Razor Directives in Blazor

Razor Directives in Blazor

Razor directives are instructions that control **how a Razor component is compiled and executed**.
They typically appear **at the top of a .razor file** and always begin with @.

Think of directives as **configuration and metadata for a component**, not UI logic.

---

1. @page — Routing Directive

Defines a component as a **routable page**.

@page "/counter"

@page "/counter/{id:int}"

Key points:

- Without @page, the file is a **component**, not a page
- Supports **route parameters** and **constraints**

@page "/product/{id:int}"

<h3>Product Id: @id</h3>

@code {

[Parameter] public int id { get; set; }

}

---

2. @using — Import Namespaces

Imports namespaces for the current component.

@using System.Text

@using MyApp.Services

Global usage (recommended):

// _Imports.razor

@using System.Net.Http

@using Microsoft.AspNetCore.Components

Best practice:

- Put common namespaces in _Imports.razor
- Use local @using only when necessary

---

3. @inject — Dependency Injection

Injects services from the **ASP.NET Core DI container**.

@inject HttpClient Http

@inject NavigationManager Nav

Equivalent C#:

[Inject] public HttpClient Http { get; set; }

Usage:

<button @onclick="Navigate">Go</button>

@code {

  void Navigate()

  {

    Nav.NavigateTo("/home");

  }

}

---

4. @code — Component Logic

Defines the **C# backing logic** of the component.

```
@code {
    int count = 0;

    void Increment()
    {
        count++;
    }
}
```

Notes:

- Compiled into a partial class
- Can be replaced by .razor.cs code-behind

---

5. @inherits — Inheritance

Allows a component to inherit from a base class.

```
@inherits MyBaseComponent
```

Example:

```
public class MyBaseComponent : ComponentBase
{
    protected string AppName => "Blazor App";
}
```

Usage:

```
<h3>@AppName</h3>
```

---

6. @implements — Implement Interfaces

Used when a component must implement an interface.

```
@implements IDisposable
```

Example:

@implements IDisposable

```
@code {
   public void Dispose()
   {
      Console.WriteLine("Component disposed");
   }
}
```

Common interfaces:

- IDisposable
- IAsyncDisposable

---

7. @layout — Page Layout

Specifies which layout a page uses.

@layout MainLayout

Default layout is set in:

```
// App.razor
<Router AppAssembly="@typeof(App).Assembly">
   <Found Context="routeData">
      <RouteView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" />
   </Found>
</Router>
```

---

8. @attribute — Add Attributes to Component

Adds attributes to the **generated component class**.

@attribute [Authorize]

Equivalent to:

[Authorize]

public partial class MyComponent : ComponentBase

Common usage:

@attribute [Authorize(Roles = "Admin")]

---

9. @typeparam — Generic Components

Defines a **generic type parameter**.

@typeparam TItem

Example:

@typeparam TItem

@foreach (var item in Items)

{

@item

}

@code {

[Parameter] public IEnumerable<TItem> Items { get; set; }

}

Usage:

<MyList TItem="string" Items="@names" />

---

10. @ref — Reference a Component or Element

Captures a reference to a component or HTML element.

HTML element

```
<input @ref="inputRef" />


@code {
   ElementReference inputRef;
}
```

Component reference

```
<MyChildComponent @ref="childRef" />


@code {
   MyChildComponent childRef;
}
```

---

11. @key — Preserve Element Identity

Improves rendering performance in loops.

```
@foreach (var item in Items)
{
   <li @key="item.Id">@item.Name</li>
}
```

Prevents:

- DOM reuse issues
- UI glitches during re-rendering

---

12. @namespace — Set Component Namespace

Defines the namespace for a Razor file.

@namespace MyApp.Components.Admin

Typically defined once in _Imports.razor.

---

13. @rendermode — Blazor Render Mode (.NET 8+)

Controls how the component is rendered.

@rendermode InteractiveServer

Other modes:

- InteractiveWebAssembly
- Static
- InteractiveAuto

Example:

@page "/chat"

@rendermode InteractiveServer

---

14. _Imports.razor — Directive Aggregation

Common directives applied automatically.

@using Microsoft.AspNetCore.Components

@using Microsoft.AspNetCore.Components.Web

@using MyApp.Shared

Reduces duplication across components.

---

15. Summary Table

Directive	Purpose
@page	Routing
@using	Import namespaces
@inject	Dependency injection
@code	Component logic
@inherits	Base class
@implements	Interface
@layout	Page layout
@attribute	Class attributes
@typeparam	Generics
@ref	Component/element reference
@key	DOM stability
@namespace	Namespace
@rendermode	Rendering mode

Tip (Important)

- Directives **do not render UI**
- Order usually does **not matter**, but conventionally placed at top
- Prefer _Imports.razor for shared directives
- Use .razor.cs for large components
