عنوان: ASP.NET MVC #22
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۰۶ ۲۲:۳۵:۰۰
آدرس: www.dotnettips.info
گروه‌ها: MVC

تهیه سایت‌های چند زبانه و بومی سازی نمایش اطلاعات در ASP.NET MVC

زمانیکه دات نت فریم ورک نیاز به انجام اعمال حساس به مسایل بومی را داشته باشد، ابتدا به مقادیر تنظیم شده دو خاصیت زیر دقت می‌کند:

الف) System.Threading.Thread.CurrentThread.CurrentCulture

بر این اساس دات نت می‌تواند تشخیص دهد که برای مثال خروجی متد DateTime.Now.ToString در کانادا و آمریکا باید با هم تفاوت داشته باشند. مثلا در آمریکا ابتدا ماه، سپس روز و در آخر سال نمایش داده می‌شود و در کانادا ابتدا سال، بعد ماه و در آخر روز نمایش داده خواهد شد. یا نمونه‌ی دیگری از این دست می‌تواند نحوه نمایش علامت واحد پولی کشورها باشد.

ب) System.Threading.Thread.CurrentThread.CurrentUICulture

مقدار CurrentUICulture بر روی بارگذاری فایل‌های مخصوصی به نام Resource، تاثیر گذار است.

این خواص را یا به صورت دستی می‌توان تنظیم کرد و یا ASP.NET، این اطلاعات را از هدر Accept-Language دریافتی از مرورگر کاربر به صورت خودکار مقدار دهی می‌کند. البته برای این منظور نیاز است یک سطر زیر را به فایل وب کانفیگ برنامه اضافه کرد:

```
<system.web>
    <globalization culture="auto" uiCulture="auto" />
```

یا اگر نیاز باشد تا برنامه را ملزم به نمایش اطلاعات Resource مرتبط با فرهنگ بومی خاصی کرد نیز می‌توان در همین قسمت مقادیر culture و uiCulture را دستی تنظیم نمود و یا اگر همانند برنامه‌هایی که چند لینک را بالای صفحه نمایش می‌دهند که برای مثال به نگارش‌های فارسی/عربی/انگلیسی اشاره می‌کند، اینکار را با کد نویسی نیز می‌توان انجام داد:

```
System.Threading.Thread.CurrentThread.CurrentCulture =
                                System.Globalization.CultureInfo.CreateSpecificCulture("fa");
```

جهت آزمایش این مطلب، ابتدا تنظیم globalization فوق را به فایل وب کانفیگ برنامه اضافه کنید. سپس به مسیر زیر در IE مراجعه کنید:

```
IE -> Tools -> Intenet options -> Genarl tab -> Languages
```

در اینجا می‌توان هدر Accept-Language را مقدار دهی کرد. برای نمونه اگر مقدار زبان پیش فرض را به فرانسه تنظیم کنیم (به عنوان اولین زبان تعریف شده در لیست) و سپس سعی در نمایش مقدار decimal زیر را داشته باشیم:

```
string.Format("{0:C}", 10.5M)
```

اگر زبان پیش فرض، انگلیسی آمریکایی باشد، $ نمایش داده خواهد شد و اگر زبان به فرانسه تنظیم شود، یورو در کنار عدد مبلغ نمایش داده می‌شود.
تا اینجا تنها با تنظیم culture=auto به این نتیجه رسیده‌ایم. اما سایر قسمت‌های صفحه چطور؟ برای مثال برچسب‌های نمایش داده شده را چگونه می‌توان به صورت خودکار بر اساس Accept-Language مرجح کاربر تنظیم کرد؟ خوشبختانه در دات نت، زیر ساخت مدیریت برنامه‌های چند زبانه به صورت توکار وجود دارد که در ادامه به بررسی آن خواهیم پرداخت.

