CarrotCakeCMS MVC: Developer's Guide

Copyright © 2015-2021 Samantha Copeland, http://www.carrotware.com/
CarrotCakeCMS is dual licensed under the MIT or GPL Version 3 licenses.

Wednesday, May 24, 2023

CarrotCakeCMS MVC Overview/Installation

Initial Configuration

Either build the solution from source or decompress a binary distribution. These instructions will assume use of the binary distribution. Extract the ZIP archive and place the contents in the location where your website files will live.

Update the web.config section for `<mailSettings>` to reflect the network delivery settings appropriate for your ISP. You can leave the password field blank if you are not required to provide one or if your mail server has whitelisted your web server. This is important for password retrieval purposes. If this is not configured and you forget your password, you will have no way to get a new password sent to you.

Locate the `<CarrotCakeCMS.Web />` section in the root web.config of the provided files (found in the admin project if you are using the source files). Create the ID to identify your site. Run the SQL statement `select NEWID() as id` or you can use the Visual Studio "Create GUID" functionality, either is fine, it is just a matter of what is more convenient. A GUID is just a long hexadecimal string, so you can even edit the string and change some of the values around, just stay with the values A-F and 0-9.

Paste the GUID into the web.config to replace the GUID entry for `SiteID` in `<Config SiteID=""/>`

```
	<CarrotCakeCMS.Web>
		<Settings>
			<Config SiteID="3BD253EA-0000-3333-BBBB-BB097C2255AA" SiteSkin="Classic" />
		</Settings>
	</CarrotCakeCMS.Web>
```

Alternately, remove the entry for `SiteID`, and open the file SiteMapping.config and add entries to map a set of domain names to different GUIDs. This will allow a single webroot to provide different content based on different domain names. This is similar to using different host headers or bindings in IIS for a single webroot has multiple domain names pointed at it. The entry for `SiteSkin` is used to designate the color scheme the admin tools will use. Valid values are: Classic, AmethystOrchid, BlueIzis, FrenchBlue, Mauve, MosaicBlue, Plum, QuetzalGreen, Rust, Sandstone, and SugarAlmond. If not specified, Classic will be used.

Note: if *www.example.com* and *example.com* need to provide the same site, you will have to add entries for both in SiteMapping.config, but they can point at the same GUID entry.

Configure the database

Go to your database server and create a new database, visit your hosting provider's database control panel, or ask your hosting provider to create a database. This should be SQL 2008 or later. This can be Express, Standard, or any higher tier version. Known and tested configurations are SQL 2008, SQL 2012R2, and SQL 2016.

Update the entry for the database connection in the web.config for `CarrotwareCMSConnectionString` to match your database credentials. See http://msdn.microsoft.com/en-us/library/jj653752.aspx and http://www.connectionstrings.com/sql-server-2008 for guidelines on editing the connection string.

If your SQL credentials used in the web.config have dbo/database owner (RECOMENDED) or higher rights, you can simply go to the logon page of the CMS and the tables will be created/updated for you. Before updating an existing site you should logout of the CMS administration. If you have no pre-existing users in the database, you will be prompted to create a new administrator. You can create whatever username/password you wish.

If you wish to manually deploy updates, or do not have sufficient rights from the credentials contained within the web.config execute the scripts found within the project **CMSDBUpdater** in the source code repository. The scripts provided were generated from SQL 2008 Express, they should also work on Standard 2008 or Standard 2008 R2 etc. and later editions. They may also work on SQL 2005, but have not been tested. Application authentication is provided by Microsoft Owin Security / AspNet Identity framework.

- **CREATE01.sql** -- will create the application tables. Run this first. It also includes the security provider schema and creates the security roles. You can just run this script and then go to the CMS logon page and create a new user.
- **MIGRATE01.sql** -- migrate data from the Web Forms edition, this is one way only, so make a backup of the database (or use a restored copy) in case you wish to return to the older version of the CMS.

Some users may experience the error "System.Data.SqlClient.SqlException: Invalid object name 'information_schema.columns'" when initially running the CMS. This is a result of SQL Server being configured for Case Sensitivity rather than Case Insensitivity. CarrotCake requires case insensitivity as not all queries will always match the database's capitalization. See also https://msdn.microsoft.com/en-us/library/ms175835.aspx

To check if this is your issue, run this SQL:

```
SELECT name, collation_name
FROM sys.databases
WHERE name = 'CarrotCMS1'     -- your database name here
```

If you see something like _CS_ in the output (ex: SQL_Latin1_General_CP1_CS_AS) you are running with case sensitivity. Your culture settings may vary.

To update the case sensitivity, run this SQL (change the CS to CI in the earlier returned culture):

```
ALTER DATABASE CarrotCMS1     -- your database name here
        COLLATE SQL_Latin1_General_CP1_CI_AS
```

This may require elevated permissions, or at least DB Owner rights.

Configure the web server (IIS)

Go to your web server and create a new website, visit your hosting provider's control panel, or ask your hosting provider to create a new site. This new site should be on a Windows server (or workstation) that has the .NET framework 4.5 installed on it. The website should be configured within an app pool that is using the .NET framework version 4.0 but only shared amongst other .NET 4.5 applications. In general, this will require either Windows Server 2008 or Windows 7, or later.

These are some more detailed websites on general IIS configuration:

- http://www.iisunderground.com/add-a-new-website-iis-7-5/
- http://www.iis.net/learn/manage/creating-websites
- https://support.microsoft.com/en-us/kb/323972
- https://support.microsoft.com/en-us/kb/308163

Point the website's root folder at the location you extracted the files into. The folder that contains the web.config and \bin\ directory is the root level of the website files.

This application must be the root level application in the website instance in which it runs. It cannot be run as a sub folder application at a deeper level within another website.

Run the website.

Initial Site Configuration

If, for security purposes, you want to use a different admin folder name than the default, you can, for example replace it with, say "/management/"as the new admin directory. You will need to place an entry in the `<CarrotCakeCMS.Web>` section in the web.config. This property should be blank unless being used for an override.

```
	<CarrotCakeCMS.Web>
		<Settings>
			<Config SiteID="3BD253EA-0000-3333-BBBB-BB097C2255AA" AdminFolderPath="/management/" />
		</Settings>
	</CarrotCakeCMS.Web>
```

Upload the files to your hosting account once you have finished updating your web.config. This CMS should be at the root level and not within a virtual folder of an existing website (though other code can co-exist with the CMS). This is an ASP.Net 4.5 web application so make sure that your IIS configuration is set up for this version of the framework.
Visit your site, you will be greeted with an Under Construction page (if you have the correct configuration) and there is no content currently deployed, or if there are pending database updates (such as when coming from when an earlier version).

