عنوان: **Globalization در ASP.NET MVC**
نویسنده: یوسف نژاد
تاریخ: ۱۳۹۱/۱۰/۲۱ ۱۶:۲۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net MVC, .NET, Localization, Globalization, Internationalization, Resource, .resx, MVC

اگر بازار هدف یک محصول شامل چندین کشور، منطقه یا زبان مختلف باشد، طراحی و پیاده سازی آن برای پشتیبانی از ویژگی‌های چندزبانه یک فاکتور مهم به حساب می‌آید. یکی از بهترین روشهای پیاده سازی این ویژگی در دات نت استفاده از فایلهای Resource است. درواقع هدف اصلی استفاده از فایلهای Resource نیز Globalization است. Globalization برابر است با Internationalization + Localization که به اختصار به آن g11n میگویند. در تعریف، Internationalization (یا به اختصار i18n) به فرایند طراحی یک محصول برای پشتیبانی از فرهنگ(culture)ها و زبانهای مختلف و Localization (یا L10n) یا بومی‌سازی به شخصی‌سازی یک برنامه برای یک فرهنگ یا زبان خاص گفته میشود. (اطلاعات بیشتر در اینجا).
استفاده از این فایلها محدود به پیاده سازی ویژگی چندزبانه نیست. شما میتوانید از این فایلها برای نگهداری تمام رشته‌های موردنیاز خود استفاده کنید. نکته دیگری که باید بدان اشاره کرد این است که تقریبا تمامی منابع مورد استفاده در یک محصول را میتوان درون این فایلها ذخیره کرد. این منابع در حالت کلی شامل موارد زیر است:
- انواع رشته‌های مورد استفاده در برنامه چون لیبل‌ها و پیغام‌ها و یا مسیرها (مثلا نشانی تصاویر یا نام کنترلرها و اکشنها) و یا حتی برخی تنظیمات ویژه برنامه (که نمیخواهیم براحتی قابل نمایش یا تغییر باشد و یا اینکه بخواهیم با تغییر زبان تغییر کنند مثل direction و امثال آن)
- تصاویر و آیکونها و یا فایلهای صوتی و انواع دیگر فایل ها
- و ...
نحوه بهره برداری از فایلهای Resource در دات نت، پیاده سازی نسبتا آسانی را در اختیار برنامه نویس قرار میدهد. برای استفاده از این فایلها نیز روشهای متنوعی وجود دارد که در مطلب جاری به چگونگی استفاده از آنها در پروژه‌های ASP.NET MVC پرداخته میشود.

Globalization در دات نت

فرمت نام یک culture دات نت (که در کلاس CultureInfo پیاده شده است) بر اساس استاندارد RFC 4646 (^ و ^) است. (در اینجا اطلاعاتی راجع به RFC یا Request for Comments آورده شده است). در این استاندارد نام یک فرهنگ (کالچر) ترکیبی از نام زبان به همراه نام کشور یا منطقه مربوطه است. نام زبان برپایه استاندارد ISO 639 که یک عبارت دوحرفی با حروف کوچک برای معرفی زبان است مثل fa برای فارسی و en برای انگلیسی و نام کشور یا منطقه نیز برپایه استاندارد ISO 3166 که به عبارت دوحرفی با حروف بزرگ برای معرفی یک کشور یا یک منطقه است مثل IR برای ایران یا US برای آمریکاست. برای نمونه میتوان به fa-IR برای زبان فارسی کشور ایران و یا en-US برای زبان انگلیسی آمریکایی اشاره کرد. البته در این روش نامگذاری یکی دو مورد استثنا هم وجود دارد (اطلاعات کامل کلیه زبانها: National Language Support (NLS) API Reference). یک فرهنگ خنثی (Neutral Culture) نیز تنها با استفاده از دو حرف نام زبان و بدون نام کشور یا منطقه معرفی میشود. مثل fa برای فارسی یا de برای آلمانی. در این بخش نیز دو استثنا وجود دارد (^).
در دات نت دو نوع culture وجود دارد: **Culture** و **UICulture** . هر دوی این مقادیر در هر Thread مقداری منحصربه فرد دارند. مقدار Culture بر روی توابع وابسته به فرهنگ (مثل فرمت رشته‌های تاریخ و اعداد و پول) تاثیر میگذارد. اما مقدار UICulture تعیین میکند که سیستم مدیریت منابع دات نت (Resource Manager) از کدام فایل Resource برای بارگذاری داده‌ها استفاده کند. درواقع در دات نت با استفاده از پراپرتی‌های موجود در کلاس استاتیک Thread برای ثرد جاری (که عبارتند از CurrentCulture و CurrentUICulture) برای فرمت کردن و یا انتخاب Resource مناسب تصمیم گیری میشود. برای تعیین کالچر جاری به صورت دستی میتوان بصورت زیر عمل کرد:

```
Thread.CurrentThread.CurrentUICulture = new CultureInfo("fa-IR");
Thread.CurrentThread.CurrentUICulture = CultureInfo.CreateSpecificCulture("fa-IR");
```

دراینجا باید اشاره کنم که کار انتخاب Resource مناسب با توجه به کالچر ثرد جاری توسط ResourceProviderFactory پیشفرض دات نت انجام میشود. در مطالب بعدی به نحوه تعریف یک پرووایدر شخصی سازی شده هم خواهم پرداخت.

پشتیبانی از زبانهای مختلف در MVC

برای استفاده از ویژگی چندزبانه در MVC دو روش کلی وجود دارد.

1. استفاده از فایلهای Resource برای تمامی رشته‌های موجود
2. استفاده از Viewهای مختلف برای هر زبان

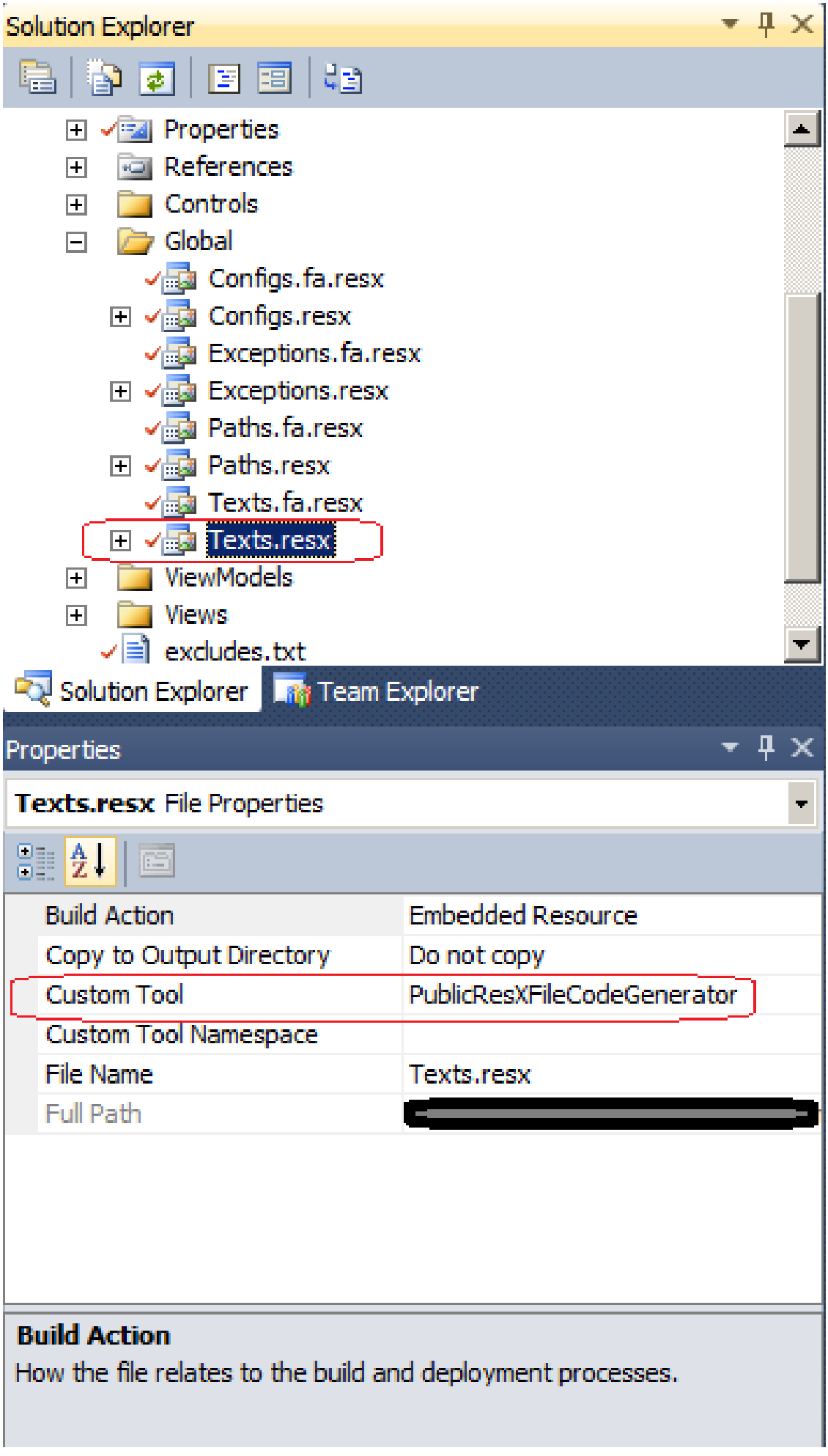
البته روش سومی هم که از ترکیب این دو روش استفاده میکند نیز وجود دارد. انتخاب روش مناسب کمی به سلیقه‌ها و عادات برنامه نویسی بستگی دارد. اگر فکر میکنید که استفاده از ویوهای مختلف به دلیل جداسازی مفاهیم درگیر در کالچرها (مثل جانمایی اجزای مختلف ویوها یا بحث Direction) باعث مدیریت بهتر و کاهش هزینه‌های پشتیبانی میشود بهتر است از روش دوم یا ترکیبی از این دو روش استفاده کنید. خودم به شخصه سعی میکنم از روش اول استفاده کنم. چون معتقدم استفاده از ویوهای مختلف باعث افزایش بیش از اندازه حجم کار میشود. اما در برخی موارد استفاده از روش دوم یا ترکیبی از دو روش میتواند بهتر باشد.