آشنایی با ساختار فایل‌های Resource

فایل‌های Resource یا منبع، در حقیقت فایل‌هایی هستند مبتنی بر XML با پسوند resx و هدف آن‌ها ذخیره سازی رشته‌های متناظر با فرهنگ‌های مختلف می‌باشد و برای استفاده از آن‌ها حداقل یک فایل منبع پیش فرض باید تعریف شود. برای نمونه فایل mydata.resx را در نظر بگیرید. برای ایجاد فایل منبع اسپانیایی متناظر، باید فایلی را به نام mydata.es.resx تولید کرد. البته نوع فرهنگ مورد استفاده را کاملتر نیز می‌توان ذکر کرد برای مثال mydata.es-mex.resx جهت فرهنگ اسپانیایی مکزیکی بکارگرفته خواهد شد، یا mydata.fr-ca.resx به فرانسوی کانادایی اشاره می‌کند. سپس مدیریت منابع دات نت فریم ورک بر اساس مقدار CurrentUICulture جاری، اطلاعات فایل متناظری را بارگذاری خواهد کرد. اگر فایل متناظری وجود نداشت، از اطلاعات همان فایل پیش فرض استفاده می‌گردد.
حین تهیه برنامه‌ها نیازی نیست تا مستقیما با فایل‌های XML منابع کار کرد. زمانیکه اولین فایل منبع تولید می‌شود، به همراه آن یک فایل cs یا vb نیز ایجاد خواهد شد که امکان دسترسی به کلیدهای تعریف شده در فایل‌های XML را به صورت strongly typed میسر می‌کند. این فایل‌های خودکار، تنها برای فایل پیش فرض mydata.resx تولید می‌شوند،از این جهت که تعاریف اطلاعات سایر فرهنگ‌های متناظر نیز باید با همان کلیدهای فایل پیش فرض آغاز شوند. تنها «مقادیر» کلیدهای تعریف شده در کلاس‌های منبع متفاوت هستند.
اگر به خواص فایل‌های resx در VS.NET دقت کنیم، نوع Build action آن‌ها به embedded resource تنظیم شده است.

مثالی جهت بررسی استفاده از فایل‌های Resource

یک پروژه جدید خالی ASP.NET MVC را آغاز کنید. فایل وب کانفیگ آن‌را ویرایش کرده و تنظیمات globalization ابتدای بحث را به آن اضافه کنید. سپس مدل، کنترلر و View متناظر با متد Index آن‌را با محتوای زیر به پروژه اضافه نمائید:

```
namespace MvcApplication19.Models
{
    public class Employee
    {
        public int Id { set; get; }
        public string Name { set; get; }
    }
}
```

```
using System.Web.Mvc;
using MvcApplication19.Models;

namespace MvcApplication19.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            var employee = new Employee { Name = "Name 1" };
            return View(employee);
        }
    }
}
```

```
@model MvcApplication19.Models.Employee
@{
    ViewBag.Title = "Index";
}
<h2>
    Index</h2>
<fieldset>
    <legend>Employee</legend>
    <div class="display-label">
        Name
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Name)
    </div>
</fieldset>
<fieldset>
    <legend>Employee Info</legend>
    @Html.DisplayForModel()
</fieldset>
```

قصد داریم در View فوق بر اساس uiCulture کاربر مراجعه کننده به سایت، برچسب Name را مقدار دهی کنیم. اگر کاربری از ایران مراجعه کند، «نام کارمند» نمایش داده شود و سایر کاربران، «Employee Name» را مشاهده کنند. همچنین این تغییرات باید بر روی متد Html.DisplayForModel نیز تاثیرگذار باشد.

برای این منظور بر روی پوشه Views/Home که محل قرارگیری فایل Index.cshtml فوق است کلیک راست کرده و گزینه Add|New Item را انتخاب کنید. سپس در صفحه ظاهر شده، گزینه «Resources file» را انتخاب کرده و برای مثال نام Index_cshtml.resx را وارد کنید.

به این ترتیب اولین فایل منبع مرتبط با View جاری که فایل پیش فرض نیز می‌باشد ایجاد خواهد شد. این فایل، به همراه فایل Index_cshtml.Designer.cs تولید می‌شود. سپس همین مراحل را طی کنید، اما اینبار نام Index_cshtml. **fa** .resx را حین افزودن فایل منبع وارد نمائید که برای تعریف اطلاعات بومی ایران مورد استفاده قرار خواهد گرفت. فایل دومی که اضافه شده است، فاقد فایل cs همراه می‌باشد.

