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: **تبدیل پلاگین‌های jQuery به کنترل‌های ASP.Net**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۰۸ ۱۵:۵۸:۱۸
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net

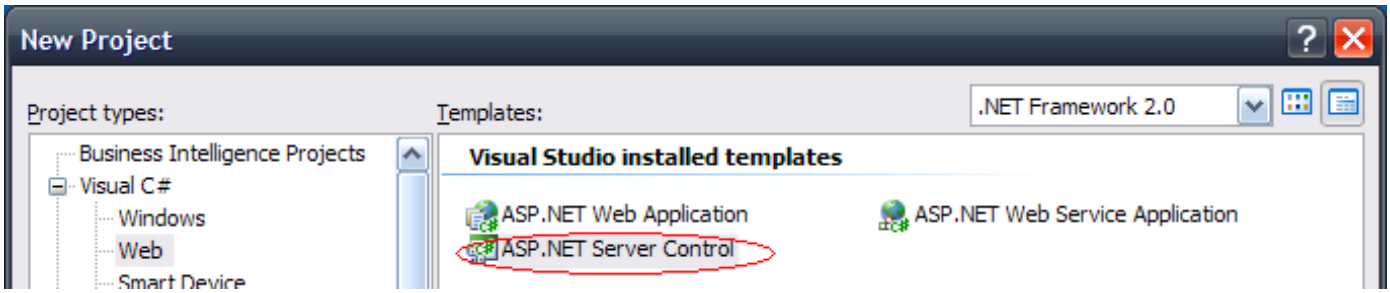
امروز داشتم یک سری از پلاگین‌های jQuery را مرور می‌کردم، مورد زیر به نظرم واقعا حرفه‌ای اومد و کمبود آن هم در بین کنترل‌های استاندارد ASP.Net محسوس است:

Masked Input Plugin

استفاده از آن به صورت معمولی بسیار ساده است. فقط کافی است اسکریپت‌های jQuery و سپس این افزونه به هدر صفحه اضافه شوند و بعد هم مطابق صفحه usage آن عمل کرد.

خیلی هم عالی! ولی این شیوه‌ی متداول کار در ASP.Net نیست. آیا بهتر نیست این مجموعه را تبدیل به یک کنترل کنیم و از این پس به سادگی با استفاده از Toolbox ویژوال استودیو آن‌را به صفحات اضافه کرده و بدون درگیر شدن با دستکاری سورس html صفحه، از آن استفاده کنیم؟ به‌عبارتی دیگر یکبار باید با جزئیات درگیر شد، آنرا بسته بندی کرد و سپس بارها از آن استفاده نمود. (مفاهیم شیءگرایی)

برای این‌کار، یک پروژه جدید ایجاد ASP.Net server control را آغاز نمائید (به نام MaskedEditCtrl).



به صورت پیش فرض یک قالب استاندارد ایجاد خواهد شد که کمی نیاز به اصلاح دارد. نام کلاس را به MaskedEdit تغییر خواهیم داد و همچنین در قسمت ToolboxData نیز نام کنترل را به MaskedEdit ویرایش می‌کنیم.

برای اینکه مجبور نشویم یک کنترل کاملا جدید را از صفر ایجاد کنیم، خواص و توانایی‌های اصلی این کنترل را از TextBox استاندارد به ارث خواهیم برد. بنابراین تا اینجای کار داریم:

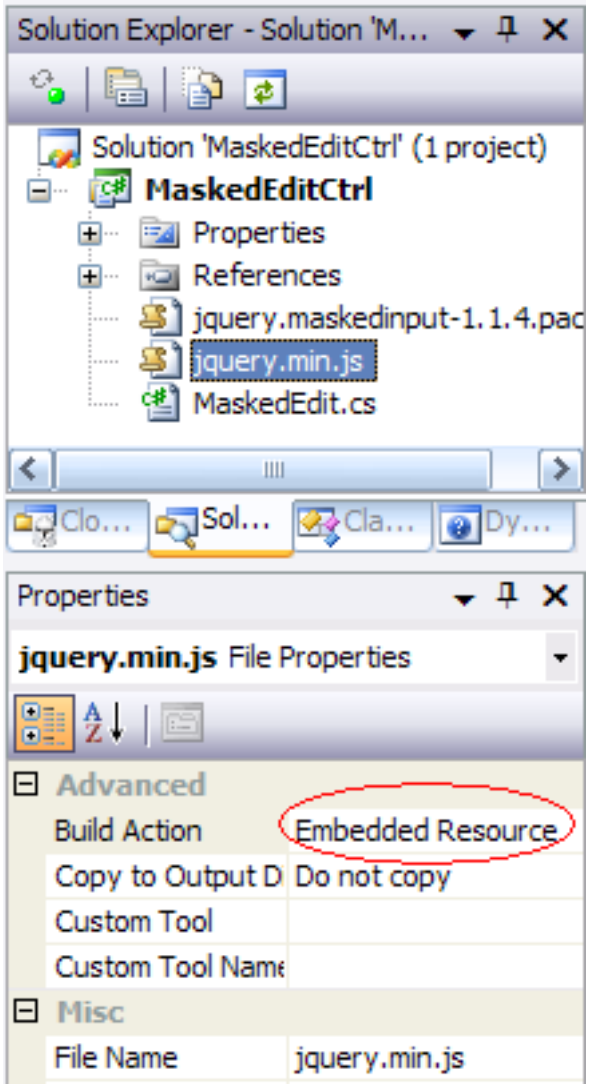
```
namespace MaskedEditCtrl
{
   [DefaultProperty("MaskFormula")]
   [ToolboxData("<{0}:MaskedEdit runat=server></{0}:MaskedEdit>")]
   [Description("MaskedEdit Control")]
   public class MaskedEdit : TextBox

   {
```

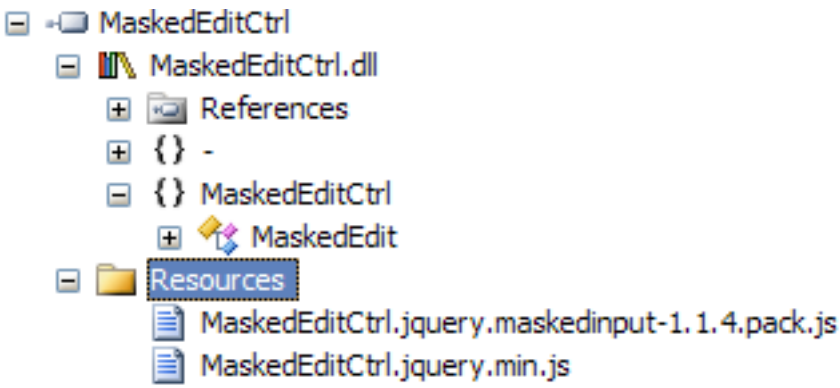
سپس باید رویداد OnPreRender را تحریف (override) کرده و در آن همان اعمالی را که هنگام افزودن اسکریپت‌ها به صورت دستی انجام می‌دادیم با برنامه نویسی پیاده سازی کنیم. چند نکته ریز در اینجا وجود دارد که در ادامه به آن‌ها اشاره خواهد شد.

از ASP.Net 2.0 به بعد، امکان قرار دادن فایل‌های اسکریپت و یا تصاویر همراه یک کنترل، درون فایل dll آن بدون نیاز به توزیع مجزای آنها به صورت WebResource مهیا شده است. برای این منظور اسکریپت‌های jQuery و افزونه mask edit را به پروژه اضافه

نمائید. سپس به قسمت خواص آنها (هر دو اسکریپت) مراجعه کرده و build action آنها را به Embedded Resource تغییر دهید (شکل زیر):



از این پس با کامپایل پروژه، این فایل‌ها به صورت resource به dll ما اضافه خواهند شد. برای تست این مورد می‌توان به برنامه reflector مراجعه کرد (تصویر زیر):



پس از افزودن مقدماتی اسکریپت‌ها و تعریف آنها به صورت resource ، باید آنها را در فایل AssemblyInfo.cs پروژه نیز تعریف کرد (به صورت زیر).

```
[assembly: WebResource("MaskedEditCtrl.jquery.min.js", "text/javascript")]
[assembly: WebResource("MaskedEditCtrl.jquery.maskedinput-1.1.4.pack.js", "text/javascript")]
```

نکته مهم: همانطور که ملاحظه می‌کنید نام فضای نام پروژه (namespace) باید به ابتدای اسکریپت‌های معرفی شده اضافه شود.