Initial Logon

Follow the management link or go directly to <*mysiteurl*>/c3-admin/ (or /management/ if using the admin override path from the earlier example) and logon using any other account you might have added to the system that is a member of the "CarrotCMS Adminstrators" or "CarrotCMS Editors" role.

If there are no users in the system, you will be prompted to create the first administrative user.

Editors may manage content, but only administrators may mange/create user accounts. "CarrotCMS Editors" can only access a site that an administrator has previously associated them with. "CarrotCMS Adminstrators" can access all sites which share the same database.

Upon logon you will be taken to the site profile. You must fill in the basic information. You will need to save this data before proceeding to add content pages. If you are going to use RSS feeds or Canonical link headers, you must enter the Site URL (such as http://www.example.com) for your primary domain name if you have multiple domain names pointed at the same site content.

Proceed to the admin menu "pages > index> add page" and fill in the information about your first page. The first page you create will be marked as the root/homepage (if you didn't check the box when performing initial site creation). You can change this by visiting the sitemap and re-ordering the pages and saving the new order.

Building out several pages and then visiting the "pages > sitemap" menu, sorting and saving will automatically number the topmost item as the homepage. You can relocate pages to be subpages of others both in the sitemap editor and by visiting the page and selecting the parent page(s) for the given page.

When you enter a new page's title, the page header and navigation text and file name will be auto generated based on the page title. You are free to go and edit the values yielded. The full filename path must be unique across a single site. Validation will happen once you leave the text box. Filenames should not include any file extensions, nor should they include a "." (period). Some characters will be escaped once you save to reduce browser incompatibilities - such as spaces being interpreted differently across different browsers ex space being either '+' or '%20'. Any invalid characters will be converted to a "-" (dash) upon save. You will be warned of duplicate file names and be unable to save until resolved.

Every content page and blog post must have a template assigned to it. The template defined in ~/Views/CmsContent/PlainPage/PlainPageView.cshtml will be used if your template file does not exist. When pages or posts are created and there is a large number of other items of the same type (page or post) sharing the same template, the most frequently used template of the working type will be preselected for new items.

Blog posts are created in much the same fashion as a content page. Blog posts support two extra pieces of meta data: Categories and Tags, but do not support sub post/parent post concepts, while content pages can be created using a parent/top/child page hierarchy. You should designate a page as the site index / blog index and apply a template with the `PagedDataSummary` component to expose an index of your blog posts (if using the blog or search features).

Incorporate customization

The goal of CarrotCake CMS is to be as simple as possible (a piece of cake) to integrate your custom code: be they contact forms, registration forms, custom data presentation etc. You can use as much or as little of the CarrotCake framework as you want. CarrotCake believes it should be a piece of cake to incorporate your code and should not force your design in a way that does not best suit your needs.

No HTML/CSS file splitting is needed to make your own page design. Take any intact HTML/CSS design template and include requisite ASP.Net components such as the main base class (`@inherits CmsWebViewPage`) as well as adding content components. You can take the page, top to bottom, as a complete picture rather than splitting up headers, footers, and page bodies into separate files. You can also leverage the MVC Razor Layout concept to share common elements across several templates.

Custom components such as widgets for the front end (the publicly viewed portion of the site) lend themselves far better to be ignorant of the CMS APIs/Interfaces, and in some cases, simply creating the configuration entry that identifies the file location is sufficient for incorporation of your front-end widget modules.

At the simplest level: take a partial view, place the CSHTML file(s) somewhere in the site's folder structure and all required assemblies for the control in the site /bin/ directory, create a grouping of controls in a subfolder of ~/Views with a Public.config, open a page in advanced edit, and drop the widget on a page in a predetermined placeholder.

The interfaces/base classes for widgets and admin widgets are almost always required. They provide the required registration that MVC needs for routing so that views in the admin area and partial views for widgets can operate.

Build your own Template

To add your own templates, study the ~/Views/CmsContent/PlainPage/PlainPageView.cshtml and/or the Citrus Island files ~/Views/Templates/citrus-island/*.cshtml. Take any HTML+CSS template and begin swapping out html content with server components like top menu, latest updates, category/tag lists, and content areas for the server components as found in the plain template.

When creating any template or template sub component, place the code `@inherits CmsWebViewPage` as the first line of the View or Layout+View template file. This exposes the page data, site data, and numerous related navigation lists (parent, child, sibling etc.) to the view file so that it can be incorporated in the design.

When you are using a layout view page, it will usually have a null layout, and may use some common Viewbag properties:

```
@{
	Layout = null;

	ViewBag.Title = CmsPage.ThePage.TitleBar;
	ViewBag.SiteName = CmsPage.TheSite.SiteName;
}
```

The view that is using your layout will point at the layout using a relative path (your filename may be different):

```
@{
	Layout = "_mainlayout.cshtml";
}
```

When using a layout view, insert the `@RenderSection` method calls as needed in your views and place the corresponding `@section` blocks in the child views as needed.

Once the template file has been set to inherit `CmsWebViewPage` you can begin fleshing out the CMS components.

Note: If using the main layout + child view template model, some of these steps will apply to the layout view and some to the child view.

In the page head block, you should place the `@CarrotCakeHtml.MetaTags()` as early as possible, a script include for jquery, jquery ui, and unobtrusive ajax (or use the `jquerybasic` helper). The component `jquerybasic` provides the ability to only deliver the javascript and the ability to omit the css such that you can generate a theme and use your own color scheme. Place the property `@CmsPage.Titlebar` in the `<title />` tag. Place a call to `@CarrotCakeHtml.IncludeHead()` as the very last thing before the head tag closes, and put in a call to `@CarrotCakeHtml.IncludeFooter()` as the very last thing before the body tag closes. The inclusion of jQuery UI, and the header/footer include methods are not optional as the CMS requires jQuery UI when in advanced edit, as well as the ability to inject edit components to support editing (the edit menu, supporting scripts and css). The meta tag function provides the injection of the page's keyword and description tags on the rendered html, as well as providing the robot nocrawl should the block search engine checkbox be selected.