اکنون فایل Index_cshtml.resx را در VS.NET باز کنید. از بالای صفحه، به کمک گزینه Access modifier، سطح دسترسی متدهای فایل cs همراه آن‌را به public تغییر دهید. پیش فرض آن internal است که برای کار ما مفید نیست. از این جهت که امکان دسترسی به متدهای استاتیک تعریف شده در فایل خودکار Index_cshtml.Designer.cs را در View های برنامه، نخواهیم داشت.

سپس دو جفت «نام-مقدار» را در فایل resx وارد کنید. مثلا نام را Name و مقدار آن‌را «Employee Name» و سپس نام دیگر را NameIsNotRight و مقدار آن‌را «Name is required» وارد نمائید.

در ادامه فایل Index_cshtml.fa.resx را باز کنید. در اینجا نیز دو جفت «نام-مقدار» متناظر با فایل پیش فرض منبع را باید وارد کرد. کلیدها یا نام‌ها یکی است اما قسمت مقدار اینبار باید فارسی وارد شود. مثلا نام را Name و مقدار آن‌را «نام کارمند» وارد نمائید. سپس کلید یا نام NameIsNotRight و مقدار «لطفا نام را وارد نمائید» را تنظیم نمائید.

تا اینجا کار تهیه فایل‌های منبع متناظر با View جاری به پایان می‌رسد.

در ادامه با کمک فایل Index_cshtml.Designer.cs که هربار پس از تغییر فایل resx متناظر آن به صورت خودکار توسط VS.NET تولید و به روز می‌شود، می‌توان به کلیدها یا نام‌هایی که تعریف کرده‌ایم، در قسمت‌های مختلف برنامه دست یافت. برای نمونه تعریف کلید Name در این فایل به نحو زیر است:

```
namespace MvcApplication19.Views.Home {
    public class Index_cshtml {
        public static string Name {
            get {
                return ResourceManager.GetString("Name", resourceCulture);
            }
        }
    }
}
```

بنابراین برای استفاده از آن در هر View ایی تنها کافی است بنویسیم:

```
@MvcApplication19.Views.Home.Index_cshtml.Name
```

به این ترتیب بر اساس تنظیمات محلی کاربر، اطلاعات به صورت خودکار از فایل‌های Index_cshtml.fa.resx فارسی یا فایل پیش فرض Index_cshtml.resx، دریافت می‌گردد.
علاوه بر امکان دسترسی مستقیم به کلیدهای تعریف شده در فایل‌های منبع، امکان استفاده از آن‌ها توسط data annotations نیز میسر است. در این حالت می‌توان مثلا پیغام‌های اعتبار سنجی را بومی کرد یا حین استفاده از متد Html.DisplayForModel، بر روی برچسب نمایش داده شده خودکار، تاثیر گذار بود. برای اینکار باید اندکی مدل برنامه را ویرایش کرد:

```
using System.ComponentModel.DataAnnotations;

namespace MvcApplication19.Models
{
    public class Employee
    {
        [ScaffoldColumn(false)]
        public int Id { set; get; }

        [Display(ResourceType = typeof(MvcApplication19.Views.Home.Index_cshtml),
                 Name = "Name")]
        [Required(ErrorMessageResourceType = typeof(MvcApplication19.Views.Home.Index_cshtml),
                  ErrorMessageResourceName = "NameIsNotRight")]
        public string Name { set; get; }
    }
}
```

همانطور که ملاحظه می‌کنید، حین تعریف ویژگی‌های Display یا Required، امکان تعریف نام کلاس متناظر با فایل resx خاصی وجود دارد. به علاوه ErrorMessageResourceName به نام یک کلید در این فایل و یا پارامتر Name ویژگی Display نیز به نام کلیدی در فایل منبع مشخص شده، اشاره می‌کنند. این اطلاعات توسط متدهای Html.DisplayForModel، Html.ValidationMessageFor، Html.LabelFor و امثال آن به صورت خودکار مورد استفاده قرار خواهند گرفت.

