# **Wisej: The Easy Button for Enterprise Software**

From: <https://www.codeproject.com/Articles/5268171/Migrate-Windows-Desktop-Apps-to-the-Web-with-wisej>

Wisej is a powerful platform that abstracts away the nuances of client/server interactions while embracing all the web has to offer, including third-party HTML5-based solutions, it is the perfect tool for building enterprise web applications.

This article covers: Obtaining the software, and does a demo walkthrough that looks at instantiating controls, real-time updates, third-party integration, data-binding at scale, embedded Media, and custom painting. Then we briefly look at features and integrations like: Pixel-Perfect WYSIWIG Designer, Theme Builder Control Library Integrations, and Native Mobile Integration.

This Review is from our sponsors at CodeProject. These reviews are intended to provide you with information on products and services that we consider useful and of value to developers.

Wisej provides a platform and set of tools designed to ease the burden of developing, testing, deploying, and maintaining complex business software. It bridges the gap between traditional WinForms-style desktop applications and the web by empowering developers to leverage their existing .NET and C# skills.

Developers are able to use the tools they are familiar with, like a pixel-perfect design surface and drag-and-drop controls with the languages they are comfortable with (C# or VB.NET) to build real-time web applications without having to learn HTML, CSS, or JavaScript. They can stay in their favorite IDE like Visual Studio to take advantage of features like IntelliSense, integrated debugging, design surfaces and source control integration. Front-end developers can leverage their knowledge of web technologies to augment applications and seamlessly integrate third-party packages like HTML5 charting libraries and ASP.NET controls.

To bridge the gap between legacy desktop applications and modern web applications, Wisej provides several features that create a straightforward path to migrate existing applications or build green field projects without having to learn a new set of technologies. Support for touch and gestures is built into the software with emulation to support the desktop experience, localization is a first-class citizen, and the applications Wisej produces are out-of-the-box responsive. That means they are readily accessible from desktop browsers, mobile smartphones and tablets.

## Obtaining the Software

To get started, visit the Wisej site:

<https://wisej.com/>

There you can browse the various pricing options and download a trial.

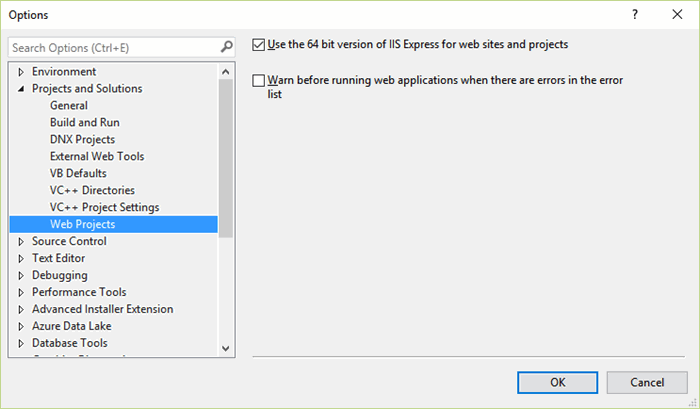
The sample application is available at:

<http://demo.wisej.com/codeproject>

You can download the source code at:

<https://wisej.com/examples>

Download the latest build of Wisej from [https://wisej.com](https://wisej.com/) to get started. You should be able to compile and debug the application. The application uses a 64-bit SQLite driver. There are two options to ensure that you are able to successfully debug. You can either run as Administrator and use the full IIS server locally, or you can set IIS Express to run in 64-bit mode as shown in the following dialog.

