Single Page Applications (SPA) and **Blazor as a SPA framework** in the .NET ecosystem.

---

1. Overview of Single Page Applications (SPA)

What is a SPA?

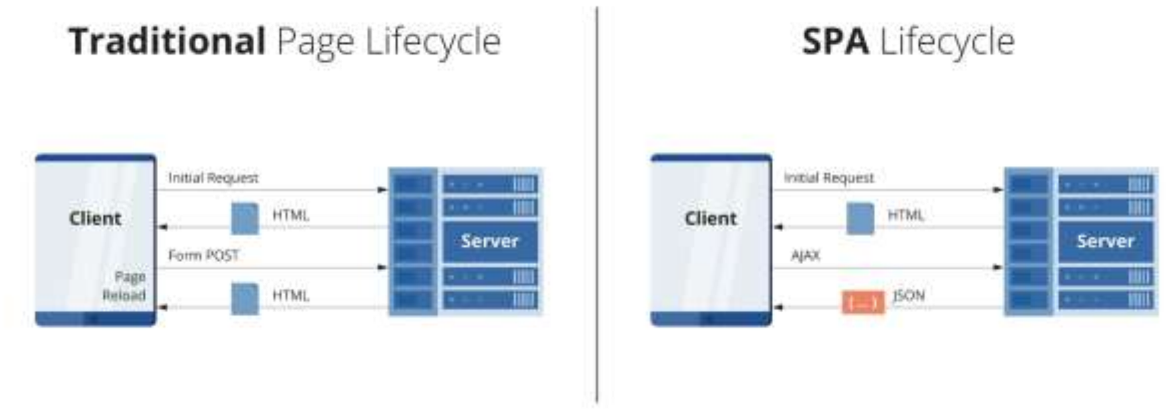
A **Single Page Application (SPA)** is a web application that **loads a single HTML page once** and then **dynamically updates the content** as the user interacts with the app—**without full page reloads**.

Instead of navigating between multiple HTML pages, the browser:

- Loads the app shell (HTML + CSS + JavaScript)
- Fetches data asynchronously (usually via APIs)
- Updates the UI dynamically using client-side logic

Traditional Web App vs SPA

Aspect	**Traditional Web App**	**SPA**
Page Load	Full page reload per request	Single initial load
UI Updates	Server-rendered HTML	Client-side rendering
Navigation	Browser refresh	Client-side routing
Performance	Slower after interactions	Faster, smoother UX
Backend	MVC / Razor Pages	API-based (REST/GraphQL)



Core Characteristics of SPA

1. **Client-Side Routing**
    - URL changes without reloading the page
    - Navigation handled inside the browser
2. **Asynchronous Communication**
    - Uses AJAX / Fetch / HTTP APIs
    - Data exchanged as JSON
3. **Stateful UI**
    - UI state managed in the client
    - Components re-render when state changes
4. **Rich User Experience**
    - Fast interactions
    - Desktop-like feel

Popular SPA Frameworks

- Angular
- React
- Vue
- **Blazor (C#-based SPA framework)**

---

2. Blazor as a Single Page Application Framework

What is Blazor?

Blazor is a **modern SPA framework from Microsoft** that allows developers to build **interactive web UIs using C# and .NET instead of JavaScript**.

Blazor enables SPA development by:

- Using **components**
- Supporting **client-side routing**
- Managing **UI state**
- Communicating with backend APIs asynchronously

Why Blazor is Considered a SPA

Blazor satisfies all SPA principles:

SPA Principle	**How Blazor Implements It**
Single Page Load	Loads once, updates dynamically
Client-Side Routing	@page directive + Router
Dynamic UI Updates	Component re-rendering
API Communication	HttpClient
State Management	Scoped/Singleton services