- Content areas are designated by the `@CarrotCakeHtml.RenderBody(CarrotCakeHtml.TextFieldZone.TextCenter)` method call, TextRight and TextLeft can also be specified. You may have one or all or none of the three content blocks in your template.
- The page heading should be taken from the `@CmsPage.Heading` property.
- Widget areas are designated by the `@CarrotCakeHtml.RenderWidget("phCenterBottom")` or `@CarrotCakeHtml.RenderWidget(CarrotCakeHtml.CommonWidgetZone.phCenterTop)` method. The names for these containers are passed in as a string or enum, `"phCenterBottom"` is just an example value, and must remain unique (not repeated) within the same template. If you don't use the same name for the same general location in every template in your site, your content may not appear when you change templates.
- It is recommended to place the `jquerybasic` server component (`@(new jquerybasic() { SelectedSkin = jquerybasic.jQueryTheme.Silver })`) as one of the first things in the page head tag so that editing in the advanced mode is least likely to have issues. You can specify jQuery versions (ex. `JQVersion = "1.9"` or `JQVersion = "1.11"`), versions 1.7 through 1.11 are available. You can disable the skin stylesheets (ex. `SelectedSkin = jquerybasic.jQueryTheme.NotUsed`) if you have generated your own theme from the jQuery UI website or do not need the jQuery UI themes that are embedded. There are several color schemes included so you can see if one of the out of the box ones works for you. Including jQuery UI in your template is not optional because of the advanced edit controls, though you may reference a CDN rather than the embedded component, or even download a full jQuery UI theme and include the JS files from that source. Note that Microsoft jQuery Unobtrusive Ajax is also included in this component and therefore should be manually included if not using the `jquerybasic` component.

By default, the system targets a three column layout, thus left, right, & center content areas are baked in, as are a top & bottom left, right, & center widget zones are assumed and are the suggested areas in example templates. Additional widget zones can also be created, but are not required. You should make your widget zone names consistent across your templates so that if you change templates you will still be able to see your widgets.

It is recommended that you hardcode the fully qualified paths to your template's asset folder within a template file. Templates are generally going to live within the ~/Views/ folder path and thus any content found there cannot be directly served to the web. Alternately, within the suggested ~/Assets/ folder you can create sub folders to contain all images, css, js etc. that your template depends on, pairing up on folder names will help organize resources. Ex . the folder ~/Assets/Carrot/ would be the assets used by the templates found in ~/Views/Templates/Carrot/ .

The CMS Skins/ Themes/ Templates can be placed in sub folders (one set per folder) in the folder named ~/Views/Templates/ . This is the default location and it can be overridden by a web.config entry (`<OverrideConfigFile TemplatePath = "~/Views/Templates/" />`). Any first level sub folder of this which has a Skin.config file can be picked up on demand and can be moved around (such as renaming the folder) and redetected, thus streamlining the upload process of the skin. Simply include a Skin.config file that enumerates the view files in the design set minus any folder information above the skin's folder level. A layout page should never be the entry in your skin file, only the resulting view files.

Example template directory:

```
~/Views/Templates/Meadow/
        _layout.cshtml
        Meadow1.cshtml
        Meadow2.cshtml
        Skin.config
```

Example asset directory:

```
~/Assets/Meadow/
        /images/ <subfolder for images in the design set>
        style.css
```

Skin.config contents:

```
<?xml version="1.0" encoding="utf-8" ?>
<tbl>
	<pagenames>
		<templatefile>Meadow1.cshtml</templatefile>
		<filedesc>Meadow 1</filedesc>
	</pagenames>
	<pagenames>
		<templatefile>Meadow2.cshtml</templatefile>
		<filedesc>Meadow 2</filedesc>
	</pagenames>
</tbl>
```

Once you have edited your template, place it within the site's template folder. It is generally recommended that your template's assets reside in its own folder with all supporting CSS & Images contained therein. Images & CSS paths should use the absolute path so that no matter where in the site you are, the paths will resolve on the rendered page and not result in broken images or missing stylesheets.

Make sure that your file name and title are XML escaped (don't use a raw ampersand the & character - use &amp; for example) Stick to alpha numeric if you are worried about non-safe characters or otherwise are unsure.

Using the Navigation Server Components

There is one two-level navigation server component `TwoLevelNavigation`. There are additional list components for listing top-most pages, child/sub pages, and sibling pages, but as they are simple lists with only a few CSS directives, they can be used intuitively. There are also extension lists available (`CmsPage.ChildNav`, `CmsPage.SiblingNav`, `CmsPage.GetSiteUpdates(10)` etc.) through the exposed properties when inheriting the CMS base page view

Within a template's view code such as the following can be used to display related data.

```
	@if (CmsPage.ChildNav.Any()) {
		<h1>Child Pages</h1>
		<ul class="sidemenu">
			@foreach (var itm in CmsPage.ChildNav) {
				<li class="child-nav"><a href="@itm.FileName">@itm.NavigationText</a></li>
			}
		</ul>
	}


	@{
		var lstUpd = CmsPage.GetSiteUpdates(10);
		if (lstUpd.Any()) {
			<h1>Recent Updates</h1>
			<ul class="sidemenu">
				@foreach (var itm in lstUpd) {
					<li class="child-nav">
						<a href="@itm.FileName">
							@itm.NavigationText
							@String.Format(" ({0:d})", itm.GoLiveDate)
						</a>
					</li>
				}
			</ul>
		}
	}


	<ul class="nav">
		@foreach (var n in CmsPage.TopNav.OrderBy(x => x.NavOrder)) {
			string cssClassState = CmsPage.NavIsInCurrentTree(n) ? "current" : "not-current";
			<li>
				<a class="@cssClassState" href="@String.Format("{0}", n.FileName)">
					@n.NavigationText
				</a>
			</li>
		}
	</ul>


	@if (CmsPage.ThePage.ContentType == ContentPageType.PageType.BlogEntry) {
		var pagecat = CmsPage.GetPageCategories(25);
		var pagetag = CmsPage.GetPageTags(25);

		<div class="meta">
			@foreach (var itm in pagecat) {
				<span class="meta-item @String.Format("meta-count{0}",
	CmsPage.GetRoundedMetaPercentage(itm))"><a href="@itm.Uri">@itm.Text</a></span>
			}
		</div>

		<div class="meta">
			@foreach (var itm in pagetag) {
				<span class="meta-item @String.Format("meta-count{0}",
	CmsPage.GetRoundedMetaPercentage(itm))"><a href="@itm.Uri">@itm.Text</a></span>
			}
		</div>
	}
```