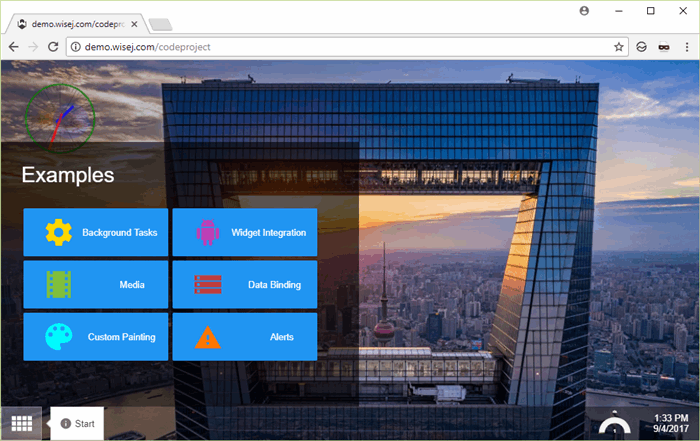


***Figure 1: 64-bit IIS Express***

One advantage of Wisej is that it can generate local executables to run the same application you access from your web browser as a standalone or self-hosted application. In the folder for the project, you will find a "Wisej.Application.exe" that you can execute to test this functionality.

## A Demo Walkthrough

The first thing you’ll notice when you access the site is the desktop experience. The code to create the desktop experience is as simple as adding a desktop control to the application. Wisej automatically handles allowing users to drag components, scale, resizing, minimizing, etc. Tap on the grid of squares in the lower left to expand a "start menu" of options.



***Figure 2: Start Menu***

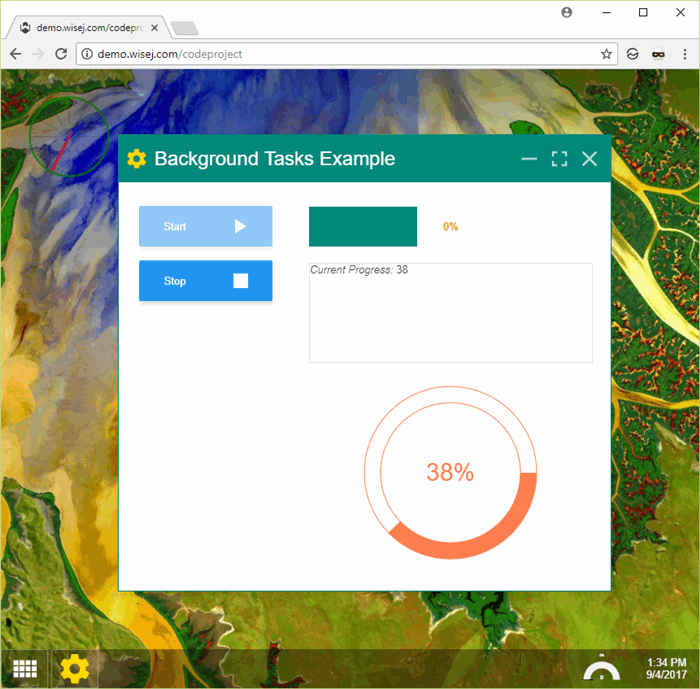
### Instantiating Controls

In Visual Studio, navigate to the "Popups" folder and double-click "StartPopup." This will open the designer experience for the pop-up menu. You can see the various buttons, click once to view properties and double-click on them to view the code-behind. Launching another component is simple. Double-click on the "Background Task" button in the designer. You will see the following code that simply creates an instance of the "BackgroundTask" control and then invokes it.



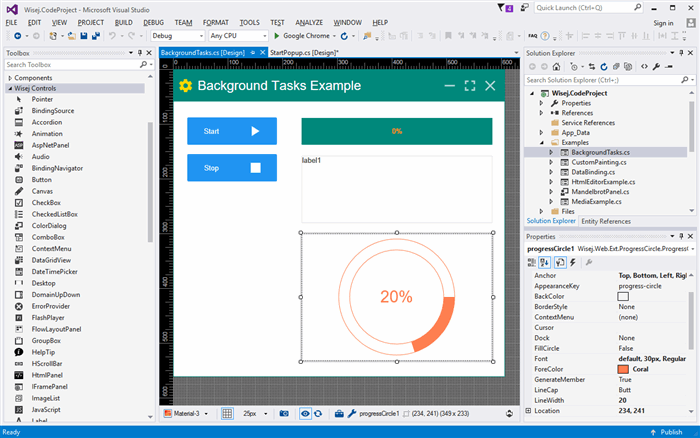
### Real-time Updates

The first example we’ll review is the "Background Tasks." Traditionally, building a real-time pipeline between a web browser client and server host involves scaffolding technologies like WebSockets or SignalR and building logic on both sides to handle dataflow and exceptions. Run the Background Tasks example press "Start" to see it running:



***Figure 3: Background Tasks***

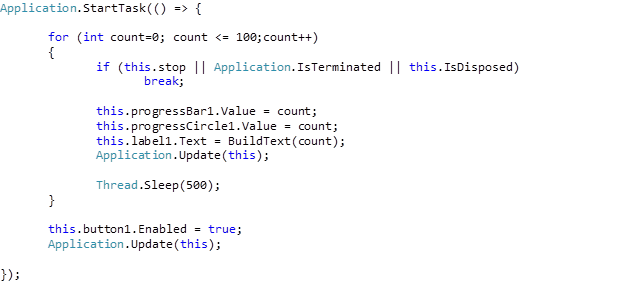
The same dialog is available in the design-time view through Visual Studio:



***Figure 4: Background Tasks in the Designer***

Note the list of controls on the left side that are ready for drag-and-drop positioning on the canvas with the same experience that WebForms developers are used to. The controls to maximize, tile, and close the dialog are all built-in and require no additional code to work. Right-click on the control to view the code-behind.

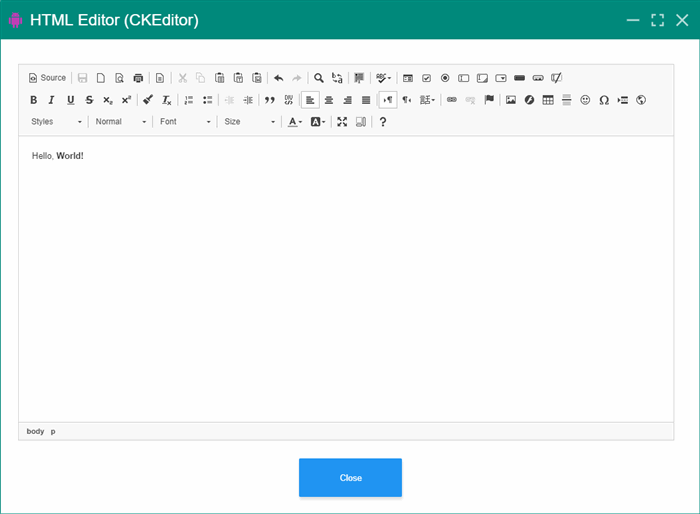
The important code is in the "button1\_click" event.



The code iterates through 100 items and pauses for one half second each loop. The browser app still runs asynchronously and is not blocked by the operation. The Wisej platform provides the "StartTask" method to initiate a background task, and refreshing data real-time is as simple as calling the "Application.Update" method! All of the real-time connections between the browser and server are handled automatically, so the developer can focus on what’s important (the core code logic) and not worry about the nuances of how the client and server connect. Wisej support for real-time communications handles Websockets, HTTP, or Websockets with HTTP fallback, all selectable via a simple configuration setting.

### Third-Party Integration

Next, click on "Widget Integration" in the demo app. You will be presented with an editing control:

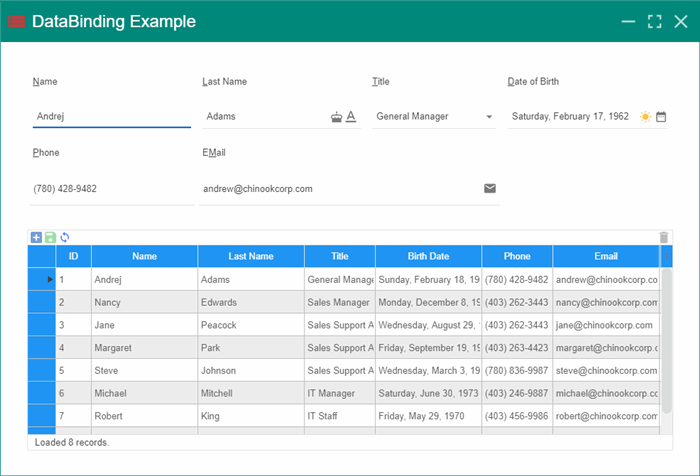


***Figure 5: Widgets Integration***

The control allows you to edit a document and use advanced features like formatting, inserting links and emojis, even embedding images. The control actually uses the open source HTML project [CKEditor](https://ckeditor.com/). This example illustrates how easy it is to integrate third-party open source libraries, even when they are not .NET-based projects.

