



+




+



Blazor per uno sviluppatore Web Form



 scansiona per
codice e slide

Andrea Dottor

Microsoft MVP Developer Technologies



Sponsor



Un po' di storia

- | | | |
|--------|--------------------|--------------------|
| • 2001 | Web Forms | .NET Framework 1.0 |
| • 2008 | ASP.NET MVC | .NET Framework 3.5 |
| • 2013 | ASP.NET MVC 5.2 | .NET Framework 4.5 |
| • 2016 | ASP.NET Core 1.0 | .NET Core 1.0 |
| • 2019 | | .NET Framework 4.8 |
| • 2019 | Blazor server | .NET Core 3.0 |
| • 2020 | Blazor WebAssembly | .NET Core 3.2 |

Perchè "Blazor per uno sviluppatore WebForm"?

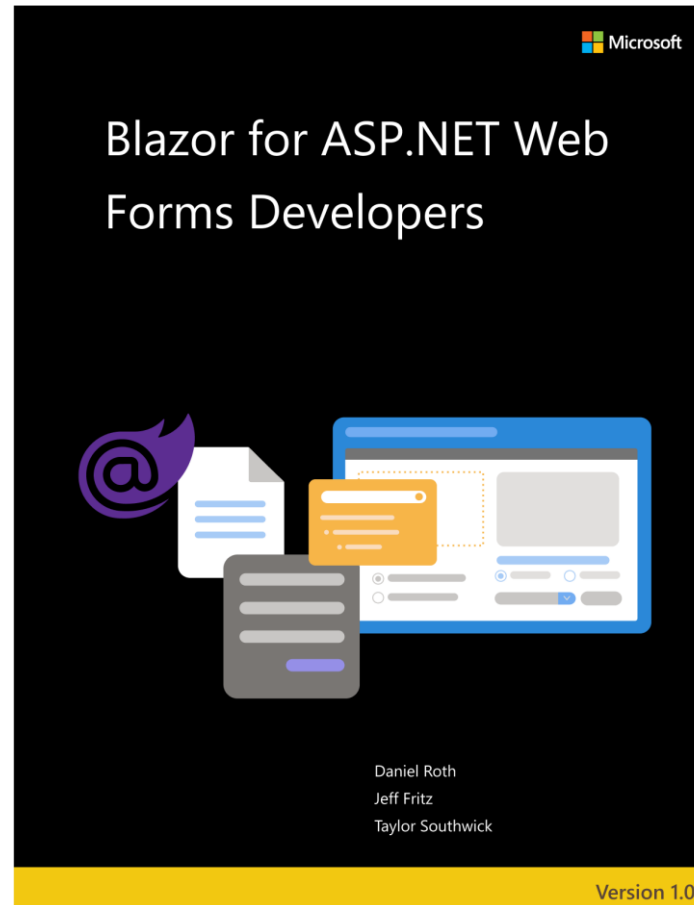
- Il .NET Framework non riceverà nuove funzionalità
- Web Forms non è supportato in ASP.NET Core
- Blazor ha un modello di sviluppo più simile a Web Form rispetto a MVC o Razor Pages

"After .NET Core 3.0 we will not port any more features from .NET Framework. If you are a Web Forms developer and want to build a new application on .NET Core, we would recommend Blazor which provides the closest programming model. "