The TwoLevelNavigation component will provide a simple UL/LI output with ability to specify colors and font size which generates a simple CSS based dropdown menu. This is a good choice if you need a quick navigation menu. The auto stylesheet generation can be turned off (AutoStylingDisabled = true) if you have already designed CSS markup to facilitate your menu. It also has many properties to specific the CSS names for various components like the CSS class for the selected state, sub menu UL CSS class etc. This should give you maximum control of re-styling the menu any way you want to.

```
TwoLevelNavigation nav = new TwoLevelNavigation() {
        AutoStylingDisabled = true,
        ElementId = "nav"
};

TwoLevelNavigation nav = new TwoLevelNavigation() {
        FontSize = new SizeUnit("10px"),
        ForeColor = System.Drawing.ColorTranslator.FromHtml("#FFFFFF"),
        BackColor = System.Drawing.ColorTranslator.FromHtml("#F4845A"),
        ElementId = "nav"
};
```

Regardless of how the TwoLevelNavigation object is configured, to output, either place an all encompassing output of @nav within the view's body. This will output all associated CSS (if any) and the HTML markup. You can also split up the output of the CSS and HTML portions so as to provide more ideal placement. Place @CarrotWeb.RenderTwoPartControlBodyCss(nav) in the head in the desired ordered location and @CarrotWeb.RenderTwoPartControlBody(nav) in the view's body. If the component is configured to not output CSS, the @CarrotWeb.RenderTwoPartControlBodyCss(nav) extension will serve no useful purpose.

Site Index Page

If you opt to allow site searches or use the blog feature, you will need to designate a page within the site as the Site Index Page/ Blog Index. This is done from the Site Info page (and also the site bulk apply template page) the same place the website identity is configured (site name, slogan, URL etc). All search results and Tag/Category/Date links will direct at this page. In order to show the matching records the PagedDataSummary component must be on this page. You can determine the number of pages, turn off the pager, specific the type of data that will load by default. If being targeted by a search result, category, or tag link, it will auto flip to the right result type.

```
var pager = new PagedDataSummary();
pager.ContentType = PagedDataSummary.SummaryContentType.Blog;
pager.PageSize = 10;
pager.IgnoreSitePath = false;
pager.FetchData();
```

Styling and formatting can be applied to present customized appearances when printing out the pager's results, CSS styling tags have been applied so that the active page can be offset from the other pages. The pager supports a querystring pattern for paginating. There is also methodology to provide marking the current page of the result set using the CarrotWeb.BeginWrappedItem method.

```
@if (pager != null) {

        foreach (var item in pager.DataSource) {
                var usr = item.BylineUser;

                <div class="post">
                        <h2 class="title"><a href="@item.FileName">@item.NavigationText</a></h2>
                        <p class="meta">
                                <span class="date">@String.Format("{0:MMMM d, yyyy}", item.GoLiveDate)</span>
                                <span class="posted">Posted by @usr.FullName_FirstLast</span>
                        </p>
                        <div class="entry">
                                <p>@item.PageTextPlainSummary</p>
                                <p class="links"><a href="@item.FileName">Read More</a> </p>
                        </div>
                </div>
        }

        <div class="pagerfooterlinks">
                @foreach (var i in pager.PageNumbers) {
                        using (CarrotWeb.BeginWrappedItem("div", i, pager.PageNumber, new { @class = "pagerlink
selectedwrap" }, new { @class = "pagerlink" })) {
                                using (CarrotWeb.BeginWrappedItem("a", i, pager.PageNumber, new { @class =
"selected", @href = pager.GetUrl(i) }, new { @href = pager.GetUrl(i) })) {
                                        @String.Format(" {0} ", i)
                                }
                        }
                }
        </div>
}
```

Using the Header Components

Each of these components should go in the page header `<head>` and `</head>` tags.

To provide a hint to search engines as to what your primary domain name is. It will use the Site URL from the site configuration. You can pass a parameter to it to force a redirect if the page that uses the template is viewed by a non-canonical URL.

```
@CarrotCakeHtml.GetSiteCanonicalURL()
```

If you want to publish an RSS feed for the site, put this component in the header. A parameter can be passed in to provide just the page or blog area to the feed.

```
@CarrotCakeHtml.Rss()
```

When the render modes specify a render as a link format, then, they can be placed in the body of the page rather than the page header. The `@CarrotCakeHtml.RssLink` method utilizes an optional parameter, `imagePath`, which will allow you to specify an icon to be shown in the link, if no value is provided, a 16px square RSS icon will be served.

```
@CarrotCakeHtml.RssLink(SiteData.RSSFeedInclude.PageOnly, null, "Page RSS", new { @class = "rssimage" })
@CarrotCakeHtml.RssLink(SiteData.RSSFeedInclude.BlogOnly, null, "Blog RSS", new { @class = "rssimage" })
```

If you are going to do some social media interaction with services that use the OpenGraph data, this control will expose some of the common page data that Open Graph often provides.

```
@CarrotCakeHtml.RenderOpenGraph()
```

Using the SiteMap Provider

Search providers such as Google allow you to provide a sitemap file to give the crawler a hint as to important urls. CarrotCake provides this functionality. Just provide (http|https)://<<domain name>>/sitemap.ashx as the url to your sitemap.

Build your own Widget

There are three main ways to deliver a widget as a controller's `PartialViewResult`, a specified view file with an optional model paired to it, and as a class which implements `IHtmlString`. There is a small bit of ground work to do when going the way of using controller view results, but it's fairly straightforward.

The project CMSInterfaces (or Carrotware.CMS.Interface.dll) should be referenced by any of your custom widgets if you want to them to have basic information injected or have communication to the CMS when being inserted in the page.

If you are only planning on building server components and not MVC views as widgets, you can just create an empty class library and optionally reference the interfaces project/assembly.

Classes for server components can also be utilized, prefixed with CLASS: and the class/assembly noted. Using this method requires implementation of the `System.Web.IHtmlString` interface from the .NET framework. Whatever text/HTML that is to be emitted from the control should be included in the `ToHtmlString()` method.
example configuration string : `CLASS:Carrotware.CMS.UI.Components.TwoLevelNavigation, Carrotware.CMS.UI.Components`

There is also a simple base class `WidgetBase` that you can use with any custom class-based widgets (the ones using `IHtmlString`) component that you opt to use the widget interface with.

If going the full view/controller route, create a new empty MVC (a .NET 4.5 and MVC5) project and reference the project CMSInterfaces (or Carrotware.CMS.Interface.dll), and at your option, the project WebComponents (or Carrotware.Web.UI.Components.dll) project as this contains a number of UI helpers within it.

