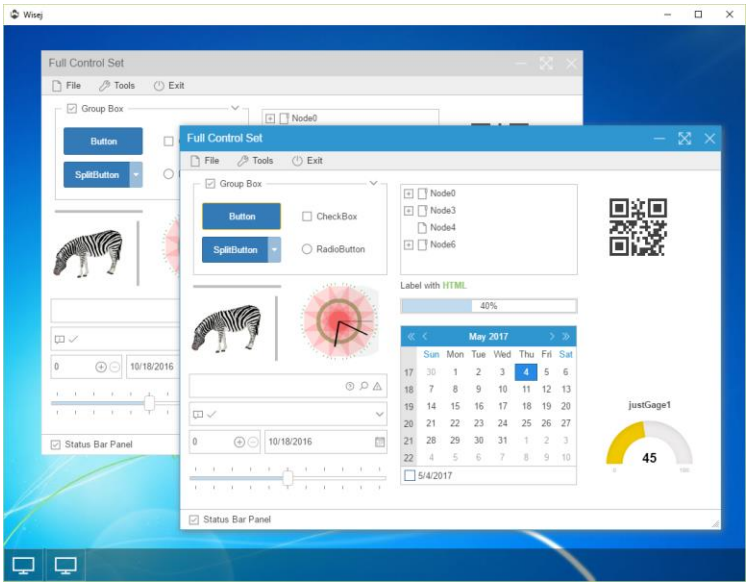
Wisej Standalone

Wisej Desktop Applications

1 Overview

Wisej allows you to run any Wisej Web Application as a traditional Windows Desktop Application without a web server or an internet connection. The application runs inside a window and it is self-hosted.



2 Browser Types

We provide 4 options:

- Wisej.Application.IE
- Wisej.Application.Chrome
- Wisej.Application.FireFox
- Wisej.Application.Edge

Wisej.Application.IE uses the Internet Explorer preinstalled with Windows configured to run at the highest possible version, usually IE10 or IE11.

The executable is only ~480K.

Wisej.Application.Chrome uses the latest build of Chromium (Chrome) and doesn't need the presence of Internet Explorer.

The executable, however, is ~133M and targeted for 64bit machines. It's relatively easy to change the Chromium libraries and retarget the executable for 32bit.

Wisej.Application.FireFox uses the latest build of GeckoFx and dosn´t need the presence of Internet Explorer. However the executable is ~102MB.

Wisej.Application.Edge uses the latest Edge browser or falls back to Internet Explorer if no Edge is installed. It´s executable is only ~658K

3 How to use it

Simply copy *Wisej.Application.exe* (either IE, Chrome, Firefox or Edge) to the root of your Wisej application, at the same level as *web.Config* and you are done. No installation or registration of any kind is needed.

Just launch the executable and the Wisej application will load automatically.

You can run multiple applications at the same time.

3.1 Command Line Arguments

Argument	Description
-p:{port} or -port:{port}	Changes the port. By default, it listens to the first available port. Use this argument if you want the standalone application to serve external users.
-d:{domain name} or -domain:{domain name}	Changes the domain that the server listens to. By default, the standalone process only accepts connections from localhost. Use this argument together with -port to server external users.
-fullscreen or -fullscreen:topmost	Starts the standalone application in full screen mode and optionally makes it the topmost window. To enable F11 and Escape to enter or leave full screen mode using the keyboard, set the MainView.AllowFullScreenMode property to true when compiling Wisej.Application.

4 CUSTOMIZATION

We provide the full source code for the *Wisej.Application.IE*, *Wisej.Application.Chrome, Wisej.Application.Edge* and *Wisej.Application.FireFox* projects. You can modify and customize the main window however you like.

4.1 POTENTIAL PERSONALIZATIONS

You can add native controls around the browser control to navigate to different parts of the Wisej application or launch different Wisej sub-applications.

The self-hosted URL is provided by *this.host.Url*. To navigate to a sub-application just add the path:

For IE:
```
this.browser.Navigate(this.host.Url + "/admin");
```

For Chrome:
```
this.browser.Load(this.host.Url + "/admin");
```

For FireFox:
```
this.browser.Navigate(this.host.Url + "/admin");
```

For Edge:
```
this.browser.Navigate(this.host.Url + "/admin");
```

You can also add arguments to the Url and process the values in the Wisej application.

And you can execute custom authentication code before launching the Wisej application.

4.2 SERVING MULTIPLE USERS

A Wisej Standalone application can also serve other users turning your machine into a web server. By default, the Wisej Host implementation is configured to accept connections only from localhost; change this line in *MainView.StartServer()* to serve external users.

```
// listens on for the localhost domain on the first available port.
this.host.Start("localhost", 0);
// listens to any domain and on port 80.
this.host.start("*", 80);
// listens to the myserver.com domain and on port 8080.
this.host.start("myserver.com", 8080);
```

4.3 Chrome Customization

You can customize the browser in the source code and recompile easily. We have included all the locales and the developer tools but have only included the en-US file and excluded the developer tools from the embedded resources.

You can choose your locale file and include it as an “Embedded Resource” and you can include the developer tools package as well and recompile.

5 How Does it Work

We didn’t recreate a new web server to replace IIS, nor we used Cassini.

Wisej Standalone and Wisej Self Hosting features use the Microsoft implementation of OWIN (Open Web Interface for .NET), called Katana: https://www.asp.net/aspnet/overview/owin-and-katana.

Wisej added a WisejMiddleware implementation to route requests to the Wisej Handler and you can add any OWIN middleware and recompile the project.

However, Wisej also added full support for ASP.NET/MVC applications, including all ASP.NET controls, WebForms, etc. Basically, our implementation can fully replace IIS in a **single executable**.

5.1 Single Executable

We like simplicity. The *Wisej.HostService* projects are configured to merge all the OWIN and Katana assemblies into a single executable using *ILMerge* and some code we added to let the host know where to load its classes.

For *Wisej.Application.Chrome* we used the *CefSharp* library but got rid of all the unnecessary complexities added to the project and deployment. We have embedded all the Chromium modules and resources as embedded resources and extract them into a temporary */CefSharp* directory at startup, making also the Chrome desktop application a single executable without requiring any installation.

6 Conclusion

Wisej Standalone is a powerful solution that open many new possibilities for web appliactions.

Last, you can use it to run pure ASP.NET/MVC (including WebForms) applications.