[.NET Core is the Future of .NET] 06.06.2019

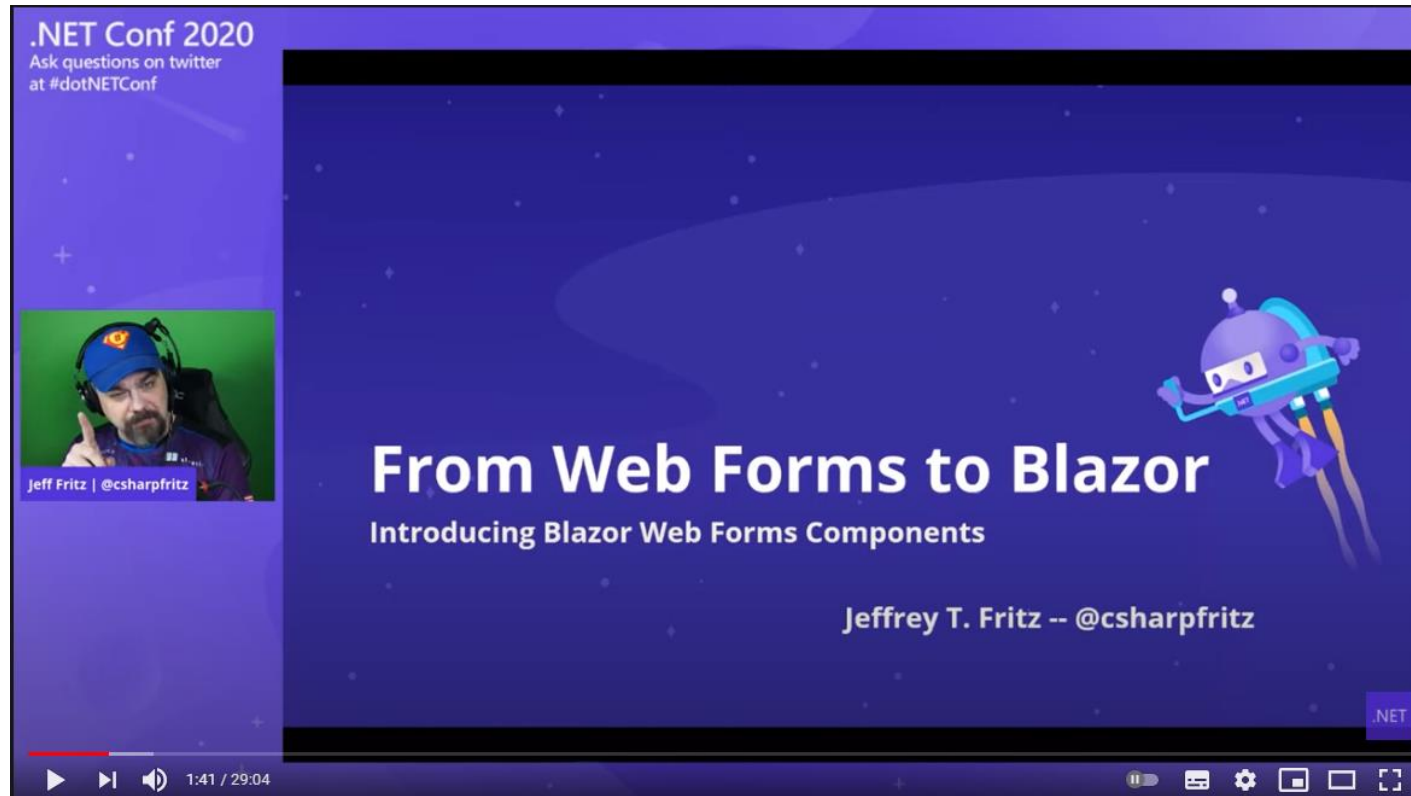
<https://devblogs.microsoft.com/dotnet/net-core-is-the-future-of-net/>