When a page that uses a controller based widget or component which uses `ToHtmlString()` is initially loaded, the widget interfaces will pass in values as part of drawing the page. On partial postback, components are on their own and are working independently of the framework as far as runtime injection goes. It is recommended that if postbacks are made but values from the initial load are needed, persist them in some fashion such as hidden fields properties of the posted model. There are several variations as examples of this in the widget projects in the CMS solution.

When using MVC views for widgets, some use of specific base classes (or implementation of specific interfaces) are required. Any public facing widget's controller should inherit `BaseDataWidgetController` or `BaseWidgetController` and any admin area's controller from `BaseAdminWidgetController`.

The project's route registration (when using MVC views) should include the namespaces from the assembly as more than one widget may have controllers named Home or Admin etc. This code can be used with little or no modification.

```
	public static void RegisterRoutes(RouteCollection routes) {
		routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

		Assembly _assembly = Assembly.GetExecutingAssembly();

		List<string> _namespaces = _assembly.GetTypes().Select(t => t.Namespace)
					.Where(x => !String.IsNullOrEmpty(x))
					.Distinct().ToList();

		routes.MapRoute(
			name: "Default",
			url: "{controller}/{action}/{id}",
			defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional },
			namespaces: _namespaces.ToArray()
		);
	}
```

You can simulate a particular SiteID being passed in to the widgets for testing purposes within a stand-alone project by wiring up the Global.asax similar to the following `Application_Start()` start function.

```
		public class MvcApplication : System.Web.HttpApplication {
			protected void Application_Start() {
				AreaRegistration.RegisterAllAreas();
				RouteConfig.RegisterRoutes(RouteTable.Routes);


			ControllerBuilder.Current.SetControllerFactory(CmsTestControllerFactory.GetFactory());

			}
		}
```

This does require use of a web.config `<appSettings />` entry to specify the test SiteID value.

```
	<add key="TestSiteID" value="3BD253EA-AC65-4EB6-A4E7-BB097C2255A0"/>
```

So that one layout can be used in test from the local dev project and in deployment use the CMS provided one, an additional web.config keys can be specified as well as using some constants in the layout cshtml file.

In the web.config `<appSettings />` these keys can be inserted, the paths reflect the local project's layout paths

```
	<add key="LayoutMain" value= "~/Views/Shared/_Layout.cshtml"/>
	<add key="LayoutPopup" value= "~/Views/Shared/_LayoutPop.cshtml"/>
	<add key="LayoutPopupOpenFunction" value= "window.open"/>
```

In the layout files, specify which layout is being used:

```
	@{
		Layout = Carrotware.CMS.Interface.CarrotLayout.Popup;
	}
```

or

```
	@{
		Layout = Carrotware.CMS.Interface.CarrotLayout.Main;
}
```

If you wish to leverage the popup window in the CMS you can specify the launch using an onclick assigned to the function call.

```
	onclick="@CarrotLayout.WritePopupLink(Url.Action("UrlActionNameGoesHere"))"
```

Controller based widgets leverage the Area concept, so your project should include a class that inherits from `BaseWidgetAreaReg`. No code needs be written, just an empty class that inherits from this base class and placed somewhere within the MVC project's codebase.

```
                    public class MyWidgetRegistration : BaseWidgetAreaReg {

                    }
```

The norm with the MVC widgets is to create a partial view (the action method a ':' (colon) and then class a ',' (comma) and the assembly file name, minus the ".dll" extension.
example : `ProductSearch:Northwind.Controllers.HomeController, Northwind`

You can also leverage the `IWidget.PublicParmValues` property so that the properties that were set using the generic property editor (if enabling widget editing - see `IWidget.EnableEdit`) will also be passed in as a `Dictionary<string, string>`. It is recommended to use the `WidgetBase` (for class library widgets) or `WidgetActionSettingModel` (for controller widgets which are using `IWidgetDataObject.WidgetPayload` and need a base class for the settings object) base classes as some parsing routines are included and should simplify assigning the values that are sent in via the interface. This will provide values that the widget can then convert at its own discretion depending on what pieces the widget developer deigns to be important.

In cases where your control doesn't play well with the edit mode use the property from `IWidget.IsBeingEdited` and when this interface passes in the edit mode, simply hide/disable those features within your component.

If you opt to enable editing of the widget by way of `IWidget.EnableEdit` interface / param combo and want to provide a drop down list or checkbox list, simply tag the property with the widget attribute to map the property to the dictionary list that will provide the values.

For a multi-value property that will be presented as a checkbox list and use a `Dictionary` that will provide the available values. In this example, the property `GalleryIDs` will have the values provided from a `Dictionary` named `lstGalleryIDs`. Note that values saved will be mapped to the property key [field]|[#] (creating a unique key for each entry)- so in this example, key values might look like `GalleryIDs|0`, `GalleryIDs|1`, `GalleryIDs|2`, so keep this in mind when consuming the data. The `Description` attribute is completely optional, but can be used to provide additional information to the end user when editing properties.

```
        [Description("Galleries to display")]
        [Widget(WidgetAttribute.FieldMode.CheckBoxList, "lstGalleryIDs")]
        public List<Guid> GalleryIDs { get; set; }
```

For a single-value property that will be presented as a drop down list and use a `Dictionary` that will provide the available values. In this example, the property `GalleryID` will have the values provided from a `Dictionary` named `lstGalleryID`.

```
        [Description("Gallery to display")]
        [Widget(WidgetAttribute.FieldMode.DropDownList, "lstGalleryID")]
        public Guid GalleryID { get; set; }
```

When needing to pass in values to an MVC View/Controller widget, some additional work is needed. All settings and properties will be passed in to the widget from a settings object. The particular class will be specified as an attribute `WidgetActionSettingModel` on the partial view method which identifies the class for the settings object. The controller should either use a base class (ex. `BaseDataWidgetController`) which has included the `IWidgetDataObject.WidgetPayload` property or implement the relevant interfaces which include this property. Either a full Assembly Qualified Name string or a Type parameter can be specified for the `WidgetActionSettingModel`.

```
[HttpGet]
[WidgetActionSettingModel("Carrotware.CMS.Interface.WidgetActionSettingModel, Carrotware.CMS.Interface")]
public PartialViewResult ProductSearch() {
        WidgetActionSettingModel settings = new WidgetActionSettingModel();

        if (this.WidgetPayload is WidgetActionSettingModel) {
                settings = (WidgetActionSettingModel)this.WidgetPayload;
                settings.LoadData();
        }

        ProductSearch model = null;
        model = InitProductSearch(model);

        return PartialView();
}


[WidgetActionSettingModel(typeof(CalendarSimpleSettings))]
public ActionResult CalendarDateInfo(DateTime? calendardate) {
        DateTime theEventDate = calendardate ?? DateTime.Now.Date;

        CalendarSimpleSettings payload = new CalendarSimpleSettings();

        if (this.WidgetPayload is CalendarSimpleSettings) {
                payload = (CalendarSimpleSettings)this.WidgetPayload;
                payload.LoadData();
        }

        DateModel model = new DateModel(theEventDate, payload.SiteID);

        if (String.IsNullOrEmpty(payload.AlternateViewFile)) {
                return PartialView(model);
        } else {
                return PartialView(payload.AlternateViewFile, model);
        }
}
```