### Data-binding at Scale

Grids are often the centerpiece of enterprise applications and building interactive grids that provide basic Create/Read/Update/Delete (CRUD) capability and advanced features like filtering, grouping, and sorting can be difficult in web-based applications. Wisej features a data-binding system for controls that is both intuitive and easy to implement. To see the results in action, choose the "Data-binding" example in the demo app. You should see a grid like this:



***Figure 6: Data-Binding Grid***

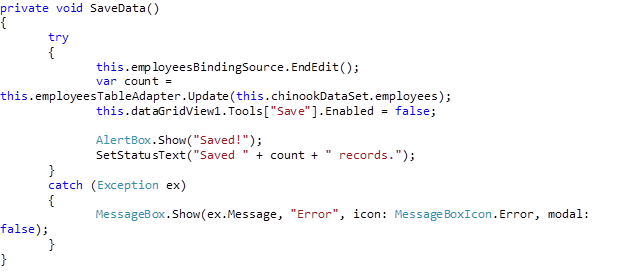
The grid is full-featured. Tap on a column header to sort, and tap again to toggle the order between ascending and descending. Tap on any row to edit in the associated dialog, or double-tap on the grid to edit inline ("Excel-style"). You can insert new rows and tweak your updates until they are ready to be saved to the server. The save features a dismissible alert notification that indicates success or a modal dialog to diagnose failure.

Open the grid in Visual Studio and right-click to view the code-behind. This is an incredibly powerful feature of Wisej that many web developers likely wish they had in the past. The grid is fully implemented through C# server-side code, and Wisej automatically handles all of the binding, updates, and dataflows without the developer having to write any client JavaScript code!

Loading the initial data set is as simple as filling a standard data adapter. The demo app uses a SQLite database to store and update sample data:

Image 9

The code to save a record provides a success alert and shows a modal error dialog when an issue is encountered, again all done with straightforward C# code that automatically controls the browser experience.

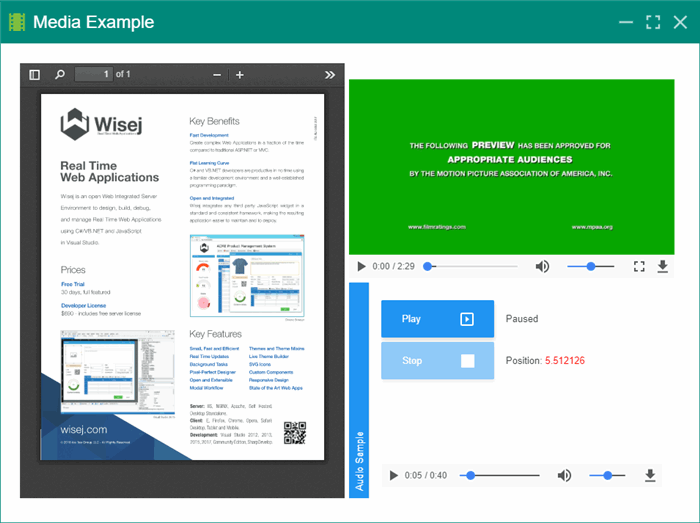


It is important to note how easy it is to provide client modals with server-side code. This is often a challenge in other frameworks due to the logic necessary to handle the asynchronous dialog and ensure that the modal successfully disables other content in the app. Notice the previous code example provides a straightforward, sequential, server-side C# workflow but successfully provides an interactive modal experience in the browser.

In more advanced scenarios, you may have to write code to page and skip records in the database and filter based on advanced criteria. Wisej simplifies this process by allowing you to focus on one set of code. If you can make the code work on the server, Wisej will connect it to the browser and display the appropriate output for you. No more writing two sets of code, one for the server and one to handle all of the events in the browser!