پس از آن نوبت به افزودن این اسکریپت‌ها به صورت خودکار در هنگام نمایش کنترل است. برای این منظور داریم:

```
protected override void OnPreRender(EventArgs e)
       {
           base.OnPreRender(e);

           //adding .js files
           if (!Page.ClientScript.IsClientScriptIncludeRegistered("jquery_base"))
           {
               string scriptUrl = Page.ClientScript.GetWebResourceUrl(this.GetType(),
                   "MaskedEditCtrl.jquery.min.js");
               Page.ClientScript.RegisterClientScriptInclude("jquery_base", scriptUrl);
           }

           if (!Page.ClientScript.IsClientScriptIncludeRegistered("edit_ctrl"))
           {
               string scriptUrl = Page.ClientScript.GetWebResourceUrl(this.GetType(),
                   "MaskedEditCtrl.jquery.maskedinput-1.1.4.pack.js");
               Page.ClientScript.RegisterClientScriptInclude("edit_ctrl", scriptUrl);
           }

           if (!Page.ClientScript.IsStartupScriptRegistered("MaskStartup" + this.ID))
           {
               // Form the script to be registered at client side.
               StringBuilder sbStartupScript = new StringBuilder();
               sbStartupScript.AppendLine("jQuery(function($){");
               sbStartupScript.AppendLine("$(\"#" + this.ClientID + "\").mask(\"" + MaskFormula +
"\");");
               sbStartupScript.AppendLine("});");
               Page.ClientScript.RegisterStartupScript(typeof(Page),
                   "MaskStartup" + this.ID, sbStartupScript.ToString(), true);
           }

       }
```

همانطور که ملاحظه می‌کنید، ابتدا WebResource دریافت شده و سپس به صفحه اضافه می‌شود. در آخر مطابق راهنمای افزونه mask edit عمل شد. یعنی اسکریپت مورد نظر را ساخته و به صفحه اضافه کردیم.

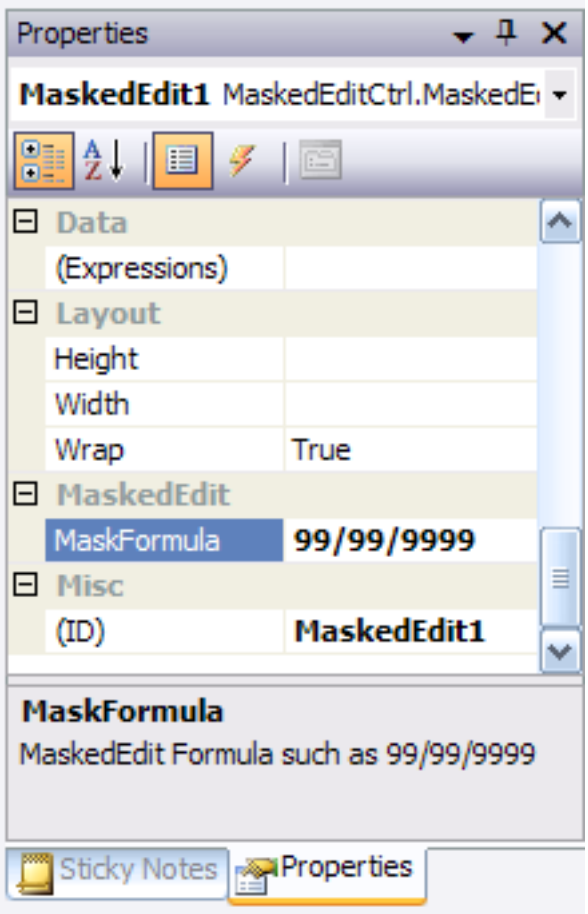
نکته جاوا اسکریپتی: علت استفاده از this.ClientID جهت معرفی نام کنترل جاری این است که هنگامیکه کنترل توسط یک master page رندر شود، ID آن توسط موتور ASP.Net کمی تغییر خواهد کرد. برای مثال myTextBox به ctl00_ContentPlaceHolder1_myTextBox تبدیل خواهد شد و اگر صرفا this.ID ذکر شده باشد دیگر دسترسی به آن توسط کدهای جاوا اسکریپت مقدور نخواهد بود. بنابراین از ClientID جهت دریافت ID نهایی رندر شده توسط ASP.Net کمک می‌گیریم.

