

بعد از ایجاد فایل Text template که [در جلسه قبل](#) با آن آشنا شدید، برای شروع قواعد زیر را در نظر بگیرید :

- تنظیمات مربوط به فایل Text template و نحوه تولید خروجی در ابتدای فایل و بین علامت <#@> و <#> قرار میگیرد.

- هر متنی که بصورت معمول در فایل tt نوشته شود، به همان صورت در فایل خروجی قرار می‌گیرد.

- هر دستوری که در بین علامت‌های <#> و <#> قرار گیرد هنگام کامپایل اجرا شده و معادل آن در همان مکان متن قرار میگیرد.

- هر دستوری که بین علامت‌های <#> و <#> قرار گیرد، هنگام کامپایل اجرا می‌شود. در این صورت دستورات نوشته شده در این

قسمت فقط اجرا می‌گردد و معمولاً برای استفاده در قسمت‌های دیگر، داخل بلوک <#> نوشته می‌شود .

- برای تعریف کلاس یا متد جدید جهت استفاده در فایل tt می‌توانیم کلاس را در بین علامت <#+> و <#> قرار دهیم. در این

صورت کلاس و متدهای نوشته شده در قسمت‌های دیگر، داخل بلوک <#> و یا <#> مورد استفاده قرار میگیرند.

اجازه دهید با یک مثال ساده قواعد اولیه را بررسی کنیم :

```
<#@ template debug="false" hostspecific="false" language="C#" #>
<#@ output extension=".txt" #>

<# var T = DateTime.Now; #>

The Time is : <# T #>

The Time is : <# DateTime.Now #>
```

در این مثال، T در واقع متغیری است که در بلوک <#> تعریف گردیده و در بلوک <#> مقدار آن استفاده میشود. خروجی فایل چیزی شبیه به دو خط زیر خواهد بود:

```
The Time is : 02/16/2014 14:17:39
```

```
The Time is : 02/16/2014 14:17:39
```

به عنوان یک مثال دیگر که قواعد توضیح داده شده را پوشش دهد به مثال زیر توجه کنید :

```
<#@ template debug="true" hostspecific="false" language="C#" #>
<#@ output extension=".cs" #>

using System;
using System.Text;

<# string ClassName = "DotnetTips"; #>
public class <# ClassName + "_" + new MyTestClass().Str #>
{
}

<#+
public class MyTestClass
{
public string Str { get{return new DateTime().DayOfWeek.ToString(); } }
}
#>
```

خروجی Text template بالا فایل Cs با محتوی شبیه کد زیر خواهد بود: (روز نگارش مطلب البته دوشنبه است)

```
using System;
using System.Text;
```

```
public class DotnetTips_Monday
{
}
```

به عنوان یک مثال ساده دیگر برای فهم بیشتر به کد زیر جهت تولید Table در Html توجه کنید:

```
<#@ template debug="false" hostspecific="false" language="C#" #>
<#@ output extension=".html" #>

<html><body>
<table>
  <# for (int i = 1; i <= 10; i++)
  { #>
    <tr>
      <td>Test name <#= i #> </td>
      <td>Test value <#= i * i #> </td>
    </tr>
  <# } #>
</table>
</body></html>
```

فکر می‌کنم این 3 مثال ساده، تا حد زیادی قواعد اولیه T4 Text Template را برای شما روشن کرده باشد. در قسمت بعدی برخی قواعد تکمیلی را در این مورد خدمتتون ارائه میدم.

نظرات خوانندگان

نویسنده: افشین عباسپور
تاریخ: ۲۳:۲۳ ۱۳۹۲/۱۱/۳۰

لینک کدهای این قسمت در ورد رو میتونید از [اینجا](#) دانلود کنید . چون رنگ کدها رو میتوند [اونجا](#) ببینید .

خوب در دو قسمت قبلی ([۱](#) و [۲](#)) با T4 و قواعد کد نویسی Text Template آشنا شدید. در این قسمت برخی مفاهیم را با یک مثال کاربردی تر بررسی می‌کنیم.

در ادامه قواعد زیر را در نظر بگیرید :

- 1 - برای استفاده از یک کتابخانه خارجی (dll) در داخل کد Text Template از بلوک `<# assembly @#>` استفاده می‌شود .
مثلا برای استفاده از System.xml کد `<# "assembly name="System.xml" @#>` رو قرار بدید .
- 2- برای import کردن یک فضای نام نیز می‌توانید از بلوک `<# @#>` استفاده نمایید .
به عنوان مثال : `<# "import namespace="System.Data" @#>`
- 3- encoding خروجی قابل تنظیم می‌باشد. مثال: `<# output extension = ".html" encoding = "utf-8" @#>`
- 4- برای استفاده از یک فایل tt درون فایل دیگر میتوان از include استفاده کرد. مثال : `<# include file=" testpath " @#>`

`<# \basetest.tt`

حالا به مثال زیر توجه کنید:

در بسیاری از پروژه‌ها، معادل تمامی جداول موجود در یک دیتابیس Class هایی به نام DTO یا Data transfer object ساخته میشود که عموماً "کلاسهای سبکی هستند که فقط شامل خصوصیت‌های معادل فیلدهای جداول می‌باشند و از آن‌ها جهت مدل کردن داده‌ها و ... استفاده می‌گردد. تولید این کلاسهای ساده می‌تواند بصورت اتوماتیک صورت گیرد و از این جهت در زمان تولید پروژه صرفه جویی شود. همچنین با تغییر ساختار دیتابیس می‌توان همواره کلاسها را بروزرسانی کرد. نمونه ای از کد T4 برای تولید تمامی کلاسهای DTO به شکل زیر میباشد . فقط برای تست کد زیر دقت کنید که ConnectionString را در داخل کد متناسب با دیتابیس خود تغییر دهید:

```
<#@ template language="C#" debug="True" hostspecific="True" #>
<#@ output extension=".cs" #>
<#@ assembly name="System.Data" #>

<#@ assembly name="System.xml" #>
<#@ import namespace="System.Collections.Generic" #>
<#@ import namespace="System.Data.SqlClient" #>
<#@ import namespace="System.Data" #>

using System;
namespace MyProject.Entities
{
    <#
        string connectionString = "Password=22125110;Persist Security Info=True;User ID=sa;Initial
Catalog=DnnDB;Data Source=ABBASPOOR299";
        SqlConnection conn = new SqlConnection(connectionString);
        conn.Open();
        System.Data.DataTable schema = conn.GetSchema("TABLES");
        string selectQuery = "select * from @tableName";
        SqlCommand command = new SqlCommand(selectQuery,conn);
        SqlDataAdapter ad = new SqlDataAdapter(command);
        System.Data.DataSet ds = new DataSet();

        foreach(System.Data.DataRow row in schema.Rows)
        {
            <#
            public class <#= row["TABLE_NAME"].ToString().Trim('s') #>
            {
                <#
                command.CommandText =
selectQuery.Replace("@tableName",row["TABLE_NAME"].ToString());
                ad.FillSchema(ds, SchemaType.Mapped, row["TABLE_NAME"].ToString());

                foreach (DataColumn dc in ds.Tables[row["TABLE_NAME"].ToString()].Columns)
                {
                    <#
                }
            }
        }
    }
}
```

```
private <# dc.DataType.Name #> _<# dc.DataType.Name #>
dc.ColumnName.Replace(dc.ColumnName[0].ToString(), dc.ColumnName[0].ToString().ToLower()) #>;
public <# dc.DataType.Name #> <# dc.ColumnName #>
{
    get { return _<# dc.ColumnName.Replace(dc.ColumnName[0].ToString(),
dc.ColumnName[0].ToString().ToLower()) #>; }
    set { _<# dc.ColumnName.Replace(dc.ColumnName[0].ToString(),
dc.ColumnName[0].ToString().ToLower()) #> = value; }
}

<# } #>

}

<#
} #>

}
```

خوب! خودتون می‌توانید تست کنید ... این یکی از رایج‌ترین استفاده‌های T4 Text Template هست که برنامه نویسان از آن استفاده می‌کنند.

نظرات خوانندگان

نویسنده: افشین عباسپور
تاریخ: ۲۳:۲۵ ۱۳۹۲/۱۱/۳۰

متن رنگی کد این قسمت رو میتونید از [اینجا](#) دانلود کنید

نویسنده: علی صداقت
تاریخ: ۱۲:۱۴ ۱۳۹۲/۱۲/۰۳

با تشکر از مطلب مفید شما. ظاهرا پس از اتمام درج اطلاعات ستونهای یک جدول باید متد `ds.Tables.Clear` را فراخوانی کرد تا جداول موجود در دیتاست پاک شوند. در حال حاضر فیلدهای جدول اول به جای فیلدهای تمامی جداول قرار می گیرند.

نویسنده: افشین عباسپور
تاریخ: ۱۵:۲ ۱۳۹۲/۱۲/۰۳

ممنون دوست عزیز و پوزش از بی دقتی من .
مشکل `Clear` شدن دیتاست نیست . حلقه دو باید تغییر کنه `ds.Tables[0].Columns ...` باید تغییر کنه و تبدیل به `ds.Tables[row["TABLE_NAME"].ToString()].Columns` بشه ! باز هم ممنون از دقت شما

نویسنده: افشین عباسپور
تاریخ: ۱۵:۳۲ ۱۳۹۲/۱۲/۰۳

کد اصلاح شد .

نویسنده: ابوالفضل علیاری
تاریخ: ۲۳:۴۳ ۱۳۹۲/۱۲/۱۶

سلام
من کامل متوجه نشدم ، وقتی برنامه اجرا میشه این کدها به صورت اتوماتیک اجرا میشه؟
ممنون

نویسنده: افشین عباسپور
تاریخ: ۱۴:۱ ۱۳۹۲/۱۲/۱۷

نه دوست عزیز .. این به اجرای برنامه ربطی نداره ! وقتی فایل رو (فایل `tt`) در ویژوال استدیو تغییر میدید و `Save` میکنید بطور اتوماتیک اجرا میشه و خروجی رو در فایل تعیین شده در `output` (بالای کد) ایجاد میکنه ... قسمت چهارم رو مطالعه کنید بیشتر متوجه میشید
ضمن اینکه مثال ساده قسمت دوم رو اجرا کنید تا متوجه عملکرد فایل [Text template](#) بشی ...

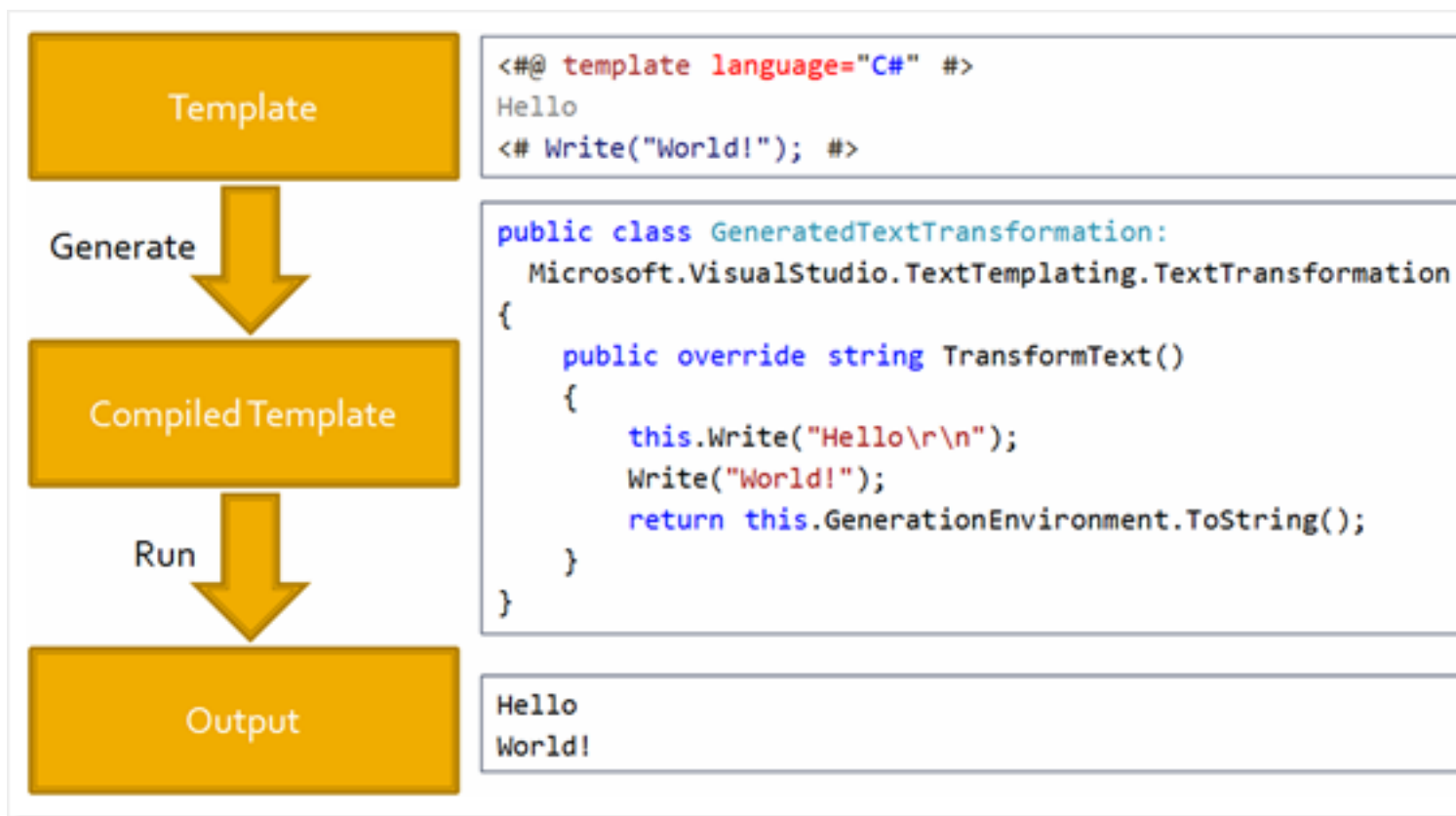
نویسنده: افشین عباسپور
تاریخ: ۱۴:۳ ۱۳۹۲/۱۲/۱۷

در مورد اجرای مثال قسمت سوم هم دقت کنید که `connectionString` رو باید متناسب دیتابیس مورد نظر خودتون در کامپیوتر تغییر بدید بعد نتیجه جادویی رو ببینید !

در قسمت‌های قبلی ([^](#) و [^](#) و [^](#)) با Text Template در Visual Studio آشنا شدید. این قسمت برای تکمیل بحث در مورد ابزاری که Microsoft از آن در برنامه‌های خود از جمله Visual Studio جهت تولید کدهای اتوماتیک استفاده می‌نماید، صحبت خواهیم کرد.

قبل از آن بد نیست که بدانید چرا این ابزار T4 نام گرفته !

T4 مخفف Text Template Transformation Toolkit می‌باشد (TTTT). شکل زیر مراحل اجرای یک کد Text Template را توسط T4 نشان می‌دهد:



پس این ابزار، یک ابزار کاربردی می‌باشد که بدون Visual Studio نیز میتوان از آن استفاده کرد. نام فایل این ابزار، TextTransform.exe است و در مسیر زیر وجود دارد :

Program Files (x86)\Common Files\microsoft shared\TextTemplating\10.0

برای اطلاع از نحوه کار با TextTransform.exe خارج از محیط Visual Studio بهتر است دستور زیر را در cmd.exe اجرا کنید تا راهنمای استفاده و پارامترهای اختیاری آن را مشاهده نمایید:

TextTransform.exe -h

برای آزمایش، یک فایل متنی کنار فایل TextTransform.exe با نام Text2.tt ایجاد نمایید و کد زیر را در داخل آن بنویسید:

```
<#@ template debug="true" hostspecific="false" language="C#" #>
<#@ output extension=".txt" #>

<#@ import namespace="System.Diagnostics" #>

Report In : <#= DateTime.Now #>

<#
Process[] Procs = Process.GetProcesses();
for (int i = 0; i < Procs.Length; i++)
{
    string Pstr = Procs[i].ProcessName + " -|- " + Procs[i].Id + Environment.NewLine ;
    #><#= Pstr #><#
}
#>
```

این مثال بعد از اجرا، لیست تمام Process های جاری سیستم را به همراه Id آنها، چاپ می‌نماید.
برای تولید فایل خروجی، دستور زیر را در cmd.exe اجرا کنید :

```
TextTransform.exe -out Report1.txt Text2.tt
```

توجه کنید که فایل Text2.tt را کنار فایل TextTransform.exe قرار دهید و بعد از اجرای دستور بالا، باید خروجی در فایل Text2.tt در همان مسیر ایجاد گردد.

نکته: اگر User شما به این پوشه دسترسی ندارد و کاربر Admin نیستید احتمالاً به مشکل بر می‌خورید. می‌توانید فایل TextTransform.exe را در مکان دیگری قرار دهید و دستور را از آن محل اجرا کنید و یا برای پوشه‌ی مذکور دسترسی ایجاد نمایید.

اگر می‌خواهید بیشتر در مورد معماری T4 بدانید بهتر است مقاله زیر را مطالعه کنید:

<http://www.olegrych.com/2008/05/t4-architecture>

نکته دیگر این که برای Visual Studio، ابزارهایی جهت بهبود کار با Text Template ها وجود دارند که با جستجوی T4 Editor، نمونه‌هایی از آنها را خواهید یافت. tangible T4 Editor نمونه ای از این Plugin ها می‌باشد که به Visual Studio افزوده می‌گردد و یا یک پروژه Open Source هم برای آشنایی بسیار بیشتر با T4 در t4toolbox.codeplex.com وجود دارد که می‌توانید مشاهده کنید.

نظرات خوانندگان

نویسنده: افشین عباسپور
تاریخ: ۱۳۹۲/۱۱/۳۰ ۲۳:۲۶

متن کد این قسمت هم میتونید از [اینجا](#) دانلود کنید