تولید فایلهای Resource

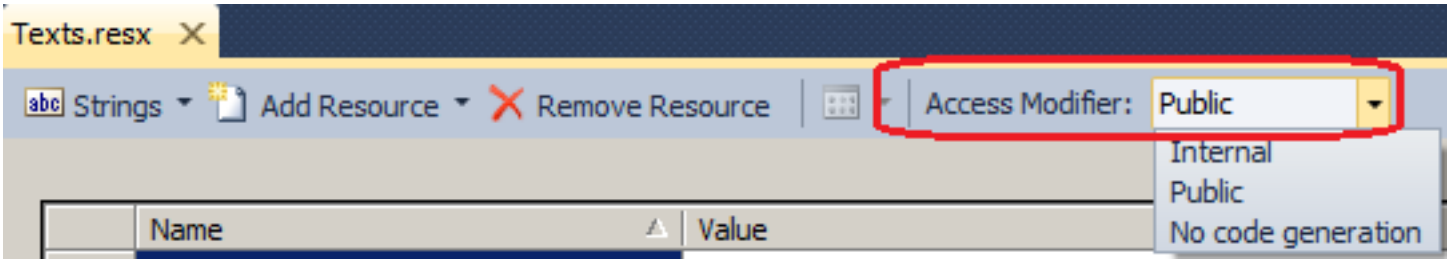
بهترین مکان برای نگهداری فایلهای Resource در یک پروژه جداگانه است. در پروژه‌های از نوع وب‌سایت پوشه‌هایی با نام App_GlobalResources یا App_LocalResources وجود دارد که میتوان از آنها برای نگهداری و مدیریت این نوع فایلها استفاده کرد. اما همانطور که در اینجا توضیح داده شده است این روش مناسب نیست. بنابراین ابتدا یک پروژه مخصوص نگهداری فایلهای Resource ایجاد کنید و سپس اقدام به تهیه این فایلها نمایید. سعی کنید که عنوان این پروژه به صورت زیر باشد. برای کسب اطلاعات بیشتر درباره نحوه نامگذاری اشیای مختلف در دات نت به این مطلب رجوع کنید.

SolutionName>.Resources<

برای افزودن فایلهای Resource به این پروژه ابتدا برای انتخاب زبان پیش فرض محصول خود تصمیم بگیرید. پیشنهاد میکنم که از زبان انگلیسی (en-US) برای اینکار استفاده کنید. ابتدا یک فایل Resource (با پسوند resx.) مثلا با نام Texts.resx به این پروژه اضافه کنید. با افزودن این فایل به پروژه، ویژوال استودیو به صورت خودکار یک فایل cs. حاوی کلاس متناظر با این فایل را به پروژه اضافه میکند. این کار توسط ابزار توکاری به نام ResXFileCodeGenerator انجام میشود. اگر به پراپرتی‌های این فایل resx. رجوع کنید میتوانید این عنوان را در پراپرتی Custom Tool ببینید. البته ابزار دیگری برای تولید این کلاسها نیز وجود دارد. این ابزارهای توکار برای سطوح دسترسی مختلف استفاده میشوند. ابزار پیش فرض در ویژوال استودیو یعنی همان ResXFileCodeGenerator، این کلاسها را با دسترسی internal تولید میکند که مناسب کار ما نیست. ابزار دیگری که برای اینکار درون ویژوال استودیو وجود دارد PublicResXFileCodeGenerator است و همانطور که از نامش پیداست از سطح دسترسی public استفاده میکند. برای تغییر این ابزار کافی است تا عنوان آن را دقیقا در پراپرتی Custom Tool تایپ کنید.


Solution Explorer
Properties
References
Controls
Global
Configs.fa.resx
Configs.resx
Exceptions.fa.resx
Exceptions.resx
Paths.fa.resx
Paths.resx
Texts.fa.resx
Texts.resx
ViewModels
Views
excludes.txt
Solution Explorer
Team Explorer
Properties
Texts.resx File Properties
Build Action
Embedded Resource
Copy to Output Directory
Do not copy
Custom Tool
PublicResXFileCodeGenerator
Custom Tool Namespace
File Name
Texts.resx
Full Path
Build Action
How the file relates to the build and deployment processes.

نکته: درباره پراپرتی مهم Build Action این فایلها در مطالب بعدی بیشتر بحث میشود.
برای تعیین سطح دسترسی Resource موردنظر به روشی دیگر، میتوانید فایل Resource را باز کرده و Access Modifier آن را به Public تغییر دهید.



سپس برای پشتیبانی از زبانی دیگر، یک فایل دیگر Resource به پروژه اضافه کنید. نام این فایل باید همنام فایل اصلی به همراه نام کالچر موردنظر باشد. مثلا برای زبان فارسی عنوان فایل باید Texts.fa-IR.resx یا به صورت ساده‌تر برای کالچر خنثی (بدون نام کشور) Texts.fa.resx باشد. دقت کنید اگر نام فایل را در همان پنجره افزودن فایل وارد کنید ویژوال استودیو این همنامی را به صورت هوشمند تشخیص داده و تغییراتی را در پراپرتی‌های پیش فرض فایل Resource ایجاد میکند.
نکته: این هوشمندی مرتبه نسبتا بالایی دارد. بدین صورت که تنها درصورتیکه عبارت بعد از نام فایل اصلی Resource (رشته بعد از نقطه مثلا fa در اینجا) متعلق به یک کالچر معتبر باشد این تغییرات اعمال خواهد شد.
مهمترین این تغییرات این است که ابزاری را برای پراپرتی Custom Tool این فایلها انتخاب نمیکند! اگر به پراپرتی فایل Texts.fa.resx مراجعه کنید این مورد کاملا مشخص است. در نتیجه دیگر فایل cs. حاوی کلاسی جداگانه برای این فایل ساخته نمیشود. همچنین اگر فایل Resource جدید را باز کنید میبینید که برای Access Modifier آن گزینه No Code Generation انتخاب شده است.