در اینجا MaskFormula مقداری است که هنگام افزودن کنترل به صفحه می‌توان تعریف کرد.

```
[Description("MaskedEdit Formula such as 99/99/9999")]
       [Bindable(true), Category("MaskedEdit"), DefaultValue(0)]
       public string MaskFormula
       {
           get
           {
               if (ViewState["MaskFormula"] == null) ViewState["MaskFormula"] = "99/99/9999";
               return (string)ViewState["MaskFormula"];
           }
           set { ViewState["MaskFormula"] = value; }

       }
```

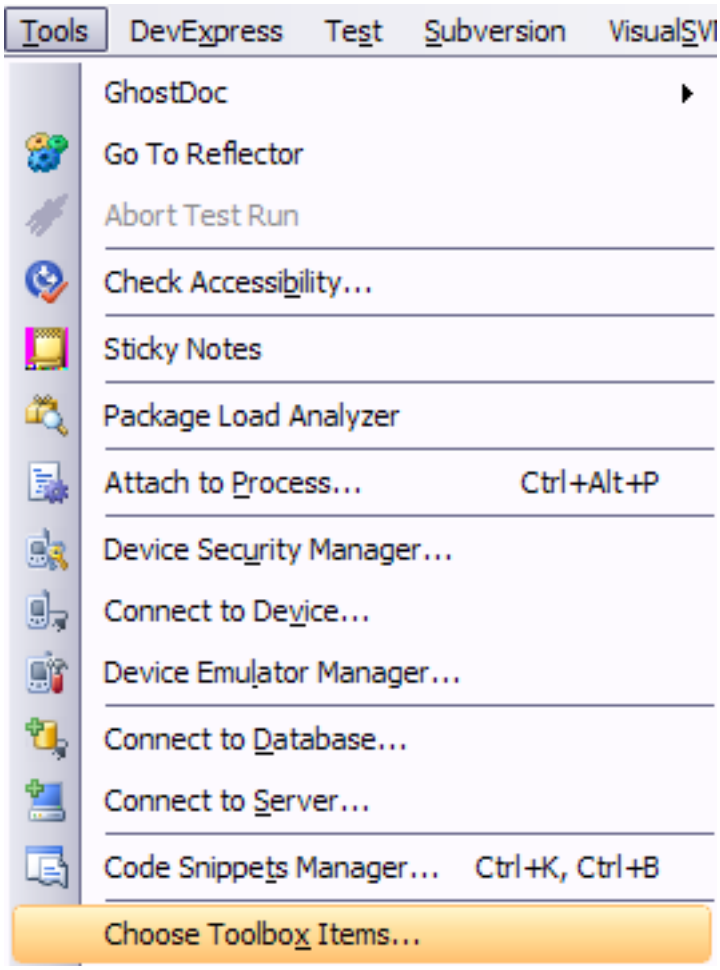
این خاصیت public هنگام نمایش در Visual studio به شکل زیر درخواهد آمد:



نکته مهم: در اینجا حتما باید از view state جهت نگهداری مقدار این خاصیت استفاده کرد تا در حین post back ها مقادیر

انتساب داده شده حفظ شوند.

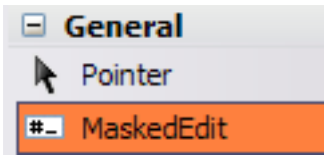
اکنون پروژه را کامپایل کنید. برای افزودن کنترل ایجاد شده به toolbox می‌توان مطابق تصویر عمل کرد:



نکته: برای افزودن آیکون به کنترل (جهت نمایش در نوار ابزار) باید: الف) تصویر مورد نظر از نوع bmp باشد با اندازه 16 در16 pixel . ب) باید آنرا به پروژه افزود و build action آن را به Embedded Resource تغییر داد. سپس آنرا در فایل AssemblyInfo.cs پروژه نیز تعریف کرد (به صورت زیر).

```
[assembly: System.Web.UI.WebResource("MaskedEditCtrl.MaskedEdit.bmp", "img/bmp")]
```

کنترل ما پس از افزوده شدن، شکل زیر را خواهد داشت:



جهت دریافت سورس کامل و فایل باینری این کنترل، اینجا کلیک کنید.

