Welcome!
Agenda

• I will share everything I know about Blazor
• Prepare to drink from the firehose
• Hands-On Lab and this slide show available at http://blazordeskshow.com
What is Blazor?

• Blazor is really a component model
• Two hosting models:
  • Blazor Client - all code runs in the browser (WebAssembly)
  • Blazor Server - all code runs on the server (Magic?)
• The same markup and code will run in either model
• You can not combine models, nor would you want to
• Blazor IS NOT Silverlight!

http://www.appvnext.com/blog/2020/2/18/is-blazor-for-everyone
Installing Blazor WebAssembly

- Install latest .NET Core SDK 3.1
- Install latest Visual Studio 2019 Preview (Community works just fine)
- Install Blazor WebAssembly Template
- Detailed Instructions
  - [https://devblogs.microsoft.com/aspnet/blazor-webassembly-3-2-0-preview-3-release-now-available/](https://devblogs.microsoft.com/aspnet/blazor-webassembly-3-2-0-preview-3-release-now-available/)
Blazor Server Hosting Model

- Persistent WebSocket for ALL client/server communication
  - No HTTP Requests and Responses
- UI Interaction -> Message to Server -> Server Executes Code -> Applies to Server Graph DOM -> Calculates Diff -> Message to Client -> Client updates DOM.
Blazor Server Hosting Model

- ASP.NET Core
- Razor Components
- .NET
- SignalR
- DOM
Blazor Server Hosting Model

• Let’s watch traffic with Browser Tools - Network Tab
  • To see SignalR Traffic, expand _blazor?Id=xxxxx

• Very little initial load sent via HTTP

• Subsequent actions result in small amounts of data passed over SignalR

• Advanced Logging of SignalR traffic:
  • http://www.appvnext.com/blog/2019/12/21/logging-signalr-traffic-in-server-side-blazor
Blazor Server - Scaling

• **Memory Pressure**
  • 85K minimum server memory required for every connection
  • Don’t keep millions of rows of data in memory for every user

• **Web Sockets**
  • SignalR has a limit
  • Scale with Azure SignalR Service
  • [https://github.com/aspnet/AzureSignalR-samples/tree/master/samples/ServerSideBlazor](https://github.com/aspnet/AzureSignalR-samples/tree/master/samples/ServerSideBlazor)
Blazor Server at scale

- We test for appropriate latency (<200ms) under load with active concurrent clients (1 click/sec)

<table>
<thead>
<tr>
<th>Instance size</th>
<th>Concurrent active users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard_D1_v2 (1 vCPU, 3.5 GB)</td>
<td>5,000+</td>
</tr>
<tr>
<td>Standard_D3_v2 (4 vCPU, 14 GB)</td>
<td>20,000+</td>
</tr>
</tbody>
</table>

- Available memory is the primary bottleneck
- Real app behavior will depend on app memory usage, client behavior, and network conditions
Blazor Server - Benefits

• Download size is significantly smaller than a Blazor WebAssembly app, and the app loads much faster.

• The app takes full advantage of server capabilities, including use of any .NET Core compatible APIs.

• Thin clients are supported. For example, Blazor Server apps work with browsers that don't support WebAssembly and on resource-constrained devices.

• The app's .NET/C# code base, including the app's component code, isn't served to clients.
Blazor Server - Downsides

• Higher latency usually exists. Every user interaction involves a network hop.
• There's no offline support. If the client connection fails, the app stops working.
• Scalability is challenging for apps with many users. The server must manage multiple client connections and handle client state.
• An ASP.NET Core server is required to serve the app. Serverless deployment scenarios aren't possible (for example, serving the app from a CDN).
Blazor Client Hosting Model

- Razor Components
- .NET
- WebAssembly

DOM
Blazor Client Hosting Model

- Let’s watch traffic with Browser Tools - Network Tab
- Initial download:
  - 6 MB without linker (debug mode)
  - 1.9 MB with linker (release mode)
  - Mono .NET Runtime CLR compiled for WebAssembly (dotnet.wasm)
  - Supporting assemblies
- UI interactions require no HTTP requests unless they call out to APIs
Blazor Client - Benefits