در ادامه شروع به افزودن عناوین موردنظر در این دو فایل کنید. در اولی (بدون نام زبان) رشته‌های مربوط به زبان انگلیسی و در دومی رشته‌های مربوط به زبان فارسی را وارد کنید. سپس در هرجایی که یک لیبل یا یک رشته برای نمایش وجود دارد از این کلیدهای Resource استفاده کنید مثل:

SolutionName>.Resources.Texts.Save>

SolutionName>.Resources.Texts.Cancel>

استفاده از Resource در ویومدل ها

دو خاصیت معروفی که در ویومدلها استفاده میشوند عبارتند از: DisplayName و Required. پشتیبانی از کلیدهای Resource به صورت توکار در خاصیت Required وجود دارد. برای استفاده از آنها باید به صورت زیر عمل کرد:

```
[Required(ErrorMessageResourceName = "ResourceKeyName", ErrorMessageResourceType =
typeof(<SolutionName>.Resources.<ResourceClassName>))]
```

در کد بالا باید از نام فایل Resource اصلی (فایل اول که بدون نام کالچر بوده و به عنوان منبع پیشفرض به همراه یک فایل cs. حاوی کلاس مربوطه نیز هست) برای معرفی ErrorMessageResourceType استفاده کرد. چون ابزار توکار ویژوال استودیو از نام این فایل برای تولید کلاس مربوطه استفاده میکند.
متاسفانه خاصیت DisplayName که در فضای نام System.ComponentModel (در فایل System.dll) قرار دارد قابلیت استفاده از کلیدهای Resource را به صورت توکار ندارد. در دات نت 4 خاصیت دیگری در فضای نام System.ComponentModel.DataAnnotations به نام Display (در فایل System.ComponentModel.DataAnnotations.dll) وجود دارد که این امکان را به صورت توکار دارد. اما قابلیت استفاده از این خاصیت تنها در MVC 3 وجود دارد. برای نسخه‌های قدیمیتر

MVC امکان استفاده از این خاصیت حتی اگر نسخه فریمورک هدف 4 باشد وجود ندارد، چون هسته این نسخه‌های قدیمی امکان استفاده از ویژگی‌های جدید فریمورک با نسخه بالاتر را ندارد. برای رفع این مشکل میتوان کلاس خاصیت DisplayName را برای استفاده از خاصیت Display به صورت زیر توسعه داد:

```
public class LocalizationDisplayNameAttribute : DisplayNameAttribute
  {
    private readonly DisplayAttribute _display;
    public LocalizationDisplayNameAttribute(string resourceName, Type resourceType)
    {
      _display = new DisplayAttribute { ResourceType = resourceType, Name = resourceName };
    }
    public override string DisplayName
    {
      get
      {
        try
        {
          return _display.GetName();
        }
        catch (Exception)
        {
          return _display.Name;
        }
      }
    }
  }
```

در این کلاس با ترکیب دو خاصیت نامبرده امکان استفاده از کلیدهای Resource فراهم شده است. در پیاده سازی این کلاس فرض شده است که نسخه فریمورک هدف حداقل برابر 4 است. اگر از نسخه‌های پایین‌تر استفاده میکنید در پیاده سازی این کلاس باید کاملا به صورت دستی کلید موردنظر را از Resource معرفی شده بدست آورید. مثلا به صورت زیر:

```
public class LocalizationDisplayNameAttribute : DisplayNameAttribute
{
    private readonly PropertyInfo nameProperty;
    public LocalizationDisplayNameAttribute(string displayNameKey, Type resourceType = null)
        : base(displayNameKey)
    {
        if (resourceType != null)
            nameProperty = resourceType.GetProperty(base.DisplayName, BindingFlags.Static |
BindingFlags.Public);
    }
    public override string DisplayName
    {
        get
        {
            if (nameProperty == null) base.DisplayName;
            return (string)nameProperty.GetValue(nameProperty.DeclaringType, null);
        }
    }
}
```

برای استفاده از این خاصیت جدید میتوان به صورت زیر عمل کرد:

```
[LocalizationDisplayName("ResourceKeyName", typeof(<SolutionName>.Resources.<ResourceClassName>))]
```

البته بیشتر خواص متداول در ویومدلها از ویژگی موردبحث پشتیبانی میکنند.

نکته: به کار گیری این روش ممکن است در پروژه‌های بزرگ کمی گیج کننده و دردسرساز بوده و باعث پیچیدگی بی‌مورد کد و نیز افزایش بیش از حد حجم کدنویسی شود. در مقاله آقای فیل هک (Model Metadata and Validation Localization using Conventions) روش بهتر و تمیزتری برای مدیریت پیامهای این خاصیت‌ها آورده شده است.

پشتیبانی از ویژگی چند زبانه

مرحله بعدی برای چندزبانه کردن پروژه‌های MVC تغییراتی است که برای مدیریت Culture جاری برنامه باید پیاده شوند. برای

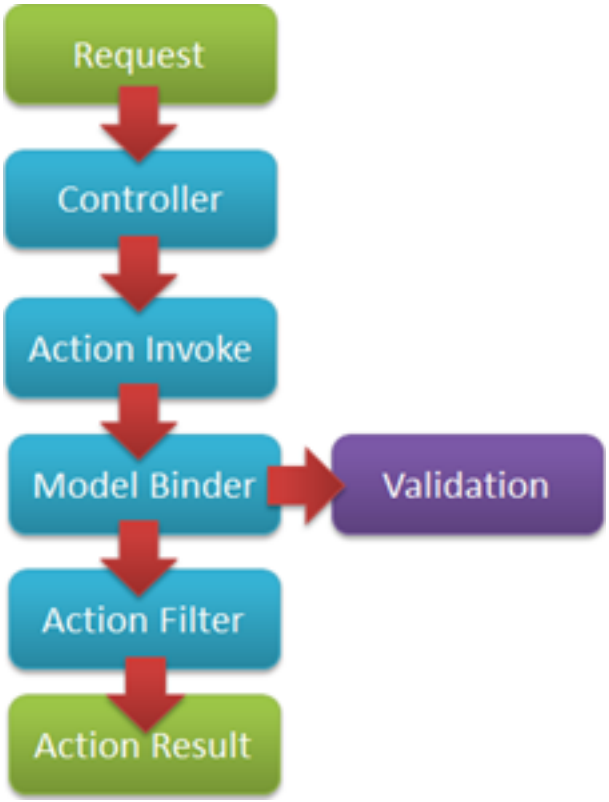
اینکار باید خاصیت CurrentUICulture در ثرد جاری کنترل و مدیریت شود. یکی از مکانهایی که برای نگهداری زبان جاری استفاده میشود کوکی است. معمولا برای اینکار از کوکی‌های دارای تاریخ انقضای طولانی استفاده میشود. میتوان از تنظیمات موجود در فایل کانفیگ برای ذخیره زبان پیش فرض سیستم نیز استفاه کرد.
روشی که معمولا برای مدیریت زبان جاری میتوان از آن استفاده کرد پیاده سازی یک کلاس پایه برای تمام کنترلرها است. کد زیر راه حل نهایی را نشان میدهد:

```
public class BaseController : Controller
  {
    private const string LanguageCookieName = "MyLanguageCookieName";
    protected override void ExecuteCore()
    {
      var cookie = HttpContext.Request.Cookies[LanguageCookieName];
      string lang;
      if (cookie != null)
      {
        lang = cookie.Value;
      }
      else
      {
        lang = ConfigurationManager.AppSettings["DefaultCulture"] ?? "fa-IR";
        var httpCookie = new HttpCookie(LanguageCookieName, lang) { Expires = DateTime.Now.AddYears(1)
};
        HttpContext.Response.SetCookie(httpCookie);
      }
      Thread.CurrentThread.CurrentUICulture = CultureInfo.CreateSpecificCulture(lang);
      base.ExecuteCore();
    }
  }
```