نظرات خوانندگان

نویسنده: شاهین کایست
تاریخ: ۱۳۸۹/۱۰/۲۳ ۲۲:۳۹:۳۰

سلام.
از کنترلی که طراحی کردید درون یک JQuery UI دیالوگ استفاده کردم (درون محتویات Dialog در یک Update Panel هست).
اما پس از قرار دادن کنترل Dialog از کار افتاد.
نکته : Dialog را از سمت Server پس از Postback اجرا کردم.
به نظرتون مشکل از کجا هست؟
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۰/۲۳ ۲۲:۵۷:۴۱

سلام،
چند مورد هست:
- یکی اینکه بهتر است نسخه‌ی جدید jQuery را به این سورس اضافه و کامپایل کنید.
- مورد دیگر آشنایی با jQuery Live است : (+) ، که پس از postback ، نیاز به تزریق یا بایند مجدد یک سری اطلاعات می‌باشد و همچنین: (+)

عنوان: **رفع تداخل jQuery با کتابخانه‌های مشابه**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۱/۲۰ ۱۰:۳۱:۱۷
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

قبل از شروع، یک خبر!
VsDoc for jQuery 1.3.1 (جهت فعال سازی intellisense آخرین نگارش جی کوئری در VS.Net)

اگر سعی کنید jQuery را به همراه سایر کتابخانه‌های جاوا اسکریپتی دیگر به صورت همزمان استفاده کنید (مثلا mootools یا ASP.Net Ajax و امثال آن)، احتمالا قسمتی و یا تمامی کدهای جاوا اسکریپتی شما کار نخواهند کرد. برای مثال update panel شما در ASP.Net Ajax از کار می‌افتد، یا کدهای mootools شما دیگر کار نمی‌کنند. علت اینجا است که تمامی این کتابخانه‌ها از نشانه $ به عنوان متغیری عمومی که بیانگر نام مستعار کتابخانه مربوطه است استفاده می‌کنند و در نهایت تمام این‌ها با هم تداخل خواهند کرد.

خوشبختانه jQuery امکان رفع این تداخل را پیش بینی کرده است که به صورت زیر می‌باشد:

```
<script type="text/javascript" language="javascript" src="jquery.min.js"></script>
<script type="text/javascript">
   jQuery.noConflict();
   jQuery(document).ready(function($) {
       //tip-1
       $("select > option").each(function() {
           var obj = $(this);
           obj.attr("title", obj.attr("value"));
       });
       //tip-1
   });
</script>
```

کد مثال فوق، به تمامی آیتم‌های drop down list های شما در یک صفحه، بر اساس value هر آیتم موجود در آن‌ها، یک tooltip اضافه می‌کند. (با IE7 به بعد و فایرفاکس سازگار است)
در اینجا ابتدا jQuery.noConflict فراخوانی شده و سپس document ready متداول هم باید اندکی مطابق کد فوق تغییر کند.
مابقی کدهای شما از این پس نیازی به تغییر نخواهند داشت. (روش‌های دیگری هم برای تغییر نام $ وجود دارند که در مستندات مربوطه قابل مشاهده است)

عنوان: **استفاده‌ی همزمان از آپدیت پنل ASP.Net و پلاگین‌های جی‌کوئری**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۶/۲۰ ۱۸:۴۷:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net

مشکل: زمانیکه یک AsyncPostback در آپدیت پنلAjax ASP.Net رخ دهد، پس از پایان کار، پلاگین جی‌کوئری که در حال استفاده از آن بودید و در هنگام بارگذاری اولیه صفحه بسیار خوب کار می‌کرد، اکنون از کار افتاده است و دیگر جواب نمی‌دهد.

