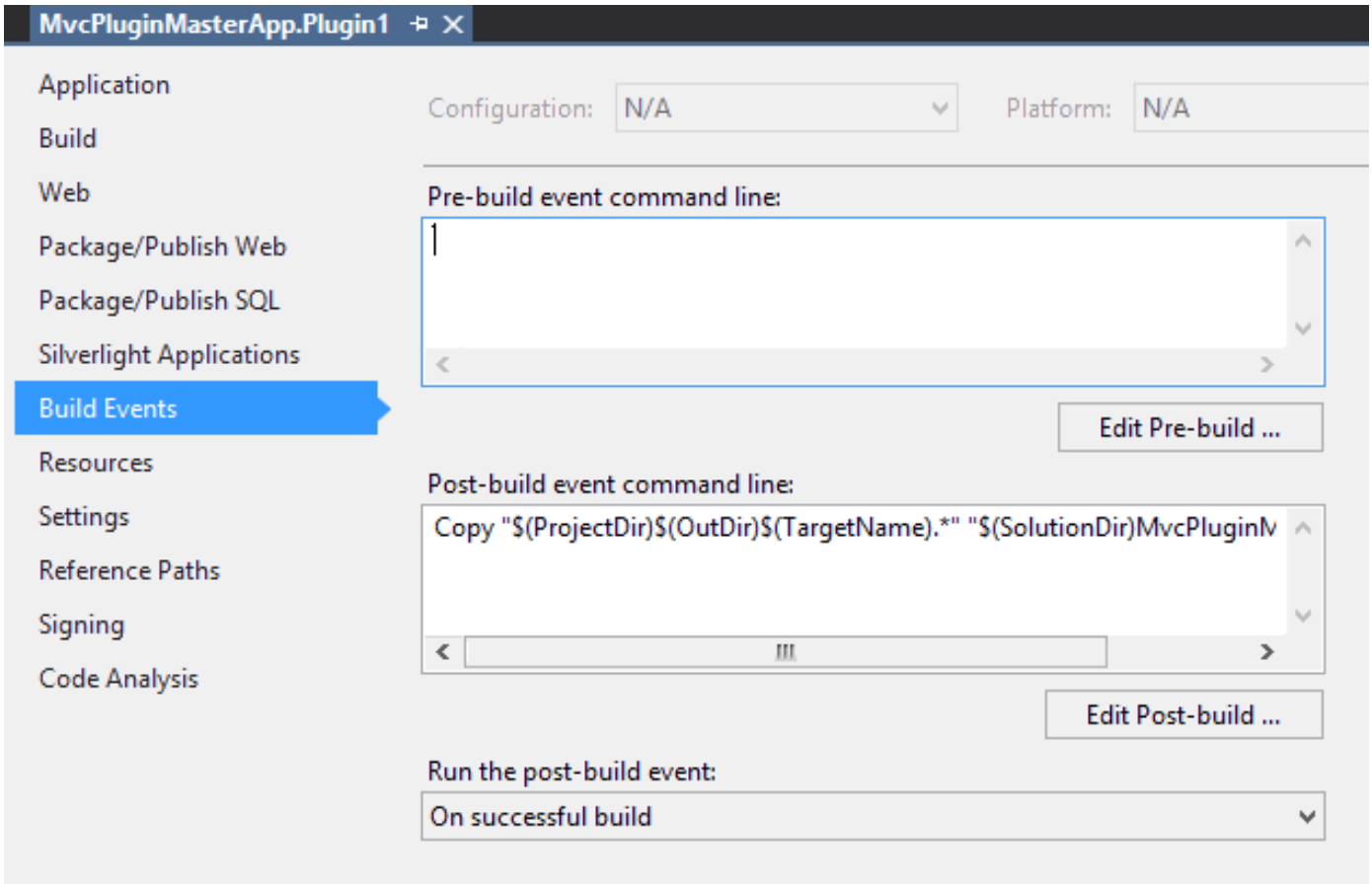
بدیهی است این دستور را نیز باید همانند تصویر فوق، یکبار بر روی پروژه‌ی اصلی و یکبار بر روی پروژه‌ی افزونه اجرا کنید.
همچنین هربار که View جدیدی اضافه می‌شود نیز باید این‌کار را تکرار کنید یا اینکه مطابق شکل زیر، به خواص View جدید مراجعه کرده و Custom tool آن‌را به صورت دستی به RazorGenerator تنظیم نمائید. دستور Enable-RazorGenerator این‌کار را به صورت خودکار انجام می‌دهد.


Solution Explorer
Search Solution Explorer (Ctrl+ک)
Solution 'MvcPluginMasterApp' (8 projects)
.nuget
Plugins
Plugin1
MvcPluginMasterApp.Plugin1
Properties
References
App_Start
Areas
NewsArea
Controllers
HomeController.cs
Models
Views
Home
Index.cshtml
Shared
_ViewStart.cshtml
Web.config
NewsAreaAreaRegistration.cs
Content
Team E...
Code A...
CodeM...
Solutio...
Class Vi...
Notific...
Properties
Index.cshtml File Properties
Advanced
Browse to URL
~/NewsArea/Home/Index
Build Action
Content
Copy to Output Directory
Do not copy
Custom Tool
RazorGenerator

تا اینجا موفق شدیم Viewهای افزونه را داخل فایل dll آن مدفون کنیم. به این ترتیب با کپی کردن افزونه به پوشه‌ی bin پروژه‌ی اصلی، دیگر نیازی به ارائه‌ی فایل‌های View آن نیست و تمام اطلاعات کنترلرها، مدل‌ها و Viewها به صورت یکجا از فایل dll افزونه‌ی ارائه شده خوانده می‌شوند.

کپی کردن خودکار افزونه به پوشه‌ی Bin پروژه‌ی اصلی

پس از اینکه ساختار اصلی کار شکل گرفت، هربار پس از کامپایل افزونه (یا افزونه‌ها)، نیاز است فایل‌های پوشه‌ی bin آن‌را به پوشه‌ی bin پروژه‌ی اصلی کپی کنیم (پروژه‌ی اصلی در این حالت هیچ ارجاع مستقیمی را به افزونه‌ی جدید نخواهد داشت). برای خودکار سازی این کار، به خواص پروژه‌ی افزونه مراجعه کرده و قسمت Build events آن‌را به نحو ذیل تنظیم کنید:



در اینجا دستور ذیل در قسمت Post-build event نوشته شده است:

```
Copy "$(ProjectDir)$(OutDir)$(TargetName).*" "$(SolutionDir)MvcPluginMasterApp\bin\"
```

و سبب خواهد شد تا پس از هر کامپایل موفق، فایل‌های اسمبلی افزونه به پوشه‌ی bin پروژه‌ی MvcPluginMasterApp به صورت خودکار کپی شوند.

تنظیم فضاهای نام کلیه مسیریابی‌های پروژه

در همین حالت اگر پروژه را اجرا کنید، موتور ASP.NET MVC به صورت خودکار اطلاعات افزونه‌ی کپی شده به پوشه‌ی bin را دریافت و به Application domain جاری اعمال می‌کند؛ برای اینکار نیازی به کد نویسی اضافه‌تری نیست و خودکار است. برای آزمایش آن فقط کافی است یک break point را داخل کلاس RazorGeneratorMvcStart افزونه قرار دهید.
اما ... پس از اجرا، بلافاصله پیام تداخل فضاهای نام را دریافت می‌کنید. خطاهای حاصل عنوان می‌کند که در App domain جاری، دو کنترلر Home وجود دارند؛ یکی در پروژه‌ی اصلی و دیگری در پروژه‌ی افزونه و مشخص نیست که مسیریابی‌ها باید به کدامیک ختم شوند.
برای رفع این مشکل، به فایل NewsAreaAreaRegistration.cs پروژه‌ی افزونه مراجعه کرده و مسیریابی آن‌را به نحو ذیل تکمیل کنید تا فضای نام اختصاصی این Area صریحا مشخص گردد.

```
using System.Web.Mvc;

namespace MvcPluginMasterApp.Plugin1.Areas.NewsArea
{
    public class NewsAreaAreaRegistration : AreaRegistration
    {
        public override string AreaName
        {
            get
            {
                return "NewsArea";
            }
        }

        public override void RegisterArea(AreaRegistrationContext context)
        {
            context.MapRoute(
                "NewsArea_default",
                "NewsArea/{controller}/{action}/{id}",
                // تکمیل نام کنترلر پیش فرض
                new { controller = "Home", action = "Index", id = UrlParameter.Optional },
                // مشخص کردن فضای نام مرتبط جهت جلوگیری از تداخل با سایر قسمت‌های برنامه
                namespaces: new[] { string.Format("{0}.Controllers", this.GetType().Namespace) }
            );
        }
    }
}
```

همینکار را باید در پروژه‌ی اصلی و هر پروژه‌ی افزونه‌ی جدیدی نیز تکرار کرد. برای مثال به فایل RouteConfig.cs پروژه‌ی اصلی مراجعه کرده و تنظیم ذیل را اعمال نمائید:

```
using System.Web.Mvc;
using System.Web.Routing;

namespace MvcPluginMasterApp
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional },
                // مشخص کردن فضای نام مرتبط جهت جلوگیری از تداخل با سایر قسمت‌های برنامه
                namespaces: new[] { string.Format("{0}.Controllers", typeof(RouteConfig).Namespace) }
            );
        }
    }
}
```

بدون تنظیم فضاهای نام هر مسیریابی، امکان استفاده‌ی بهینه و بدون خطا از Areaها وجود نخواهد داشت.

طراحی قرارداد پایه افزونه‌ها

تا اینجا با نحوه‌ی تشکیل ساختار هر پروژه‌ی افزونه آشنا شدیم. اما هر افزونه در آینده نیاز به مواردی مانند منوی اختصاصی در منوی اصلی سایت، تنظیمات مسیریابی اختصاصی، تنظیمات EF و امثال آن نیز خواهد داشت. به همین منظور، یک پروژه‌ی class library جدید را به نام MvcPluginMasterApp.PluginsBase آغاز کنید.
سپس قرار داد IPlugin را به نحو ذیل به آن اضافه نمائید:

```
using System;
using System.Reflection;
using System.Web.Optimization;
using System.Web.Routing;
using StructureMap;

namespace MvcPluginMasterApp.PluginsBase
{
    public interface IPlugin
    {
        EfBootstrapper GetEfBootstrapper();
        MenuItem GetMenuItem(RequestContext requestContext);
        void RegisterBundles(BundleCollection bundles);
        void RegisterRoutes(RouteCollection routes);
        void RegisterServices(IContainer container);
    }

    public class EfBootstrapper
    {
        /// <summary>
        /// Assemblies containing EntityTypeConfiguration classes.
        /// </summary>
        public Assembly[] ConfigurationsAssemblies { get; set; }

        /// <summary>
        /// Domain classes.
        /// </summary>
        public Type[] DomainEntities { get; set; }

        /// <summary>
        /// Custom Seed method.
        /// </summary>
        //public Action<IUnitOfWork> DatabaseSeeder { get; set; }
    }

    public class MenuItem
    {
        public string Name { set; get; }
        public string Url { set; get; }
    }
}
```

پروژه‌ی این قرارداد برای کامپایل شدن، نیاز به بسته‌های نیوگت ذیل دارد:

```
PM> install-package EntityFramework
PM> install-package Microsoft.AspNet.Web.Optimization
PM> install-package structuremap.web
```

همچنین باید به صورت دستی، در قسمت ارجاعات پروژه، ارجاعی را به اسمبلی استاندارد System.Web نیز به آن اضافه نمائید.

