عنوان مقاله: مفهوم Data Binding در Blazor چیست؟

نویسنده مقاله: تیم فنی نیک‌آموز

تاریخ انتشار: ۲ تیر ۱۴۰۳

منبع: https://nikamooz.com/data-binding-in-blazor/

Data Binding در Blazor حکم قلب تپنده برنامه‌های Blazor را دارد. این ویژگی کلیدی به شما امکان می‌دهد تا داده را به‌راحتی بین اجزای مختلف رابط کاربری انتقال دهید. در این مقاله، گام به گام با انواع مختلف Data Binding، مزایا و معایب هر یک و نحوه استفاده از آن‌ها آشنا خواهید شد. علاوه‌براین، به بررسی چالش‌های Data Binding می‌پردازیم. درنهایت، با مثال‌های پیشرفته‌ای از این ویژگی کلیدی در Blazor، می‌توانید مهارت خود را در این زمینه افزایش دهید و برای ساخت برنامه‌های آن آماده شوید.

Blazor چیست؟ فریم ورک ساخت اپلیکیشن های تک صفحه ای

Blazor یک فریم‌ورک مدرن و مبتنی‌بر وب است که با استفاده از HTML و CSS و C# به شما در ساخت برنامه‌های وب قدرتمند و کاربرپسند کمک می‌کند. این فریم‌ورک جدید مایکروسافت به شما امکان می‌دهد تا روابط کاربری وب تک‌صفحه‌ای (SPA) را با استفاده از سی‌شارپ و HTML بسازید.

نام Blazor ترکیبی از Browser (مرورگر) و Razor (موتور تولید خروجی HTML در دات‌نت) است. به‌عبارتی، Blazor به جای اینکه کدهای HTML را در سرور اجرا کند، آن‌ها را مستقیماً در مرورگر اجرا می‌کند؛ این یعنی سرعت بالاتر، کارایی بهتر و توسعه‌ای لذت‌بخش‌تر برای شما به‌عنوان توسعه‌دهنده.

اهمیت و کاربرد Blazor در توسعه وب

با Blazor دیگر نگران پلاگین‌های اضافی نخواهید بود. برخلاف Silverlight، فناوری قدیمی مایکروسافت، Blazor نیازی به نصب هیچ پلاگینی ندارد. این فریم‌ورک می‌تواند هم روی سرور و هم به‌طور مستقیم روی مرورگر با استفاده از WebAssembly اجرا شود. WebAssembly یک استاندارد وب است. به‌عبارتی، اپلیکیشن‌های Blazor روی تمام مرورگرهای مطرح ویندوز، لینوکس، مک، اندروید و iOS قابل اجرا هستند. با Blazor، دیگر نیازی به نوشتن خطوط طولانی کد جاوااسکریپت نیست. این فریم‌ورک از موتور قدرتمند Razor برای تولید خروجی HTML بهره می‌برد و اطمینان می‌دهد کد شما خوانا و قابل نگهداری است.

Data Binding چیست؟ جادوی هماهنگی بین کد و رابط کاربری

برای درک نقش Data Binding در Blazor سراغ ویژگی کلیدی دیگری می‌رویم. تصور کنید درحال طراحی صفحه‌ای هستید که به‌راحتی کاربرانتان با آن ارتباط برقرار می‌کنند. در Blazor، یک ویژگی کلیدی به نام data binding به کمک

شما می‌آید تا دیتای برنامه‌ای را که با C# نوشته شده است، به‌سادگی با رابط کاربری هماهنگ کنید. این همگام‌سازی دوطرفه است؛ یعنی هم می‌توانید اطلاعات را از برنامه به رابط کاربری منتقل کنید و هم برعکس.

اهمیت و کاربرد Data Binding در توسعه وب

با Data Binding در Blazor رابط کاربری شما به‌طور خودکار به تغییرات در کد C# واکنش نشان می‌دهد و کد شما نیز از تمام تعاملات کاربر با رابط کاربری مطلع می‌شود. این همگام‌سازی، وب‌سایت شما را پویاتر و کاربرپسندتر کرده و درنهایت، تجربه کاربری لذت‌بخش‌تری برای کاربران رقم می‌زند.

