禪

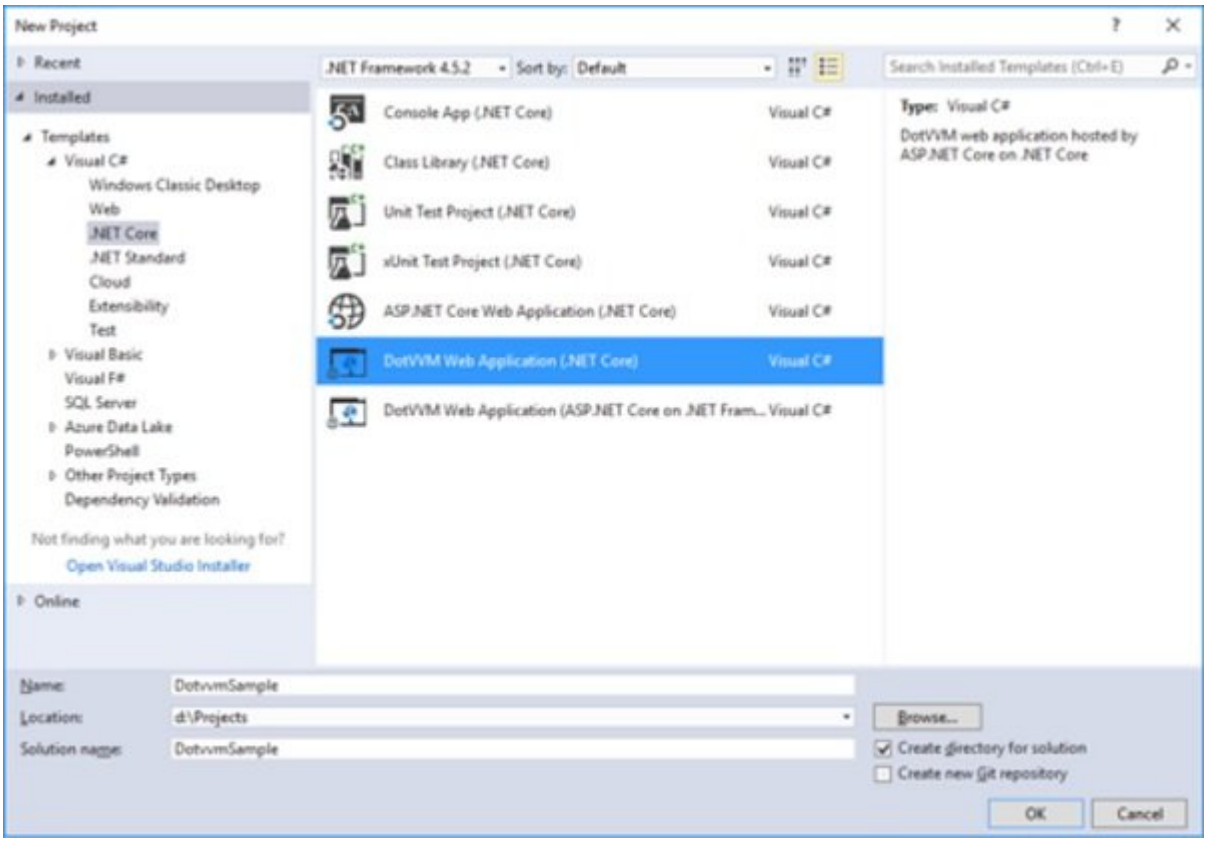
[redacted]

[redacted]

[redacted]

[redacted] DotVVM Riganti sro [redacted] .NET [redacted] DotVVM [redacted] MVVM [redacted] Visual Studio 2015 [redacted] C#, HTML [redacted] CSS [redacted] DotVVM. [redacted] Visual Studio 2017 [redacted] IntelliSense [redacted] Visual Studio Marketplace [redacted] DotVVM [redacted] .NET [redacted] DotVVM [redacted] Visual Studio 2017 [redacted] DotVVM [redacted]:

- [redacted] DotVVM Web Application (.NET Core) [redacted] .NET [redacted]. [redacted] DotVVM [redacted].
- [redacted] DotVVM Web Application (ASP.NET Core on .NET Framework) [redacted] .NET [redacted]





Framework 4.5. ایجاد و از کتابخانههای جدید ASP.NET استفاده میکند . این یک ترکیب رایج است، زیرا به شما امکان میدهد از کتابخانههایی که در حال حاضر توسط .NET پشتیبانی نمیشوند استفاده کنید .

- قالب DotVVM Web Application (OWIN on .NET Framework) یک پروژه اپلیکیشن وب کلاسیک ایجاد میکند که از .NET Framework 4.5.1 و OWIN استفاده میکند . شما میتوانید از این شیوه برای یکپارچه کردن DotVVM با MVC یا اپلیکیشنهای فرم وب موجود استفاده کنید .