توضیحات قرار داد IPlugin

از این پس هر افزونه باید دارای کلاسی باشد که از اینترفیس IPlugin مشتق می‌شود. برای مثال فعلا کلاس ذیل را به افزونه‌ی پروژه اضافه نمائید:

```
using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;
using MvcPluginMasterApp.PluginsBase;
using StructureMap;

namespace MvcPluginMasterApp.Plugin1
{
    public class Plugin1 : IPlugin
```

```
    {
        public EfBootstrapper GetEfBootstrapper()
        {
            return null;
        }

        public MenuItem GetMenuItem(RequestContext requestContext)
        {
            return new MenuItem
            {
                Name = "Plugin 1",
                Url = new UrlHelper(requestContext).Action("Index", "Home", new { area = "NewsArea" })
            };
        }

        public void RegisterBundles(BundleCollection bundles)
        {
            //todo: ...
        }

        public void RegisterRoutes(RouteCollection routes)
        {
            //todo: add custom routes.
        }

        public void RegisterServices(IContainer container)
        {
            // todo: add custom services.

            container.Configure(cfg =>
            {
                //cfg.For<INewsService>().Use<EfNewsService>();
            });
        }
    }
}
```

در قسمت جاری فقط از متد GetMenuItem آن استفاده خواهیم کرد. در قسمت‌های بعد، تنظیمات EF، تنظیمات مسیریابی‌ها و Bundling و همچنین ثبت سرویس‌های افزونه را نیز بررسی خواهیم کرد.
برای اینکه هر افزونه در منوی اصلی ظاهر شود، نیاز به یک نام، به همراه آدرسی به صفحه‌ی اصلی آن خواهد داشت. به همین جهت در متد GetMenuItem نحوه‌ی ساخت آدرسی را به اکشن متد Index کنترلر Home واقع در Areaایی به نام NewsArea، مشاهده می‌کنید.

بارگذاری و تشخیص خودکار افزونه‌ها

پس از اینکه هر افزونه دارای کلاسی مشتق شده از قرارداد IPlugin شد، نیاز است آن‌ها را به صورت خودکار یافته و سپس پردازش کنیم. این‌کار را به کتابخانه‌ی StructureMap واگذار خواهیم کرد. برای این منظور پروژه‌ی جدیدی را به نام MvcPluginMasterApp.IoCConfig آغاز کرده و سپس تنظیمات آن‌را به نحو ذیل تغییر دهید:

```
using System;
using System.IO;
using System.Threading;
using System.Web;
using MvcPluginMasterApp.PluginsBase;
using StructureMap;
using StructureMap.Graph;

namespace MvcPluginMasterApp.IoCConfig
{
    public static class SmObjectFactory
    {
        private static readonly Lazy<Container> _containerBuilder =
            new Lazy<Container>(defaultContainer, LazyThreadSafetyMode.ExecutionAndPublication);

        public static IContainer Container
        {
            get { return _containerBuilder.Value; }
        }

        private static Container defaultContainer()
```

```
        {
            return new Container(cfg =>
            {
                cfg.Scan(scanner =>
                {
                    scanner.AssembliesFromPath(
                        path: Path.Combine(HttpRuntime.AppDomainAppPath, "bin"),
                            // یک اسمبلی نباید دوبار بارگذاری شود //
                        assemblyFilter: assembly =>
                        {
                            return !assembly.FullName.Equals(typeof(IPlugin).Assembly.FullName);
                        });

                    scanner.WithDefaultConventions(); //Connects 'IName' interface to 'Name' class
automatically.
                    scanner.AddAllTypesOf<IPlugin>().NameBy(item => item.FullName);
                });
            });
        }
    }
}
```

این پروژه‌ی class library جدید برای کامپایل شدن نیاز به بسته‌های نیوگت ذیل دارد:

```
PM> install-package EntityFramework
PM> install-package structuremap.web
```

همچنین باید به صورت دستی، در قسمت ارجاعات پروژه، ارجاعی را به اسمبلی استاندارد System.Web نیز به آن اضافه نمائید.

کاری که در کلاس SmObjectFactory انجام شده، بسیار ساده است. مسیر پوشه‌ی Bin پروژه‌ی اصلی به structuremap معرفی شده‌است. سپس به آن گفته‌ایم که تنها اسمبلی‌هایی را که دارای اینترفیس IPlugin هستند، به صورت خودکار بارگذاری کن. در ادامه تمام نوع‌های IPlugin را نیز به صورت خودکار یافته و در مخزن تنظیمات خود، اضافه کن.

تامین نیازهای مسیریابی و Bundling هر افزونه به صورت خودکار

در ادامه به پروژه‌ی اصلی مراجعه کرده و در پوشه‌ی App_Start آن کلاس ذیل را اضافه کنید:

```
using System.Linq;
using System.Web.Optimization;
using System.Web.Routing;
using MvcPluginMasterApp;
using MvcPluginMasterApp.IoCConfig;
using MvcPluginMasterApp.PluginsBase;

[assembly: WebActivatorEx.PostApplicationStartMethod(typeof(PluginsStart), "Start")]

namespace MvcPluginMasterApp
{
    public static class PluginsStart
    {
        public static void Start()
        {
            var plugins = SmObjectFactory.Container.GetAllInstances<IPlugin>().ToList();
            foreach (var plugin in plugins)
            {
                plugin.RegisterServices(SmObjectFactory.Container);
                plugin.RegisterRoutes(RouteTable.Routes);
                plugin.RegisterBundles(BundleTable.Bundles);
            }
        }
    }
}
```

بدیهی است در این حالت نیاز است ارجاعی را به پروژه‌ی MvcPluginMasterApp.PluginsBase به پروژه‌ی اصلی اضافه کنیم.
دراینجا با استفاده از کتابخانه‌ای به نام WebActivatorEx (که باز هم توسط نویسندگان اصلی Razor Generator تهیه شده‌است)، یک متد PostApplicationStartMethod سفارشی را تعریف کرده‌ایم.

مزیت استفاده از اینکار این است که فایل Global.asax.cs برنامه شلوغ نخواهد شد. در غیر اینصورت باید تمام این کدها را در انتهای متد Application_Start قرار می‌دادیم.
در اینجا با استفاده از structuremap، تمام افزونه‌های موجود به صورت خودکار بررسی شده و سپس پیشنیازهای مسیریابی و Bundling و همچنین تنظیمات IoC Container مورد نیاز آن‌ها به هر افزونه به صورت مستقل، تزریق خواهد شد.

اضافه کردن منوهای خودکار افزونه‌ها به پروژه‌ی اصلی

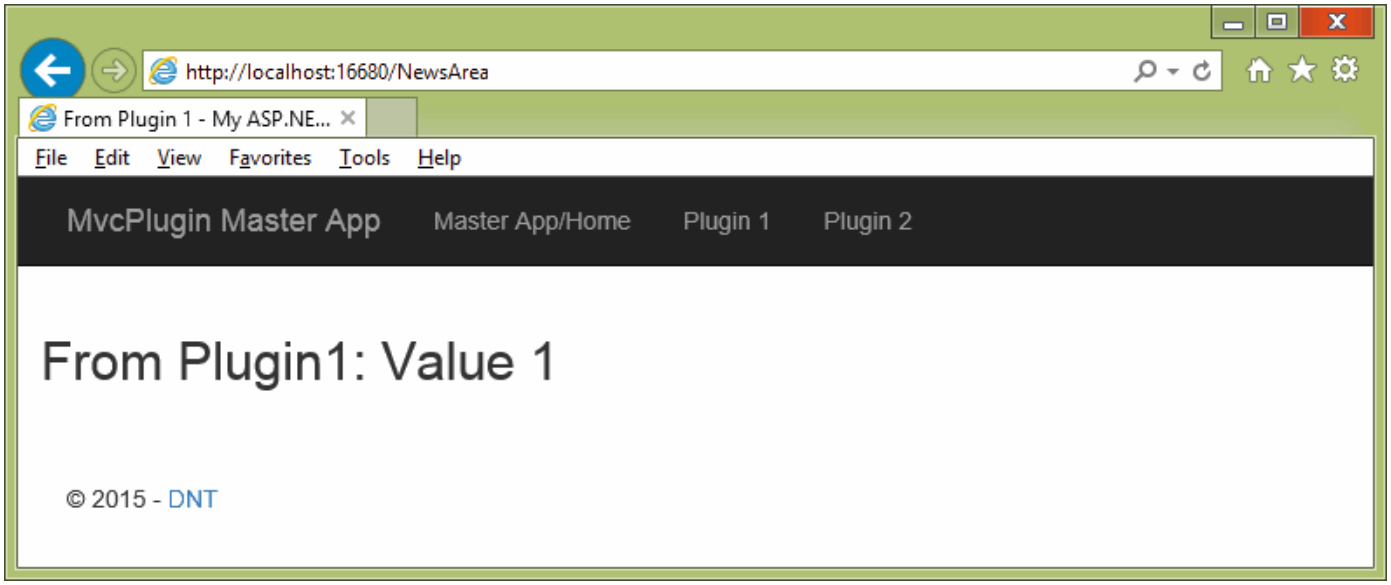
پس از اینکه کار پردازش اولیه‌ی IPluginها به پایان رسید، اکنون نوبت به نمایش آدرس اختصاصی هر افزونه در منوی اصلی سایت است. برای این منظور فایل جدیدی را به نام _PluginsMenu.cshtml، در پوشه‌ی shared پروژه‌ی اصلی اضافه کنید؛ با این محتوا:

```
@using MvcPluginMasterApp.IoCConfig
@using MvcPluginMasterApp.PluginsBase
@{
    var plugins = SmObjectFactory.Container.GetAllInstances<IPlugin>().ToList();
}

@foreach (var plugin in plugins)
{
    var menuItem = plugin.GetMenuItem(this.Request.RequestContext);
    <li>
        <a href="@menuItem.Url">@menuItem.Name</a>
    </li>
}
```