If the widget developer has decided to allow alternate views to be provided to the widget and has implemented `IWidgetView.AlternateViewFile` code can be written similar to this. The alternate view value is provided from the widget configuration string.

```
[HttpGet]
[WidgetActionSettingModel("Carrotware.CMS.Interface.WidgetActionSettingModel, Carrotware.CMS.Interface")]
public PartialViewResult ProductSearch() {
        WidgetActionSettingModel settings = new WidgetActionSettingModel();

        if (this.WidgetPayload is WidgetActionSettingModel) {
                settings = (WidgetActionSettingModel)this.WidgetPayload;
                settings.LoadData();
        }

        ProductSearch model = null;
        model = InitProductSearch(model);

        if (String.IsNullOrEmpty(settings.AlternateViewFile)) {
                return PartialView(model);
        } else {
                model.AltViewName = settings.AlternateViewFile;
                return PartialView(settings.AlternateViewFile, model);
        }
}
```

This widget configuration value is similar to the earlier controller action widget and follows the pattern Action : (colon) Class + Assembly as before, but with another : (colon) and the View Name. This can be specified in two ways: just the view name or a partially/fully qualified path including the extension cshtml or vbhtml. Specifying just the view name expects that the view file resides in the same directory as the original/default view for the corresponding action. The other way provides the ability to place the view in another location.

ex 1: `ProductSearchMulti:Northwind.Controllers.HomeController, Northwind:ProductSearchAlt2Multi`

ex 2: `ProductSearchMulti:Northwind.Controllers.HomeController, Northwind:/Home/ProductSearchAltMulti.cshtml`

```
        [HttpGet]
        [WidgetActionSettingModel("Carrotware.CMS.Interface.WidgetActionSettingModel, Carrotware.CMS.Interface")]
        public PartialViewResult ProductSearch() {
                WidgetActionSettingModel settings = new WidgetActionSettingModel();

                if (this.WidgetPayload is WidgetActionSettingModel) {
                        settings = (WidgetActionSettingModel)this.WidgetPayload;
                        settings.LoadData();
                }

                ProductSearch model = null;
                model = InitProductSearch(model);

                if (String.IsNullOrEmpty(settings.AlternateViewFile)) {
                        return PartialView(model);
                } else {
                        model.AltViewName = settings.AlternateViewFile;
                        return PartialView(settings.AlternateViewFile, model);
                }
        }
```

The settings object used with the MVC controller is configured in the same way as one of the class library based widgets. This object is loaded up with data which the particular partial view can pick off values from. Because the controller is using the datatype of `Object` for the settings property `WidgetPayload` will need to be cast to whatever specific datatype was designated by the action. The use of the `Object` is deliberate as it can allow multiple actions in the same controller to each have data injected at run time rather than forcing only one action per controller.

For the class or settings object, all attributes that are desired to be shown in the generic property editor will need to have the `[Widget]` attribute set otherwise that property won't be shown in the generic property editor.

```
        [Description("Display gallery heading")]
        [Widget]
        public bool ShowHeading { get; set; }
```

Two other methods of using a view as a widget involve either specifying a razor view file (cshtml or vbhtml) with or without a model being passed. These will be "|" (pipe) delimited if providing a model class. The model class, if used, should be created and implemented similarly to that of an class being used in the context of one of the controller resulting widgets. If passing just a view file, the view should not expect a model to be passed, if a model is being passed, the model type should match what the view is expecting.

ex 1: `Home/Hello.cshtml`
ex 2: `Home/MultiProdList.cshtml|Northwind.MultiOptions, Northwind`

Once you have built your widget, add it to the Public.config file within the widget's view folder. Entries must be well-formed XML. All widgets will live within the ~/Views/ path within a subdirectory named according to the assembly file which corresponds to the controller's namespace. As with Skins, do not include the path for folder configured widgets except as relative to the location of the config file . If the view is in the Home controller's view folder listing "Home/Index.cshtml" is valid, or simply referencing the action and assembly information, depending on the type of widget being created.

If building view-only widgets, the widget's folder name is not enforced the way it would be if there was an assembly involved. If you wish, you can even define logic/models and place them in the App_Code folder and pass these in. Note that these classes should have a namespace for scope not just a class name. These classes will only have a class name and no assembly path passed in as the model class string ex (`<filepath>FeedTest.cshtml|Carrotware.CMS.Mvc.UI.App_Code.FeedDataList</filepath>`). If a model is being defined for a view in this fashion, it can still use any of the widget interfaces.

Ex. the widget's project is set up to use `CarrotCake.CMS.Plugins.PhotoGallery` as the assembly name. So the widget's views should live within ~/Views/CarrotCake.CMS.Plugins.PhotoGallery/ with copies of the usual ~/Views/<<Controller

Name>> from the widget's project. The Public.config and Admin.config files for the widget will live in the widget's root directory.

```
<ctrlfile>
       <filepath>widget path or class name</filepath>
       <crtldesc>title to show in toolbar</crtldesc>
</ctrlfile>
```

To add your widget to a page, login to the management backend. Once logged in, view the page you want to insert the component into. You should see some light green boxes in the margins and in the footer. Follow the advanced edit link when you are going to load your widgets into the site.

You will get a floating toolbar which lists your widgets. You can drag and drop these into any of the widget placeholders. Each placeholder's name will appear in a dark green bar, each of your content areas will appear in a light green bar.

Individual widget toolbars can be dragged & dropped to reorder within a container or from one container to another. Widgets that expose custom edit links or just provide info that they are editable will expose edit links. If you drop the widget in one container and need it to be placed elsewhere, you can simply drag it from one container into another.

The floating toolbar also provides the ability to modify some of the core page information, like navigation link caption, page heading, and page title attributes.