قبل از شروع، نیاز به یک سری پیش زمینه هست (شاید بر اساس روش استفاده شما از آن پلاگین جی‌کوئری، مشکل را حل کنند):
الف) رفع تداخل جی‌کوئری با سایر کتابخانه‌های مشابه.
ب) آشنایی با jQuery Live جهت بایند رخدادها به عناصری که بعدا به صفحه اضافه خواهند شد.
ج) تزریق اسکریپت به صفحه در حین یک AsyncPostback رخ داده در آپدیت پنل

علت بروز مشکل:
علت رخدادن این مشکل (علاوه بر قسمت الف ذکر شده)، عدم فراخوانی document.ready تعریف شده، جهت بایند مجدد پلاگین jQuery مورد استفاده شما پس از هر AsyncPostback رخ داده در آپدیت پنل ASP.Net Ajax است. راه حل استاندارد جی‌کوئری هم همان مورد (ب) فوق می‌باشد، اما ممکن است جهت استفاده از آن نیاز به بازنویسی یک پلاگین موجود خاص وجود داشته باشد، که آنچنان مقرون به صرفه نیست.

مثالی جهت مشاهده‌ی این مشکل در عمل:
می‌خواهیم افزونه‌ی Colorize - jQuery Table را به یک گرید ویوو ASP.Net قرار گرفته درون یک آپدیت پنل اعمال کنیم.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="UpdatePanelTest.aspx.cs"
   Inherits="TestJQueryAjax.UpdatePanelTest" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
   <asp:ScriptManager ID="sm" runat="server">
       <Scripts>
           <asp:ScriptReference Path="~/js/jquery.js" />
           <asp:ScriptReference Path="~/js/jquery.colorize-1.6.0.js" />
       </Scripts>
   </asp:ScriptManager>
   <asp:UpdatePanel ID="uppnl" runat="server">
       <ContentTemplate>
           <asp:GridView ID="GridView1" runat="server" AllowPaging="True"
            OnPageIndexChanging="GridView1_PageIndexChanging">
           </asp:GridView>
       </ContentTemplate>
   </asp:UpdatePanel>
   </form>

   <script type="text/javascript">
       $(document).ready(function() {
           $('#<%=GridView1.ClientID %>').colorize();
       });
   </script>
</body>
</html>
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;

namespace TestJQueryAjax
{
   public partial class UpdatePanelTest : System.Web.UI.Page
   {
       void BindTo()
       {
           List<string> rows = new List<string>();
           for (int i = 0; i < 1000; i++)
               rows.Add(string.Format("row{0}", i));

           GridView1.DataSource = rows;
           GridView1.DataBind();
       }

       protected void Page_Load(object sender, EventArgs e)
       {
           if (!Page.IsPostBack)
           {
               BindTo();
           }
       }

       protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
       {
           GridView1.PageIndex = e.NewPageIndex;
           BindTo();
       }
   }
}
```

مثال بسیار ساده‌ای است جهت اعمال این افرونه به یک گریدویو و مشاهده کار کردن این افزونه در هنگام بارگذاری اولیه صفحه و سپس از کار افتادن آن پس از مشاهده صفحه دوم گرید. در این مثال از نکته " اسکریپت‌های خود را یکی کنید " استفاده شده است.

راه حل:

از ویژگی‌های ذاتی ASP.Net Ajax باید کمک گرفت برای مثال:

```
<script type="text/javascript">
       $(document).ready(function() {
           $('#<%=GridView1.ClientID %>').colorize();
       });

       function pageLoad(sender, args) {
           if (args.get_isPartialLoad()) {
               $('#<%=GridView1.ClientID %>').colorize();
           }
       }
   </script>
```

متد استاندارد pageLoad به صورت خودکار پس از هر AsyncPostback رخ داده در آپدیت پنل ASP.Net Ajax فراخوانی می‌شود (و همچنین پس از پایان پردازش و بارگذاری اولیه DOM صفحه). در این متد بررسی می‌کنیم که آیا یک partial postback رخ داده است؟ اگر بله، مجددا عملیات باید افزونه به گرید را انجام می‌دهیم و مشکل برطرف خواهد شد.

برای مطالعه بیشتر

نظرات خوانندگان

نویسنده: نیما
تاریخ: ۱۳۸۸/۰۶/۲۱ ۰۱:۲۸:۴۲

سلام استاد عزیز
ممنون از مطلب بسیار مفیدتون.