در اینجا تمام افزونه‌ها به کمک structuremap یافت شده و سپس آیتم‌های منوی آن‌ها به صورت خودکار دریافت و اضافه می‌شوند.
سپس به فایل _Layout.cshtml پروژه‌ی اصلی مراجعه و توسط فراخوانی Html.RenderPartial، آن‌را در بین سایر آیتم‌های منوی اصلی اضافه می‌کنیم:

```
<div class="navbar navbar-inverse navbar-fixed-top">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-
collapse">
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            @Html.ActionLink("MvcPlugin Master App", "Index", "Home", new { area = "" }, new { @class =
"navbar-brand" })
        </div>
        <div class="navbar-collapse collapse">
            <ul class="nav navbar-nav">
                <li>@Html.ActionLink("Master App/Home", "Index", "Home", new {area = ""}, null)</li>
                @{ Html.RenderPartial("_PluginsMenu"); }
            </ul>
        </div>
    </div>
</div>
```

اکنون اگر پروژه را اجرا کنیم، یک چنین شکلی را خواهد داشت:



بنابراین به صورت خلاصه

1) هر افزونه، یک پروژه‌ی کامل ASP.NET MVC است که پوشه‌های ریشه‌ی اصلی آن حذف شده‌اند و اطلاعات آن توسط یک Area جدید تامین می‌شوند.
2) تنظیم فضای نام مسیریابی‌های تمام پروژه‌ها را فراموش نکنید. در غیر اینصورت شاهد تداخل پردازش کنترلرهای هم نام خواهید بود.
3) جهت سهولت کار، می‌توان فایل‌های bin هر افزونه را توسط رخداد post-build، به پوشه‌ی bin پروژه‌ی اصلی کپی کرد.
4) Viewهای هر افزونه توسط Razor Generator در فایل dll آن مدفون خواهند شد.
5) هر افزونه باید دارای کلاسی باشد که اینترفیس IPlugin را پیاده سازی می‌کند. از این اینترفیس برای ثبت اطلاعات هر افزونه یا دریافت اطلاعات سفارشی از آن کمک می‌گیریم.
6) با استفاده از استراکچرمپ و قرارداد IPlugin، منوهای هر افزونه را به صورت خودکار یافته و سپس به فایل layout اصلی اضافه می‌کنیم.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

MvcPluginMasterApp-Part1.zip

نظرات خوانندگان

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۴/۰۱/۲۷ ۱۶:۲۱

یک سوال، هنگام حذف افزونه با توجه به اینکه ممکنه کاربری در حال کار با بخش‌های مختلف اون باشه چه اتفاقی برای حذف ارجاع‌های اون به برنامه می‌افتد؟ آیا اجازه حذف لازم است؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۷ ۱۶:۲۵

- برنامه‌ی اصلی ارجاع مستقیمی را به هیچ افزونه‌ای ندارد.
+ هر نوع تغییری در پوشه‌ی bin برنامه سبب ری استارت آن خواهد شد. بنابراین اگر افزونه‌ای اضافه شود، برنامه به صورت خودکار ری استارت شده و بلافاصله افزونه‌ی جدید، قابل استفاده خواهد بود. اگر فایل افزونه‌ای از پوشه‌ی bin حذف شود، باز هم سبب ری استارت برنامه و بارگذاری خودکار منوها و محاسبه‌ی مجدد آن‌ها می‌گردد که اینبار دیگر شامل اطلاعات افزونه‌ی حذف شده نیست.

نویسنده: حامد67
تاریخ: ۱۳۹۴/۰۱/۲۷ ۲۱:۴۱

سلام؛ یه سوال امنیتی، آیا راهکاری دارید که کسی به طور غیر مجاز برای برنامه پلاگین ننویسه منظور این هستش که فردی که پلاگین رو نوشته فقط با تایید بتونه فعالش کنه و از لحاظ امنیتی قابل چک باشه و بدون تایید اجرایی نشه چون من نگران هستم فردی پلاگین بنویسد و عمدا یا غیر عمد پلاگینی توسعه دهد که اطلاعات و روند فعالیت برنامه را جاسوسی کند
خودم این ذهنیت رو دارم که هش کد هر پلاگین باید توسط مدیر تایید بشه و سپس قابل اجرا باشه تا کسی نتونه بعدا پلاگین را تغییر بده و امنیت سیستم را به خطر بنداره
در کل ملاحظات امنیتی پاگین‌ها را چگونه در نظر بگیریم ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۷ ۲۲:۵

از مطلب « تهیه XML امضاء شده جهت تولید مجوز استفاده از برنامه » ایده بگیرید. یک متد GetLicense به اینترفیس IPlugin اضافه کنید و در آن مجوز ارائه شده توسط افزونه را در برنامه‌ی اصلی بررسی کنید (در کلاس PluginsStart و همچنین فایل _PluginsMenu.cshtml). فقط کسانی می‌توانند «XML امضاء شده» تولید کنند که دسترسی به کلیدهای خصوصی و امن شما را داشته باشند.

نویسنده: میثم99
تاریخ: ۱۳۹۴/۰۱/۳۰ ۱۵:۵۳

سلام؛ اگر بخواهیم مسیر یابی پروژه را به attribute routing تغییر بدهیم چه کارهایی باید انجام دهیم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۳۰ ۱۸:۲۶

از این مطالب تکمیلی استفاده کنید:
- « قابلیت Attribute Routing در ASP.NET MVC 5 »
- « Attribute Routing در ASP.NET MVC 5 »

عنوان: **طراحی افزونه پذیر با ASP.NET MVC 4.x/5.x - قسمت دوم**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۸ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Plugin

در مطلب « طراحی افزونه پذیر با ASP.NET MVC 4.x/5.x - قسمت اول » با ساختار کلی یک پروژه‌ی افزونه‌ی پذیر ASP.NET MVC آشنا شدیم. پس از راه اندازی آن و مدتی کار کردن با این نوع پروژه‌ها، این سؤال پیش خواهد آمد که ... خوب، اگر هر افزونه تصاویر یا فایل‌های CSS و JS اختصاصی خودش را بخواهد داشته باشد، چطور؟ موارد عمومی مانند بوت استرپ و جی‌کوئری را می‌توان در پروژه‌ی پایه قرار داد تا تمام افزونه‌ها به صورت یکسانی از آن‌ها استفاده کنند، اما هدف، ماژولار شدن برنامه است و جدا کردن فایل‌های ویژه‌ی هر پروژه، از پروژه‌ای دیگر و همچنین بالا بردن سهولت کار تیمی، با شکستن اجزای یک پروژه به صورت افزونه‌هایی مختلف، بین اعضای یک تیم. در این قسمت نحوه‌ی مدفون سازی انواع فایل‌های استاتیک افزونه‌ها را درون فایل‌های DLL آن‌ها بررسی خواهیم کرد. به این ترتیب دیگر نیازی به ارائه‌ی مجزای آن‌ها و یا کپی کردن آن‌ها در پوشه‌های پروژه‌ی اصلی نخواهد بود.

مدفون سازی فایل‌های CSS و JS هر افزونه درون فایل DLL آن

به solution جاری، یک class library جدید را به نام MvcPluginMasterApp.Common اضافه کنید. از آن جهت قرار دادن کلاس‌های عمومی و مشترک بین افزونه‌ها استفاده خواهیم کرد. برای مثال قصد نداریم کلاس‌های سفارشی و عمومی ذیل را هربار به صورت مستقیم در افزونه‌ای جدید کپی کنیم. کتابخانه‌ی Common، امکان استفاده‌ی مجدد از یک سری کدهای تکراری را در بین افزونه‌ها میسر می‌کند.
این پروژه برای کامپایل شدن نیاز به بسته‌ی نیوگت ذیل دارد:

```
PM> install-package Microsoft.AspNet.Web.Optimization
```

همچنین باید به صورت دستی، در قسمت ارجاعات پروژه، ارجاعی را به اسمبلی استاندارد System.Web نیز به آن اضافه نمائید.
پس از این مقدمات، کلاس ذیل را به این پروژه‌ی class library جدید اضافه کنید:

```
using System.Collections.Generic;
using System.IO;
using System.Reflection;
using System.Text;
using System.Web.Optimization;

namespace MvcPluginMasterApp.Common.WebToolkit
{
    public class EmbeddedResourceTransform : IBundleTransform
    {
        private readonly IList<string> _resourceFiles;
        private readonly string _contentType;
        private readonly Assembly _assembly;

        public EmbeddedResourceTransform(IList<string> resourceFiles, string contentType, Assembly
assembly)
        {
            _resourceFiles = resourceFiles;
            _contentType = contentType;
            _assembly = assembly;
        }

        public void Process(BundleContext context, BundleResponse response)
        {
            var result = new StringBuilder();

            foreach (var resource in _resourceFiles)
            {
                using (var stream = _assembly.GetManifestResourceStream(resource))
                {
                    if (stream == null)
                    {
                        throw new KeyNotFoundException(string.Format("Embedded resource key: '{0}' not
found in the '{1}' assembly.", resource, _assembly.FullName));
                    }
```

```
                    using (var reader = new StreamReader(stream))
                    {
                        result.Append(reader.ReadToEnd());
                    }
                }
            }

            response.ContentType = _contentType;
            response.Content = result.ToString();
        }
    }
}
```

اگر با سیستم bundling & minification کار کرده باشید، با تعاریفی مانند new Bundle("~/Plugin1/Scripts") آشنا هستید. سازنده‌ی کلاس Bundle، پارامتر دومی را نیز می‌پذیرد که از نوع IBundleTransform است. با پیاده سازی اینترفیس IBundleTransform می‌توان محل ارائه‌ی فایل‌های استاتیک CSS و JS را بجای فایل سیستم متداول و پیش فرض، به منابع مدفون شده‌ی در اسمبلی جاری هدایت و تنظیم کرد.

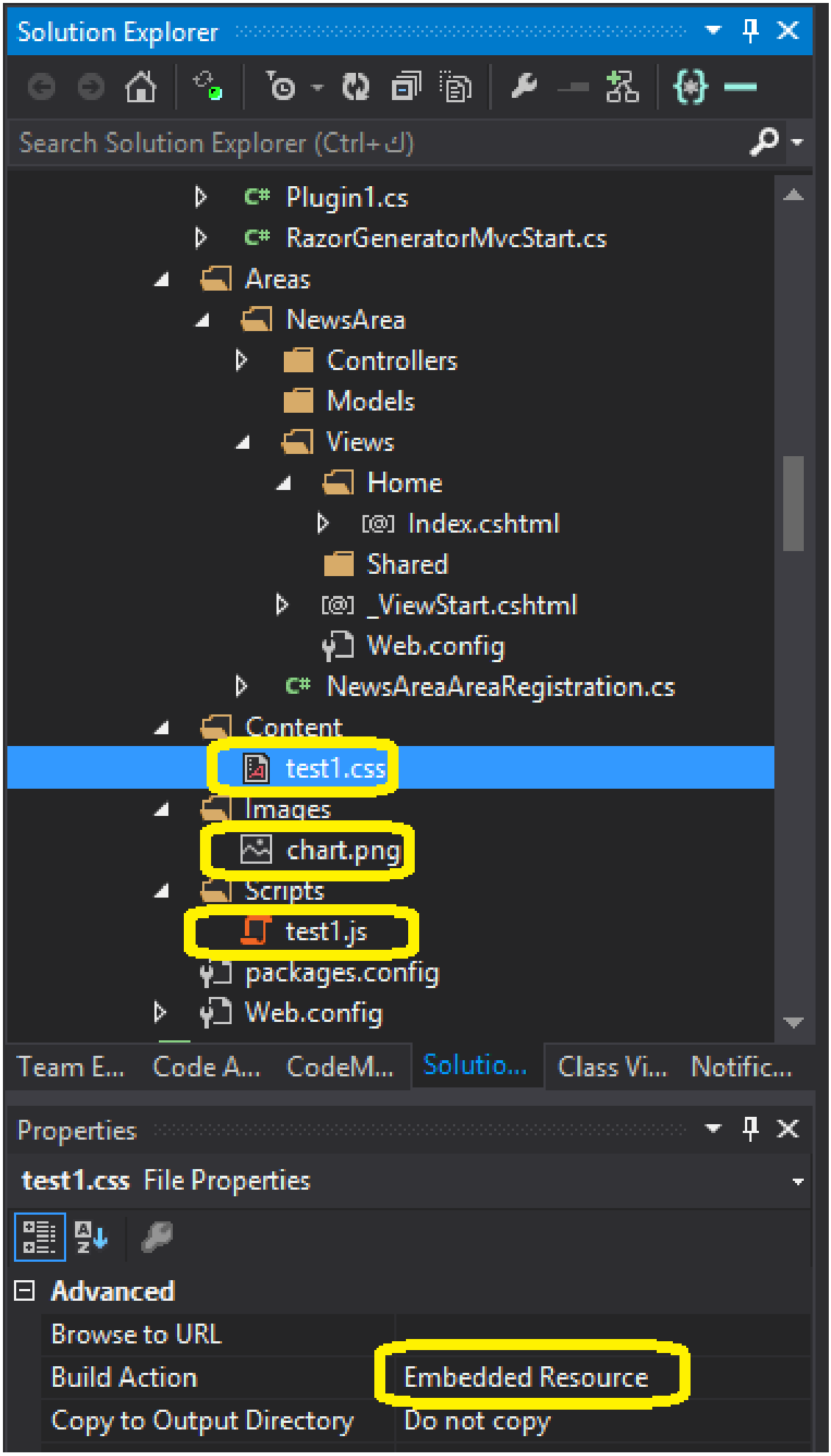
کلاس فوق در اسمبلی معرفی شده به آن، توسط متد GetManifestResourceStream به دنبال فایل‌ها و منابع مدفون شده گشته و سپس محتوای آن‌ها را بازگشت می‌دهد.

اکنون برای استفاده‌ی از آن، به پروژه‌ی MvcPluginMasterApp.Plugin1 مراجعه کرده و ارجاعی را به پروژه‌ی MvcPluginMasterApp.Common فوق اضافه نمائید. سپس در فایل Plugin1.cs، متد RegisterBundles آن‌را به نحو ذیل تکمیل کنید:

```
namespace MvcPluginMasterApp.Plugin1
{
    public class Plugin1 : IPlugin
    {
        public EfBootstrapper GetEfBootstrapper()
        {
            return null;
        }

        public MenuItem GetMenuItem(RequestContext requestContext)
        {
            return new MenuItem
            {
                Name = "Plugin 1",
                Url = new UrlHelper(requestContext).Action("Index", "Home", new { area = "NewsArea" })
            };
        }

        public void RegisterBundles(BundleCollection bundles)
        {
            var executingAssembly = Assembly.GetExecutingAssembly();
            // Mostly the default namespace and assembly name are the same
            var assemblyNameSpace = executingAssembly.GetName().Name;
            var scriptsBundle = new Bundle("~/Plugin1/Scripts",
                new EmbeddedResourceTransform(new List<string>
                {
                    assemblyNameSpace + ".Scripts.test1.js"
                }, "application/javascript", executingAssembly));
            if (!HttpContext.Current.IsDebuggingEnabled)
            {
                scriptsBundle.Transforms.Add(new JsMinify());
            }
            bundles.Add(scriptsBundle);
            var cssBundle = new Bundle("~/Plugin1/Content",
                new EmbeddedResourceTransform(new List<string>
                {
                    assemblyNameSpace + ".Content.test1.css"
                }, "text/css", executingAssembly));
            if (!HttpContext.Current.IsDebuggingEnabled)
            {
                cssBundle.Transforms.Add(new CssMinify());
            }
            bundles.Add(cssBundle);
            BundleTable.EnableOptimizations = true;
        }

        public void RegisterRoutes(RouteCollection routes)
        {
        }
```

```
        public void RegisterServices(IContainer container)
        {
        }
    }
}
```

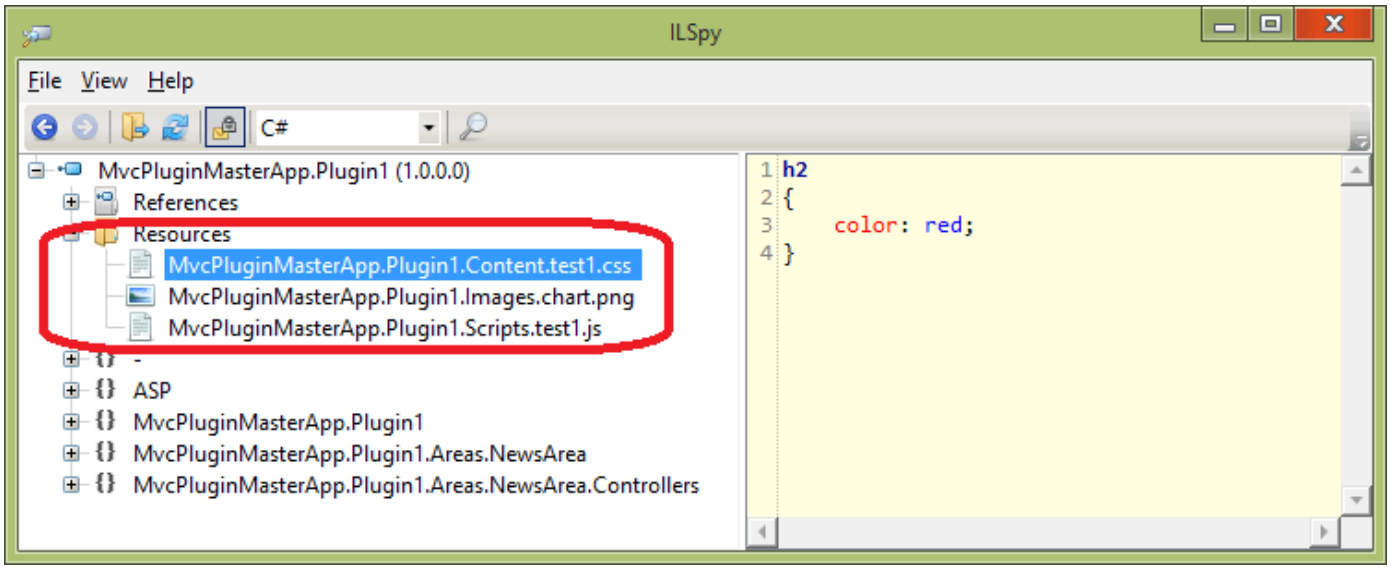
در اینجا نحوه‌ی کار با کلاس سفارشی EmbeddedResourceTransform را مشاهده می‌کنید. ابتدا فایل‌های js و سپس فایل‌های css برنامه به سیستم Bundling برنامه اضافه شده‌اند.
این فایل‌ها به صورت ذیل در پروژه تعریف گردیده‌اند:


Solution Explorer
Search Solution Explorer (Ctrl+ک)
Plugin1.cs
RazorGeneratorMvcStart.cs
Areas
NewsArea
Controllers
Models
Views
Home
Index.cshtml
Shared
_ViewStart.cshtml
Web.config
NewsAreaAreaRegistration.cs
Content
test1.css
Images
chart.png
Scripts
test1.js
packages.config
Web.config
Team E...
Code A...
CodeM...
Solutio...
Class Vi...
Notific...
Properties
test1.css File Properties
Advanced
Browse to URL
Build Action
Embedded Resource
Copy to Output Directory
Do not copy

همانطور که مشاهده می‌کنید، باید به خواص هر کدام مراجعه کرد و سپس Build action آن‌ها را به embedded resource تغییر داد، تا در حین کامپایل، به صورت خودکار در قسمت منابع اسمبلی ذخیره شوند.

یک نکته‌ی مهم

اینبار برای مسیردهی منابع، باید بجای / فایل سیستم، از «نقطه» استفاده کرد. زیرا منابع با نام‌هایی مانند namespace.folder.name در قسمت resources یک اسمبلی ذخیره می‌شوند:



مدفون سازی تصاویر ثابت هر افزونه درون فایل DLL آن

مجددا به اسمبلی مشترک MvcPluginMasterApp.Common مراجعه کرده و اینبار کلاس جدید ذیل را به آن اضافه کنید:

```
using System;
using System.Collections.Generic;
using System.Reflection;
using System.Web;
using System.Web.Routing;

namespace MvcPluginMasterApp.Common.WebToolkit
{
    public class EmbeddedResourceRouteHandler : IRouteHandler
    {
        private readonly Assembly _assembly;
        private readonly string _resourcePath;
        private readonly TimeSpan _cacheDuration;

        public EmbeddedResourceRouteHandler(Assembly assembly, string resourcePath, TimeSpan
cacheDuration)
        {
            _assembly = assembly;
            _resourcePath = resourcePath;
            _cacheDuration = cacheDuration;
        }

        IHttpHandler IRouteHandler.GetHttpHandler(RequestContext requestContext)
        {
            return new EmbeddedResourceHttpHandler(requestContext.RouteData, _assembly, _resourcePath,
_cacheDuration);
        }
    }
```

```
    public class EmbeddedResourceHttpHandler : IHttpHandler
    {
        private readonly RouteData _routeData;
        private readonly Assembly _assembly;
        private readonly string _resourcePath;
        private readonly TimeSpan _cacheDuration;

        public EmbeddedResourceHttpHandler(
            RouteData routeData, Assembly assembly, string resourcePath, TimeSpan cacheDuration)
        {
            _routeData = routeData;
            _assembly = assembly;
            _resourcePath = resourcePath;
            _cacheDuration = cacheDuration;
        }

        public bool IsReusable
        {
            get { return false; }
        }

        public void ProcessRequest(HttpContext context)
        {
            var routeDataValues = _routeData.Values;
            var fileName = routeDataValues["file"].ToString();
            var fileExtension = routeDataValues["extension"].ToString();

            var manifestResourceName = string.Format("{0}.{1}.{2}", _resourcePath, fileName,
fileExtension);
            var stream = _assembly.GetManifestResourceStream(manifestResourceName);
            if (stream == null)
            {
                throw new KeyNotFoundException(string.Format("Embedded resource key: '{0}' not found in
the '{1}' assembly.", manifestResourceName, _assembly.FullName));
            }

            context.Response.Clear();
            context.Response.ContentType = "application/octet-stream";
            cacheIt(context.Response, _cacheDuration);
            stream.CopyTo(context.Response.OutputStream);
        }

        private static void cacheIt(HttpResponse response, TimeSpan duration)
        {
            var cache = response.Cache;

            var maxAgeField = cache.GetType().GetField("_maxAge", BindingFlags.Instance |
BindingFlags.NonPublic);
            if (maxAgeField != null) maxAgeField.SetValue(cache, duration);

            cache.SetCacheability(HttpCacheability.Public);
            cache.SetExpires(DateTime.Now.Add(duration));
            cache.SetMaxAge(duration);
            cache.AppendCacheExtension("must-revalidate, proxy-revalidate");
        }
    }
}
```

تصاویر پروژه‌ی افزونه نیز به صورت embedded resource در اسمبلی آن قرار خواهند گرفت. به همین جهت باید سیستم مسیریابی را پس درخواست رسیده‌ی جهت نمایش تصاویر، به منابع ذخیره شده‌ی در اسمبلی آن هدایت نمود. اینکار را با پیاده سازی یک IRouteHandler سفارشی، می‌توان به نحو فوق مدیریت کرد.

این IRouteHandler، نام و پسوند فایل را دریافت کرده و سپس به قسمت منابع اسمبلی رجوع، فایل مرتبط را استخراج و سپس بازگشت می‌دهد. همچنین برای کاهش سربار سیستم، امکان کش شدن منابع استاتیک نیز در آن درنظر گرفته شده‌است و هدرهای خاص caching را به صورت خودکار اضافه می‌کند.

سیستم bundling نیز هدرهای کش کردن را به صورت خودکار و توکار اضافه می‌کند.

اکنون به تعاریف Plugin1 مراجعه کنید و سپس این IRouteHandler سفارشی را به نحو ذیل به آن معرفی نمائید:

```
namespace MvcPluginMasterApp.Plugin1
{
    public class Plugin1 : IPlugin
    {
```

```
        public void RegisterRoutes(RouteCollection routes)
        {
            //todo: add custom routes.

            var assembly = Assembly.GetExecutingAssembly();
            // Mostly the default namespace and assembly name are the same
            var nameSpace = assembly.GetName().Name;
            var resourcePath = string.Format("{0}.Images", nameSpace);

            routes.Insert(0,
                new Route("NewsArea/Images/{file}.{extension}",
                    new RouteValueDictionary(new { }),
                    new RouteValueDictionary(new { extension = "png|jpg" }),
                    new EmbeddedResourceRouteHandler(assembly, resourcePath, cacheDuration:
TimeSpan.FromDays(30))
                ));
        }
    }
}
```

در مسیریابی تعریف شده، تمام درخواست‌های رسیده‌ی به مسیر /Images **NewsArea** به EmbeddedResourceRouteHandler هدایت می‌شوند.
مطابق تعریف آن، file و extension به صورت خودکار جدا شده و توسط routeData.Values در متد ProcessRequest کلاس EmbeddedResourceHttpHandler قابل دسترسی خواهند شد.
پسوندهایی که توسط آن بررسی می‌شوند از نوع png یا jpg تعریف شده‌اند. همچنین مدت زمان کش کردن هر منبع استاتیک تصویری به یک ماه تنظیم شده‌است.

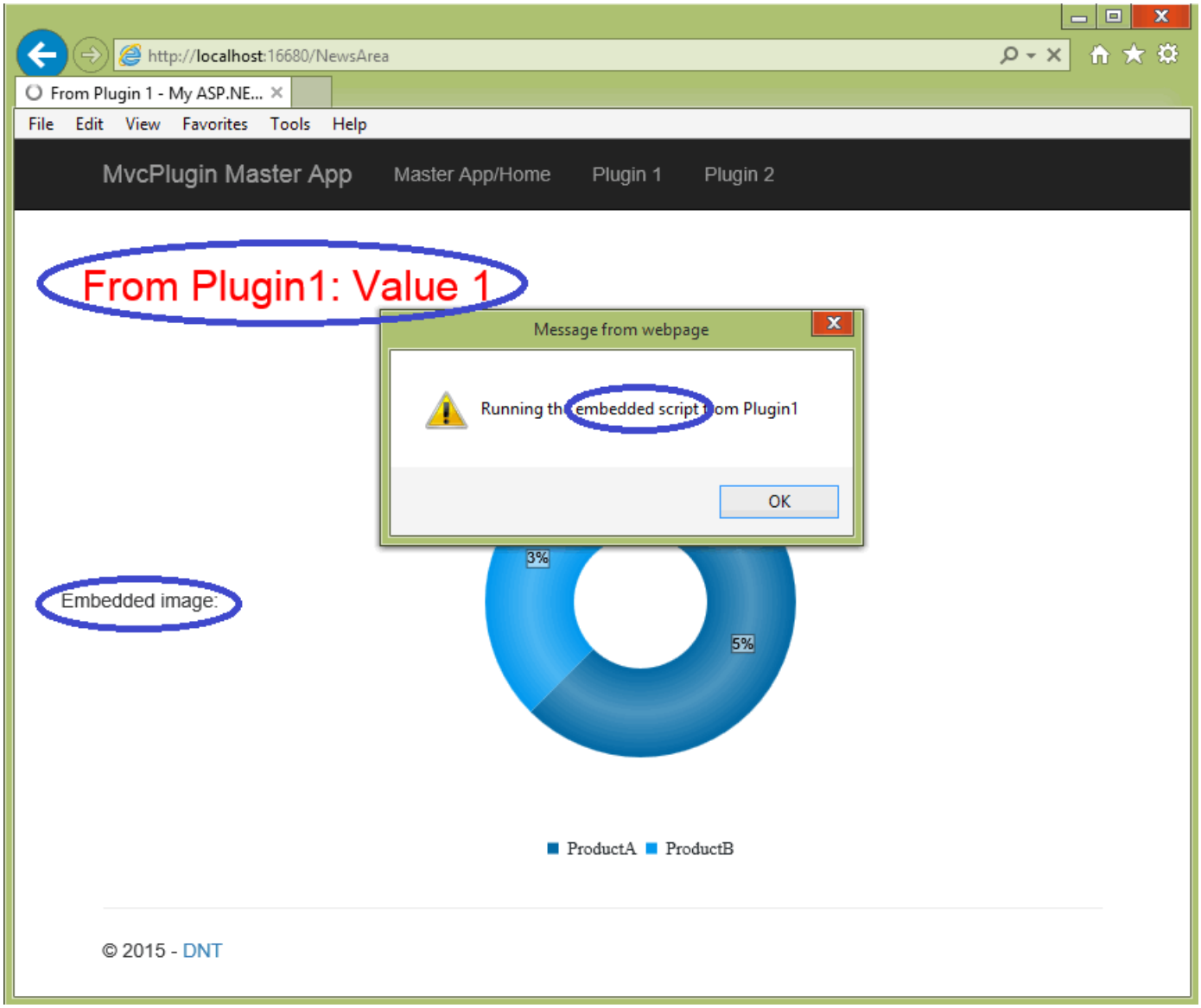
استفاده‌ی نهایی از تنظیمات فوق در یک View افزونه

پس از اینکه تصاویر و فایل‌های css و js را به صورت embedded resource تعریف کردیم و همچنین تنظیمات مسیریابی و bundling خاص آن‌ها را نیز مشخص نمودیم، اکنون نوبت به استفاده‌ی از آن‌ها در یک View است:

```
@{
    ViewBag.Title = "From Plugin 1";
}
@Styles.Render("~/Plugin1/Content")

<h2>@ViewBag.Message</h2>

<div class="row">
    Embedded image:
    <img src="@Url.Content("~/NewsArea/Images/chart.png")" alt="clock" />
</div>

@section scripts
{
    @Scripts.Render("~/Plugin1/Scripts")
}
```

در اینجا نحوه‌ی تعریف فایل‌های CSS و JS ارائه شده‌ی توسط سیستم Bundling را مشاهده می‌کنید.
همچنین مسیر تصویر مشخص شده‌ی در آن، اینبار یک **NewsArea** اضافه‌تر دارد. فایل اصلی تصویر، در مسیر Images/chart.png قرار گرفته‌است اما می‌خواهیم این درخواست‌ها را به مسیریابی جدید { extension}.{file}/Images/ **NewsArea** هدایت کنیم.
بنابراین نیاز است به این نکته نیز دقت داشت.

اینبار اگر برنامه را اجرا کنیم، می‌توان به سه نکته در آن دقت داشت:



الف) alert اجرا شده از فایل js مدفون شده خوانده شده‌است.
ب) رنگ قرمز متن (تگ h2) از فایل css مدفون شده، گرفته شده‌است.
ج) تصویر نمایش داده شده، همان تصویر مدفون شده‌ی در فایل DLL برنامه است.
و هیچکدام از این فایل‌ها، به پوشه‌های پروژه‌ی اصلی برنامه، کپی نشده‌اند.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

MvcPluginMasterApp-Part2.zip

نظرات خوانندگان

نویسنده: رضا ح
تاریخ: ۱۳۹۴/۰۱/۲۸ ۱۵:۲۸

سلام.
یه مشکلی برای من پیش اومده که هنگام استفاده از روتر در افزونه وقتی Home رو می‌زنم به Home پروژه می‌ره نه افزونه و وقتی یه اسم دیگه برای کنترلر افزونه می‌زنم خطا می‌ده که نتونسته ویوها رو پیدا کنه. namespaceهای روترها رو هم زدم ولی کار نمی‌کنه.
ممنون می‌شم اگه کمک کنین

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۸ ۱۶:۱۰

در قسمت اول ، لینک به منوی Area جاری به این صورت تعریف شد:

```
public MenuItem GetMenuItem(RequestContext requestContext)
        {
            return new MenuItem
            {
                Name = "Plugin 1",
                Url = new UrlHelper(requestContext).Action("Index", "Home", new { area = "NewsArea" })
            };
        }
```

در این لینک، ذکر نام صحیح area جاری، در آدرس ساخته شده الزامی است. اساسا کلیه لینک‌های ختم به هر area در ASP.NET MVC باید دارای قسمت الزامی {"......name....." = area new} باشند.
همین نکته در پلاگین دوم هم بکار رفته‌است ({"**ArticlesArea**" = area new }).