Changes will be in memory (serialized to the database) until the save button from the toolbar is clicked. If you abandon your edit session for more than 2 hours, your changes will be lost. Each time you add/remove or edit a widget, the clock will get reset.

When a page is being edited, there is a "heartbeat" which will update your claim on the page so as to block other users from editing the page and overwriting your changes. If you exit the edit mode of a page or otherwise lose connection with the website more than 2 minutes, another user may then edit the page.

If your widget does not appear in the toolbar, you can visit the management homepage (the page which has the site identity information) and click the "Refresh Configs" button.

Once your widget has an entry in a Public.config in a sub folder of ~/Views/, copy the view file(s) to the pre-determined location, and copy its DLL to the site's \bin\ folder.

Build your own Admin Module

The project CMSInterfaces should be referenced by any of your custom admin module controllers. Modules that will implement the interfaces need only reference the Carrotware.CMS.Interface.dll assembly.

There is also a base class `BaseAdminWidgetController` that you can use with any admin controller which will help with interaction and registration of the controller.

Admin views leverage the Area concept, so your project should include a class that inherits from `BaseWidgetAreaReg`. No code needs be written, just an empty class that inherits from this base class. If this is the admin area for a public facing widget, this has possibly been done already.

```
public class MyWidgetRegistration : BaseWidgetAreaReg {

}
```

Once you have built your module, add it to the Admin.config file for the widget's main folder. Again, entries must be well-formed XML. As with Skins, do not include the path for folder configured widgets.

Ex. the widget's admin project is set up to use `CarrotCake.CMS.Plugins.PhotoGallery` as the assembly name. So the admin widget's views should live within ~/Views/CarrotCake.CMS.Plugins.PhotoGallery/ with copies of the usual ~/Views/<<Controller Name>> from the widget's project. The Public.config and Admin.config files for the widget will live in the widget's root directory.

The XML format has two tiers, one tier is the top level menu to group your widgets according to functionality. You can have multiple views within a family of modules. The `<area>` node corresponds to the assembly name/directory the admin widget is hosted in. This value should be the same for all widgets within a widget set.

```
<?xml version="1.0" encoding="utf-8" ?>
<tbls>

        <pluginlist>
                <area>Northwind</area>
                <caption>Northwind Sampler</caption>
        </pluginlist>

        <plugincontrols>
                <area>Northwind</area>
                <menuorder>1</menuorder>
                <action>Products</action>
                <controller>Admin</controller>
                <pluginlabel>Product List</pluginlabel>
                <visible>true</visible>
        </plugincontrols>

        <plugincontrols>
                <area>Northwind</area>
                <menuorder>2</menuorder>
                <action>CreateProduct</action>
                <controller>Admin</controller>
                <pluginlabel>Add Product</pluginlabel>
                <visible>true</visible>
        </plugincontrols>

</tbls>
```

Once the entries for your controls have been saved to the config file, you can click the modules menu and a list of admin modules will appear. Expanding the menu will show the one or more components that make up a module.

While you do not have to implement any of the admin interfaces, you should so that the admin area is properly registered.

If your module does not appear in the list on the module page, you can visit the site info page (the page which has the site identity information) and click the "Refresh Configs" button.

Once your module has an entry in the Admin.config in a sub folder of ~/Views/, copy the razor views (cshtml or vbhtml) files to the pre-determined location, and copy its DLL to the sites \bin\ folder.

Build your own Text Widget

Sometimes it is necessary to escape or otherwise massage text content found in the content body of a page. To this end, there is an interface (`ITextBodyUpdate`) and a configuration file (by default TextContentProcessors.config). See the example class Carrotware.CMS.UI.Components.EmailEscapeInBody which escapes email addresses into their ASCII codes.

Create a class which implements the interface `ITextBodyUpdate`

Add an entry into the config file specifying the class and assembly location as in the example implementation

Build your DLL class and copy to the website \bin\ directory

Visit the text widget menu and turn on/off the areas you want to have the content evaluated

Using a Content Snippet

Sometimes you have some content that is fairly static and/or used in many places and would otherwise include in a hard coded fashion in your template. Rather than hard coding the content, you can use the content snippet component to create a sluggable piece of content that will be looked up at run time. This content is versioned and can either be hard coded in a content template file or dragged and dropped as a widget into individual page by page basis. To include in a template/razor view, place markup similar to the below. The `SnippetSlug = "first-snippet"` represents the corresponding block of content and will be taken from the corresponding snippet, this value is just an example.

```
@(new ContentSnippetText { SnippetSlug = "first-snippet" })
```

The markup for the snippet's tag may be placed in any view you have been using as part of a template family.

The snippet's slug is required to be unique (the admin UI will validate that this is so) but the name is not so constrained and is used as a hint only when trying to determine which snippet to select. They can be turned on or off, deleted, or time activated/deactivated.

Sitemap

The website can serve an XML sitemap <website>/sitemap.ashx ex: http://www.carrotware.com/sitemap.ashx

Titlebar Placeholders

Under Site > Site Info there is a field labeled “Site Titlebar Pattern”. This can dynamically populate the page title in the browser titlebar.

[[CARROT_SITENAME]] – Site Name
[[CARROT_SITE_NAME]] – Site Name (alternate format)
[[CARROT_SITE_SLOGAN]] – Site Slogan

[[CARROT_PAGE_TITLEBAR]] – Webpage Title
[[CARROT_PAGE_PAGEHEAD]] – Webpage Heading
[[CARROT_PAGE_NAVMENUTEXT]] – Webpage Navigation Link Text
[[CARROT_PAGE_DATE_GOLIVE]] – Webpage Go Live Date, default format
[[CARROT_PAGE_DATE_EDIT]] – Webpage Last Edited Date, default format
[[CARROT_PAGE_DATE_GOLIVE:X]] – Webpage Go Live Date, X is a string that would be appropriate for String.Format of a date/time object: ex [[CARROT_PAGE_DATE_GOLIVE:MMMM d, yyyy]]
[[CARROT_PAGE_DATE_EDIT:X]] – Webpage Last Edited Date, X is a string that would be appropriate for String.Format of a date/time object: ex [[CARROT_PAGE_DATE_EDIT:MMMM d, yyyy]]

Credits & Licensing

CarrotCakeCMS is dual licensed under the MIT or GPL Version 3 licenses.

