Blazor – A Dream Come True for C# Developers

Tismo Technology Solutions (P) Ltd
Bangalore, India

JavaScript is the spine of any modern web app. Over the last few years, it has become one of the most widely used programming languages, helping front end developers build beautiful user interfaces on mobile and web. But this left .NET developers at a disadvantage. Now imagine this - **.**NET running on the browser. No add-ons, no plugins, no transpilation wizardry. Not possible you say. Well, say hello to Blazor from Microsoft!

What is Blazor?

Blazor is a .NET web framework that runs on the browser. Think React or Angular, but propelled by C# and Razor, a mark-up syntax that lets you embed server-based code into web pages. Blazor executes .NET assemblies using the Mono .NET runtime, a free and open-source project, implemented via WebAssembly. It uses the latest web standards and does not require any additional plugins or add-ons to run**.** It allows full stack web development with the stability, consistency and productivity of .NET.

Blazor offers all the benefits of the rich, modern single-page application (SPA) platform. Additionally, it allows developers to write the code for the client and server in the same technology (.NET). Also, the same classes can be shared by both client and server code.

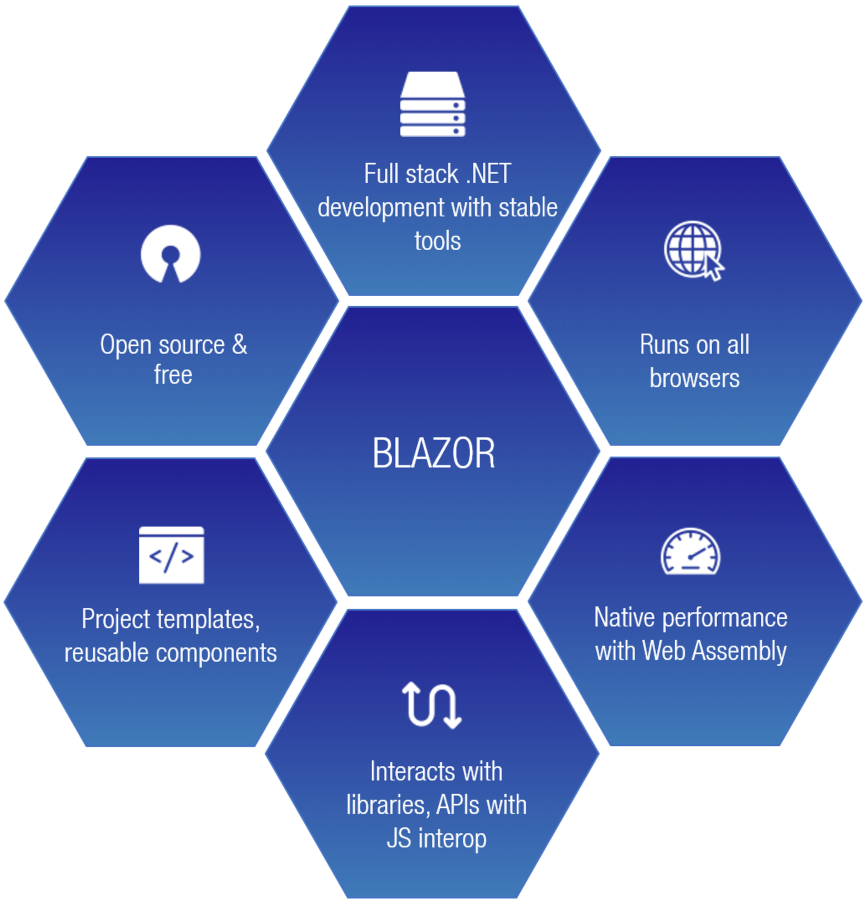

Full stack .NET development with stable tools
Open source & free
Runs on all browsers
BLAZOR
Project templates, reusable components
Native performance with Web Assembly
Interacts with libraries, APIs with JS interop

Figure 1: Highlights of Blazor

WebAssembly - The Secret Sauce that makes Blazor Possible

WebAssembly is a web standard similar to lower level assembly language. It enables execution of code almost as fast as executing native machine code. High level languages can be converted to WebAssembly and run on the browser at native speeds. This is a groundbreaking advancement for web development since traditional server-based languages like C#, F# can now be run on the browser.

WebAssembly is supported by all the major browsers. Then what about legacy browsers like older versions of internet explorer, firefox or stock mobile browsers on old phones? No worries, WebAssembly has got you covered. It will seamlessly fall back to JavaScript on older browsers and run without a hitch. It is efficient and fast and provides a memory-safe, sandboxed execution environment that enforces the security policies of the browser when it is embedded in the web. WebAssembly is open, can be easily debugged and is part of the open web platform.