Blazor for ASP.NET Web Forms Developers



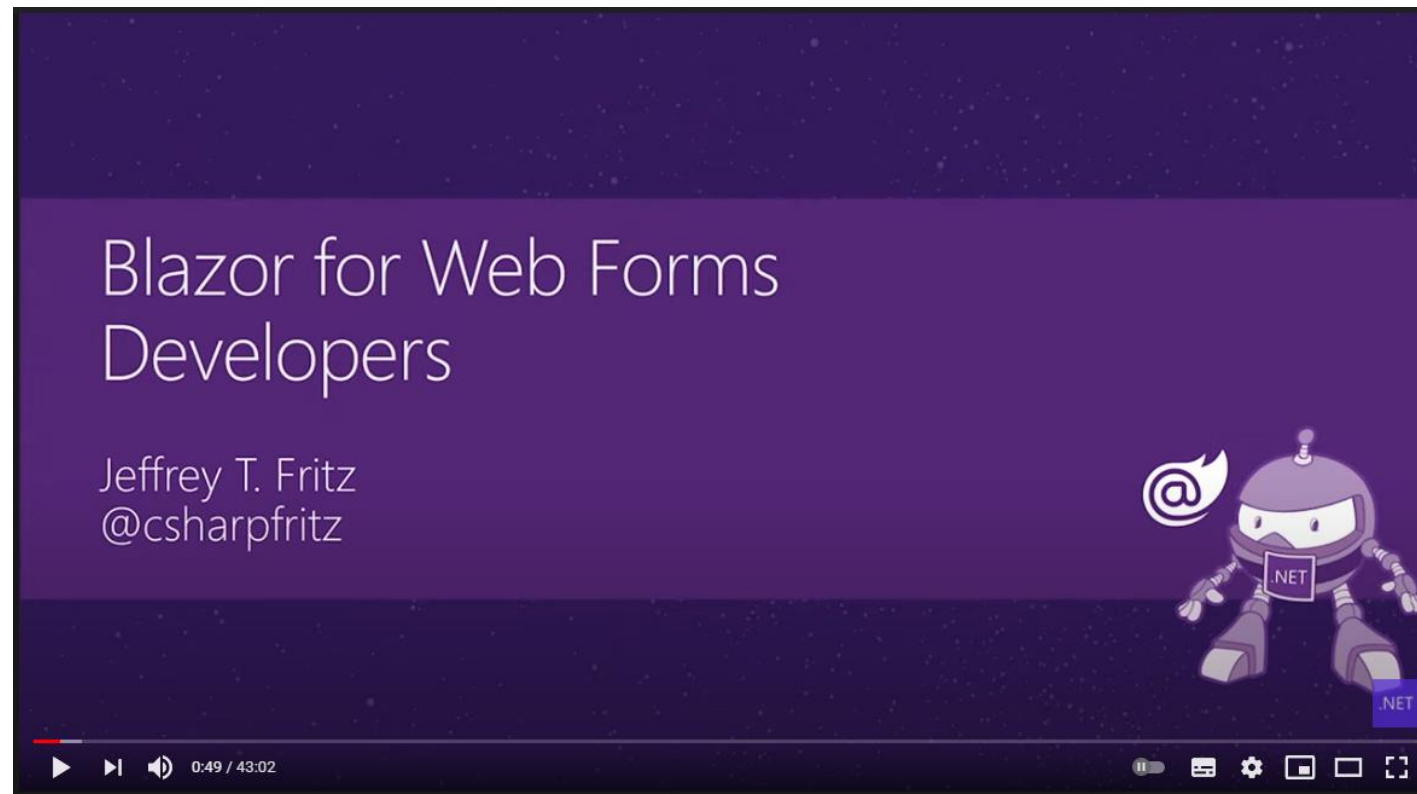
<https://docs.microsoft.com/en-us/dotnet/architecture/blazor-for-web-forms-developers/>

(.NET Conf 2020) From Web Forms to Blazor Introducing the Blazor Web Forms Components



<https://www.youtube.com/watch?v=ceGzm-pBhx4>

Blazor for Web Form and C# Developers



<https://www.youtube.com/watch?v=gmpoA3LEEE>

Migrating a Windows Forms App to Blazor: The Amazing and True Story of GIFBot



<https://www.youtube.com/watch?v=NRDdu67VJH0>

Pagine e Componenti

- Blazor ha il concetto di componente
 - File con estensione .razor
 - **Corrispondenza con i file ascx di Web Form**
- Un componente può essere anche una pagina
 - **Stesso uso dei file aspx in Web Form**
- I componenti sono scritti utilizzando la sintassi Razor, combinando codice HTML e C#
 - Il codice C# può essere inserito in un blocco "@code" o in un file separato
 - **Stesso uso dei file MyPage.aspx.cs, MyComponent.ascx.cs**

Components → WebForm UserControl

- Un componente può venir richiamato utilizzando il suo nome come fosse un tag html
- Possono avere parametri (in ingresso)
 - **E' possibile passare oggetti complessi** e non solo tipi semplici come capitava in Web Form
- Possono esporre eventi

```
<Details  
  Brands="@catalogBrands"  
  Types="@catalogTypes"  
  OnEditClick="EditClick"  
  @ref="DetailsComponent"></Details>
```

BlazorWebFormsComponents

Editor Controls

- AdRotator
- Button
- HiddenField
- HyperLink
- Image
- ImageButton
- Label
- LinkButton
- Literal

Data Controls

- DataList

- FormView
- GridView
- ListView
- Repeater

Validation Controls

- CompareValidator
- CustomValidator
- RegularExpressionValidator
- RequiredFieldValidator
- ValidationSummary

Navigation Controls

- TreeView

```
<asp:Repeater
  DataMember="string"
  DataSource="string"
  DataSourceID="string"
  EnableTheming="True|False"
  EnableViewState="True|False"
  ID="string"
  OnDataBinding="DataBinding event handler"
  OnDisposed="Disposed event handler"
  OnInit="Init event handler"
  OnItemCommand="ItemCommand event handler"
  OnItemCreated="ItemCreated event handler"
  OnItemDataBound="ItemDataBound event handler"
  OnLoad="Load event handler"
  OnPreRender="PreRender event handler"
  OnUnload="Unload event handler"
  runat="server"
  Visible="True|False"
>
  <AlternatingItemTemplate>
    <!-- child controls -->
  </AlternatingItemTemplate>
  <FooterTemplate>
    <!-- child controls -->
  </FooterTemplate>
  <HeaderTemplate>
    <!-- child controls -->
  </HeaderTemplate>
  <ItemTemplate>
    <!-- child controls -->
  </ItemTemplate>
  <SeparatorTemplate>
    <!-- child controls -->
  </SeparatorTemplate>
</asp:Repeater>
```

<https://github.com/FritzAndFriends/BlazorWebFormsComponents>

Modello Event-Driven

- I componenti hanno un proprio ciclo di vita basato su eventi
 - OnInitialized{Async}
 - OnParametersSet{Async}
 - OnAfterRender{Async}
- I componenti possono scatenare eventi
 - @on{EVENT}
- Possiamo gestire gli eventi dei controlli associando un nostro metodo

```
<button class="btn btn-primary" @onclick="UpdateHeading">
    Update heading
</button>

@code {
    private void UpdateHeading(MouseEventArgs e)
    {
        ...
    }
}
```

Mantenimento dello stato → ViewState

- WebForm fa uso del ViewState per mantenere i valori tra un postback e l'altro
- Un componente Blazor mantiene i propri valori fino a quando non viene rimosso o la pagina non viene ricaricata
 - Ogni dato salvato in una proprietà o field viene mantenuto

Confronto Web Form - Blazor





Cosa c'è da imparare

Razor engine

- Il carattere @ permette di richiamare codice C# all'interno del markup HTML
- Implicit Razor expressions:

```
<p>@DateTime.Now</p>  
<p>@DateTime.IsLeapYear(2016)</p>  
  
<p>@GenericMethod<int>()</p>  
  
<p>@await DoSomething("hello", "world")</p>
```

- Explicit Razor expressions:

```
<p>Last week this time: @(DateTime.Now - TimeSpan.FromDays(7))</p>
```