نکته‌ای در مورد کش کردن اطلاعات

در این مثال اگر فیلتر OutputCache را بر روی متد Index تعریف کنیم، حتما نیاز است به هدر Accept-Language نیز دقت داشت. در غیراینصورت تمام کاربران، صرفنظر از تنظیمات بومی آن‌ها، یک صفحه را مشاهده خواهند کرد:

```
[OutputCache(Duration = 60, VaryByHeader = "Accept-Language")]
public ActionResult Index()
```

نظرات خوانندگان

نویسنده: Salehi
تاریخ: ۱۳۹۱/۰۲/۰۷ ۲۲:۳۹:۴۳

با تشکر از این آموزش جالب
-1 در مورد نامگذاری و محل قرارگیری فایل های resx نکته و ملاحظه خاصی وجود نداره؟
-2 اگه بخوایم قالب بندی و direction رو هم بر اساس cultureInof تعیین کنیم، می تونیم داخل resx مثلا خاصیتی به اسم dir تعریف کنیم و مثلا برای فارسی به اون مقدار rtl بدیم درسته؟ یا مثلا اسم و مسیر فایل css مربوطه رو هم به همین صورت. کلا این روش رو توصیه می کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۰۷ ۲۲:۵۹:۱۱

- در حالت کلی خیر. فقط اگر فایل پایه xyz.resx باشه، فایل‌های بعدی باید xyz.lang.resx باشند.
ضمنا برای این مسایل پوشه‌ی مخصوصی به نام «App_GlobalResources» در ASP.NET وجود دارد، ولی در ASP.NET MVC توصیه نمی‌شود. چون حاصل این فایل‌ها در زمان اجرا توسط موتور ASP.NET کامپایل می‌شوند و یک dll جداگانه را درست می‌کنند که مناسب استفاده در شرایط Unit test نیست (مقادیر نال خواهند بود). بنابراین بهتر است فایل‌های منبع هر View، داخل همان پوشه مربوطه تعریف شود تا بتوان ارتباط منطقی آن‌هارا پیدا کرد و همچنین در این حالت مشکلی با آزمون‌های واحد هم نخواهد بود.
- این روش هم خوبه. چون نهایتا میشه رشته رو توسط Razor خوند و در View یا master page درج کرد.
ضمن اینکه با استفاده از فایل منبع زبان فارسی (xyz.fa.resx)، این راست به چپ سازی حداقل به صفحات زرد خطای ASP.NET به صورت خودکار اعمال می‌شود. یکبار تست کنید، جالب است.

نویسنده: A. Karimi
تاریخ: ۱۳۹۱/۰۲/۰۸ ۱۱:۵۳:۲۶

چون مدل‌های ما در یک Assembly جداگانه قرار گرفته، با استفاده از typeof نمی توانیم به Resource داخل پروژه Web دسترسی داشته باشیم. به همین خاطر از Attribute های مربوط به DataAnnonation ارثبری کردم و با گرفتن رشته حاوی مشخصات Resource با استفاده از Reflection به Reousrce Type دسترسی پیدا کردم. به نظر شما راه بهتری هست؟ (با بردن Resourceها به Assemblyها دیگر هم مشکل داشتیم.)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۰۸ ۱۳:۰۷:۱۳

یک پروژه نمونه که مدل آن به اسمبلی دیگری منتقل شده. همچنین منابع آن هم در اسمبلی جداگانه‌ای قرار دارند و نهایتا از تمام این‌ها در پروژه اصلی استفاده شده و بدون مشکل کار می‌کند: (^)
حدس من این است که از نکته تغییر «Access modifier» که در متن اشاره کردم، استفاده نکردید و هنوز سطح دسترسی منابع شما internal است که باید public شود.

نویسنده: A. Karimi
تاریخ: ۱۳۹۱/۰۲/۰۹ ۰۲:۱۲:۵۹

بله این نمونه به خوبی کار کرد. با تغییر Culture در Action فارسی هم به خوبی کار کرد. فراموش کردم کجای کار مشکل داشت که این روش برایمان مشکل ساز شده بود! اما قریب به یقین مشکل internal نبود.