Advantages

Using .NET in the browser offers many advantages and makes web application development easier and faster. Some of the main advantages of using .NET for full stack web application development are:

- Leverage existing C# skills for full stack development
- Better productivity due to code sharing between client and server applications
- Interoperability with Java Script
- Consistent programming framework across platforms and browsers

Types of Blazor Project Templates

- Blazor (Client Side)

What do you do when a new Web UI is required to be developed completely in C#? What if an existing Web UI, developed using Java Script, needs to be converted to C# based Web UI for ease of maintenance? Instead of the traditional route using HTML, CSS, and JavaScript, Blazor offers a well-rounded alternative by supplementing JavaScript with .NET and C# / Razor via WebAssembly. Processing and rendering of the Web UI happens completely in the browser as static pages. This type of project can be hosted on any type of web server that can serve static pages

- Blazor (Full Stack - Client Side and Server Side)

Full stack developers have forever dreamt of a way to develop the complete web application (both client side and server side) in C#. With Blazor, their dreams are finally a reality. Templates for this type of project include Blazor.Client (Client side of the web application), Blazor.Server (ASP.NET Core server application) and Blazor.Shared (Common application logic between client and server side). The server application (Blazor.Server) is responsible for serving the requests and providing Web API endpoints for the client side of the application (Blazor.Client)

- Blazor (Server Side)

With Blazor, C# can now be used when real time applications such as data streaming, need to be developed and network traffic needs to be utilized very efficiently. The templates for this type of project include Blazor.App (Server side app) and Blazor.Server (ASP.NET Core server application) that hosts the server side app. In this type of project, the entire web application runs on the server. This is a completely different approach to how user interface (UI) is delivered and how it interacts with the browser. WebAssembly is not used with this and instead, the browser is used as a thin client by deploying a SignalR JavaScript application to the client. The SignalR hub on the server side communicates with the client using web sockets.

Tismo Rolls Up its sleeve to take Blazor for a spin

Blazor is the new kid on the block and true to the company culture, Tismo got down and dirty to understand more about the implications of Blazor. Let's catch up with Rahul, the full stack wizard of Tismo Technology Solutions, who led the effort. "The common practice among Web developers coming from the .NET stack is to use C# for back-end programming and to write the services, and then to use an excellent Client-side framework like Angular or React to develop the UI. Being a C# guy, I find it difficult to switch between C# and JavaScript. In this scenario, Blazor is going to be a game changer" he says.

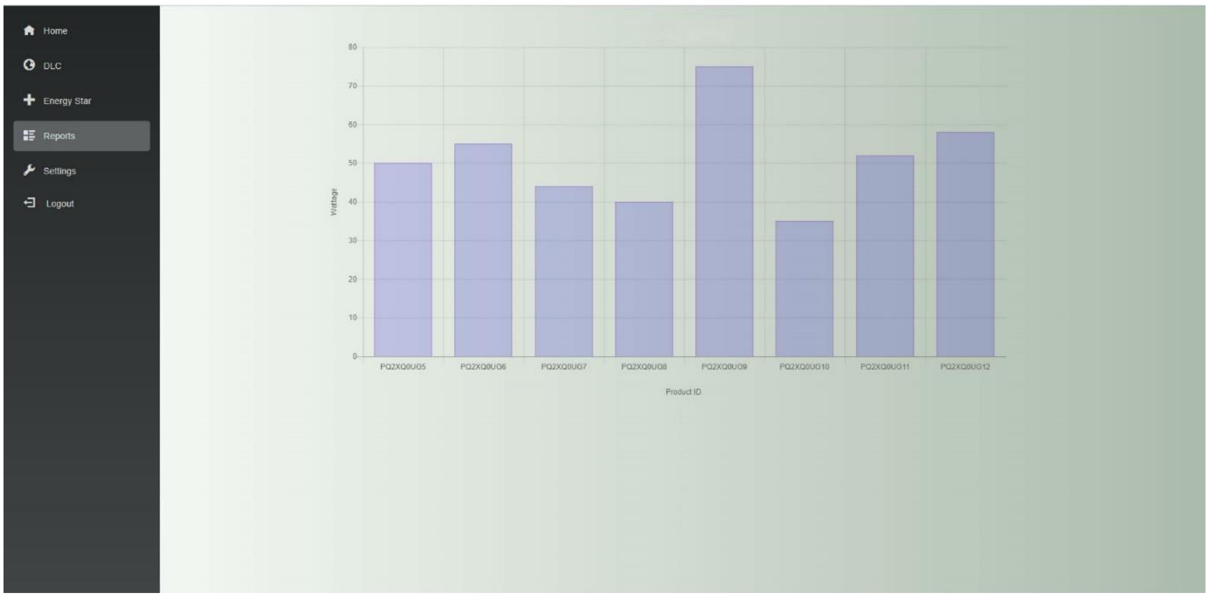
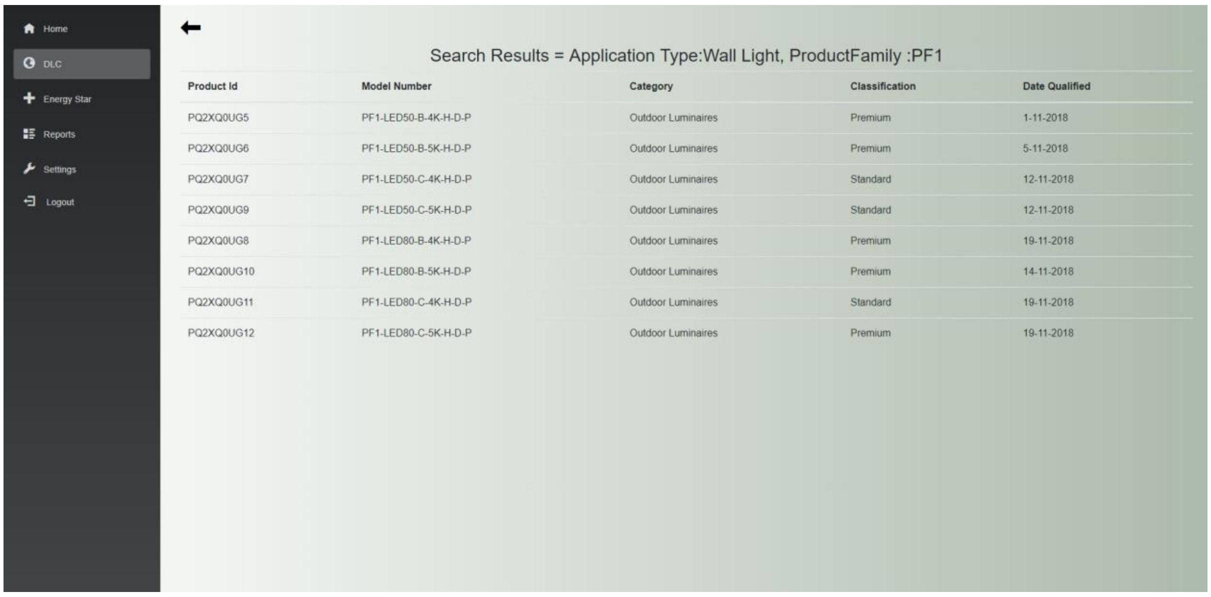


