عنوان: **ارائه کاربری ساده‌تر انتخاب چندین آیتم از یک لیست به کمک افزونه TagIt در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۵ ۱۳:۰
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, jQuery, MVC, JavaScript

چندی قبل مطلبی را در مورد بررسی تفصیلی رابطه چند به چند در این سایت مطالعه کردید. در آن مطلب صرفا به بحث ذخیره سازی اطلاعات دریافتی از کاربر اشاره شد. برای مثال اگر مطلبی چندین برچسب دارد، چگونه باید این‌ها را در بانک اطلاعاتی به نحو صحیحی ذخیره کرد.
در مطلب جاری قصد داریم با نحوه ارائه یک UI کاربر پسند برای این منظور آشنا شویم و سؤال مهم هم این است: «چگونه می‌توان کار کاربر را در حین وارد کردن تعدادی از برچسب‌های مرتبط با یک مطلب ساده‌تر کرد؟». برای این منظور یکی از راه حل‌هایی که در بسیاری از سایت‌ها مرسوم شده است، استفاده از افزونه‌هایی مانند jQuery TagIt می‌باشد که در ادامه با نحوه استفاده از آن در ASP.NET MVC آشنا خواهیم شد.

پیشنیازها:
دریافت افزونه TagIt
همچنین دریافت jQuery UI (افزونه TagIt برای نمایش لیست Auto Complete آیتم‌ها از jQuery UI در پشت صحنه استفاده می‌کند)

```
<head>
    <title>@ViewBag.Title</title>
    <link href="@Url.Content("~/Content/TagIt/jquery-ui-1.8.23.custom.css")" rel="stylesheet"
type="text/css" />
    <link href="@Url.Content("~/Content/TagIt/tagit-simple-blue.css")" rel="stylesheet" type="text/css"
/>
    <link href="@Url.Content("Content/Site.css")" rel="stylesheet" type="text/css" />
    <script src="@Url.Content("~/Scripts/jquery-1.9.1.min.js")" type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.validate.min.js")" type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.unobtrusive-ajax.min.js")"
```

```
type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>
    <script src="@Url.Content("~/Content/TagIt/jquery-ui-1.8.23.custom.min.js")"
type="text/javascript"></script>
    <script src="@Url.Content("~/Content/TagIt/tagit.js")" type="text/javascript"></script>
    @RenderSection("JavaScript", required: false)
</head>
```

که نهایتا نیاز است یک چنین تعاریفی را به فایل layout برنامه اضافه کنیم.

آشنایی با مدل برنامه

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace jQueryMvcSample04.Models
{
    public class BlogPostViewModel
    {
        [DisplayName("عنوان"), Required(ErrorMessage = "*")]
        public string Title { set; get; }

        [DisplayName("متن"), Required(ErrorMessage = "*")]
        public string Body { set; get; }

        /// <summary>
        /// آرایه‌ای محدود از برچسب‌های این مطلب خاص به صورت جی‌سون که پیشتر ثبت شده است
        /// هدف استفاده در حین ویرایش مطلب
        /// </summary>
        public string InitialTags { set; get; }

        /// <summary>
        /// آرایه‌ای جی‌سونی از تمام برچسب‌های موجود در سیستم
        /// هدف نمایش منوی انتخاب برچسب‌ها از لیست
        /// </summary>
        public string TagsSource { set; get; }

        /// <summary>
        /// آرایه‌ای از برچسب‌های وارد شده توسط کاربر در حین ثبت مطلب
        /// </summary>
        [DisplayName("برچسب‌ها"), Required(ErrorMessage = "*")]
        public string[] Tags { set; get; }

        public int? Id { set; get; }
    }
}
```

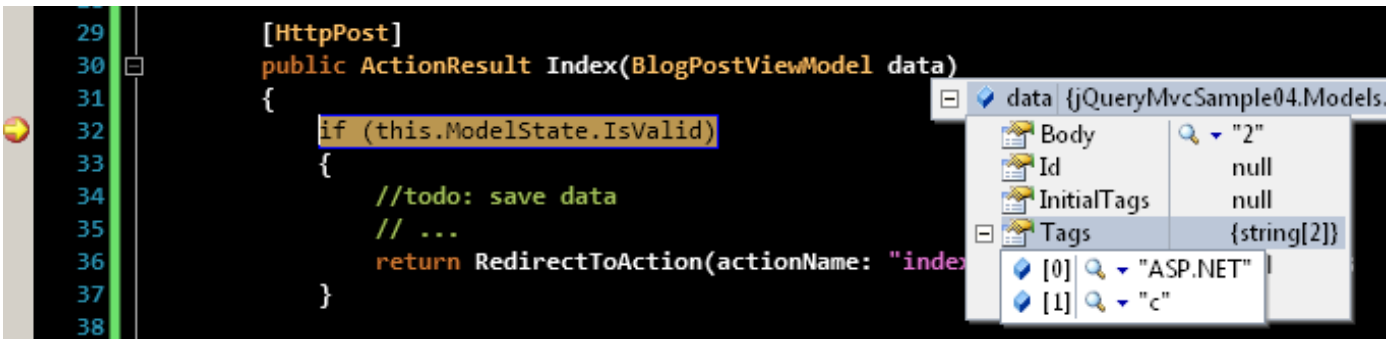
اگر به نام این کلاس دقت کنید، به ViewModel ختم شده است. از این لحاظ که حاوی خواصی می‌باشد که عموما جهت رندر کردن صحیح UI مورد استفاده قرار می‌گیرند و معادلی در سمت بانک اطلاعاتی نخواهند داشت.
افزونه TagIt برای نمایش اطلاعات خود به دو منبع داده نیاز دارد:
الف) TagsSource : لیستی است به فرمت JSON، از هر آنچه که در سیستم پیشتر به عنوان یک برچسب ثبت شده است. از این لیست برای نمایش منوی خودکار انتخاب آیتم‌ها استفاده می‌شود.
ب) InitialTags : لیستی است به فرمت JSON، از تمام برچسب‌های مرتبط با یک مطلب. از این اطلاعات در حین ویرایش یک مطلب استفاده خواهد شد.

در این ViewModel یک خاصیت دیگر به شکل آرایه، به نام Tags تعریف شده است که لیست برچسب‌های وارد شده توسط کاربر را دریافت خواهد کرد.

معرفی کنترلر برنامه

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