گذشته از این موضوع؛ کلاً جدا کردن Resource ها در یک Assembly دیگر بهتر است یا در پروژه UI؟ البته مطمئناً کمترین حسن آن استفاده مجدد Resourceها در پروژه‌هایی است که پلتفرمهای مختلف دارند مثل Win و Web یا حتی Mobile. به طور عملی در پروژه‌ای این جداسازی Resource را انجام ندادم. شما تجربه این کار را در پروژه‌های بزرگ داشتید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۰۹ ۰۹:۳۵:۱۴

فکر نمی‌کنم کسی این دور و اطراف بزرگتر از SharePoint پروژه‌ای داشته باشه. SharePoint هم به وفور از فایل‌های Resx استفاده می‌کنه. محل آن‌ها هم پوشه‌های زیر داخل خود پروژه است و یا در اسمبلی‌های جداگانه:

C:\inetpub\wwwroot\wss\VirtualDirectories\80\App_GlobalResources

C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\Resources

C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\CONFIG\Resources

نویسنده: محسن.د
تاریخ: ۱۳۹۱/۰۷/۲۴ ۹:۵۴

سلام
سوالی که برای من پیش اومده اینه که مگه این فایل‌های Resource برای استفاده در UI (فرم‌ها و viewها) ساخته نشدن (حداقل استفاده اصلی اونها در این قسمت هاست) پس این قضیه سطح دسترسی internal چیه ، متوجه نشدم چرا ؟

ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۴ ۱۲:۲۳

فایل Index_cshtml.Designer.cs ذکر شده، یک فایل code behind متناظر و همراه فایل منبع است که باید بتوان بنابر سلیقه یا طراحی خاصی، سطوح دسترسی مختلفی را به آن اعمال کرد.

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۲۳ ۱۷:۳۳

با سلام
اگر ما بخواهیم به عنوان مثال یک سایت خبری بنویسیم که چند زبانه باشه باید داده هایی هم که مدیر سایت در database وارد میکنه دارای این ویژگی باشه تا به هر زبانی که کاربر میخواد تبدیل بشه
ایا میشه از همین روش در این پروژه فرضی استفاده کرد
اگر امکان پذیر هست ایا میشه روی ترجمه سایت حساب باز کرد.
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۲۳ ۱۷:۴۲

- نسخه کاملتر مطلب جاری رو اینجا می‌تونید مطالعه کنید.
- هدف از این قسمت ارائه راهکاری برای refactoring رشته‌های ثابت بکار رفته در قسمت‌های مختلف برنامه است.
- برای ارائه اخبار چند زبانه که اطلاعات آن پویا است و هر روز تغییر می‌کند نیاز به طراحی دیتابیس، کاربر ویژه و سازمانی برای مدیریت و ترجمه مطالب خواهید داشت.

نویسنده: ایمان
تاریخ: ۱۳۹۱/۱۱/۲۳ ۱۹:۵۲

جناب نصیری به نظر شما بهتر نیست ریسورس‌های متنی به صورت key،Value در دیتابیس ذخیره شوند و برای export، import از xml استفاده کنیم؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۱/۱۱/۲۳ ۲۰:۳۱

شما برنامه نویس هستید. روشی خاصی نمی‌تونه شما رو محدود کنه. فقط دست آخر باید همین فریم ورک و جزئیات اون رو از صفربازنویسی کنید. به علاوه بحث caching رو هم اضافه کنید تا بار دیتابیس رو کم کنید. در حالیکه این اطلاعات چیزی نیستند که هر روز تغییر کنند؛ جزو ثوابت برنامه به شمار می‌روند.
به علاوه روش بحث جاری هم آماده است و تست شده، هم استاندارد، سازگار با اجزای مختلف MVC با سربار حداقل و همچنین فقط منحصر به MVC نیست و با کل دات نت و فناوری‌های وابسته به آن یکپارچه است.