Razor engine

- Razor code blocks:
- Control structures:

```
@{  
    var quote = "The future depends on what you do today. - Mahatma Gandhi";  
}  
  
<p>@quote</p>  
  
@{  
    quote = "Hate cannot drive out hate, only love can do that. - Martin Luther King, Jr.";  
}  
  
<p>@quote</p>
```

```
@if (value % 2 == 0)  
{  
    <p>The value was even.</p>  
}  
else if (value >= 1337)  
{  
    <p>The value is large.</p>  
}  
else  
{  
    <p>The value is odd and small.</p>  
}
```

Startup.cs → Global.asax

- La classe **Startup** viene richiamata all'avvio dell'applicazione e permette di configurare quali funzionalità l'applicazione utilizza
 - [Simile al Global.asax \(Application_Start, Application_BeginRequest, ...\)](#)
- **ConfigureServices**: aggiunta delle funzionalità/servizi necessari all'applicazione, andando a configurare la Dependency Injection
- **Configure**: permette di specificare i middleware necessari all'applicazione
 - Ogni richiesta HTTP viene elaborata da questi middleware nel preciso ordine con il quale li andiamo a inserire

Startup.ConfigureServices()

```
// This method gets called by the runtime. Use this method to add services to the container.  
// For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940  
0 references  
public void ConfigureServices(IServiceCollection services)  
{  
    services.AddRazorPages();  
    services.AddServerSideBlazor();  
    services.AddSingleton<WeatherForecastService>();  
}
```

Startup .Configure()

```
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Error");
        // The default HSTS value is 30 days. You may want to change this for production scenarios, see https:// aka.ms/aspnetcore-hsts.
        app.UseHsts();
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.UseRouting();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapBlazorHub();
        endpoints.MapFallbackToPage("/_Host");
    });
}
```

Differente modalità per Blazor WebAssembly

- Nella classe Program è possibile configurare l'applicazione e definire i servizi utilizzando la Dependency Injection
- Non c'è una definizione di pipeline

```
1 reference
public class Program
{
    0 references
    public static async Task Main(string[] args)
    {
        var builder = WebAssemblyHostBuilder.CreateDefault(args);
        builder.RootComponents.Add<App>("#app");

        builder.Services.AddScoped(sp => new HttpClient { BaseAddress = new Uri(builder.HostEnvironment.BaseAddress) });

        await builder.Build().RunAsync();
    }
}
```

Dependency Injection

- **Transient:** il servizio viene creato ogni volta che viene richiesto.
- **Scoped:** il servizio viene creato una sola volta per "scope". In un'applicazione web corrisponde ad una richiesta http.
 - **In Blazor Scoped si comporta come un Singleton*
- **Singleton:** il servizio viene creato la prima volta che viene richiesto. Tutte le volte successive, viene passata l'istanza già creata.

```
services.AddScoped<IMyDependency, MyDependency>();  
services.AddSingleton<IMyDependency, MyDependency>();
```

Dependency Injection

- In un componente blazor:
`@inject IDataAccess DataRepository`
- Nel caso venga utilizzata una classe base (codice C# in un file separato) è possibile utilizzare l'InjectAttribute:
`[Inject]`
`protected IDataAccess DataRepository { get; set; }`
- Per utilizzarla in classi create tramite DI, utilizzare la **Constructor injection**

Dependency Injection - Blazor default services

- **HttpClient**: permette di eseguire richieste HTTP
 - Scoped
- **IJSRuntime**: fornisce i metodi per invocare funzioni JavaScript
 - Blazor WebAssembly: Singleton
 - Blazor Server: Scoped
- **NavigationManager**: helper per gestire gli url e per navigare
 - Blazor WebAssembly: Singleton
 - Blazor Server: Scoped

Gestione della configurazione (Blazor Server)

- Non esiste più la classe ConfigurationManager
- Il file di configurazione utilizza **JSON** e non XML
- La configurazione viene letta utilizzando la Dependency Injection
 - Servizio **IConfiguration** passato tramite il costruttore
 - **Options pattern**