راه حل دیگر استفاده از یک ActionFilter است که نحوه پیاده سازی یک نمونه از آن در زیر آورده شده است:

```
public class LocalizationActionFilterAttribute : ActionFilterAttribute
  {
    private const string LanguageCookieName = "MyLanguageCookieName";
    public override void OnActionExecuting(ActionExecutingContext filterContext)
    {
      var cookie = filterContext.HttpContext.Request.Cookies[LanguageCookieName];
      string lang;
      if (cookie != null)
      {
        lang = cookie.Value;
      }
      else
      {
        lang = ConfigurationManager.AppSettings["DefaultCulture"] ?? "fa-IR";
        var httpCookie = new HttpCookie(LanguageCookieName, lang) { Expires = DateTime.Now.AddYears(1)
};
        filterContext.HttpContext.Response.SetCookie(httpCookie);
      }
      Thread.CurrentThread.CurrentUICulture = CultureInfo.CreateSpecificCulture(lang);
      base.OnActionExecuting(filterContext);
    }
  }
```

نکته مهم: تعیین زبان جاری (یعنی همان مقداردهی پراپرتی CurrentCulture ثرد جاری) در یک اکشن فیلتر بدرستی عمل نمیکند. برای بررسی بیشتر این مسئله ابتدا به تصویر زیر که ترتیب رخدادن رویدادهای مهم در ASP.NET MVC را نشان میدهد دقت کنید:



همانطور که در تصویر فوق مشاهده میکنید رویداد OnActionExecuting که در یک اکشن فیلتر به کار میرود بعد از عملیات مدل بایندینگ رخ میدهد. بنابراین قبل از تعیین کالچر جاری، عملیات validation و یافتن متن خطاها از فایلهای Resource انجام میشود که منجر به انتخاب کلیدهای مربوط به کالچر پیشفرض سرور (و نه آنچه که کاربر تنظیم کرده) خواهد شد. بنابراین استفاده از یک اکشن فیلتر برای تعیین کالچر جاری مناسب نیست. راه حل مناسب استفاده از همان کنترلر پایه است، زیرا متد ExecuteCore قبل از تمامی این عملیات صدا زده میشود. بنابراین همیشه کالچر تنظیم شده توسط کاربر به عنوان مقدار جاری آن در ثرد ثبت میشود.

امکان تعیین/تغییر زبان توسط کاربر

برای تعیین یا تغییر زبان جاری سیستم نیز روشهای گوناگونی وجود دارد. استفاده از زبان تنظیم شده در مرورگر کاربر، استفاده از عنوان زبان در آدرس صفحات درخواستی و یا تعیین زبان توسط کاربر در تنظیمات برنامه/سایت و ذخیره آن در کوکی یا دیتابیس و مواردی از این دست روشهایی است که معمولا برای تعیین زبان جاری از آن استفاده میشود. در کدهای نمونه ای که در بخشهای قبل آورده شده است فرض شده است که زبان جاری سیستم درون یک کوکی ذخیره میشود بنابراین برای استفاده از این روش میتوان از قطعه کدی مشابه زیر (مثلا در فایل _Layout.cshtml) برای تعیین و تغییر زبان استفاه کرد:

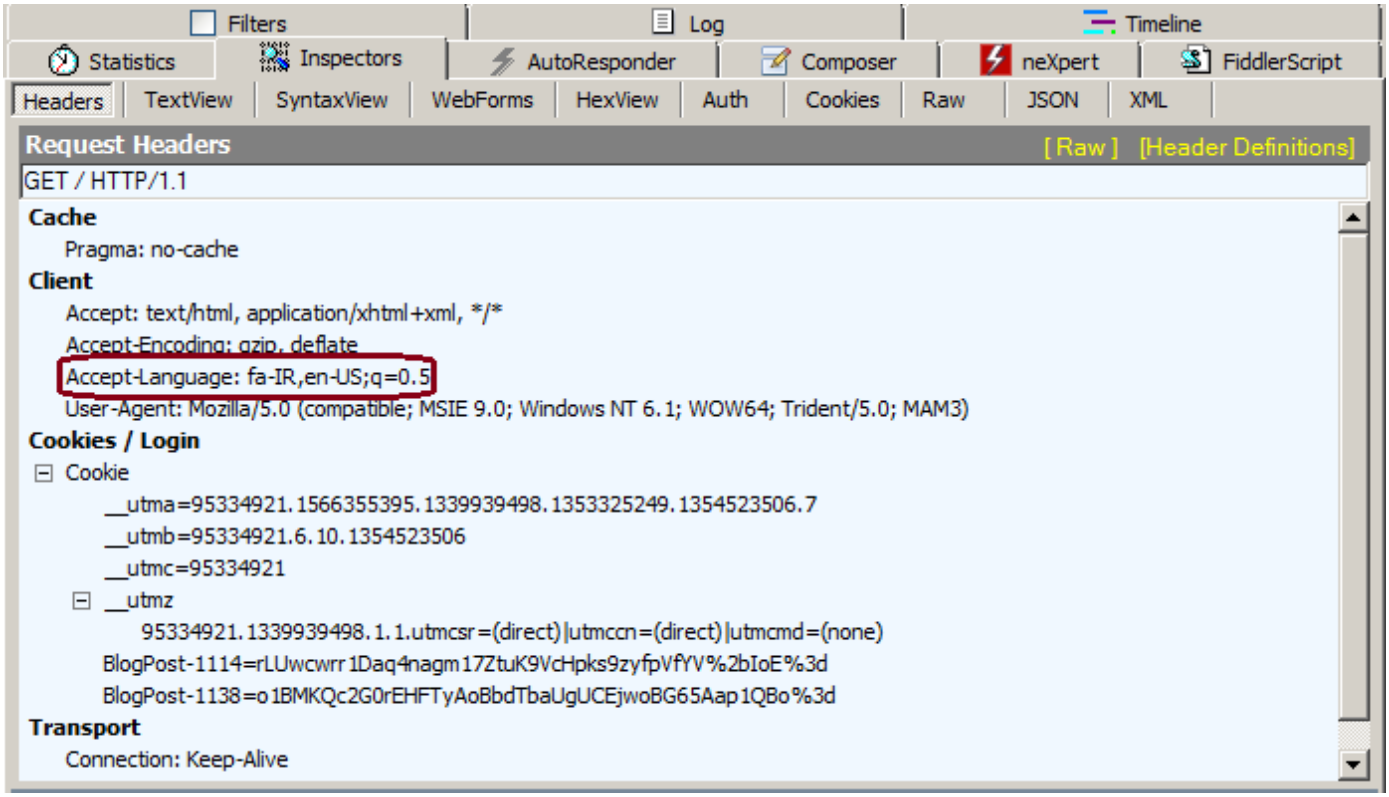
```
<select id="langs" onchange="languageChanged()">
  <option value="fa-IR">فارسی</option>
  <option value="en-US">انگلیسی</option>
</select>
<script type="text/javascript">
  function languageChanged() {
    setCookie("MyLanguageCookieName", $('#langs').val(), 365);
    window.location.reload();
  }
  document.ready = function () {
    $('#langs').val(getCookie("MyLanguageCookieName"));
  };
  function setCookie(name, value, exdays, path) {
    var exdate = new Date();
    exdate.setDate(exdate.getDate() + exdays);
    var newValue = escape(value) + ((exdays == null) ? "" : "; expires=" + exdate.toUTCString()) +
((path == null) ? "" : "; path=" + path) ;
    document.cookie = name + "=" + newValue;
  }
```

```
  function getCookie(name) {
    var i, x, y, cookies = document.cookie.split(";");
    for (i = 0; i < cookies.length; i++) {
      x = cookies[i].substr(0, cookies[i].indexOf("="));
      y = cookies[i].substr(cookies[i].indexOf("=") + 1);
      x = x.replace(/^\s+|\s+$/g, "");
      if (x == name) {
        return unescape(y);
      }
    }
  }
</script>
```

