# Wisej Extensions

# Extending Wisej

## 1 Overview

Wisej was designed to be extensible and open to all sorts of technologies from the start. The extensions published on the web site are a fraction of what is possible and what will be available during the product's lifetime.

This short document outlines the different extension types, how to use the published projects, and how to build new extensions.

Note that all the extensions that are currently published are also already precompiled and included in the installer and are automatically added to Visual Studio's toolbox.

## 2 Extension Types

Extensions in Wisej follow a simple naming convention:

**Wisej.Ext.{Name}** for non-visual extensions that are not specific for Wisej.Web and can be used also with Wisej.Mobile. Examples are: **Wisej.Ext.Translation** and **Wisej.Ext.Geolocation**.

**Wisej.Web.Ext.{Name}** for extensions (visual and non) that are specific to Wisej.Web.

**Wisej.Mobile.Ext.{Name}** for extensions (visual and non) that are specific to Wisej.Mobile.

*Note that Wisej.Mobile is not a product yet and it's in pre-alpha stage.*

Wisej supports 4 kinds of extensions:

- Wisej Control
- Wisej Widget
- Wisej Component
- Wisej Extender Provider

### 2.1 Wisej Control

The native Wisej Control is a control that implements both sides of the technology: client and server. The client side is a specific JavaScript class (i.e. "wisej.web.TextBox" derived from one of the qooxdoo widgets) while the server side is the corresponding server component (i.e. Wisej.Web.TextBox derived from Wisej.Web.Control or a subclass).

A new Wisej Control must extend any of the Wisej.Web classes but it's not required to provide a new JavaScript class.

**Wisej.Web.Ext.ProgressCircle** is an example of a Wisej Control that doesn't provide its own JavaScript class. It extends Wisej.Web.Canvas and draws the component using HTML5 canvas instructions.

**Wisej.Web.Ext.JustGage** instead is an example of a Wisej (Extension) Control that provides its own JavaScript class, "wisej.web.ext.JustGage".

The main difference is that the first control is limited on the client side by the implementation of the base class. While the second (JustGage) can provide any kind of client side functionality in its JavaScript class fully controlled by the server side of the control.

Look at the two projects to get a better sense of these two controls.

## 2.2 WISEJ WIDGET

Wisej Widgets are always derived from the Wisej.Web.Widget class and never provide a specific JavaScript class. In fact the Wisej.Web.Widget can also be using by itself on any container.

A Wisej Widget is basically a general purpose client side container that can server as the containing element for any third party JavaScript widget. The control provides a simple way to load JavaScript packages from any sources, in sequence, and still asynchronously.

The two most important properties are Packages and InitScript. Packages is a collection of libraries while InitScript is the initialization JavaScript. The packages are cached and organized by name, so if multiple widgets use the same package (i.e. jQuery), it will be loaded only once.

**Wisej.Web.Ext.CoolClock** and **Wisej.Web.Ext.jQueryKnob** are examples of Wisej Widget extensions. The first loads the external resources from its own embedded repository, the second loads jQuery from "https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js" and the jQueryKnob widget from its own embedded resources.

## 2.3 WISEJ COMPONENT

Wisej Components are components that may live only the server or also on the client. They may specify a server side JavaScript object (not a widget) or be limited to the server.

The two relevant examples are Wisej.Ext.Translation and Wisej.Ext.Geolocation.

**Wisej.Ext.Translation** is a server side only component, derived from System.ComponentModel.Component and usable on any container. It lives only on the server side and never interacts with the client.

**Wisej.Ext.Geolocation** is a server and client component, derived from Wisej.Base.Component, and it provides its own JavaScript class “wisej.ext.Geolocation” which is not a widget but simply a qooxdoo object. One of the most interesting features of this component is the capability to callback the server asynchronously to provide the return value of the GetCurrentPosition function.

### 2.4 WISEJ EXTENDER PROVIDER

Extender providers are a completely different type of extension to Wisej. They all implement the System.ComponentModel.IExtenderProvider interface and extend Wisej.Web.Component.

Extender providers extend other components: when an extender provider is dropped on a container it will add properties and functionality to the other controls.

Optionally, a Wisej Extender Provider, may implement instead the Wisej.core.IWisejExtenderProvider interface to extend Wisej controls in design mode on the designer.

**Wisej.Web.Ext.Bubbles**, **Wisej.Web.Ext.SpeechRecognition**, and **Wisej.Web.SpeechSynthesis** are examples of extender providers that add bubble notifications, speech recognition, and speech synthesis functionalities to all the controls in the same container. The source code for these extenders is provided in the extensions download page.

**Wisej.Web.JavaScript**, **Wisej.Web.Animation**, **Wisej.Web.ToolTip**, **Wisej.Web.HelpTip**, and **Wisej.Web.ErrorProvider** are also extenders but are part of the core Wisej.Web product and the source code is not available in the download page.

**Wisej.Web.Rotation**, and **Wisej.Web.StyleSheet**, are all examples of extenders that are capable of extending components also at design time: if you add the Rotation extender to a container it will add rotation capabilities to all the other controls and if you change one of the Rotation properties it will be reflected immediately in the designer. The source code is not available.

## 3 HOW TO USE THE EXTENSIONS

All the Wisej extensions are components that have to be added to Visual Studio’s toolbox to be usable in the designer. In alternative, for testing or other reasons, you can drop a simple panel control (or a label control) and then change the class name in the .designer.cs file.

If the specific extension is already installed in the toolbox and you want to use a new build that you have created using the projects that we have provided, you have to first delete the component from the toolbox and then re add it using your new assembly.

Components can be added to Visual Studio toolboxes quite easily by adding the path to the assembly to the registry like this:

```
[HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\14.0_Config\ToolboxCo
ntrolsInstaller\Wisej.Web.Ext.Bubbles]

@="Wisej Extensions"

"CodeBase"="{Your Bin Folder}\Wisej.Web.Ext.Bubbles.dll"

"TargetFramework"=".NETFramework,Version=v4.5"

"Index"=dword:00000001
```

You can change the name of the tab in the default value (@=”Wisej Extensions”) and select the version of Visual Studio. The snippet above is for Visual Studio 2015. Use “11.0_Config” for VS 2012, “12.0_Config” for VS 2013, and “14.0_Config” for VS 2015.

In each project you will find the file Install-Template.reg. You can use it (must edit it first) to install your build of the extensions.