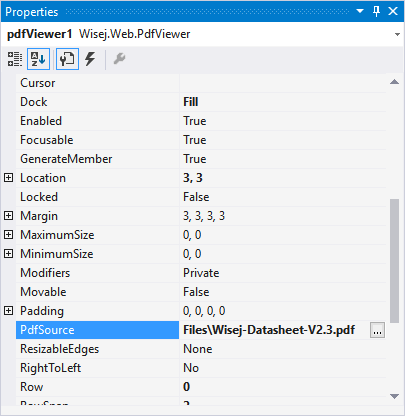
### Embedded Media

Click on the "Media" option from the start menu to view the media experience. The dialog box includes embedded video with position tracking and an embedded PDF document that you can view and navigate without leaving the current webpage.



***Figure 7: Embedded Media***

You can open the media control in Visual Studio and click on the individual components to see their properties. Wisej has built-in "Video" and "PDFViewer" controls that can simply be dragged onto the design surface to use and are bound to their related documents. The properties control shows how the PDFViewer control is bound to a document that is included with the project:



***Figure 8: PDF Viewer Properties***

The PDF viewer control is flexible and offers support for pdf.js, native PDF documents, Google Docs and custom viewers.

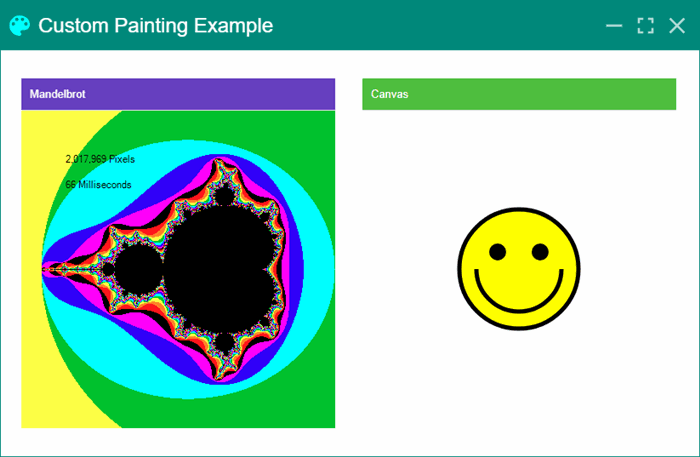
For the position tracking in the media element, a simple event is wired in to trigger when the time position changes and handled with the following code:



The code is handled just like a desktop app and all of the updates to the browser are coordinated automatically by Wisej.

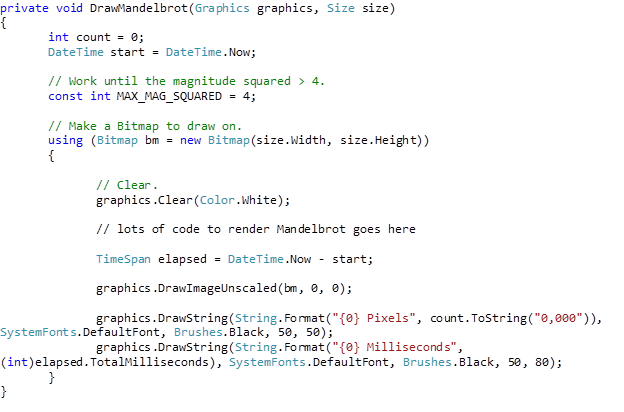
### Custom Painting

The final dialog to cover in this review is "Custom Painting." Tap on the option and you’ll see a modal dialog with a fully rendered Mandelbrot set and a smiling face.