متدهای setCookie و getCookie جاوا اسکریپتی در کد بالا از اینجا گرفته شده اند البته پس از کمی تغییر.
نکته : مطلب Cookieها بحثی نسبتا مفصل است که در جای خودش باید به صورت کامل آورده شود. اما در اینجا تنها به همین نکته اشاره کنم که عدم توجه به پراپرتی path کوکی‌ها در این مورد خاص برای خود من بسیار گیج‌کننده و دردسرساز بود.
به عنوان راهی دیگر میتوان به جای روش ساده استفاده از کوکی، تنظیماتی در اختیار کاربر قرار داد تا بتواند زبان تنظیم شده را درون یک فایل یا دیتابیس ذخیره کرد البته با درنظر گرفتن مسائل مربوط به کش کردن این تنظیمات.
راه حل بعدی میتواند استفاده از تنظیمات مرورگر کاربر برای دریافت زبان جاری تنظیم شده است. مرورگرها تنظیمات مربوط به زبان را در قسمت Accept-Languages در HTTP Header درخواست ارسالی به سمت سرور قرار میدهند. بصورت زیر:

```
GET http://www.dotnettips.info HTTP/1.1
...
Accept-Language: fa-IR,en-US;q=0.5
...
```

این هم تصویر مربوط به Fiddler آن:

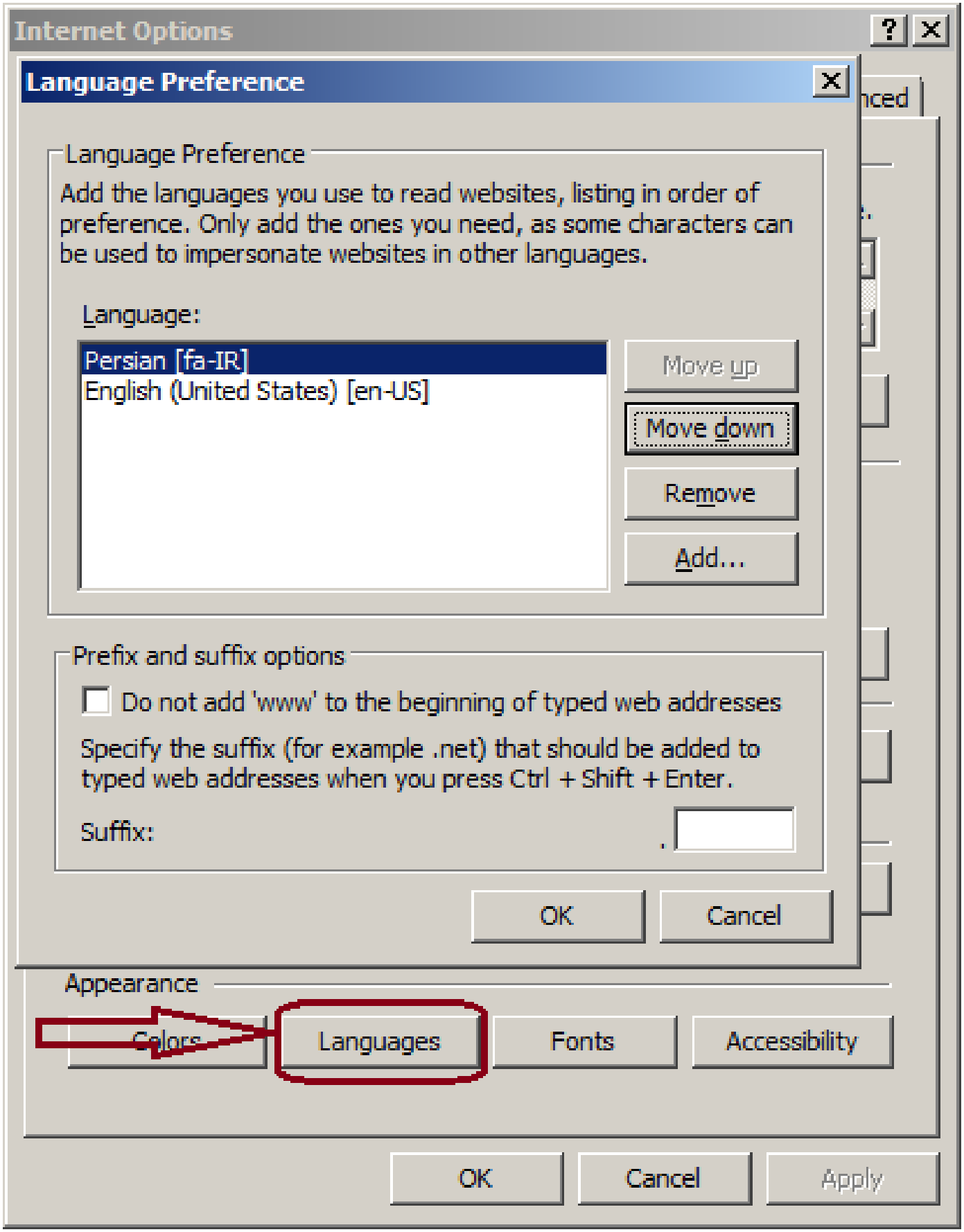


نکته: پارامتر q در عبارت مشخص شده در تصویر فوق relative quality factor نام دارد و به نوعی مشخص کننده اولویت زبان مربوطه است. مقدار آن بین 0 و 1 است و مقدار پیش فرض آن 1 است. هرچه مقدار این پارامتر بیشتر باشد زبان مربوطه اولویت

بالاتری دارد. مثلا عبارت زیر را درنظر بگیرید:

```
Accept-Language: fa-IR,fa;q=0.8,en-US;q=0.5,ar-BH;q=0.3
```

در این حالت اولویت زبان fa-IR برابر 1 و fa برابر 0.8 (fa;q=0.8) است. اولویت دیگر زبانهای تنظیم شده نیز همانطور که نشان داده شده است در مراتب بعدی قرار دارند. در تنظیم نمایش داده شده برای تغییر این تنظیمات در IE میتوان همانند تصویر زیر اقدام کرد:



در تصویر بالا زبان فارسی اولویت بالاتری نسبت به انگلیسی دارد. برای اینکه سیستم g11n دات نت به صورت خودکار از این مقادیر جهت زبان ثرد جاری استفاده کند میتوان از تنظیم زیر در فایل کانفیگ استفاده کرد:

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

در سمت سرور نیز برای دریافت این مقادیر تنظیم شده در مرورگر کاربر میتوان از کدهای زیر استفاه کرد. مثلا در یک اکشن فیلتر:

```
var langs = filterContext.HttpContext.Request.UserLanguages;
```

پراپرتی UserLanguages از کلاس Request حاوی آرایه‌ای از استرینگ است. این آرایه درواقع از Split کردن مقدار Accept-Languages با کاراکتر ',' بدست می‌آید. بنابراین اعضای این آرایه رشته‌ای از نام زبان به همراه پارامتر q مربوطه خواهند بود (مثل "fa;q=0.8").

راه دیگر مدیریت زبانها استفاده از عنوان زبان در مسیر درخواستی صفحات است. مثلا آدرسی شبیه به www.MySite.com/fa/Employees نشان میدهد کاربر درخواست نسخه فارسی از صفحه Employees را دارد. نحوه استفاده از این عناوین و نیز موقعیت فیزیکی این عناوین در مسیر صفحات درخواستی کاملا به سلیقه برنامه نویس و یا کارفرما بستگی دارد. روش کلی بهره برداری از این روش در تمام موارد تقریبا یکسان است.

برای پیاده سازی این روش ابتدا باید یک route جدید در فایل Global.asax.cs اضافه کرد:

```
routes.MapRoute(
    "Localization", // Route name
    "{lang}/{controller}/{action}/{id}", // URL with parameters
    new { controller = "Home", action = "Index", id = UrlParameter.Optional } // Parameter defaults
);
```

دقت کنید که این route باید قبل از تمام routeهای دیگر ثبت شود. سپس باید کلاس پایه کنترلر را به صورت زیر پیاده سازی کرد:

```
public class BaseController : Controller
{
  protected override void ExecuteCore()
  {
    var lang = RouteData.Values["lang"];
    if (lang != null && !string.IsNullOrWhiteSpace(lang.ToString()))
    {
      Thread.CurrentThread.CurrentUICulture = CultureInfo.CreateSpecificCulture(lang.ToString());
    }
    base.ExecuteCore();
  }
}
```

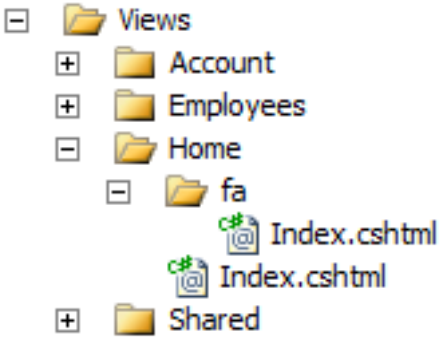
این کار را در یک اکشن فیلتر هم میتوان انجام داد اما با توجه به توضیحاتی که در قسمت قبل داده شد استفاده از اکشن فیلتر برای تعیین زبان جاری کار مناسبی نیست.

نکته: به دلیل آوردن عنوان زبان در مسیر درخواستها باید کنترل دقیقتری بر کلیه مسیرهای موجود داشت!

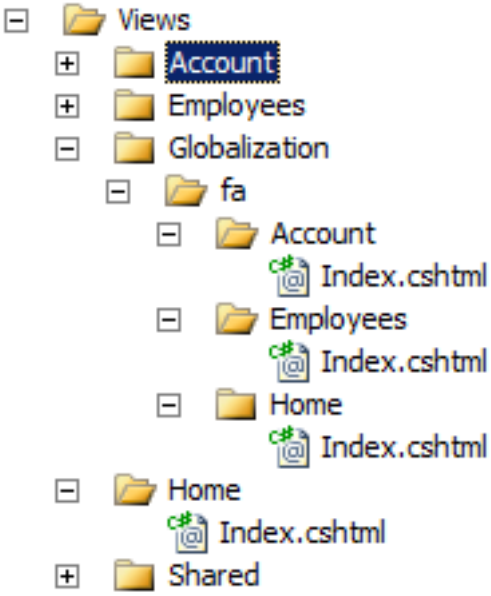
استفاده از ویوهای جداگانه برای زبانهای مختلف

برای اینکار ابتدا ساختار مناسبی را برای نگهداری از ویوهای مختلف خود درنظر بگیرید. مثلا میتوانید همانند نامگذاری فایلهای Resource از نام زبان یا کالچر به عنوان بخشی از نام فایلهای ویو استفاده کنید و تمام ویوها را در یک مسیر ذخیره کنید. همانند تصویر زیر:

البته اینکار ممکن است به مدیریت این فایلها را کمی مشکل کند چون به مرور زمان تعداد فایلهای ویو در یک فولدر زیاد خواهد شد. روش دیگری که برای نگهداری این ویوها میتوان به کار برد استفاده از فولدرهای جداگانه با عناوین زبانهای موردنظر است. مانند تصویر زیر:



روش دیگری که برای نگهداری و مدیریت بهتر ویوهای زبانهای مختلف از آن استفاده میشود به شکل زیر است:



استفاه از هرکدام از این روشها کاملا به سلیقه و راحتی مدیریت فایلها برای برنامه نویس بستگی دارد. درهر صورت پس از

انتخاب یکی از این روشها باید اپلیکشن خود را طوری تنظیم کنیم که با توجه به زبان جاری سیستم، ویوی مربوطه را جهت نمایش انتخاب کند.

مثلا برای روش اول نامگذاری ویوها میتوان از روش دستکاری متد OnActionExecuted در کلاس پایه کنترلر استفاده کرد:

```
public class BaseController : Controller
{
  protected override void OnActionExecuted(ActionExecutedContext context)
  {
    var view = context.Result as ViewResultBase;
    if (view == null) return; // not a view
    var viewName = view.ViewName;
    view.ViewName = GetGlobalizationViewName(viewName, context);
    base.OnActionExecuted(context);
  }
  private static string GetGlobalizationViewName(string viewName, ControllerContext context)
  {
    var cultureName = Thread.CurrentThread.CurrentUICulture.Name;
    if (cultureName == "en-US") return viewName; // default culture
    if (string.IsNullOrEmpty(viewName))
      return context.RouteData.Values["action"] + "." + cultureName; // "Index.fa"
    int i;
    if ((i = viewName.IndexOf('.')) > 0) // ex: Index.cshtml
      return viewName.Substring(0, i + 1) + cultureName + viewName.Substring(i); // "Index.fa.cshtml"
    return viewName + "." + cultureName; // "Index" ==> "Index.fa"
  }
}
```

همانطور که قبلا نیز شرح داده شد، چون متد ExecuteCore قبل از OnActionExecuted صدا زده میشود بنابراین از تنظیم درست مقدار کالچر در ثرد جاری اطمینان داریم.

روش دیگری که برای مدیریت انتخاب ویوهای مناسب استفاده از یک ویوانجین شخصی سازی شده است. مثلا برای روش سوم نامگذاری ویوها میتوان از کد زیر استفاده کرد:

```
public sealed class RazorGlobalizationViewEngine : RazorViewEngine
  {
    protected override IView CreatePartialView(ControllerContext controllerContext, string partialPath)
    {
      return base.CreatePartialView(controllerContext, GetGlobalizationViewPath(controllerContext,
partialPath));
    }
    protected override IView CreateView(ControllerContext controllerContext, string viewPath, string
masterPath)
    {
      return base.CreateView(controllerContext, GetGlobalizationViewPath(controllerContext, viewPath),
masterPath);
    }
    private static string GetGlobalizationViewPath(ControllerContext controllerContext, string
viewPath)
    {
      //var controllerName = controllerContext.RouteData.GetRequiredString("controller");
      var request = controllerContext.HttpContext.Request;
      var lang = request.Cookies["MyLanguageCookie"];
      if (lang != null && !string.IsNullOrEmpty(lang.Value) && lang.Value != "en-US")
      {
        var localizedViewPath = Regex.Replace(viewPath, "^~/Views/",
string.Format("~/Views/Globalization/{0}/", lang.Value));
        if (File.Exists(request.MapPath(localizedViewPath))) viewPath = localizedViewPath;
      }
      return viewPath;
    }
```

و برای ثبت این ViewEngine در فایل Global.asax.cs خواهیم داشت:

```
protected void Application_Start()
{
  ViewEngines.Engines.Clear();
  ViewEngines.Engines.Add(new RazorGlobalizationViewEngine());
}
```

محتوای یک فایل Resource

ساختار یک فایل resx. به صورت XML استاندارد است. در زیر محتوای یک نمونه فایل Resource با پسوند resx. را مشاهده میکنید:

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <!--
    Microsoft ResX Schema ...
    -->
  <xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
   ...
  </xsd:schema>
  <resheader name="resmimetype">
    <value>text/microsoft-resx</value>
  </resheader>
  <resheader name="version">
    <value>2.0</value>
  </resheader>
  <resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <resheader name="writer">
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <data name="RightToLeft" xml:space="preserve">
    <value>false</value>
    <comment>RightToleft is false in English!</comment>
  </data>
</root>
```

در قسمت ابتدایی تمام فایلهای resx. که توسط ویژوال استودیو تولید میشود کامنتی طولانی وجود دارد که به صورت خلاصه به شرح محتوا و ساختار یک فایل Resource میپردازد. در ادامه تگ نسبتا طولانی xsd:schema قرار دارد. از این قسمت برای معرفی ساختار داده ای فایلهای XML استفاده میشود. برای آشنایی بیشتر با XSD (یا XML Schema) به اینجا مراجعه کنید. به صورت خلاصه میتوان گفت که XSD برای تعیین ساختار دادهها یا تعیین نوع داده ای اطلاعات موجود در یک فایل XML به کار میرود. درواقع تگهای XSD به نوعی فایل XML ما را Strongly Typed میکند. با توجه به اطلاعات این قسمت، فایلهای resx. شامل 4 نوع گره اصلی هستند که عبارتند از: metadata و assembly و data و resheader. در تعریف هر یک از گرهها در این قسمت مشخصاتی چون نام زیر گرههای قابل تعریف در هر گره و نام و نوع خاصیتهای هر یک معرفی شده است.

بخش موردنظر ما در این مطلب قسمت انتهایی این فایلهاست (تگهای resheader و data). همانطور در بالا مشاهده میکنید تگهای reheader شامل تنظیمات مربوط به فایل resx. با ساختاری ساده به صورت name/value است. یکی از این تنظیمات resmimetype فایل resource را معرفی میکند که درواقع مشخص کننده نوع محتوای (Content Type) فایل XML است(^). برای فایلهای resx این مقدار برابر text/microsoft-resx است. تنظیم بعدی نسخه مربوط به فایل resx (یا Microsoft ResX Schema) را نشان میدهد. در حال حاضر نسخه جاری (در VS 2010) برابر 2.0 است. تنظیم بعدی مربوط به کلاسهای reader و writer تعریف شده برای استفاده از این فایلهاست. به نوع این کلاسهای خواننده و نویسنده فایلهای resx. و مکان فیزیکی و فضای نام آنها دقت کنید که در مطالب بعدی از آنها برای ویرایش و بروزرسانی فایلهای resource در زمان اجرا استفاده خواهیم کرد.

در پایان نیز تگهای data که برای نگهداری دادهها از آنها استفاده میشود. هر گره data شامل یک خاصیت نام (name) و یک زیرگره مقدار (value) است. البته امکان تعیین یک کامنت در زیرگره comment نیز وجود دارد که اختیاری است. هر گره data مینواند شامل خاصیت type و یا mimetype نیز باشد. خاصیت type مشخص کننده نوعی است که تبدیل text/value را با استفاده از ساختار TypeConverter پشتیبانی میکند. البته اگر در نوع مشخص شده این پشتیبانی وجود نداشته باشد، داده موردنظر پس از سریالایز شدن با فرمت مشخص شده در خاصیت mimetype ذخیره میشود. این mimetype اطلاعات موردنیاز را برای کلاس خواننده این فایلها (ResXResourceReader به صورت پیشفرض) جهت چگونگی بازیابی آبجکت موردنظر فراهم میکند. مشخص کردن این دو خاصیت برای انواع رشته ای نیاز نیست. انواع mimetype قابل استفاده عبارتند از:

- application/x-microsoft.net.object.binary.base64: آبجکت موردنظر باید با استفاده از کلاس System.Runtime.Serialization.Formatters.Binary.BinaryFormatter سریالایز شده و سپس با فرمت base64 به یک رشته انکد شود (راجع به انکدینگ base64 ^ و ^).

- application/x-microsoft.net.object.soap.base64: آبجکت موردنظر باید با استفاده از کلاس

System.Runtime.Serialization.Formatters.Soap.SoapFormatter سریالایز شده و سپس با فرمت base64 به یک رشته انکد شود.

- application/x-microsoft.net.object.bytearray.base64: آبجکت ابتدا باید با استفاده از یک System.ComponentModel.TypeConverter به آرایه ای از بایت سریالایز شده و سپس با فرمت base64 به یک رشته انکد شود.

نکته : امکان جاسازی کردن (embed) فایلهای resx. در یک اسمبلی یا کامپایل مستقیم آن به یک سَتِلایت اسمبلی (ترجمه مناسبی برای satellite assembly پیدا نکردم، چیزی شبیه به اسمبلی قمری یا وابسته و از این قبیل ...) وجود ندارد. ابتدا باید این فایلهای resx. به فایلهای resources. تبدیل شوند. اینکار با استفاده از ابزار Resource File Generator (نام فایل اجرایی آن resgen.exe است) انجام میشود (^ و ^). سپس میتوان با استفاده از Assembly Linker ستلایت اسمبلی مربوطه را تولید کرد (^). کل این عملیات در ویژوال استودیو با استفاده از ابزار msbuild به صورت خودکار انجام میشود!

نحوه یافتن کلیدهای Resource در بین فایلهای مختلف Resx توسط پرووایدر پیش فرض در دات نت

عملیات ابتدا با بررسی خاصیت CurrentUICulture از ثرد جاری آغاز میشود. سپس با استفاده از عنوان استاندارد کالچر جاری، فایل مناسب Resource یافته میشود. در نهایت بهترین گزینه موجود برای کلید درخواستی از منابع موجود انتخاب میشود. مثلا اگر کالچر جاری fa-IR و کلید درخواستی از کلاس Texts باشد ابتدا جستجو برای یافتن فایل Texts.fa-IR.resx آغاز میشود و اگر فایل موردنظر یا کلید درخواستی در این فایل یافته نشد جستجو در فایل Texts.fa.resx ادامه می‌یابد. اگر باز هم یافته نشد درنهایت این عملیات جستجو در فایل resource اصلی خاتمه می‌یابد و مقدار کلید منبع پیش فرض به عنوان نتیجه برگشت داده میشود. یعنی در تمامی حالات سعی میشود تا دقیقترین و بهترین و نزدیکترین نتیجه انتخاب شود. البته درصورتیکه از یک پرووایدر شخصی سازی شده برای کار خود استفاده میکنید باید چنین الگوریتمی را جهت یافتن کلیدهای منابع خود از فایلهای Resource (یا هرمنبع دیگر مثل دیتابیس یا حتی یک وب سرویس) درنظر بگیرید.

Globalization در کلاینت (javascript g11n)

یکی دیگر از موارد استفاده g11n در برنامه نویسی سمت کلاینت است. با وجود استفاده گسترده از جاوا اسکریپت در برنامه نویسی سمت کلاینت در وب اپلیکیشنها، متاسفانه تا همین اواخر عملا ابزار یا کتابخانه مناسبی برای مدیریت g11n در این زمینه وجود نداشته است. یکی از اولین کتابخانه‌های تولید شده در این زمینه کتابخانه jQuery Globalization است که توسط مایکروسافت توسعه داده شده است (برای آشنایی بیشتر با این کتابخانه به ^ و ^ مراجعه کنید). این کتابخانه بعدا تغییر نام داده و اکنون با عنوان Globalize شناخته میشود. Globalize یک کتابخانه کاملا مستقل است که وابستگی به هیچ کتابخانه دیگر ندارد (یعنی برای استفاده از آن نیازی به jQuery نیست). این کتابخانه حاوی کالچرهای بسیاری است که عملیات مختلفی چون فرمت و parse انواع داده‌ها را نیز در سمت کلاینت مدیریت میکند. همچنین با فراهم کردن منابعی حاوی جفتهای key/culture میتوان از مزایایی مشابه مواردی که در این مطلب بحث شد در سمت کلاینت نیز بهره برد. نشانی این کتابخانه در github اینجا است. با اینکه خود این کتابخانه ابزار کاملی است اما در بین کالچرهای موجود در فایلهای آن متاسفانه پشتیبانی کاملی از زبان فارسی نشده است. ابزار دیگری که برای اینکار وجود دارد پلاگین jquery localize است که برای بحث g11n رشته‌ها پیاده‌سازی بهتر و کاملتری دارد.

در مطالب بعدی به مباحث تغییر مقادیر کلیدهای فایلهای resource در هنگام اجرا با استفاده از روش مستقیم تغییر محتوای فایلها و کامپایل دوباره توسط ابزار msbuild و نیز استفاده از یک ResourceProvider شخصی سازی شده به عنوان یک راه حل بهتر برای اینکار میپردازم.

در تهیه این مطلب از منابع زیر استفاده شده است: Localization in ASP.NET MVC – 3 Days Investigation, 1 Day Job

ASP.NET MVC 3 Internationalization

Localization and skinning in ASP.NET MVC 3 web applications Simple ASP.Net MVC Globalization with Graceful Fallback

Globalization, Internationalization and Localization in ASP.NET MVC 3, JavaScript and jQuery - Part 1

نظرات خوانندگان

نویسنده: امیرحسین مرجانی
تاریخ: ۱۳۹۱/۱۰/۲۱ ۲۳:۵

سلام آقای یوسف نژاد
من بعد از تلاش‌های زیاد توی پروژه‌های مختلف این مطالبی که شما نوشته اید رو پیاده سازی کردم ، ولی خیلی پراکنده.
ولی حالا می‌بینم شما به زیبایی این مطالب رو کنار هم قرار دادید.
می خواستم بابت مطلب خوب و مفیدتون و همچنین وقتی که گذاشتید تشکر کنم.
ممنونم بابت زحمات شما

اگر ممکنه برچسب MVC رو هم به مطلبتون اضافه کنید.

نویسنده: یوسف نژاد
تاریخ: ۱۳۹۱/۱۰/۲۱ ۲۳:۲۴

با سلام و تشکر بابت نظر لطف شما.
البته باید بگم که همه دوستانی که اینجا به عنوان نویسنده کمک میکنند هدفشون اشتراک مطالبی هست که یاد گرفته اند تا سایر دوستان هم استفاده کنند.

برچسب MVC هم اضافه شد. با تشکر از دقت نظر شما.

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۱/۱۰/۲۲ ۱:۳

کاملا مشخصه که مطلب از روی تجربه‌ی کاریه و بسیار عالی جمع آوری شده ... ممنون ... به طرز عجیبی منتظر قسمت بعدم :دی

نویسنده: پندار
تاریخ: ۱۳۹۱/۱۲/۰۸ ۲۱:۳۸

گویا در MVC 4 این روش پاسخ نمیدهد. لطفا در این مورد برای MVC 4 راه حلی بدهید

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۲/۰۸ ۲۳:۶

MVC 4 فقط یک سری افزونه بیشتر از MVC3 داره. مثلا razor آن بهبود پیدا کرده، فشرده سازی فایل‌های CSS به اون اضافه شده یا Web API رو به صورت یکپارچه داره. از لحاظ کار با فایل‌های منبع فرقی نکرده.

نویسنده: پندار
تاریخ: ۱۳۹۱/۱۲/۰۹ ۹:۲۱

متن نشانی زیر را مطالعه کنید

http://geekswithblogs.net/shaunxu/archive/2012/09/04/localization-in-asp.net-mvc-ndash-upgraded.aspx

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۲/۰۹ ۹:۴۳

مطلبی که لینک دادی در مورد آپدیت یک helper شخصی توسعه داده شده توسط شخص ثالث است از MVC2 به MVC4. اگر کسی

از این راه حل شخصی و خاص استفاده نکرده باشه، اصول فوق فرقی نکرده.

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۲/۱۴ ۱۶:۵

مطلب خیلی خوبی بود کلی استفاده کردیم.
مهندس کالچر زبان کردی چی میشه؟ توی لیست منابعی که دادین گیر نیاوردم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۱۴ ۱۷:۲۲

kur هست مطابق استاندارد .

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۲/۱۷ ۲:۱۱

سلام
اما مهندس کلاس Culture Info این مقدار قبول نمی‌کنه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۱۷ ۹:۳

می‌تونید کلاس فرهنگ سفارشی را ایجاد و استفاده کنید.

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۲/۱۷ ۱۰:۱۲

اما روش گفته شده نیاز به دسترسی مدیریت دارد که روی سرورهای اشتراکی ممکن نیست

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۱۷ ۱۱:۱۹

نحوه توسعه اکثر برنامه‌ها و کتابخانه‌ها در طول زمان، بر اساس تقاضا و پیگیری مصرف کننده است. اگر بعد از بیش از 10 سال، چنین فرهنگی اضافه نشده یعنی درخواستی نداشته. مراجعه کنید به محل پیگیری این نوع مسایل .

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۲/۱۹ ۱۰:۱۴

سلام مهندس یوسف نژاد (ابتدا ممنونم از پست خوب شما)
با پیروی از پست شما
ابتدا فایل‌های ریسورس در پروژه جاری فولدر App_GlobalResources گذاشتم و پروژه در صفحات aspx با قالب زیر به راحتی تغییر زبان داده میشد:

```
<asp:Literal ID="Literal1" Text='<%$ Resources:resource, Title %>' runat="server" />
```

اما بعدش فایل هارو توی یک پروژه کتابخانه ای جدید گذاشتم و Build Action فایل‌های ریسورس روی Embedded Resource تنظیم کردم، پروژه با موفقیت اجرا شد و در سمت سرور با کد زیر راحت به مقادیر دسترسی دارم:

```
Literal1.Text=ResourcesManager.Resource.Title;
```

اما در سمت صفحات aspx با کد قبلی به شکل زیر نمایش نمیده و خطا صادر میشه:

```
<asp:Literal ID="Literal1"  runat="server" Text='<%$ ResourcesManager.Resource:resource, Title %>'/>
```

و خطای زیر صادر میشه:

```
Parser Error
Description: An error occurred during the parsing of a resource required to service this request.
Please review the following specific parse error details and modify your source file appropriately.

Parser Error Message: The expression prefix 'ResourcesManager.Resource' was not recognized.  Please
correct the prefix or register the prefix in the <expressionBuilders> section of configuration.

Source Error:
```

مراحل این پست روی هم دنبال کردم اما بازم نشد.
چه تنظیماتی ست نکردم ؟

نویسنده: یوسف نژاد
تاریخ: ۱۳۹۲/۰۱/۳۱ ۱۲:۴۴

ببخشید یه چند وقتی فعال نبودم و پاسخ این سوال رو دیر دارم میدم.
امکان استفاده از کلیدهای Resource برای مقداردهی خواص سمت سرور کنترلها در صفحات aspx به صورت مستقیم وجود ندارد. بنابراین برای استفاده از این کلیدها همانند روش پیش فرض موجود در ASP.NET باید از یکسری ExpressionBuilder استفاده شود که کار Parse عبارت وارده برای این خواص را در سمت سرور انجام میدهد. کلاس پیش فرض برای اینکار در ASP.NET Web Form که از پیشوند Resources استفاده میکند تنها برای Resourceهای محلی (Local) موجود در فولدرهای پیش فرض (App_GlobalResources و App_LocalResources) کاربرد دارد و برای استفاده از Resourceهای موجود در منابع ریفرنس داده شده به پروژه باید از روشی مثل اونچه که خود شما لینکش رو دادین استفاده کرد.
من این روش رو استفاده کردم و پیاده سازی موفقی داشتم. نمیدونم مشکل شما چیه...

نویسنده: یوسف نژاد
تاریخ: ۱۳۹۲/۰۱/۳۱ ۱۲:۵۲

اگر مشکلی در پیاده سازی روش بالا دارین، تمام مراحلی که من طی کردم دقیقا اینجا میارم:
ابتدا کلاس ExpressionBuilder رو به صورت زیر مثلا در خود پروژه Resources اضافه میکنیم:

```
using System.Web.Compilation;
using System.CodeDom;
namespace Resources
{
  [ExpressionPrefix("MyResource")]
  public class ResourceExpressionBuilder : ExpressionBuilder
  {
    public override System.CodeDom.CodeExpression GetCodeExpression(System.Web.UI.BoundPropertyEntry
entry, object parsedData, System.Web.Compilation.ExpressionBuilderContext context)
    {
      return new CodeSnippetExpression(entry.Expression);
    }
  }
}
```

سپس تنظیمات زیر رو به Web.config اضافه میکنیم:

```
<compilation debug="true" targetFramework="4.0">
    <expressionBuilders>
        <add expressionPrefix="MyResource" type="Resources.ResourceExpressionBuilder, Resources" />
```

```
    </expressionBuilders>
</compilation>
```

درنهایت به صورت زیر میتوان از این کلاس استفاده کرد:

```
<asp:Literal ID="Literal1"  runat="server" Text="<%$ MyResource: Resources.Resource1.String2 %>" />
```

هرچند ظاهرا مقدار پیشوند معرفی شده در Attribute کلاس ResourceExpressionBuilder اهمیت چندانی ندارد! امیدوارم مشکلتون حل بشه.

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۲/۰۱ ۲:۲۹

ممنونم از پاسخ شما
همون روش شمارو دنبال کردم پاسخ گرفتم، اشکال از خودم بود
با تشکر از شما

نویسنده: صادق نجاتی
تاریخ: ۱۳۹۲/۱۲/۲۷ ۱۲:۰

با سلام
ضمن تشکر از مطلب بسیار خوبتون
خاصیت DisplayFormat قابلیت استفاده از کلیدهای Resource را ندارد !
لطفا راهنمایی فرمایید که چطور میشه از این خاصیت برای DisplayFormat استفاده کرد؟
من می‌خوام برای تاریخ در زبانه فارسی از فرمت {yyyy-MM-dd} و در زبانه انگلیسی از {yyyy-dd-MM} استفاده کنم.
با سپاس فراوان