---

3. Blazor SPA Architecture

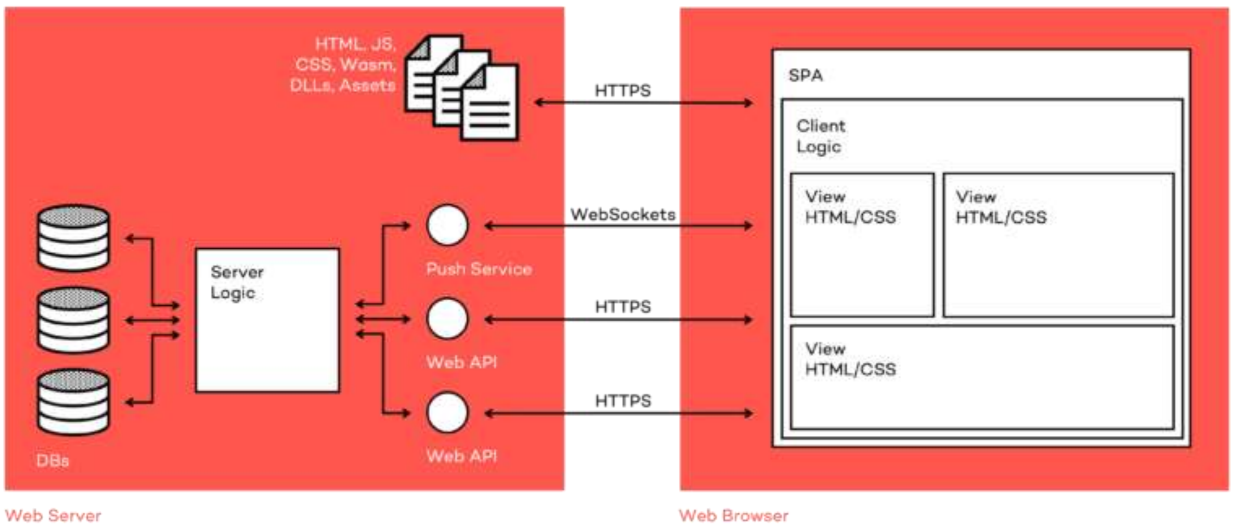
High-Level Architecture

```
Browser
 ├── Blazor App (Components)
 │    ├── Razor Components (.razor)
 │    ├── Routing & Navigation
 │    ├── State Management
 │
 ├── API Calls (HTTP / gRPC)
 │
Backend (ASP.NET Core)
 ├── Web API / Minimal API
 ├── Authentication / Authorization
 └── Database
```

---

4. Blazor Hosting Models and SPA Behavior

1. Blazor WebAssembly (Client-Side SPA)



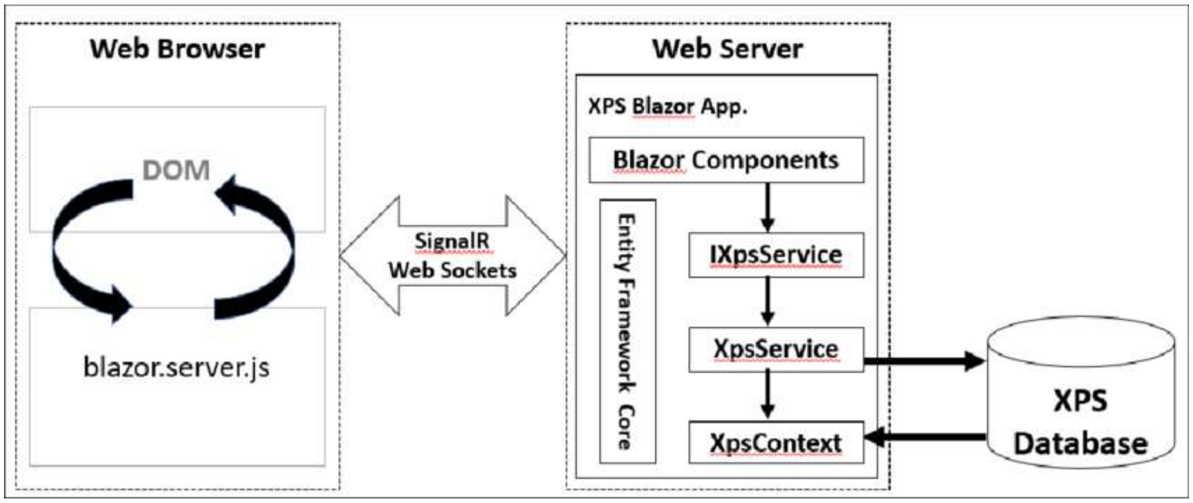
- Runs **entirely in the browser**
- .NET runtime runs on **WebAssembly**
- Communicates with backend APIs
- Closest to React/Angular SPA model

