**انجام مراحل لازم از مرحله طراحی تا پياده‏سازی**

**تهيه و تنظيم: داريوش تصديقی**

نسخه: 1.7 - تاريخ: 28/08/1393

**توصيه‏های کلی**

1. تمام Namespace ها، Class ها، Method ها، Field ها، Property ها و پوشه های پروژه، بايد با اسامی بامعنی و کامل و با قانون Pascal (به غير از Field ها که بايد با قانون Camel Case و با يک Underline در ابتدای آن نوشته شود) نوشته شوند و به هيچ عنوان از (\_) در اين اسامی نبايد استفاده کنيد.
2. تنها کلمه‏ای که به صورت مختصر نوشته می‏شود، Id بوده و بايد دقيقا به همين شکل نوشته شود، يعنی I بزرگ و d کوچک.
3. در زمان تعريف متغيری از جنس Collection، بايد نام متغير، اسم جمع باشد.
4. هر آنچه که اگر ننويسيم، Compiler به طور اتوماتيک می‏نويسد، بايد به صراحت نوشته شود.
5. ساختار هر پروژه از قسمت های ذيل تشکيل شده است:
   1. يک پروژه به نام Dtx از جنس Class Library که در داخل آن تمام Library های لازم قرار دارد.
   2. يک پروزه به نام Resources از جنس Class Library که در داخل آن تمام Resource های پروژه ها در داخل آن قرار دارد.
   3. يک پروژه به نام Models از جنس Class Library که در داخل آن تمام Model ها يا اصطلاحا Entity ها قرار می گيرند.

**کارهايی که بايد در Class Library، مربوط به Models انجام گيرد:**

1. ايجاد Entity (Model) مورد نظر در Root پروژه:
   1. دقت کنيد، در صورتی که پروژه بزرگی در دست داريد و اين پروژه از زيرسيستم‏های مختلف و بعضا مستقلی تشکيل شده‏اند، بهتر است که با توجه به نام آن زيرسيستم‏ها، در داخل Root پروژه، پوشه‏ای به همان نام ايجاد کرده و مدل خود را داخل آن قرار دهيد. (نمونه‏ای از پوشه‏ها در اين خصوص: HumanResources، EShop، OfficeAutomation، ...) و صرفا مدل‏های عمومی، در همه پروژه ها، در Root قرار می گيرند، مانند User, Gender, Role و غيره...) ولی Entity هايی که در داخل پروژه احتمال می دهيم و يا اطمينان داريم که در دو يا چند زيرسيستم مشترک می باشند، در داخل پوشه ای به نام Common آنرا قرار می دهيم.
   2. دقت کنيد که اگر به عنوان مثال مدلی به نام Religion (دين/مذهب) در داخل پوشه‌ای به نام HumanResources ايجاد می‌کنيد، بايد Namespace آنرا نيز (که البته به طور اتوماتيک درست می‌شود) به صورت تکميلی بنويسيد: Models.HumanResources
   3. دقت کنيد که اساسا اين کار صرفا به دليل دسته‌بندی نيست! برای همين اگر فقط به همين توصيه ها عمل کنيد، جدول Religions در Schema‌ی پيش فرض يعنی dbo ايجاد می گردد. در صورتی که در زيرسيستم‌های مختلف، ممکن است که مدل‌های هم نام داشته باشيم که شايد ارتباطی به زيرسيستم‌های مختلف نداشته باشد، در اين صورت در زمان توليد جداول مربوط به اينگونه مدل‌ها، دچار مشکل خواهيم شد. برای حل اين مشکل، برای کلاس مدل، از Attribute ذيل استفاده می‌کنيم:

[System.ComponentModel.DataAnnotations.Schema.Table("Religions", Schema = "HumanResources")]

در اين صورت، پس از آنکه جدول مربوطه، در بانک اطلاعاتی ايجاد گرديد، Schema‌ی آن ديگر مقدار پيش‌فرض dbo نبوده و مقدار آن به HumanResources تغيير می‌کند.

1. Inherit کردن مدل از BaseEntity لازم.
2. درج Fluent API Configuration.
3. درج Property های لازم
4. ايجاد Resource مربوط به مدل و به زبان انگليسی، در داخل پروژه Resources، و پوشه Models. دقت کنيد که ساختار فايل ها و پوشه های موجود در پوشه Models مربوط به پروژه Resources، بايد دقيقا با ساختار فايل ها و پوشه های موجود در پروژه Models يکسان باشد.
   1. دقت داشته باشيد که در بالای پنجره Resource، حتما Access Modifier را به گزينه Public، تغيير دهيد.
   2. به ازای هر Entity، فارغ از نام Property هايی که در داخل آن نوشته ايم، چهار Property در داخل Resource مربوط به همان Entity ايجاد می کنيم.
      1. EntityName
      2. EntityNameOf
      3. EntitiesName
      4. EntitiesNameOf
   3. دقت کنيد که Property هايی که نام آنها در بين Entity ها، خيلی تکراری می باشد، در داخل Resource ی به نام General آنها را قرار می دهيم. (مانند Name, Description, Hits و غيره)
   4. سپس مابالتفاوت Property های Entity مورد نظر را در داخل Resource مربوط به همان Entity، قرار می دهيم.
5. ايجاد Resource به زبان فارسی (Religion.fa-IR.resx)
6. ايجاد فايل tt مربوط به Resource (Religion.tt)
   1. برای اين منظور، از پوشه Documents، (که نکاتی کليدی و آموزشی را در داخل آن قرار می‌دهيم) فايل SmartStringResourcesTemplate.txt را به محل مورد نظر کپی کرده و نام آنرا به نام مدل و پسوند آنرا به tt تغيير می‌دهيم. (Religion.tt)
   2. سپس بر روی فايل tt، **ابتدا Double Click کرده** تا فايل مربوطه باز گرديده و سپس بر روی فايل tt دکمه سمت راست را زده و گزينه Run Custom Tool را انتخاب می کنيم.
7. درج Attribute های لازم در داخل مدل.
8. ايجاد ارتباط بين Entity های ديگر.
9. اصلاح Fluent API Configuration مدل.
10. درج يک Collection از Entityمربوطه در داخل کلاس DatabaseContext
11. درج Configuration Entity در تابع OnModelCreating داخل کلاس DatabaseContext
12. در صورت نياز، درج اطلاعات لازم اوليه (اطلاعات پايه و/يا تستی) در داخل تابع Seed، در کلاس DatabaseContextInitializer

**توصيه‏هايی در رابطه با ايجاد Models Class Library**

1. اصطلاحا به کلاس‏هايی که از آنها Inherit نمی‏شود، Entity می‏گويند.
2. اصطلاحا به کلاس‏هايی که از آنها Inherit می‏شود، BaseEntity می‏گويند.
3. **به هيچ عنوان در داخل های Base، Property ای از جنس Entity و يا کلاس Base ديگری استفاده نمی‏کنيم، مگر با نظارت و توصيه مسوول فنی!**

**کارهايی که بايد در Class Library مربوط به DAL انجام گيرد:**

1. درج Interface
2. درج توابع لازم در Interface مربوطه
3. درج کلاس
4. درج يک Property در IUnitOfWork
5. درج يک Property در UnitOfWork