انواع روش های Data Binding در Blazor

Blazor در دنیای Data Binding مثل یک پل ارتباطی بین داده‌ها و رابط کاربری عمل می‌کند. روش‌های مختلفی برای نمایش داده‌ها به شکل دلخواه وجود دارد. رایج‌ترین روش‌های Data Binding در Blazor سه طریق زیر است:

- Data Binding یک‌طرفه
- Data Binding دوطرفه
- Event Binding

Data Binding یک طرفه: جهتی واحد برای انتقال اطلاعات

در Data Binding یک‌طرفه، اطلاعات فقط از مدل داده به رابط کاربری سرازیر می‌شوند. به عبارت دیگر، هرگونه تغییر در مدل داده، به‌طور خودکار در رابط کاربری منعکس می‌شود، اما برعکس آن صادق نیست.

مزایا

- سادگی و سهولت استفاده
- ایده‌آل برای نمایش اطلاعات ثابت یا به‌روزرسانی‌شده ازطریق منبع خارجی

معایب

- عدم امکان دریافت ورودی از کاربر
- محدودیت در تعامل پویا با رابط کاربری

مثال: نمایش نام کاربری در یک برچسب HTML

```
<p>@user.Name</p>
```

Data Binding دوطرفه: گفتگوی دوطرفه بین داده و رابط کاربری

Data Binding دوطرفه، راهکار جذاب دنیای Data Binding در Blazor است. در این روش، اطلاعات می‌توانند هم از مدل داده به رابط کاربری و هم از رابط کاربری به مدل داده انتقال پیدا کنند؛ یعنی هر تغییری در یکی از دو طرف، به‌طور خودکار در طرف دیگر نیز اعمال می‌شود.

مزایا

- امکان ایجاد رابط کاربری پویا و تعاملی
- سهولت دریافت ورودی از کاربر و به‌روزرسانی مدل داده

معایب

- پیچیدگی بیشتر نسبت به Data Binding یک‌طرفه
- نیاز به مدیریت دقیق رویدادها و به‌روزرسانی‌ها

مثال: ایجاد یک فرم ورود به سیستم با استفاده از TextBox برای دریافت نام کاربری و رمز عبور

```
<input type="text" @bind="username" />
<input type="password" @bind="password" />
```

Event Binding: واکنش به رویدادهای کاربر

در ادامه روش‌های Data Binding در Blazor روش Event Binding به شما امکان می‌دهد تا به رویدادهای کاربر، مانند کلیک، تغییر یا فوکوس، در عناصر رابط کاربری واکنش نشان دهید.

مزایا

- امکان ایجاد رابط کاربری پویا و تعاملی
- کنترل دقیق رفتار رابط کاربری در پاسخ به actions کاربر

معایب

- افزایش پیچیدگی برنامه
- نیاز به نوشتن کد بیشتر برای مدیریت رویدادها

مثال: ارسال اطلاعات فرم ورود به سیستم به سرور با کلیک برروی دکمه «ورود»

```
<button @onclick="SubmitForm">ورود</button>
```

Data Binding با استفاده از Component Parameters

برای درک بهتر اهمیت Data Binding در Blazor ، در این قسمت می‌خواهیم نحوه ساخت یک کامپوننت ساده به نام MyFirstComponent در HTML را بررسی کنیم که یک عنوان ثابت نمایش می‌دهد. سپس، نحوه ایجاد یک کامپوننت دیگر به نام MySecondComponent را توضیح می‌دهیم که یک مقدار شمارنده را به‌صورت پویا نمایش می‌دهد. این مثال، مفهوم اتصال یک‌طرفه داده (One-way Data Binding) در Blazor را نشان می‌دهد.

۱. ایجاد یک کامپوننت ساده

- فایلی به نام Components را در پوشه Client پروژه Blazor خود ایجاد کنید.
- فایل دیگری با نام MyFirstComponent.razor در پوشه Components ایجاد کنید و کد زیر را در آن قرار دهید:

```
<div>
<h2>This is my first component</h2>
</div>
```

این کد، یک کامپوننت ساده ایجاد می‌کند که عنوان «This is my first component» را نمایش می‌دهد.