Figure 2: DLC certification app developed on Blazor - Report screen



Search Results = Application Type:Wall Light, ProductFamily :PF1

Product Id	Model Number	Category	Classification	Date Qualified
PQ2XQ0UG5	PF1-LED50-B-4K-H-D-P	Outdoor Luminaires	Premium	1-11-2018
PQ2XQ0UG6	PF1-LED50-B-5K-H-D-P	Outdoor Luminaires	Premium	5-11-2018
PQ2XQ0UG7	PF1-LED50-C-4K-H-D-P	Outdoor Luminaires	Standard	12-11-2018
PQ2XQ0UG9	PF1-LED50-C-5K-H-D-P	Outdoor Luminaires	Standard	12-11-2018
PQ2XQ0UG8	PF1-LED80-B-4K-H-D-P	Outdoor Luminaires	Premium	19-11-2018
PQ2XQ0UG10	PF1-LED80-B-5K-H-D-P	Outdoor Luminaires	Premium	14-11-2018
PQ2XQ0UG11	PF1-LED80-C-4K-H-D-P	Outdoor Luminaires	Standard	19-11-2018
PQ2XQ0UG12	PF1-LED80-C-5K-H-D-P	Outdoor Luminaires	Premium	19-11-2018

Figure 3: DLC certification app developed on Blazor - Search result screen

Blazor allows .NET Developers to use existing skills for Client-side development. They can write fast, Single Page Applications, built on a Component-based Model based on open web standards. “We built the client and server side of a DLC certification application, used to certify outdoor illuminating lamps. Since we're already using ASP.NET Core with MVC, setting up a Blazor project was very simple and easy to develop. It's very interesting to build applications that run directly on the browser with C#. With Blazor we can get full control over which part of our program runs on the server and which part runs on the client. There can also be shared code for both parts! Blazor enables code sharing by allowing the same class or model to be reused in the Server-side as well as Client-side. For example, I was able to create a DLL defining the class and then use the class in Client-side and Server-side by referencing the shared DLL. Handling call back in JavaScript is a headache. Right now, the only obstacle for developers in adopting Blazor is debugging.” says an elated Rahul.

Features

Blazor draws inspiration from modern, single page app frameworks, like React, Angular, and Vue, but is a new framework in its own right. It is fast, reusable and is open source, paving the way for great support from the community. Blazor supports features such as:

- Server-side rendering
- A component model for building composable UI
- Routing
- JavaScript interop
- Forms and validation
- Dependency injection
- Layouts
- Publishing and app size trimming
- Rich IntelliSense and tooling
- Live reloading in the browser during development
- Ability to run on older browsers via asm.js, a subset of JavaScript designed to allow software written in languages such as C to be run as web applications

What’s Next for Blazor?

“Blazor is an experimental project and there are still many questions to answer about its viability. The interest from the developer community is growing but Microsoft has not officially released it as a product. As a result, there is no support for it from Microsoft and everything is changing at a quick pace. As of now, there are a few kinks in the armour, but it is indeed the start of something special” says Rahul. The groundwork is already well underway to make it a fully robust web framework which supports all the features one would expect. It was a pipedream to use C# on the Client-side. But it looks like Blazor is going to make that dream come true for C# developers.