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۱/۲۸ ۱۹:۵۶

سلام،
خیلی ممنون واقعاً از خوندن این سه قسمت لذت بردم. خیلی ممنون
طبق مطالب گفته شده پروژه رو ایجاد کردم اما هنگام اجرا استثناء FileNotFoundException صادر میشه:

```
at MvcPluginMasterApp.Plugin1.Plugin1.RegisterRoutes(RouteCollection routes)
   at MvcPluginMasterApp.PluginsStart.Start() in c:\Users\Sirwan-Afifi\Desktop\MvcPluginMasterApp-
Part2\MvcPluginMasterApp\App_Start\PluginsStart.cs:line 20
```

کدهای این قسمت هم رو دانلود و اجرا کردم، باز هم همین خطا رو دریافت کردم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۸ ۲۰:۱۱

- لطفا stack trace کامل را ارسال کنید.
- اگر در PluginsStart پیام FileNotFoundException را دریافت می‌کنید، احتمالا یکی از وابستگی‌ها و ارجاعات پروژه یا افزونه‌ها، در پوشه‌ی bin برنامه وجود ندارند یا کپی نشده‌اند.

نکته 1

```
Copy "$(ProjectDir)$(OutDir)$(TargetName).*" "$(SolutionDir)MvcPluginMasterApp\bin\"
```

دستور post build event عنوان شده در قسمت اول ، فقط اسمبلی‌های اصلی افزونه را به پوشه‌ی bin پروژه‌ی اصلی کپی می‌کند. اگر این افزونه ارجاعات بیشتری دارد که در پروژه‌ی اصلی وجود ندارند، باید کل پوشه‌ی bin افزونه را کپی کنید و نه فقط فایل‌های اصلی آن‌را.
برای مثال فایل **MvcPluginMasterApp.Common** هم باید به پوشه‌ی bin کپی شود.

نکته 2
اگر پیام FileNotFoundException توسط استراکچرمپ صادر می‌شود، نیاز است inner exception آن‌را بررسی کنید. اصل خطای رخ داده را در inner exception ارائه می‌دهد.

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۱/۲۸ ۲۱:۵۳

این مورد رو فراموش کرده بودم :)
در پروژه MvcPluginMasterApp باید ارجاعی به پروژه MvcPluginMasterApp.Common داشته باشیم.

نویسنده: پوریا انوشیروانی
تاریخ: ۱۳۹۴/۰۱/۲۹ ۱۵:۲۳

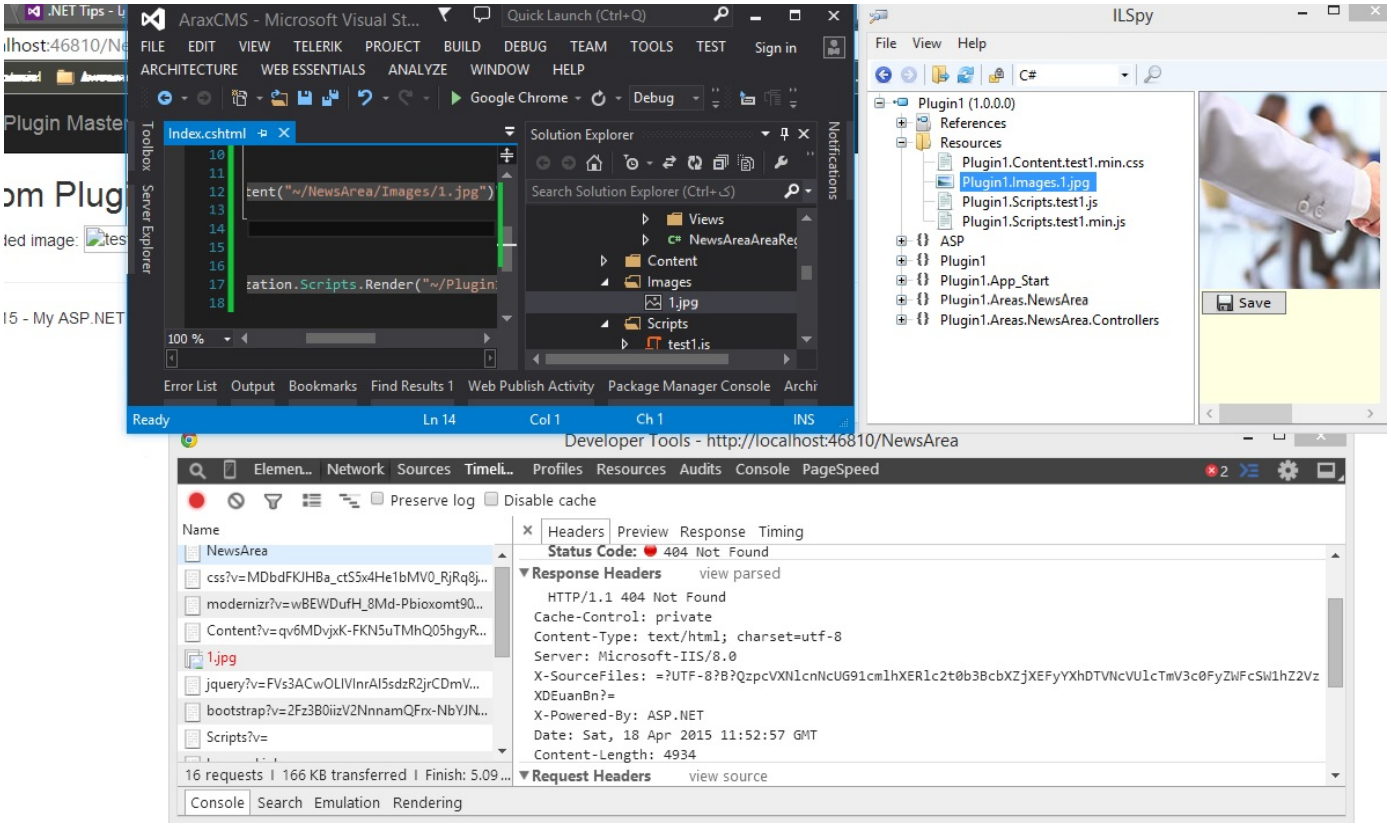
سلام مرسی از آموزش برای من فایل‌های CSS و JS درست کار میکنن ولی عکس رو ارور 404 میده نمی‌دونم مشکل از کجاست

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۹ ۱۵:۴۵

برگه‌ی web developer مرورگر را باز کنید (و یا از فایرباگ استفاده کنید). بعد محتوای response مرتبط با درخواست تصویر را بررسی کنید. اصل خطا در آنجا قابل مشاهده‌است.

نویسنده: پوریا انوشیروانی
تاریخ: ۱۳۹۴/۰۱/۲۹ ۱۶:۲۸

ممنون از راهنماییتون . یک اسکرین از صفحات گذاشتم .



نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۹ ۱۶:۴۴

در متد RegisterRoutes ایی که در مثال فوق هست:

```
public void RegisterRoutes(RouteCollection routes)
        {
            //....
            routes.Insert(0,
                new Route("NewsArea/Images/{file}.{extension}",
                    new RouteValueDictionary(new { }),
                    new RouteValueDictionary(new { extension = "png|jpg" }),
                    new EmbeddedResourceRouteHandler(assembly, resourcePath, cacheDuration:
TimeSpan.FromDays(30))
                ));
        }
```

آدرسی‌هایی با فرمت /file **NewsArea/Images** به EmbeddedResourceRouteHandler هدایت می‌شوند.

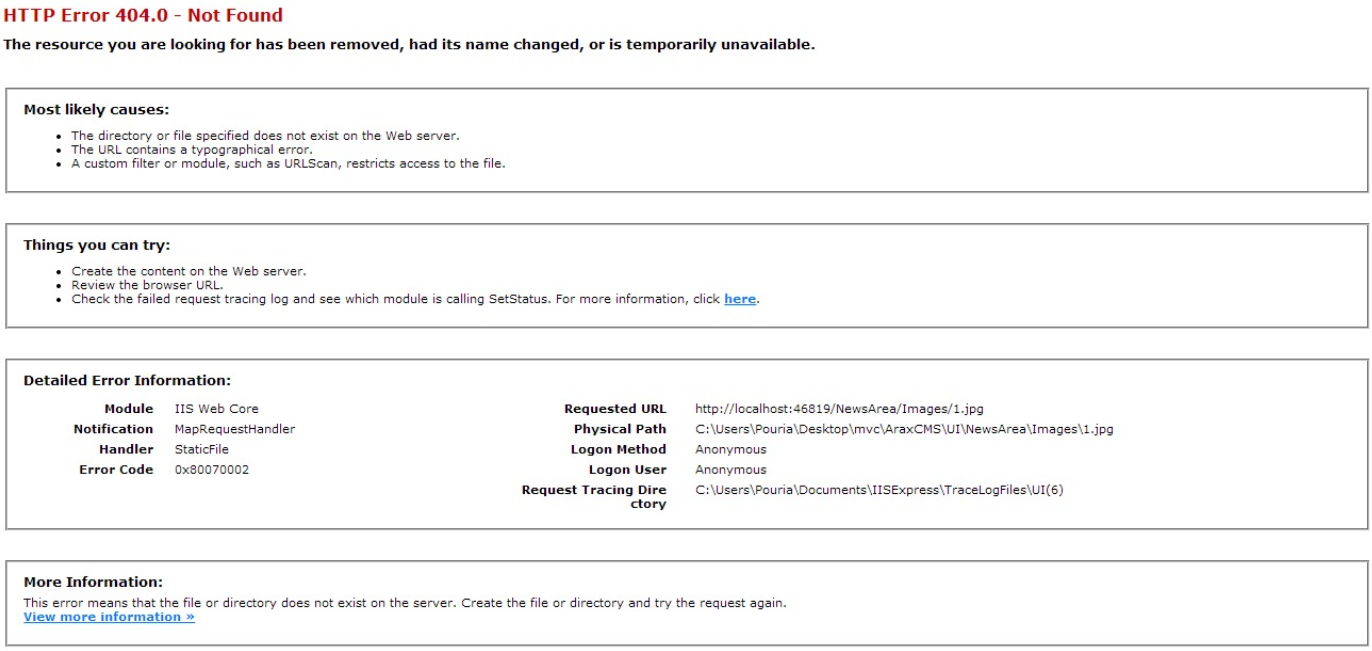
- بررسی کنید آدرس کاملی که به 404 ختم شده چیست؟ آیا آدرس درخواستی با **NewsArea/Images** شروع می‌شود؟
- در برگه‌ی response آن چه خروجی را مشاهده می‌کنید؟