۲. استفاده از کامپوننت در صفحه

برای استفاده از کامپوننت MyFirstComponent در این صفحه از دستور @using در فایل _Imports.razor استفاده می‌کنیم. کافیست کد زیر را به فایل Index.razor اضافه کنید:

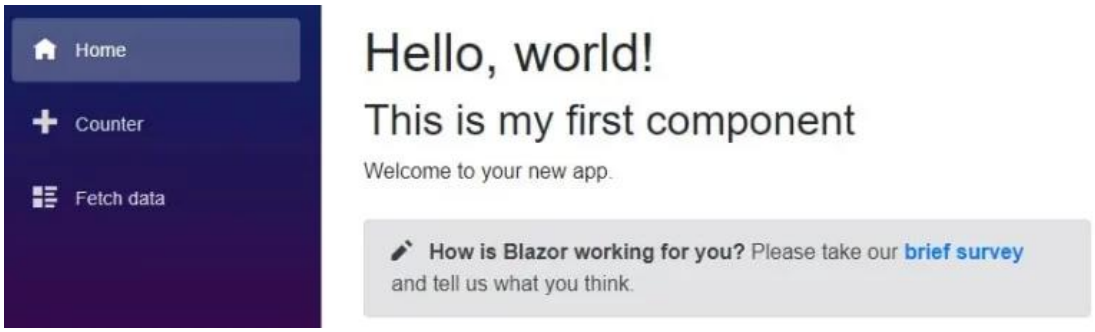
```
@page "/"

<h1>Hello, world!</h1>
<MyFirstComponent />

Welcome to your new app.

<SurveyPrompt Title="How is Blazor working for you?" />
```

با اجرای برنامه، عنوان ایجاد شده توسط کامپوننت MyFirstComponent را مشاهده خواهید کرد.



تاکنون، مقدار نمایش‌داده‌شده توسط کامپوننت ما ثابت بوده است. اکنون می‌خواهیم مقداری را به‌صورت پویا نمایش دهیم.

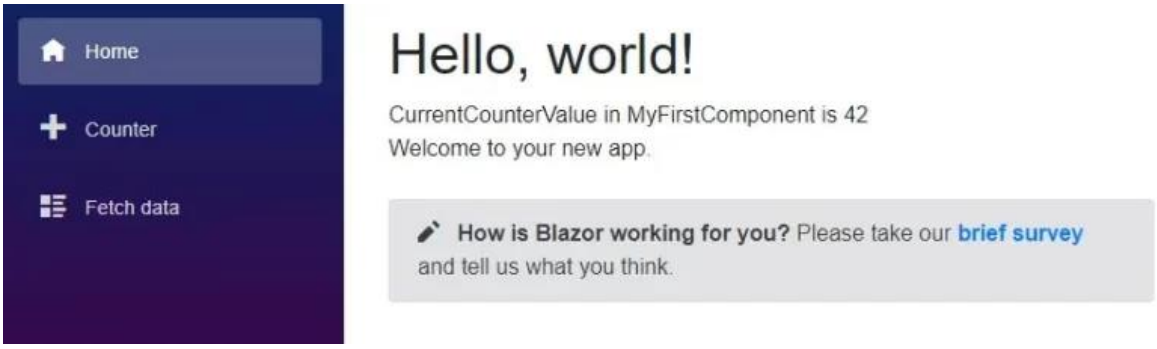
۳. نمایش داده به‌صورت پویا

فایل MyFirstComponent.razor را ویرایش کرده و کد زیر را در آن قرار دهید:

```
<div>
    CurrentCounterValue in MyFirstComponent is @CurrentCounterValue
</div>

@code {
    private int CurrentCounterValue = 42;
}
```

از این کد یک مقدار شمارنده به نام CurrentCounterValue را با مقدار اولیه ۴۲ تعریف می‌کند. مقدار این شمارنده درون کامپوننت با استفاده از نماد @ نمایش داده می‌شود.



در مثال فعلی، مقدار شمارنده درون خود کامپوننت تعریف شده است. اما می‌خواهیم این مقدار را از یک کامپوننت دیگر دریافت کنیم.

۴. دریافت مقادیر ازطریق پارامترها در Blazor

اگر بخواهید کامپوننت والد بگوید چه عددی را نمایش دهید، به‌صورت زیر اقدام کنید:

- یک کامپوننت جدید به نام MySecondComponent بسازید.
- کد HTML آن را از MyFirstComponent کپی کنید.
- سپس private member را به یک public property تغییر دهید.

این کار باعث می‌شود تا کامپوننت جدید بتواند مقدار دریافتی را به‌عنوان یک public property در اختیار داشته باشد.

کد MySecondComponent به شکل زیر خواهد شد:

```
<div>
  CurrentCounterValue in MySecondComponent is @CurrentCounterValue
</div>

@code {
  public int CurrentCounterValue { get; set; }
}
```

- در صفحه Counter یک کامپوننت MySecondComponent اضافه کنید و مقدار CurrentCounterValue آن را با مقدار currentCount تنظیم کنید:

```
<MySecondComponent CurrentCounterValue=@currentCount/>
```

با اجرای برنامه و رفتن به صفحهCounter ، یک خطا در کنسول مرورگر ظاهر می‌شود.

ASM: System.InvalidOperationException: Object of type 'OneWayBinding.Client.Components.MySecondComponent' has a property matching the name 'CurrentCounterValue', but it does not have [ParameterAttribute] or [CascadingParameterAttribute] applied.

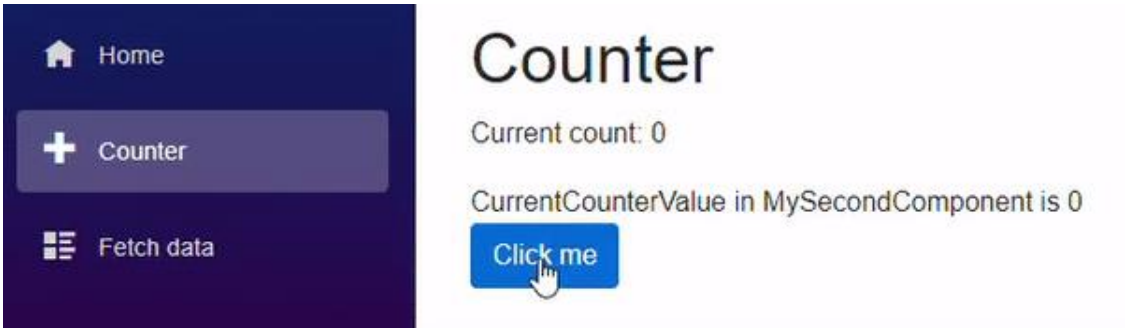
این خطا به‌طور واضح به ما می‌گوید که چه چیزی کم است. برای اضافه‌کردن یک پارامتر به کامپوننت، باید خاصیت کامپوننت خود را با ویژگی [Parameter] تزئین کنیم.

```
<div>
  CurrentCounterValue in MySecondComponent is @CurrentCounterValue
</div>

@code {
  [Parameter]
  public int CurrentCounterValue { get; set; }
}
```

این کد به Blazor اطلاع می‌دهد ما یک پارامتر در کامپوننت خود می‌خواهیم که ازطریق چیزی شبیه به یک ویژگی HTML قابل تنظیم باشد. هر زمان که کامپوننت والد رندر شد، Blazorهر کامپوننت فرزندی که مقادیر پارامتر را برای آن فراهم می‌کند، دوباره رندر کند. این کد تضمین می‌کند کامپوننت فرزند دوباره رندر می‌شود تا هر تغییر احتمالی را در وضعیتی نشان دهد که ازطریق یک property با [Parameter] به کامپوننت منتقل می‌شود.

اگر برنامه را دوباره اجرا کنید و به صفحه Counter بروید، خواهید دید هر زمان currentCount در صفحه Counter تغییر کند، آن تغییر ازطریق خاصیت CurrentCounterValue به کامپوننت Child منتقل می‌شود.



با این مثال احتمالا پی برده باشید چگونه می‌توانید در Blazor از قابلیت Data Binding برای ارتباط بین کامپوننت‌ها و به‌روزرسانی محتوای پویا استفاده کنید.

Data Binding در فرم های Blazor

در این قسمت از آموزش Data Binding در Blazor ، چند مثال از نحوه استفاده از Data Binding در فرم‌های Blazor برایتان آوردیم. برای ایجاد یک کامپونت Blazor که یک فرم ساده شامل فیلدهای نام، سن و یک دکمه ارسال است، از کد زیر استفاده می‌کنیم. این کد شامل تگ‌های HTML برای نمایش فیلدها و دستورات C# برای مدیریت ارسال فرم است.

```
@page "/simple-form"

<h3>فرم کاربری</h3>

<form @onsubmit="HandleSubmit">
    <div>
        <label for="firstName">نام:</label>
        <input type="text" id="firstName" @bind-value="Model.FirstName" />
    </div>
    <div>
        <label for="age">سن:</label>
        <input type="number" id="age" @bind-value="Model.Age" />
    </div>
    <button type="submit">ارسال</button>
</form>

@code {
    private UserModel Model = new UserModel();

    private void HandleSubmit()
    {
        // Save the data in the Model
        // This is where you can add your logic to handle the form
submission
        Console.WriteLine($"Name: {Model.FirstName}, Age: {Model.Age}");
    }
```

```
    public class UserModel
    {
        public string FirstName { get; set; }
        public int Age { get; set; }
    }
}
```

۱. نمایش یک رشته از مدل داده در یک برچسب (برای نام):

```
<label for="firstName">نام:</label>
<input type="text" id="firstName" @bind-value="Model.FirstName" />
```

۲. ویرایش یک عدد در مدل داده با استفاده از یک ورودی عددی (برای سن):

```
<label for="age">سن:</label>
<input type="number" id="age" @bind-value="Model.Age" />
```

۳. ارسال فرم و ذخیره داده‌ها در مدل داده (برای ارسال):

```
<form @submit="HandleSubmit">
  <input type="text" @bind-value="Model.Name" />
  <button type="submit">ارسال</button>
</form>

@code {
  private void HandleSubmit()
  {
    // Save the data in the Model
  }
}
```

چالش های Data Binding در Blazor

استفاده از Data Binding در Blazor برای برنامه‌های پیچیده، به خصوص زمانی که با چندین منبع داده و کامپوننت‌های تودرتو سروکار دارید، ممکن است چالش‌برانگیز شود. طوری که ممکن است حتی ابزارهای عیب‌یابی Blazor هنگام بروز خطا نیز به کارتان نیاید. علاوه‌بر بروز تداخل هنگام استفاده از چندین کامپوننت، استفاده بیش از حد از Data Binding ، خوانایی کدها را نیز سخت‌تر می‌کند. استفاده از روش‌های مناسب برای Data Binding و سازماندهی کد، به بهبود خوانایی و نگهداری آن کمک می‌کند.

مثال های پیشرفته از Data Binding در Blazor

در نوع One way در Data Binding درBlazor ، اطلاعات از کد C# شما به رابط کاربری منتقل می‌شوند و کاربر فقط می‌تواند آن‌ها را ببیند، نه اینکه تغییری ایجاد کند.

فرض کنید می‌خواهیم عنوانی را با تگ <h1> در صفحه نمایش دهیم. برای این کار می‌توانیم از خاصیتی به نام Title در بخش کد @code استفاده کنیم. سپس با استفاده از علامت @ آن را به تگ <h1> در بخش نمایش @ وصل کنیم. به‌این‌صورت:

```
<h1>@Title</h1>

@code {
  public string Title { get; set; }
}
```

با تغییر مقدار خاصیت Title در بخش کد، مقدار عنوان نمایش‌داده‌شده در صفحه هم به‌طور خودکار تغییر می‌کند. این سادگی Binding یک‌طرفه در Blazor را نشان می‌دهد.

از همین تکنیک می‌توانیم برای ارسال مقادیر به کامپوننت‌های دیگر نیز استفاده کنیم. برای مثال، فرض کنید یک کامپوننت به نام MyComponent داریم که یک پارامتر به نام Title دارد. می‌توانیم در کامپوننت اصلی، مقدار خاصیت Name را به‌عنوان آرگومان به پارامتر Title در کامپوننت Child ارسال کنیم:

```
@page "/MyPage"
<MyComponent Title="@Name" />

@code {
  public string Name { get; set; }
}
```

با این کار، مقدار خاصیت Name از کامپوننت اصلی به عنوان Title در کامپوننت Child نمایش داده می‌شود.

مثالی از روش Two-Way Data Binding

در این روش از Data Binding در Blazor، ماجرا هیجان‌انگیز می‌شود. در حالت Two-Way Data Binding می‌توانیم مقداری در مدل داده خودمان تعریف کنیم و آن را به یک فیلد ورودی در رابط کاربری متصل کنیم. هر تغییری در این فیلد ورودی، مقدار را در مدل داده نیز به‌روزرسانی می‌کند و بالعکس.

به بیان ساده‌تر، کاربر می‌تواند با رابط کاربری (مثلاً با واردکردن اطلاعات در یک فرم) داده‌ها را تغییر دهد و این تغییرات بلافاصله در کد C# شما هم اعمال می‌شوند. مثل یک خیابان دوطرفه که برای ردوبدل کردن اطلاعات بین کاربر و برنامه، شکل گرفته است. کد HTML زیر را ببینید:

```
<input @bind="Username" />

@code {
    public string Username { get; set; }
}
```

در این مثال، ما یک فیلد ورودی داریم که خاصیت Username را به خاصیت value این عنصر ورودی متصل کردیم. عبارت @bind یک میانبر برای اتصال است. هر کامپوننت (component) در Blazor به‌صورت پیش‌فرض یک خاصیت قابل اتصال دارد. ما می‌توانیم از روش طولانی‌تر و با ذکر صریح خاصیت value نیز استفاده کنیم.

```
<input @bind-value="Username" />

@code {
    public string Username { get; set; }
}
```

در اینجا، از bind-value برای اتصال صریح خاصیت Username به خاصیت value فیلد ورودی استفاده کردیم.

همچنین در مثال زیر، از کامپوننت ازپیش‌تعریف‌شده InputText استفاده می‌کنیم:

```
<InputText @bind-Value="Username" />
```

توجه داشته باشید حروف بزرگ و کوچک در سینتکس Data Binding در Blazor اهمیت دارد. برای مثال، در کامپوننت ازپیش‌تعریف‌شده InputText باید از Value با V بزرگ استفاده کنیم. به‌عنوان یک قاعده کلی، اتصال به تگ‌های استاندارد HTML معمولاً با حروف کوچک و اتصال به کامپوننت‌های Blazor با حروف بزرگ نوشته می‌شود.

مثالی از روش Event Binding

علاوه‌بر اتصال یک‌طرفه و دوطرفه Data Binding در Blazor ، می‌توانید یک متد C# را به یک رویداد متصل کنید.

کد C# زیر نمونه‌ای از یک کامپوننت Blazor پیچیده‌تر را نشان می‌دهد که شامل یک فرم HTML با یک فیلد متنی برای ورود نام کاربری است.

```
<h3>User</h3>

<EditForm Model="@Model" OnSubmit="@Save">
  <InputText @bind-Value="Model.Username" />
  <input type="submit" value="Submit" />
</EditForm>


@code {
  public User Model { get; set; } = new User();

  public void Save()
  {
    // Save the User model
  }

  public class User
  {
    public string Username { get; set; }
  }
}
```

در این مثال، خاصیت Model را به پارامتر Model کامپوننت داخلی EditForm و خاصیت Value کامپوننت InputText را به Property Username کلاس User وصل می‌کنیم. در بخش کد کامپوننت، متدی به نام Save داریم. آن را به رویداد OnSubmit کامپوننت EditForm متصل می‌کنیم. هر زمان که کاربر با استفاده از دکمه ارسال، فرم را ارسال کند، رویداد OnSubmit فعال شده و متد Save فراخوانی می‌شود.

درون متد Save می‌توانیم با استفاده از خاصیت Model به نام کاربری دسترسی پیدا کنیم و آن را در پایگاه داده ذخیره کنیم.

کد زیر، نمونه‌ای از اتصال رویداد برای یک عنصر HTML را نشان می‌دهد:

```
<input type="button" @onclick="Save" />
```

در این مثال، ما از حروف کوچک برای اتصال متد Save به رویداد onclick عنصر دکمه استفاده می‌کنیم. همانطور که می‌بینید، نحوه اتصال Data Binding با استفاده از Component Parameter Blazor و عناصر HTML کمی متفاوت است.

جمع بندی: به Data Binding درBlazor

Data Binding در Blazor موتور پویایی برنامه‌های شما است و این امکان را می‌دهد تا به‌سادگی داده‌ها را به رابط کاربری متصل کنید. Data Binding کلید اتصال دنیای داده‌ها به رابط کاربری شماست. با استفاده از این ویژگی در Blazor می‌توانید به‌راحتی داده‌ها را از مدل‌های داده خود به عناصر UI منتقل کرده و تغییرات کاربر در رابط کاربری را به مدل‌های داده بازتاب دهید. این امر، پویایی و تعامل را به اپلیکیشن‌های Blazor شما می‌بخشد و تجربه کاربری بی‌نظیری را برای کاربران رقم می‌زند.