نویسنده: حسام محمدی
تاریخ: ۱۳۹۱/۱۱/۳۰ ۱۱:۵۲

سلام. در این روش شما گفتید که برای هر View یک فایل منبع درست کنیم حالا به نظر شما بهتر نیست تا یک فایل منبع مثلاً به اسم EntitiesResource درست کرد و پروپرتی‌ها رو تماماً داخل اون گذاشت و تمام Viewها از اون استفاده کنند؟
و برای کنترل‌ها هم به همین شکل (ControlsResource).
میخواستم بدونم این روش اشتباهه؟

نویسنده: میثم خوشقدم
تاریخ: ۱۳۹۲/۰۲/۲۹ ۱۶:۴۰

سلام

ResourceType یکبار می‌بایست برای Display ست شود و یکبار هم برای ErrorMessageResourceType

که هر دوی این‌ها معمولا یکی است . همچنین در کلاس هر پراپرتی (Poco) ممکن است یک ErrorMessageResourceType داشته باشد که برای کل کلاس تکراری است.

میشه ResourceType را برای کل یک کلاس یا یک NameSpace مشخص کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۲۹ ۱۷:۴۷

حالت پیش فرض همین هست. اگر مطابق سلیقه شما نیست، می‌شود اون رو سفارشی کرد:

```
public class GlobalResourceTypeResourceDataAnnotationsModelValidator :
DataAnnotationsModelValidator<ValidationAttribute>
{
    public GlobalResourceTypeResourceDataAnnotationsModelValidator(
        ModelMetadata metadata,
        ControllerContext context,
        ValidationAttribute attribute
    )
        : base(metadata, context, attribute)
    {
        if (Attribute.ErrorMessageResourceType == null)
        {
            Attribute.ErrorMessageResourceType = typeof(ModelValidationMessages);
        }
    }
}
```

و بعد در Application_Start برنامه به ازای هر ویژگی مورد نظر یکبار باید معرفی و ثبت شود:

```
DataAnnotationsModelValidatorProvider.RegisterAdapter(typeof(RequiredAttribute),
    typeof(GlobalResourceTypeResourceDataAnnotationsModelValidator));
DataAnnotationsModelValidatorProvider.RegisterAdapter(typeof(StringLengthAttribute),
   typeof(GlobalResourceTypeResourceDataAnnotationsModelValidator));
```

نویسنده: آروین
تاریخ: ۱۳۹۲/۰۹/۲۱ ۱۲:۵۳

با عرض سلام و خسته نباشید.
من دو تا فایل Resource داشتم برای زبان انگلیسی و فارسی آیا مقادیر فایل‌های Resource را می‌توان داخل کد C# مقداردهی کرد مثلا یک مقدار String همراه با مقدار آن را بصورت Dynamic در کد C# به ازای هر کدام از این زبانها وارد کرد و در فایل Resource متناظر آنها اضافه شوند.
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۱ ۱۳:۳

برای کارهای پویا گروه Globalization را در سایت پیگیری کنید.

نویسنده: میثم فغفوری
تاریخ: ۱۳۹۲/۱۱/۱۵ ۲۲:۳۹

ممنون؛ من مثلا یه ریسورس ساختم به اسم lang.resx که دیفالت هستش و یدونه هم برای فارسی به اسم lang.fa.resx یا خود مثال‌های شما. سوالم اینه که اصلا ASP.NET از کجا متوجه میشه فایل lang.fa متعلق به زبان فارسی هستش؟ استاندارد توکاری وجود داره که مثلا آخر اسم‌ها به fa یا en یا ... ختم بشه؟ همچنین گزینه Access modifier برای ریسورس‌های من کلا غیر فعال هستش.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۵ ۲۳:۵۴

- این خلاصه نام‌ها، استاندارد جهانی هستند.
- احتمالا گزینه‌ی Custom Tool فایل منبع را تغییر داده‌اید.