نویسنده: پوریا انوشیروانی
تاریخ: ۱۳۹۴/۰۱/۲۹ ۱۶:۵۵

منظورتون رو از برگه ریسپانس متوجه نمیشم .

```
http://localhost:46819/NewsArea/Images/1.jpg
```

```
C:\Users\Pouria\Desktop\mvc\AraxCMS\UI\NewsArea\Images\1.jpg
```



نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۹ ۱۷:۱۲

- در همان تصویر اولی که ارسال کردید، برگه‌ی اول headers هست و برگه‌ی سوم آن response، که صفحه‌ی زرد رنگ معروف خطاهای ASP.NET را نمایش می‌دهد.
- ابتدا بررسی کنید response ایی که در حالت دیباگ مشاهده می‌کنید چی هست.
- سپس بررسی کنید اصلا متد RegisterRoutes ایی که عنوان شد، فراخوانی می‌شود و مسیریابی آن در سیستم ثبت می‌شود یا خیر؟ (یک break point داخل آن قرار دهید)

اگر فراخوانی نمی‌شود، بررسی کنید آیا فایل‌های پوشه‌ی bin این افزونه، به پوشه‌ی bin پروژه‌ی اصلی کپی شده‌اند یا خیر؟

نویسنده: پوریا انوشیروانی
تاریخ: ۱۳۹۴/۰۲/۰۱ ۱۳:۱

مرسی مشکل یکجای دیگه بود. کلا برنامم با نقطه مشکل داشت و قبل از اینکه کاری انجام بده ارور میداد. دوستانی که مشکل منو داشتن با کد زیر داخل وب کانفیگ حل میشه.

```
<system.webServer>
    <modules  runAllManagedModulesForAllRequests="true">
```

اگه میشه توضیحی بدید کد بالا چی هست و چرا با این درست شد؟!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۰۱ ۱۳:۳۰

این کد (در حالت تنظیمات integrated mode) باعث خواهد شد تا تمام درخواست‌های رسیده (منجمله درخواست دریافت فایل‌های استاتیک)، توسط موتور ASP.NET پردازش شود و IIS پیش از آن راسا اقدام نکند.

عنوان: طراحی افزونه پذیر با ASP.NET MVC 4.x/5.x - قسمت سوم
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۸ ۱۲:۱۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, MVC, Plugin

پس از ب ررسی ساختار یک پروژه‌ی افزونه پذیر و همچنین بهبود توزیع فایل‌های استاتیک آن ، اکنون نوبت به کار با داده‌ها است. هدف اصلی آن نیز داشتن مدل‌های اختصاصی و مستقل Entity framework code-first به ازای هر افزونه است و سپس بارگذاری و تشخیص خودکار آن‌ها در Context مرکزی برنامه.

پیشنیازها

- آشنایی با مباحث Migrations در EF Code first
- آشنایی با مباحث الگوی واحد کار
- چگونه مدل‌های EF را به صورت خودکار به Context اضافه کنیم؟
- چگونه تنظیمات مدل‌های EF را به صورت خودکار به Context اضافه کنیم؟

کدهایی را که در این قسمت مشاهده خواهید کرد، در حقیقت همان برنامه‌ی توسعه یافته « آشنایی با مباحث الگوی واحد کار » است و از ذکر قسمت‌های تکراری آن جهت طولانی نشدن مبحث، صرفنظر خواهد شد. برای مثال Context و مدل‌های محصولات و گروه‌های آن‌ها به همراه کلاس‌های لایه سرویس برنامه‌ی اصلی، دقیقا همان کدهای مطلب « آشنایی با مباحث الگوی واحد کار » است.

تعریف domain classes مخصوص افزونه‌ها

در ادامه‌ی پروژه‌ی افزونه پذیر فعلی، پروژه‌ی class library جدیدی را به نام MvcPluginMasterApp.Plugin1.DomainClasses اضافه خواهیم کرد. از آن جهت تعریف کلاس‌های مدل افزونه‌ی یک استفاده می‌کنیم. برای مثال کلاس News را به همراه تنظیمات Fluent آن به این پروژه‌ی جدید اضافه کنید:

```
using System.Data.Entity.ModelConfiguration;

namespace MvcPluginMasterApp.Plugin1.DomainClasses
{
    public class News
    {
        public int Id { set; get; }

        public string Title { set; get; }

        public string Body { set; get; }
    }

    public class NewsConfig : EntityTypeConfiguration<News>
    {
        public NewsConfig()
        {
            this.ToTable("Plugin1_News");
            this.HasKey(news => news.Id);
            this.Property(news => news.Title).IsRequired().HasMaxLength(500);
            this.Property(news => news.Body).IsOptional().IsMaxLength();
        }
    }
}
```

این پروژه برای کامپایل شدن نیاز به بسته‌ی نیوگت ذیل دارد:

```
PM> install-package EntityFramework
```

مشکل! برنامه‌ی اصلی، همانند مطلب « آشنایی با مباحث الگوی واحد کار » دارای domain classes خاص خودش است به همراه تنظیمات Context ایی که صریحا در آن مدل‌های متناظر با این پروژه در معرض دید EF قرار گرفته‌اند:

```
public class MvcPluginMasterAppContext : DbContext, IUnitOfWork
{
    public DbSet<Category> Categories { set; get; }
    public DbSet<Product> Products { set; get; }
```

اکنون برنامه‌ی اصلی چگونه باید مدل‌ها و تنظیمات سایر افزونه‌ها را یافته و به صورت خودکار به این Context اضافه کند؟ با توجه به اینکه این برنامه هیچ ارجاع مستقیمی را به افزونه‌ها ندارد.

تغییرات اینترفیس Unit of work جهت افزونه پذیری

در ادامه، اینترفیس بهبود یافته‌ی IUnitOfWork را جهت پذیرش DbSetهای پویا و همچنین EntityTypeConfigurationهای پویا، ملاحظه می‌کنید:

```
namespace MvcPluginMasterApp.PluginsBase
{
    public interface IUnitOfWork : IDisposable
    {
        IDbSet<TEntity> Set<TEntity>() where TEntity : class;
        int SaveAllChanges();
        void MarkAsChanged<TEntity>(TEntity entity) where TEntity : class;
        IList<T> GetRows<T>(string sql, params object[] parameters) where T : class;
        IEnumerable<TEntity> AddThisRange<TEntity>(IEnumerable<TEntity> entities) where TEntity :
class;
        void SetDynamicEntities(Type[] dynamicTypes);
        void ForceDatabaseInitialize();
        void SetConfigurationsAssemblies(Assembly[] assembly);
    }
}
```

متدهای جدید آن:

SetDynamicEntities : توسط این متد در ابتدای برنامه، نوع‌های مدل‌های جدید افزونه‌ها به صورت خودکار به Context اضافه خواهند شد.

SetConfigurationsAssemblies : کار افزودن اسمبلی‌های حاوی تعاریف EntityTypeConfigurationهای جدید و پویا را به عهده دارد.

ForceDatabaseInitialize: سبب خواهد شد تا مباحث migrations، پیش از شروع به کار برنامه، اعمال شوند.

در کلاس Context ذیل، نحوه‌ی پیاده سازی این متدهای جدید را ملاحظه می‌کنید:

```
namespace MvcPluginMasterApp.DataLayer.Context
{
    public class MvcPluginMasterAppContext : DbContext, IUnitOfWork
    {
        private readonly IList<Assembly> _configurationsAssemblies = new List<Assembly>();
        private readonly IList<Type[]> _dynamicTypes = new List<Type[]>();

        public void ForceDatabaseInitialize()
        {
            Database.Initialize(force: true);
        }

        public void SetConfigurationsAssemblies(Assembly[] assemblies)
        {
            if (assemblies == null) return;

            foreach (var assembly in assemblies)
            {
                _configurationsAssemblies.Add(assembly);
            }
        }

        public void SetDynamicEntities(Type[] dynamicTypes)
        {
            if (dynamicTypes == null) return;
            _dynamicTypes.Add(dynamicTypes);
        }
```

```
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            addConfigurationsFromAssemblies(modelBuilder);
            addPluginsEntitiesDynamically(modelBuilder);
            base.OnModelCreating(modelBuilder);
        }

        private void addConfigurationsFromAssemblies(DbModelBuilder modelBuilder)
        {
            foreach (var assembly in _configurationsAssemblies)
            {
                modelBuilder.Configurations.AddFromAssembly(assembly);
            }
        }

        private void addPluginsEntitiesDynamically(DbModelBuilder modelBuilder)
        {
            foreach (var types in _dynamicTypes)
            {
                foreach (var type in types)
                {
                    modelBuilder.RegisterEntityType(type);
                }
            }
        }
    }
}
```

در متد استاندارد OnModelCreating، فرصت افزودن نوع‌های پویا و همچنین تنظیمات پویای آن‌ها وجود دارد. برای این منظور می‌توان از متدهای modelBuilder.RegisterEntityType و modelBuilder.Configurations.AddFromAssembly کمک گرفت.

بهبود اینترفیس IPlugin جهت پذیرش نوع‌های پویای EF

در قسمت اول، با اینترفیس IPlugin آشنا شدیم. هر افزونه باید دارای کلاسی باشد که این اینترفیس را پیاده سازی می‌کند. از آن جهت دریافت تنظیمات و یا ثبت تنظیمات مسیریابی و امثال آن استفاده می‌شود.
در اینجا متد GetEfBootstrapper آن کار دریافت تنظیمات EF هر افزونه را به عهد دارد.

```
namespace MvcPluginMasterApp.PluginsBase
{
    public interface IPlugin
    {
        EfBootstrapper GetEfBootstrapper();
        //...به همراه سایر متدهای مورد نیاز
    }

    public class EfBootstrapper
    {
        /// <summary>
        /// Assemblies containing EntityTypeConfiguration classes.
        /// </summary>
        public Assembly[] ConfigurationsAssemblies { get; set; }

        /// <summary>
        /// Domain classes.
        /// </summary>
        public Type[] DomainEntities { get; set; }

        /// <summary>
        /// Custom Seed method.
        /// </summary>
        public Action<IUnitOfWork> DatabaseSeeder { get; set; }
    }
}
```

ConfigurationsAssemblies مشخص کننده‌ی اسمبلی‌هایی است که حاوی تعاریف EntityTypeConfigurationهای افزونه‌ی جاری هستند.
DomainEntities بیانگر لیست مدل‌ها و موجودیت‌های هر افزونه است.