- Code executes more quickly in the browser. Lower latency
- Offline support. PWA + Wasm is a beautiful thing
- Can be distributed via CDN
- Any .NET Standard 2.0 C# will run in the browser!
Blazor Client - Downsides

• Blazor Client can't directly access your database
  • Requires API access
• Debugging in Visual Studio not yet baked
• Still in preview until May. Shipping with .NET 5 in November
• Your code is shipped to the client. Sensitive code should run on server, just as you would with JavaScript
• In the same boat as JavaScript - Sandboxed.
Binding

- **One-Way**: `<span>@MyVariable</span>`
- **Two-Way**: Bind input elements to properties:
  - `@bind="MyVariable"`
  - `@bind-[PropertyName]="MyVariable"`
  - `@bind-[PropertyName]:event="oninput"` (onchange is default)

- **Two-Way to Component**:
  - Expose EventCallback Parameter in child component
  - Bind property in parent component
  - Bind property to child component event in parent component
  - Invoke event in child component when data changes
Handling State

• Pages reinitialize on navigation
  • Module-level variables are reset
• Move state into a scoped service (class) outside the page
• Inject service into pages to access state
• Optionally you can persist and recall state with your own code
• Blog Post: Shopping Cart stored in Local Storage:
Develop Blazor Client and Blazor Server apps simultaneously

- Trick I learned from Steve Sanderson at NDC London (Jan, 2020)
- Add Wasm project to Server solution
- Reference Wasm project in Server project
- Change App reference to client App reference
- Ensure both Wasm and Server have access to same resources
- Set Multiple Startup Projects (Wasm Server and Server)
- Debugging Wasm project in Visual Studio!
BandBooker

• App to allow bands to schedule gigs
• Models:
  • Instrument, Musician, Band, Venue, Booking
• We will limit our demo to Instruments and Musicians
• Purchase a 2.5-hour Blazor Workshop Video and Hands-On Lab ($99)
  • [http://blazor.appvnext.com](http://blazor.appvnext.com)
BandBooker - Techniques

• Code Behind
  • Use the power of Visual Studio Editor
• Validation
• JavaScript Interop
• Generic Components
BandBooker - Authentication/Authorization

• ASP.NET Core Identity
• Wasm Server uses API Controllers for Auth
• JWT Tokens
  • Local Storage Access Required
Deployment - Prerequisites

- Move databases to production
- Move credentials to config files
- Change URLs and connection strings accordingly
Deploy to Azure

- Create new Azure App Service
  - Enable WebSockets
- Benefits:
  - Azure SQL accepts connections by default
  - HTTPS provided by default
DevExpress Free Blazor Components

• Enterprise Developers want to get s#!t done ASAP!
• DevExpress Controls = maximum productivity
  • Data Grid
  • Data Pivot Grid
  • Scheduler
  • Charting
  • Primatives
Two ways to run DevExpress Blazor Demos

- Online
  - https://demos.devexpress.com/blazor/
- Locally
  - https://github.com/DevExpress/Blazor#how-to-run-this-demo-locally
To use DevExpress Tools

- Configure Private NuGet Feed
- Install NuGet Package(s)
- Modify _Hosts.cshtml (server) or wwwroot/index.html (client)
  - <link href="_content/DevExpress.Blazor/dx-blazor.css" rel="stylesheet" />
- Add to _Imports.razor
  - @using DevExpress.Blazor
DevExpress Blazor Resources

• Blazor Roadmap
  • Keep up with the latest developments

• Blazor Community Portal
  • Blog posts, new discoveries
  • https://community.devexpress.com/Blogs/

• Get In Touch!
  • Email: blazorroadshow@devexpress.com

Carl Franklin's
BLAZOR DESKSHOW
**Free Blazor Components**

Create High-Impact User Experiences with DevExpress UI for Blazor

Download Your Free Copy Today
devexpress.com/freeblazor

For a complete list of controls included in this offer, please visit devexpress.com/freeblazor. This free offer is subject to change and expires on June 1, 2020.
Free Blazor Components
Create High-Impact User Experiences with DevExpress UI for Blazor

THANK YOU!

Download Your Free Copy Today
devexpress.com/freeblazor

For a complete list of controls included in this offer, please visit devexpress.com/freeblazor. This free offer is subject to change and expires on June 1, 2020.