```
using jQueryMvcSample04.Extensions;
using jQueryMvcSample04.Models;

namespace jQueryMvcSample04.Controllers
{
    public class HomeController : Controller
    {
        [HttpGet]
        public ActionResult Index(int? id)
        {
            //در ابتدای کار تمام تگ‌های موجود در سیستم از بانک اطلاعاتی دریافت خواهند شد
            //از این تگ‌ها برای تشکیل منوی انتخاب برچسب‌ها استفاده می‌شود
            var tagsSource = new[] { "C#", "C++", "C", "ASP.NET", "MVC" }.ToJson();

            //همچنین صرفا برچسب‌های مطلب جاری که پیشتر ثبت شده‌اند نیز باید از بانک اطلاعاتی دریافت
گردند
            //از این برچسب‌ها برای ویرایش یک مطلب موجود استفاده خواهد شد
            var init = new[] { "ASP.NET" }.ToJson();

            var model = new BlogPostViewModel
            {
                TagsSource = tagsSource,
                InitialTags = init,
                Id = id
            };
            return View(model);
        }

        [HttpPost]
        public ActionResult Index(BlogPostViewModel data)
        {
            if (this.ModelState.IsValid)
            {
                //todo: save data
                // ...
                return RedirectToAction(actionName: "index", controllerName: "home");
            }

            //در صورت بروز خطا مجددا اطلاعات موجود نمایش داده خواهند شد
            data.TagsSource = new[] { "C#", "C++", "C", "ASP.NET", "MVC" }.ToJson();
            data.InitialTags = data.Tags.ToJson();
            return View(data);
        }
    }
}
```



با توجه به توضیحاتی که ارائه شد، کنترلر برنامه ساختار واضح‌تری را یافته است. در اولین بار نمایش صفحه، لیست منبع داده تگ‌ها و همچنین تگ‌های مرتبط با یک مطلب (در صورت وجود) به View ارائه خواهند شد.

از همین ViewModel، در عملیات Post نیز استفاده گردیده و اطلاعات دریافت می‌گردد.

تعریف متد الحاقی ToJson مورد استفاده را نیز در ادامه ملاحظه می‌نمائید:

```
using System.Linq;
```

```
using System.Web.Script.Serialization;

namespace jQueryMvcSample04.Extensions
{
    public static class JsonExt
    {
        public static string ToJson(this string[] initialTags)
        {
            if (initialTags == null || !initialTags.Any())
                return "[]";
            else
                return new JavaScriptSerializer().Serialize(initialTags);
        }
    }
}
```

و مرحله آخر تعریف View متناظر است

```
@model jQueryMvcSample04.Models.BlogPostViewModel
@{
    ViewBag.Title = "Index";
}
@using (Html.BeginForm())
{
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>ثبت مطلب جدید</legend>
        @Html.HiddenFor(model => model.Id)
        <div class="editor-label">
            @Html.LabelFor(model => model.Title)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>
        <div class="editor-label">
            @Html.LabelFor(model => model.Body)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Body)
            @Html.ValidationMessageFor(model => model.Body)
        </div>
        <div class="editor-label">
            @Html.LabelFor(model => model.Tags)
        </div>
        <div class="editor-field">
            <ul id="tagsList" dir="ltr" name="Tags">
            </ul>
            @Html.ValidationMessageFor(model => model.Tags)
        </div>
        <p>
            <input type="submit" value="Create" />
        </p>
    </fieldset>
}
@section JavaScript
{
    <script type="text/javascript">
    $(document).ready(function () {
            var tagsSource = @Html.Raw(Model.TagsSource);
            $('#tagsList').tagit({
                 tagSource: tagsSource,
                 select: true,
                 triggerKeys: ['enter', 'comma', 'tab'],
                 initialTags:  @Html.Raw(Model.InitialTags)
              });
});
    </script>
}
```

در این View دو نکته حائز اهمیت هستند:

الف) برای نمایش افزونه TagIt از یک ul با id مساوی tagsList استفاده شده است.
ب) خواص اضافی موجود در ViewModel که اطلاعات JSON ایی مورد نیاز را بازگشت می‌دهند در قسمت اسکریپت صفحه مورد استفاده قرار گرفته‌اند. در اینجا نیاز است از Html.Raw استفاده شود تا اطلاعات مرتبط با JSON اشتباها encode نشده و قابل استفاده باشند.

دریافت مثال و پروژه کامل این قسمت
jQueryMvcSample04.zip

نظرات خوانندگان

نویسنده: محسن.د
تاریخ: ۱۳۹۲/۰۱/۱۳ ۱۱:۶

بسیار عالی .
بعد از خوندن مطلب ، اول فکر کردم که استفاده از jQuery UI ضروری نیست و فقط در صورتی که بخوایم امکان autoComplete رو فعال کنیم باید رفرنسی به اون بدیم اما بعد از حذف کردن متوجه شدم که اینطور نیست و کلا این plugin هم به jQuery و هم به jQuery UI وابسته است و بدون UI کار نمی‌کنه .