DatabaseSeeder کار دریافت منطق متد Seed را بر عهده دارد. برای مثال اگر افزونه‌ای نیاز است در آغاز کار تشکیل جداول آن، دیتای پیش فرض و خاصی را در بانک اطلاعاتی ثبت کند، می‌توان از این متد استفاده کرد. اگر دقت کنید این Action یک وهله از IUnitOfWork را به افزونه ارسال می‌کند. بنابراین در این طراحی جدید، اینترفیس IUnitOfWork به پروژه‌ی MvcPluginMasterApp.PluginsBase منتقل می‌شود. به این ترتیب دیگر نیازی نیست تا تک تک افزونه‌ها ارجاع مستقیمی را به DataLayer پروژه‌ی اصلی پیدا کنند.

تکمیل متد GetEfBootstrapper در افزونه‌ها

اکنون جهت معرفی مدل‌ها و تنظیمات EF آن‌ها، تنها کافی است متد GetEfBootstrapper هر افزونه را تکمیل کنیم:

```
namespace MvcPluginMasterApp.Plugin1
{
    public class Plugin1 : IPlugin
    {
        public EfBootstrapper GetEfBootstrapper()
        {
            return new EfBootstrapper
            {
                DomainEntities = new[] { typeof(News) },
                ConfigurationsAssemblies = new[] { typeof(NewsConfig).Assembly },
                DatabaseSeeder = uow =>
                {
                    var news = uow.Set<News>();
                    if (news.Any())
                    {
                        return;
                    }

                    news.Add(new News
                    {
                        Title = "News 1",
                        Body = "news 1 news 1 news 1 ...."
                    });

                    news.Add(new News
                    {
                        Title = "News 2",
                        Body = "news 2 news 2 news 2 ...."
                    });
                }
            };
        }
```

در اینجا نحوه‌ی معرفی مدل‌های جدید را توسط خاصیت DomainEntities و تنظیمات متناظر را به کمک خاصیت ConfigurationsAssemblies مشاهده می‌کنید. باید دقت داشت که هر اسمبلی فقط باید یکبار معرفی شود و مهم نیست که چه تعداد تنظیمی در آن وجود دارند. کار یافتن کلیه‌ی تنظیمات از نوع EntityTypeConfigurationها به صورت خودکار توسط EF صورت می‌گیرد.
همچنین توسط delegate ایی به نام DatabaseSeeder، نحوه‌ی دسترسی به متد Set واحد کار و سپس استفاده‌ی از آن، برای تعریف متد Seed سفارشی نیز تکمیل شده‌است.

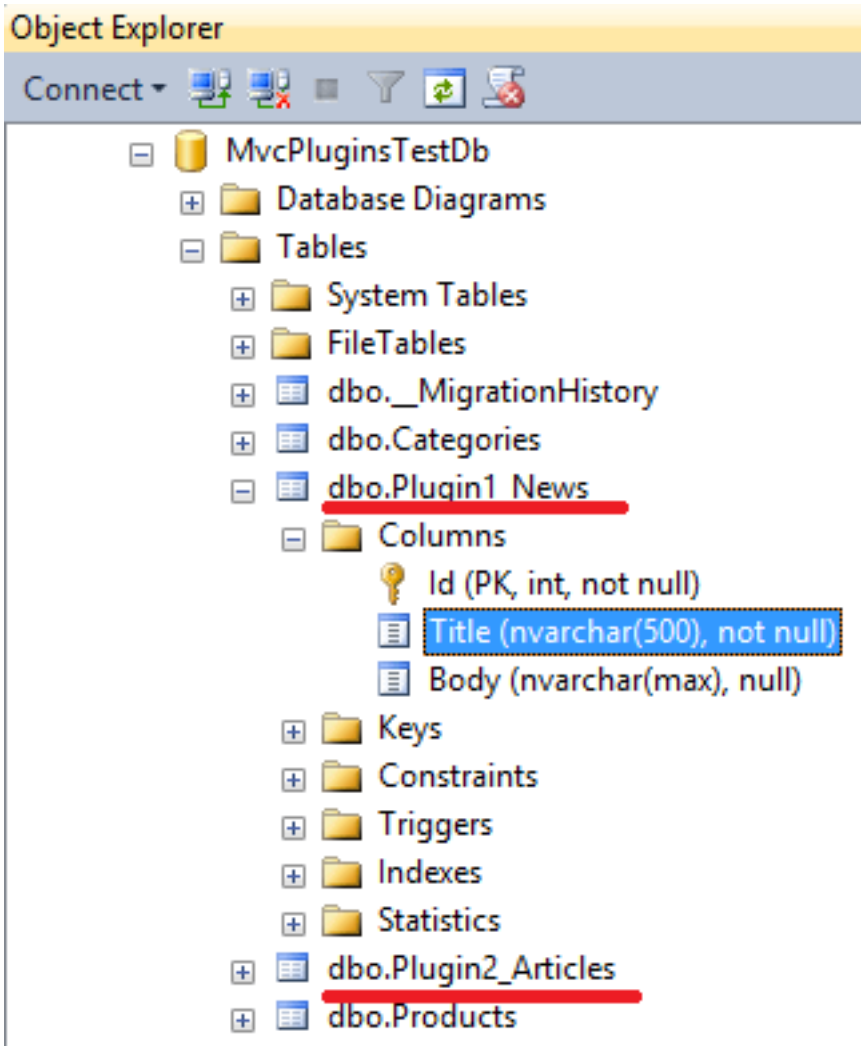
تدارک یک راه انداز EF، پیش از شروع به کار برنامه

در پوشه‌ی App_Start پروژه‌ی اصلی یا همان MvcPluginMasterApp، کلاس جدید EFBootstrapperStart را با کدهای ذیل اضافه کنید:

```
[assembly: PreApplicationStartMethod(typeof(EFBootstrapperStart), "Start")]
namespace MvcPluginMasterApp
{
    public static class EFBootstrapperStart
    {
        public static void Start()
        {
            var plugins = SmObjectFactory.Container.GetAllInstances<IPlugin>().ToList();
            using (var uow = SmObjectFactory.Container.GetInstance<IUnitOfWork>())
```

```
            {
                initDatabase(uow, plugins);
                runDatabaseSeeders(uow, plugins);
            }
        }

        private static void initDatabase(IUnitOfWork uow, IEnumerable<IPlugin> plugins)
        {
            foreach (var plugin in plugins)
            {
                var efBootstrapper = plugin.GetEfBootstrapper();
                if (efBootstrapper == null) continue;

                uow.SetDynamicEntities(efBootstrapper.DomainEntities);
                uow.SetConfigurationsAssemblies(efBootstrapper.ConfigurationsAssemblies);
            }

            Database.SetInitializer(new MigrateDatabaseToLatestVersion<MvcPluginMasterAppContext,
Configuration>());
            uow.ForceDatabaseInitialize();
        }

        private static void runDatabaseSeeders(IUnitOfWork uow, IEnumerable<IPlugin> plugins)
        {
            foreach (var plugin in plugins)
            {
                var efBootstrapper = plugin.GetEfBootstrapper();
                if (efBootstrapper == null || efBootstrapper.DatabaseSeeder == null) continue;

                efBootstrapper.DatabaseSeeder(uow);
                uow.SaveAllChanges();
            }
        }
    }
}
```

در اینجا یک راه انداز سفارشی از نوع PreApplicationStartMethod تهیه شده‌است. Pre بودن آن به معنای اجرای کدهای متد Start این کلاس، پیش از آغاز به کار برنامه و پیش از فراخوانی متد Application_Start فایل Global.asax.cs است.
همانطور که ملاحظه می‌کنید، ابتدا لیست تمام افزونه‌های موجود، به کمک StructureMap دریافت می‌شوند. سپس می‌توان در متد initDatabase به متد GetEfBootstrapper هر افزونه دسترسی یافت و توسط آن تنظیمات مدل‌ها را یافته و به Context اصلی برنامه اضافه کرد. سپس با فراخوانی ForceDatabaseInitialize تمام این موارد به صورت خودکار به بانک اطلاعاتی اعمال خواهند شد.
کار متد runDatabaseSeeders، یافتن DatabaseSeeder هر افزونه، اجرای آن‌ها و سپس فراخوانی متد SaveAllChanges در آخر کار است.



کدهای کامل این سری را از اینجا می‌توانید دریافت کنید:

MvcPlugin

نظرات خوانندگان

نویسنده: غلامرضا ربال
تاریخ: ۱۳۹۴/۰۱/۲۸ ۱۵:۴۲

با تشکر.
اگر لازم باشد پلاگین ما به مدل موجود در پروژه اصلی دسترسی داشته باشد باید به چه شکلی عمل کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۸ ۱۶:۲۳

- در عمل کل برنامه و تمام افزونه‌های آن از یک IUnitOfWork استفاده می‌کنند؛ یعنی تمام آن‌ها به تمام مدل‌های اضافه شده‌ی به Context اصلی برنامه دسترسی دارند. بنابراین هر پلاگین در صورت نیاز امکان دسترسی به مدل‌های برنامه‌ی اصلی یا سایر افزونه‌ها را دارا است. تمام این افزونه‌ها در کنار هم یک سیستم را تشکیل می‌دهند و مانند شکل انتهای بحث، از یک بانک اطلاعاتی استفاده می‌کنند.
- به همین جهت تنها کاری که باید انجام داد، افزودن ارجاعی به کلاس‌های مدل مورد نظر هست. پس از آن شبیه به کاری که در DatabaseSeeder انجام شده، می‌توان با استفاده از متد Set، به کلیه امکانات مدلی خاص دسترسی یافت:

```
DatabaseSeeder = uow =>
{
     var news = uow.Set<News>();
```

اگر نمی‌خواهید ارجاعی را به کلاس‌های مدل مورد نظر اضافه کنید، با توجه به اینکه این کلاس‌ها هم اکنون جزئی از وهله‌ی Context ارائه شده‌ی توسط IUnitOfWork هستند، باید متوسل به Reflection و تدارک متد Set ویژه‌ای شوید که بجای News، معادل رشته‌ای آن‌را دریافت کند.
ولی در کل افزودن ارجاعی به کلاس‌های مدل دیگر، مشکل ساز نیست؛ چون این کلاس‌ها عملا منطق خاصی را پیاده سازی نمی‌کنند و همچنین وابستگی خاصی هم به پروژه‌ی خاصی ندارند. یک سری کلاس دارای خاصیت‌های get/set دار معمولی هستند به همراه تنظیمات آن‌ها.