در این مقاله ما از اولین گزینه (شکل 1) استفاده میکنیم . اگر شما از Visual Studio 2015 استفاده میکنید، امکان پشتیبانی DotVVM از .NET را نخواهید داشت، بهجای آن میتوانید از قالب پروژه OWIN استفاده کنید . کدهای مورد استفاده در این مقاله نیز در هر دو نسخه کار میکند . (شکل 1) اگر میخواهید اپلیکیشنهای DotVVM را روی لینوکس یا macOS توسعه دهید، میتوانید قالب جدید دات نت را نصب کنید و اپلیکیشن DotVVM را از خط فرمان بسازید . وقتی کادر محاورهای را تأیید کردید، یک پروژه DotVVM تولید میشود . شما چندین فایل مختلف در این پروژه مشاهده خواهید کرد : (شکل 2)

New Project
?
Recent
Installed
Templates
Visual C#
Windows Classic Desktop
Web
.NET Core
.NET Standard
Cloud
Extensibility
Test
Visual Basic
Visual F#
SQL Server
Azure Data Lake
PowerShell
Other Project Types
Dependency Validation
Not finding what you are looking for?
Open Visual Studio Installer
Online
.NET Framework 4.5.2
Sort by: Default
Search Installed Templates (Ctrl+E)
Console App (.NET Core) Visual C#
Class Library (.NET Core) Visual C#
Unit Test Project (.NET Core) Visual C#
xUnit Test Project (.NET Core) Visual C#
ASP.NET Core Web Application (.NET Core) Visual C#
DotVVM Web Application (.NET Core) Visual C#
DotVVM Web Application (ASP.NET Core on .NET Fram... Visual C#
Browse...
OK
Cancel

Type: Visual C#

DotVVM web application hosted by ASP.NET Core on .NET Core

Name: DotvvmSample

Location: d:\Projects

Solution name: DotvvmSample

☑ Create directory for solution

☐ Create new Git repository

شکل **1**

مطلب پیشنهادی

JS
JavaScript

جاوا اسکریپت بیحد و مرز

آینده توسعه برنامههای کاربردی در دستان جاوااسکریپت است

- Program.cs که مدخل اصلی مورد اشاره به این پروژه است و در تمام اپلیکیشنهای .NET موجود است. اگر شما داخل این فایل را نگاه کنید پیکربندی Kestrel، یکپارچهسازی با IIS، تنظیمات محتوای

دایرکتوری ریشه و شروع به کار میزبان وب را مشاهده خواهید کرد .

Solution Explorer
Search Solution Explorer (Ctrl+ö)
Solution 'DotvvmSample' (1 project)
DotvvmSample
Connected Services
Dependencies
Properties
wwwroot
ViewModels
DefaultViewModel.cs
Views
default.dothtml
DotvvmStartup.cs
Program.cs
Startup.cs
Web.config

شکل **2**

- Startup.cs که شامل دو متد است و در زمان شروع اپلیکیشن فراخوانی میشود. متد ConfigureServices برای ثبت سرویسها در آبجکت IServiceCollection استفاده میشود . در اینجا برای ثبت سرویس مورد نیاز DotVVM فراخوانی services.AddDotVVM را انجام میدهیم . متد Configure تمام میانافزارهای ASP.NET را پیکربندی میکند . برای ثبت میانافزار DotVVM که درخواستهای HTTP را پردازش میکند، به فراخوانی app.UseDotVVM<DotvvmStartup> احتیاج داریم .
- DotvvmStartup.cs که یک کلاس شامل پیکربندی DotVVM است. میتوانید از طریق خطوطی که در فهرست 1 مشاهده میکنید که مسیر پیشفرض را ثبت میکند، متد ConfigureRoutes را پیدا کنید :

```
config.RouteTable.Add(“Default”, “”, “Views/default.dothtml”);
```

فهرست **1**

وقتی کاربر از صفحه اصلی بازدید میکند، DotVVM از صفحه Views/default.dothtml استفاده خواهد کرد . علاوه بر این، این پروژه شامل فولدرهای Views و ViewModels است. اولین فولدر شامل یک فایل بهنام default.dothtml و دومی شامل کلاس DefaultViewModel.cs است. این دو فایل با یکدیگر یک صفحه وب DotVVM را میسازند . DotVVM از الگوی طراحی Model-View-ViewModel (MVVM) استفاده میکند . View یک فایل با پسوند .dothtml است که در اصل یک فایل HTML با چند شیوه کدنویسی است.

ViewModel یک کلاس C# است که دو هدف را دنبال میکند:

1. وضعیت صفحه را نگهداری میکند که شامل هر چیزی است که در صفحه توسط کاربر قابل تغییر باشد. اگر شما در یک صفحه از TextBox استفاده میکنید، باید مقدار آن را در جایی ذخیره کنید . در DotVVM شما تنها باید یک خاصیت رشتهای عمومی را در کلاس ViewModel تعیین و آن را به TextBox متصل کنید.

2. ViewModel تمام فرامینی را که در صفحه به آنها نیاز است (مثل کلیک کردن یک دکمه) را اداره میکند . در DotVVM شما باید برای هر فرمان یک متد عمومی مشخص کنید.

پروژه عملی : تشکیل یک فهرست کاری

برای نشان دادن اصول پایه DotVVM قصد داریم یک اپلیکیشن فهرست کاری ایجاد کنیم . در این پروژه کاربر قادر خواهد بود چندین وظیفه مشخص را اضافه و آنها را بهعنوان انجام شده نشانهگذاری کند . اول ما با ViewModel شروع میکنیم. باید یک کلاس تعریف کنیم که اطلاعات مربوط به هر وظیفه مشخص را

نگهداری میکند . برای ساده نگه داشتن آن تنها متن هر وظیفه (برای مثال رفتن به خرید) و تاریخ انجام آن (که درباره وظایفی که هنوز انجام نشده است null خواهد بود) را ذخیره میکنیم .

```
public class TaskData
{
  public string Text { get; set; }

  public DateTime? CompletedDate { get; set; }
}
```

فهرست **2**

در مرحله بعد ما یک فهرست از وظایف را به کلاس DefaultViewModel اضافه میکنیم .

```
// Stores the list of tasks
public List<TaskData> Tasks { get; set; } = new List<TaskData>();
```

فهرست **3**

برای اینکه کاربر بتواند یک وظیفه جدید اضافه کند، باید دو چیز را به ViewModel اضافه کنیم . ما در صفحه خود یک TextBox و یک Button داریم، بنابراین باید محتوای TextBox در ViewModel ذخیره شود و همچنین باید یک متد برای فراخوانی در زمان کلیک کردن دکمه تعریف شود .

```
// Stores the text of a new task
public string NewTaskText { get; set; }

public void AddNewTask()
{
  Tasks.Add(new TaskData()
  {
    Text = NewTaskText
  });
  NewTaskText = string.Empty;
}
```

فهرست **4**

مشاهده میکنید که متد AddNewTask یک آبجکت TaskData را به این مجموعه اضافه و خاصیت NewTaskText را تخلیه میکند تا کاربر بتواند یک وظیفه دیگر را اضافه کند . حالا ما View را میسازیم . اگر به فایل default.dothtml نگاه کنید، مشاهده خواهید کرد با دایرکتیو @viewModel شروع شده است. این دایرکتیو به DotVVM میگوید کدام کلاس بهعنوان یک ViewModel استفاده خواهد شد .

```
@viewModel DotvvmSample.ViewModels.DefaultViewModel, DotvvmSample
```

این دایرکتیو توسط عنصر <html> دنبال میشود . اجازه دهید کد زیر را داخل <body> اضافه کنیم .

```
<h1>Task List</h1>
<div>
  <dot:TextBox Text="{value: NewTaskText}" />
  <dot:Button Text="Add Task" Click="{command: AddNewTask()}" />
</div>
```

فهرست **5**

[ссылка]

[ссылка]

Click="{command: NewTask()}"

<dot:Repeater DataSource="{value: Tasks}">

    {{value: Text}}

</dot:Repeater>

6

عبارت به عنصر اضافه کند . در غیر این صورت، این کلاس CSS به عنصر اضافه نخواهد شد و مسلماً هر زمان مقدار این عبارت تغییر میکند، کلاس CSS اضافه شده به عنصر بلافاصله بهروزرسانی خواهد شد . میتوانید با استفاده از دستور زیر به یک عملکرد مشابه دست پیدا کنید :

```
class=”{value: CompletedDate != null ? “completed” : “”}”
```

اما خواندن این نوع از عبارات دشوار است، بهویژه وقتی به ترکیب چندین کلاس CSS با شروط مختلف احتیاج دارید . حالا میتوانید این دستور CSS را اضافه کنید :

```
.completed {
  text-decoration: line-through;
}
```

برای مشخص کردن وظایف بهعنوان تکمیل شده، تابع زیر را در ViewModel اضافه میکنیم . (فهرست 7)

```
public void CompleteTask(TaskData task)
{
  task.CompletedDate = DateTime.Now;
}
```

فهرست 7

مشاهده میکنید که متد CompleteTask یک پارامتر را تعریف میکند . در فرامین ViewModel میتوانید هر تعداد پارامتر که مایل هستید مشخص کنید . حالا اجازه دهید دکمه زیر را به Repeater اضافه کنیم :

```
<dot:LinkButton Text=”Completed”
          Click=”{command: _root.CompleteTask(_this)}”
          Visible=”{value: CompletedDate == null}”/>
```

مطلب پیشنهادی

دنیای فناوری در تسخیر جاوا اسکریپت

دلایلی که ثابت می کند جاوا اسکریپت می تواند کشتی بزرگ داده ها را هدایت کند _

از آنجا که تمام الصاقهای داخل کنترل Repeater بهشکل خاصی از آیتم Repeater مورد ارزیابی قرار گرفته است (آبجکت TaskData)، برای دسترسی به متد CompleteTask به استفاده از _root.CompleteTask نیاز دارید . فراموش نکنید که این متد را بهجای کلاس TaskData در کلاس DefaultViewModel تعریف کردهایم . متغیر _root خودش به ViewModel دسترسی پیدا میکند .
توجه داشته باشید که _this را بهعنوان یک آرگمان در این متد در نظر گرفتهایم . نشانگر _this مشخصکننده محتوای الصاق جاری است که همان آیتم Repeater جاری محسوب میشود. در کنترلهای سلسله مراتبی پیچیده ممکن است به متغیر _parent نیز احتیاج داشته باشید . همچنین، خصوصیت Visible را نیز تنظیم کردهایم تا دکمه تنها برای وظایف انجام نشده نمایش داده شود . اگر Visible مقدار نادرست را نشان دهد، DotVVM با تنظیم خصوصیت نمایش در CSS این عنصر را پنهان میکند . برای نمایش CompletedDate میتوانیم از کتابخانه Literal استفاده کنیم و فرمت داده را تعیین کنیم . DotVVM از همان فرمت ساختار دستور شناخته شده .NET استفاده میکند :

```
<dot:Literal Text=”{value: CompletedDate}” FormatString=”yyyy/MM/dd” />
```

جمعبندی

DotVVM از تعداد زیادی کنترل و قابلیت داخلی تشکیل شده است . میتوانید از فرمهای کنترل پایه مثل ComboBox، CheckBox یا RadioButton استفاده کنید . مجموعهای از کنترلهای پیشرفته نیز مثل GridView یا FileUpload به همراه کنترل SpaContentPlaceHolder وجود دارد که اپلیکیشن وب شما را به یک اپلیکیشن تکصفحهای تبدیل میکند . میتوانید از ویژگیهای تأیید اعتبار .NET، محلیسازی RESX و فیلترهای کاربردی استفاده کنید .

DotVVM به طور همزمان از .NET Framework و .NET Core پشتیبانی میکند و میتوان از آن در پروژههای موجود ASP.NET به همراه ASP.NET MVC یا Web API استفاده کرد . درنهایت، امکانات بسیار دیگری نیز وجود دارد که طی ماههای آینده به DotVVM اضافه خواهد شد. یکی از ضروریترین سناریوهای موجود یکپارچهسازی با REST API است. چنین قابلیتی به شما اجازه میدهد تا بخشهای ViewModel (برای مثال، دادههای موجود در کنترل GridView) را با استفاده از REST API بارگذاری و بهجای درگیر شدن با متدهای ViewModel پارامترهای REST API را فراخوانی کنید .

نویسنده :
محسن آقاجانی
منبع :
Visual Studio Magazine
دسته بندی:
کارگاه
برنامه نویسی
تاریخ انتشار :
13:30 - 25/12/1396
برچسب:
ساختِ اپلیکیشنِ وپ بدونِ جاوا اِسکریپت - ساختِ اپلیکیشن وِب - ساختِ اپلیکیشن - جاوا اِسکریپت - DotVVM

نشانی
□□□□:https://www.shabakeh-mag.com/workshop/11458/%DA%86%D9%87%D8%A7%D8%B1%DA%86%D9%88%D8%A8-%DA%A9%D8%A7%D8%B1%D8%A8%D8%B1%D8%AF%DB%8C-%D8%A8%D8%B1%D8%A7%DB%8C-%D8%B3%D8%A7%D8%AE%D8%AA-%D8%A7%D9%BE%D9%84%DB%8C%DA%A9%DB%8C%D8%B4%D9%86%E2%80%8C%D9%87%D8%A7%DB%8C-%D9%88%D8%A8-%D8%A8%D8%AF%D9%88%D9%86-%D9%86%DB%8C%D8%A7%D8%B2-%D8%A8%D9%87-%D8%AC%D8%A7%D9%88%D8%A7%D8%A7%D8%B3%DA%A9%D8%B1%DB%8C%D9%BE%D8%AA