JSInterop

- Tramite il servizio **IJSRuntime** che ci viene iniettato dalla DI è possibile invocare funzioni JavaScript
 - Il tipo di ritorno può specificando il **TValue** nel metodo **InvokeAsync**

```
/// <summary>  
/// Chiamo la funzione javascript getGravatar che mi ritorna l'url da utilizzare  
/// per visualizzare l'immagine Gravatar  
/// </summary>  
1 reference | Andrea Dottor, 150 days ago | 1 author, 1 change  
public async Task<string> GetGravatarUrl(string email)  
..... => await _jsRuntime.InvokeAsync<string>("getGravatar", email, 100);
```

Demo progetto completo





Migrare applicazioni esistenti



nessuna scorciatoia da poter prendere



Migrate from ASP.NET Web Forms to Blazor

- <https://docs.microsoft.com/en-us/dotnet/architecture/blazor-for-web-forms-developers/migration>
 - Server-side versus client-side hosting
 - Create a new project
 - Enable startup process
 - Migrate HTTP modules and handlers to middleware
 - Migrate static files
 - Migrate runtime bundling and minification setup
 - Migrate ASPX pages
 - Migrate configuration
 - Migrate data access
 - Architectural changes
 - Migration conclusion

Blazor Server o WebAssembly?

- Blazor Server permette di richiamare eventuale logica di business e/o accesso ai dati direttamente dai componenti
 - Con Blazor Web Assembly si deve comunicare con il backend utilizzando delle API Rest, gRPC o SignalR
- Se la logica dell'applicazione non è nella UI
 - Viene riscritto "*solo*" il progetto di UI e non l'intero applicativo
 - Con Blazor Web Assembly la logica di business può essere richiamata da delle API Rest
 - Con Blazor Server la logica di business può essere richiamata direttamente dai componenti Blazor

Una terza scelta: RazorComponent

- Uso di componenti Blazor in applicazioni Razor Pages
- Permettono di non vincolarsi ad una SPA Blazor
- Possibilità di inserire Blazor per le sole parti dinamiche dell'applicazione

```
@await Html.RenderComponentAsync<Counter>(RenderMode.Server,
    new {
        Label = "Blazor Conf 2021 - Counter",
        StartIndex = 42
    })
```


RazorComponent



Controlli di terze parti

- Gran parte delle aziende che sviluppa controlli ha realizzato una propria libreria per Blazor
- Prestare attenzione alla maturità di queste librerie
 - Non sempre sono complete di tutti i controlli
 - Non tutti i controlli hanno le stesse funzionalità dei rispettivi per ASP.NET Core

Conclusioni

- Blazor velocizza la migrazione di applicativi Web Form (rispetto all'uso delle sole Razor Pages o MVC)
- La curva di apprendimento si abbassa
- Sviluppatori desktop (WPF, Windows Forms) possono passare allo sviluppo web con maggiore facilità

Domande? Dubbi? Perplessità?





Andrea Dottor

Microsoft MVP Developer Technologies



www.dottor.net



andrea@dottor.net



[@dottor](https://twitter.com/dottor)



Partecipate alle discussioni



<https://fb.me/groups/998755440506950/>



<https://www.linkedin.com/groups/8896269/>



<https://github.com/blazordevita>



Blazor Developer Italiani



blazordev.it



blazordevita



blazordevita



fb.me/blazordeveloperitaliani

Alla prossima!



Seguite la community!

BLAZOR DEVELOPER ITALIANI

La Community italiana su Blazor



<https://blazordev.it/>



<https://twitter.com/blazordevita>



<https://github.com/blazordevita>



<https://fb.me/blazordeveloperitaliani>



Blazor Developer Italiani



blazordev.it



[blazordevita](https://github.com/blazordevita)



[blazordevita](https://twitter.com/blazordevita)



fb.me/blazordeveloperitaliani