Key SPA Traits

- Offline capability
- Reduced server load
- Rich client-side interactivity

---

2. Blazor Server (Server-Driven SPA)

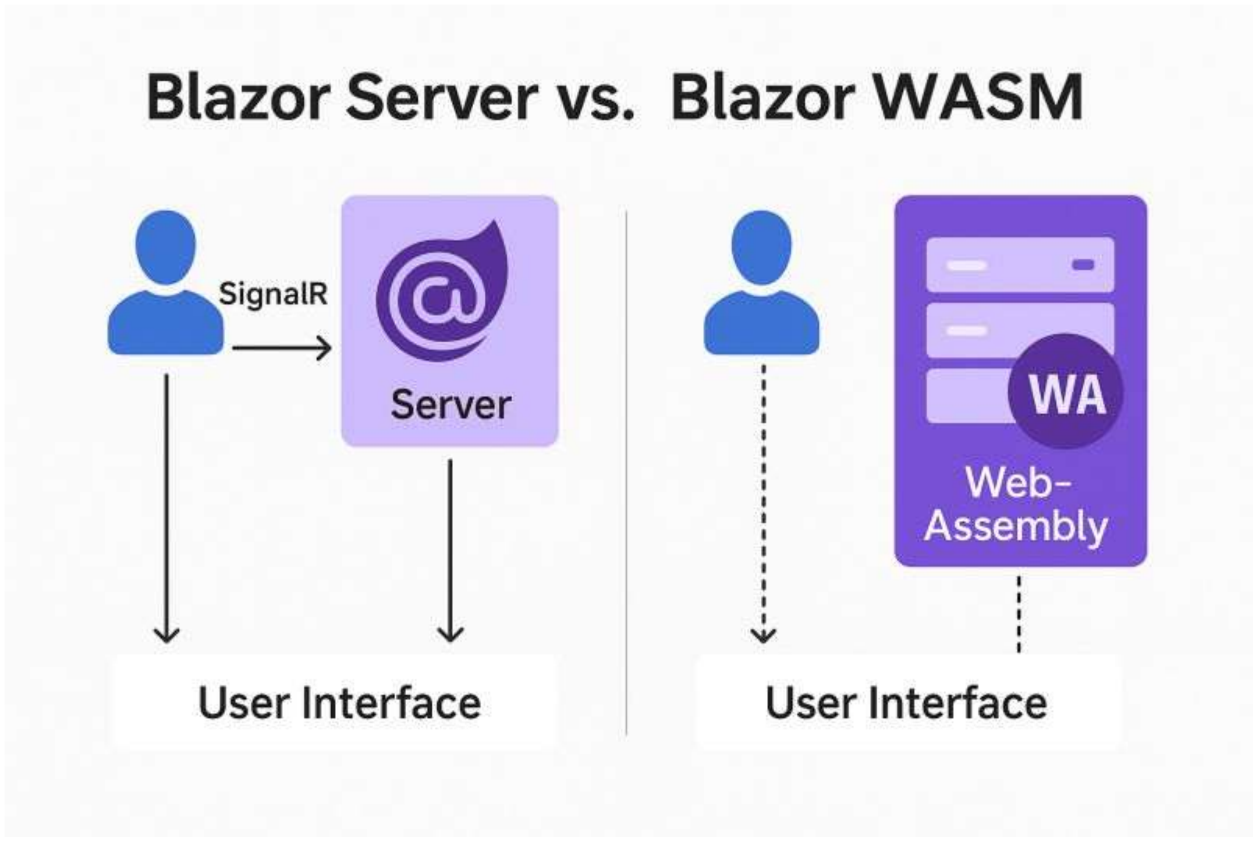


- UI logic runs on the **server**
- Browser maintains a **SignalR connection**
- DOM updates are streamed to the client

Still a SPA because

- No full page reloads
- Client-side routing

- Stateful UI over a persistent connection



---

5. Blazor Component Model (SPA Core Concept)

Components are the Building Blocks

```
@page "/counter"


<h3>Counter</h3>


<p>Current count: @count</p>


<button @onclick="Increment">Click me</button>


@code {
    int count = 0;


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

SPA Behavior

- UI updates instantly
- No page refresh
- State-driven rendering

---

6. Advantages of Blazor as a SPA Framework

Technical Benefits

- Single language: **C# for frontend + backend**
- Strong typing & compile-time checks
- Shared models between UI and API
- First-class .NET tooling

Enterprise Benefits

- Ideal for **line-of-business applications**
- Long-term maintainability
- Easier onboarding for .NET teams

---

7. When to Use Blazor as a SPA

Best Fit Scenarios

- Enterprise dashboards

- Admin portals
- Internal business applications
- Teams with strong .NET background

---

8. Summary

SPA

- A web app model with single-page loading and dynamic updates

Blazor as a SPA

- Fully compliant SPA framework
- Uses C# and .NET instead of JavaScript
- Supports both client-side and server-driven SPA models