نویسنده: میثم فغفوری
تاریخ: ۱۳۹۲/۱۱/۱۶ ۱۳:۴۱

چند زبانه کردن کلاس‌های POCO توسط Localresourcesها امکان پذیر نیست؟ برای مثال من توی پوشه Models مربوط به کلاس‌های POCO به ازای هر کلاس یک پوشه ساختم و توی اون پوشه هم یک پوشه App_LocalResource که نام فیلدها و اعتبار سنجی‌های اون کلاس رو هم به صورت فارسی Fields.resx و هم به صورت انگلیسی Fields.en.resx ایجاد کردم در رویداد BeginRequest هم سعی کردم با این کدها سوئیچ کنم بین زبان‌ها برای تست مساله.

```
Thread.CurrentThread.CurrentUICulture = new CultureInfo("en-US");
            Thread.CurrentThread.CurrentUICulture = CultureInfo.CreateSpecificCulture("en-US");
```

ولی هیچ تغییری صورت نمیگیره و فقط فیلدهارو به اسم فارسیشون میاره ممنون میشم راهنماییم کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۶ ۱۴:۰

کمی بالاتر در بین نظرات «یک پروژه نمونه قابل دریافت که مدل آن به اسمبلی دیگری منتقل شده. همچنین منابع آن هم در اسمبلی جداگانه‌ای قرار دارند» ارسال شده .

نویسنده: بهزاد
تاریخ: ۱۳۹۲/۱۱/۳۰ ۱۷:۴۵

سلام ممنون آموزش خیلی مفیدی بود.
یک مشکل! در خصوصیات Data annotation وقتی خصوصیات رو به ریسورس‌ها مرتبط میکنیم خصوصیت فوق دیگه تاثیری روی پایگاه داده نمیذاره مثلا وقتی روی فیلدی خصوصیت [Requied] رو به تنهایی تنظیم میکنیم توی دیتابیس هم اعمال میشه ولی همین خصوصیتو با ریسورس‌ها اضافه میکنیم مثلا : [Required(ErrorMessageResourceType = typeof(MvcApplication19.Views.Home.Index_cshtml), ErrorMessageResourceName = "NameIsNotRight")] دیگه فیلد فوق توی دیتابیس Not NULL نمیشه چند بار هم تست کردم نشد توی این موارد حتما باید از Fluent استفاده کنیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۳۰ ۲۰:۱۲

با EF 6.0.2 مشکلی مشاهده نشد. یک مثال برای آزمایش: Sample31.cs

نویسنده: بهزاد
تاریخ: ۱۳۹۲/۱۱/۳۰ ۲۰:۴۷

ممنون درست شد مشکل این بود که من با ابزار ef power tools خروجی SQL میگرفتم و بعد روی دیتابیس اجرا میکردم تا ساخته شه ظاهرا این ابزار این موارد رو پیش بینی نکرده و اعمال نمیکنه چون با دستور update-database که اجرا میکنم به طور صحیح اعمال میشه.

نویسنده: رشوند
تاریخ: ۱۳۹۳/۰۱/۲۸ ۱۵:۵۸

با سلام؛ بلافاصله بعد از استفاده (ایجاد) **fa**.resx Index_cshtml. خطای زیر به وجود می‌آید

```
|Error1Task could not find "AL.exe" using the SdkToolsPath "" or the registry key
"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SDKs\Windows\v8.0A\WinSDK-NetFx40Tools-x86".
 Make sure the SdkToolsPath is set and the tool exists in the correct processor specific location
under the SdkToolsPath and that the Microsoft Windows SDK is installed MvcAppProject
```

به نظرتان اشکال کجاست؟ راه حل چیست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۸ ۱۷:۴۳

فایل al.exe پوشه C:\Program Files (x86)\Microsoft SDKs\Windows\v8.0A\bin\NETFX 4.0 Tools را نمی‌تواند پیدا کند.
بررسی کنید آیا وجود دارد یا خیر؟ با نصب کامل VS.NET، باید موجود باشد. اگر خیر، باید دستی SDK ویندوز را نصب کنید: برای ویندوز 7 ، برای ویندوز 8
ضمنا باز هم مسیر C:\Program Files (x86)\Microsoft SDKs\Windows را برای یافتن al.exe جستجو کنید. اگر موجود است، مسیر یاد شده را دستی درست کرده و فایل al.exe و کانفیگ‌اش را در آنجا کپی کنید.
اطلاعات بیشتر