***Figure 9: Custom Painting***

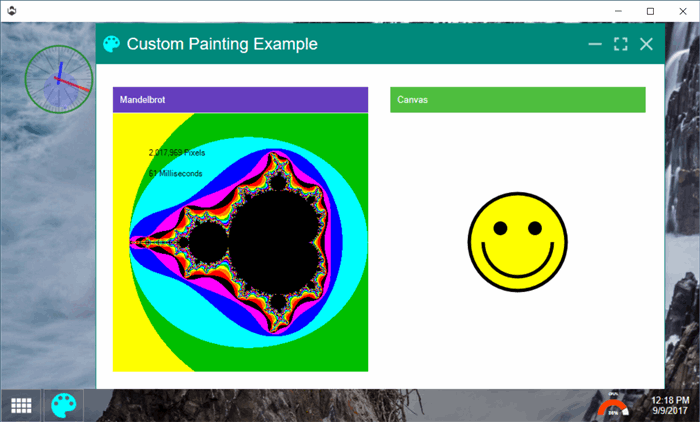
Despite the number of pixels and level of detail, you can see the Mandelbrot renders exceptionally fast – just over 50ms on my machine. If you look at the code behind, the image is rendered on the server using a bitmapped object. Once it is completed, a special "DrawImageUnscaled" call is made to transfer the bitmap contents onto the browser-based canvas (again, without the developer having to learn or create any client-side code).



The smiling face is a "canvas" control that enables you to write server-side code to render a client-side HTML5 canvas. This allows you to create complex custom web controls entirely in .NET that are fully HTML5 compliant.

## Standalone Executable

In addition to supporting web-based deployments, the Wisej platform allows you to package your application as a local executable that can be run from the desktop. It supports identical functionality to the web-based project as a standalone program. This enables distribution without a web server. The executable is generated in the root directory of the project, and this is an example of running it with the canvas widget.



***Figure 10: Standalone Executable***

## Features and Integrations

### Pixel-Perfect WYSIWIG Designer

Everything done with Wisej revolves around the framework’s patented pixel-perfect designer. The WYSIWIG designer allows you to see exactly what the application will look like at runtime. Drag and drop controls from the toolbox onto the page and see for yourself. The designer also allows developers to create responsive profiles: pages that will fit any device.

### Theme Builder

Wisej comes with a powerful theme builder that allows developers to modify the look and feel of any control within their application. Different themes and styles can be applied to debug applications running in production with ease. Each Wisej control also comes with a “CssStyle” attribute that applies the custom style during design-time.

### Control Library Integrations

In addition to the third-party widgets integrated in the sample, Wisej now features the Syncfusion, DevExpress, and Infragistics JavaScript component libraries available as drag-and-drop controls in the designer. The integration simplifies the migration process for developers concerned about losing their native desktop application’s charts, graphs, and other controls.

### Native Mobile Integration

The Wisej Mobile Integration package for Android and iOS offers web developers the power to integrate native device functionalities into their Wisej applications. Native scrolling and a custom theme will give the user a responsive experience. Use the device’s biometric authentication capabilities, native toolbars, tabbars, and more. Handle any native device event in Wisej using a custom handler.

### Benefits of Wisej

* Load Balancer support
* Support for native mobile integrations (iOS and Android)
* Support for progressive web applications (PWA)
* Robust architecture that is resistant to some of the top OWASP attacks

## Conclusion

For developers, the challenge of building a complex enterprise web application with features like large customizable grids and real-time data updates can be daunting. It may seem even more complex when the requirement is to migrate a legacy WinForms-style application to the web without losing the features business users have grown accustomed to, like being able to use their keyboard for data entry and editing values inside a live grid. Fortunately, Wisej is a tool that eases the burden of the challenge and provides a direct path from desktop applications to the web. As a powerful platform that abstracts away the nuances of client/server interactions while embracing all the web has to offer, including third-party HTML5-based solutions, it is the perfect tool for building enterprise web applications.

This review just scratched the surface of the capabilities Wisej provides. In addition to what was covered here, there are dozens more out of the box controls and features like comprehensive theming that can be applied while the application is running. It is possible to target both a web application and a desktop application from the same codebase. All of the functionality is available as a drag-and-drop, designer-based .NET application that leverages existing C# or VB.NET skills. Why not see for yourself how powerful the Wisej platform is? Head over to the website and start your free trial if you haven’t already to follow through this review.

*Note: This is an update version of Jeremy Likness’ original article on Wisej:*[*https://www.codeproject.com/Articles/1208339/Wisej-The-Easy-Button-for-Enterprise-Software*](https://www.codeproject.com/Articles/1208339/Wisej-The-Easy-Button-for-Enterprise-Software#_comments)