The **CarrotCakeCMS** logo is © 2011-2012 Samantha Copeland

The **Carrotware** carrot is © 1997, 2002 Samantha Copeland

The **Powered by Carrotware** badge is © 2005, 2009 Samantha Copeland

These works are licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.
http://creativecommons.org/licenses/by-nc-nd/3.0/

These images may freely be used in conjunction with promotion of the CarrotCakeCMS.

Components Utilized

Microsoft .NET framework version 4.5, Microsoft MVC 5, Razor 3, Web Pages 3, Microsoft jQuery Unobtrusive Ajax, Microsoft Owin, AspNet Identity, AspNet Web Optimization, Web Infrastructure, and Entity Framework 6

http://www.asp.net/mvc/mvc5
http://www.asp.net/identity
https://msdn.microsoft.com/en-us/data/ef

Microsoft software license for the Microsoft .NET library:
    http://www.microsoft.com/web/webpi/eula/net_library_eula_enu.htm

Copyright (c) Microsoft Open Technologies, Inc. All rights reserved.
Apache 2.0 license: http://www.apache.org/licenses/LICENSE-2.0
    https://github.com/ASP-NET-MVC/aspnetwebstack/blob/master/License.txt
    https://aspnetwebstack.codeplex.com/license
See also framework source:
    https://aspnetwebstack.codeplex.com/
    https://github.com/ASP-NET-MVC/aspnetwebstack

jQuery JavaScript Library

http://jquery.com/

Copyright 2011, John Resig
Dual licensed under the MIT or GPL Version 2 licenses.
http://jquery.org/license

Includes Sizzle.js
http://sizzlejs.com/
Copyright 2011, The Dojo Foundation
Released under the MIT, BSD, and GPL Licenses.

jQuery UI
http://jqueryui.com/

Copyright 2011 (http://jqueryui.com/about)
Dual licensed under the MIT or GPL Version 2 licenses.
http://jquery.org/license

http://docs.jquery.com/UI

Chosen
http://harvesthq.github.io/chosen/

Chosen, a Select Box Enhancer for jQuery and Prototype
by Patrick Filler for Harvest, http://getharvest.com
MIT License, https://github.com/harvesthq/chosen/blob/master/LICENSE.md

iCheck
Damir Sultanov, http://fronteed.com/iCheck/
iCheck plugin is released under the MIT License. Feel free to use it in personal and commercial projects.

jQuery Upload File Plugin
Copyright (c) 2013 Ravishanker Kusuma
http://hayageek.com/
MIT License, https://github.com/hayageek/jquery-upload-file/blob/master/MIT-License.txt

jQuery UI Nested Sortable - jQuery Plugin
Copyright (c) 2010-2012 Manuele J Sarfatti
https://github.com/ilikenwf/nestedSortable
Licensed under the MIT License

LinqToSqlExtensions
https://terryaney.wordpress.com/2008/04/14/batch-updates-and-deletes-with-linq-to-sql/
Copyright 2008, 2015 Terry Aney
Licensed under the MIT License https://bitbucket.org/terryaney/linqtosqlextensions/

Silk Icon Set
Mark James, http://www.famfamfam.com/lab/icons/silk/
This work is licensed under a Creative Commons Attribution 2.5 License.
http://creativecommons.org/licenses/by/2.5/

Json.NET
http://www.newtonsoft.com/json
MIT License (MIT), Copyright (c) 2007 James Newton-King
https://raw.githubusercontent.com/JamesNK/Newtonsoft.Json/master/LICENSE.md

Antlr
https://github.com/antlr/antlrcs
BSD License (3-clause), Copyright (c) 2011 Terence Parr
Conversion to C#: Copyright (c) 2011 Sam Harwell, Pixel Mine, Inc.
http://www.antlr3.org/license.html

OWIN
https://github.com/owin-contrib/owin-hosting
OWIN hosting components
Copyright 2012 Louis DeJardin
Copyright 2012 Chris Ross
Apache 2 https://github.com/owin-contrib/owin-hosting/blob/master/LICENSE.txt

Preloaders.net
AJAX Spinners. All animated GIF and APNG images are completely free to use in all projects (web and desktop applications, freeware and commercial projects).
http://preloaders.net/

ajaxload.info
Ajaxload - Ajax loading gif generator. Generated gifs are totally free for use.
http://ajaxload.info/

normalize.css
https://github.com/necolas/normalize.css
MIT License, Copyright (c) Nicolas Gallagher and Jonathan Neal

Base64 encode / decode
https://github.com/client9/stringencoders/tree/master/javascript
Copyright (c) 2010 Nick Galbreath
MIT License https://github.com/client9/stringencoders/blob/master/javascript/base64.js

Tooltipster
http://iamceege.github.io/tooltipster/
The MIT License (MIT) Copyright (c) 2015 Caleb Jacob
https://github.com/iamceege/tooltipster

jQuery MiniColors: A tiny color picker built on jQuery
Licensed under the MIT license - Developed by Cory LaViska for A Beautiful Site, LLC
https://github.com/claviska/jquery-minicolors

jQuery blockUI - jQuery Plugin
Examples at: http://malsup.com/jquery/block/
Copyright (c) 2007-2010 M. Alsup
Dual licensed under the MIT and GPL licenses:
http://opensource.org/licenses/MIT
http://www.gnu.org/licenses/gpl.html

jQuery Form Plugin
Copyright (c) 2014 M. Alsup, http://malsup.com/jquery/form/
Dual licensed under the MIT and GPL licenses. https://github.com/malsup/form#copyright-and-license

SimpleModal - jQuery Plugin
http://www.ericmmartin.com/projects/simplemodal/
Copyright (c) 2010 Eric Martin
Dual licensed under the MIT and GPL licenses

jQuery UI Timepicker (By François Gélinas)
This is a jQuery UI time picker plugin build to match with other official jQuery UI widgets.
Licensed under the same license as jQuery : MIT and GPL licenses
http://fgelinas.com/code/timepicker/

TinyMCE
Copyright (c) Tiny Technologies, Inc. All rights reserved.
Released under LGPL License.
License: http://tinymce.moxiecode.com/license

The MIT License (MIT)

Copyright (c) 2011, 2015 Samantha Copeland, http://www.carrotware.com/
<https://opensource.org/licenses/MIT>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>
<http://www.gnu.org/licenses/gpl-3.0.html> or <https://opensource.org/licenses/GPL-3.0>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

The BSD 3-Clause License

Copyright (c) <YEAR>, <OWNER>
All rights reserved. <https://opensource.org/licenses/BSD-3-Clause>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING

IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Apache License, Version 2.0

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/
<https://opensource.org/licenses/Apache-2.0>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on

behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS