# This article explains how to apply [Timer — Free Responsive Multi Page Personal Bootstrap Template](https://themewagon.com/themes/timer-free-responsive-multi-page-personal-bootstrap-template/) to a Sitefinity project using Feather with a Bootstrap package.

**Project and package setup**

1. Create a Sitefinity project.

2. Install the latest Feather version, following this article: <http://docs.sitefinity.com/feather-get-started> .

3. Make a copy of Bootstrap package folder under *<root>/ResourcePackages* and rename it BootstrapTimer (because the Bootstrap template is called Timer).  
This will automatically create a default page template in Sitefinity under *Design > Page Templates*. The template is based on default.cshtml file located in *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/Layouts*.

4. Copy Timer template front-end assets to *<root>/ResourcePackages/BootstrapTimer/assets/*. The template doesn’t use preprocesors (less, sass, stylus etc), that's why the final files are placed in the "dist" folder, not in the "src" (source) folder.

Create a folder with the name of the template Timer under *<root>/ResourcePackages/BootstrapTimer/assets/dist* and copy all front-end assets there (*<root>/ResourcePackages/BootstrapTimer/assets/dist/Timer/*).

5. Go to *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/* where all widget views are located and move all folders except for Layouts folder from *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/* to *<root>/Mvc/Views/*.

# As per Feather specification, if a folder with the name of a widget (for example News) doesn't exist in BootstrapTimer package under *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/*, Sitefinity will take the view from *<root>/Mvc/Views*.

# This is done to simplify upgrade process because modified files will not be automatically overridden and less manual update will be required. The custom package BootstrapTimer (*<root>/ResourcePackages/BootstrapTimer/Mvc/Views/*) will contain only package specific widget views.

Layouts folder *(<root>/ResourcePackages/BootstrapTimer/Mvc/Views/ Layouts/*) contains the razor view, which is the base for the Timer template page template in Sitefinity under *Design > Page Templates*.

More about priorities can be found here: <http://docs.sitefinity.com/feather-priorities-for-resolving-views>.

6. All grid widgets reside in a *<root>/ResourcePackages/BootstrapTimer/GridSystem/* folder. In order to preserve the default ones and have only custom grid widgets in BootstrapTimer (same as with the views of the widgets), first move all files from *<root>/ResourcePackages/BootstrapTimer/GridSystem/* to *<root>/GridSystem/* (preserving the same hierarchy, as in the package so GridSystem is on the same level as MVC folder). The default grid widgets will be pulled from *<root>/GridSystem/* and the package specific grid widgets will be taken from *<root>/ResourcePackages/BootstrapTimer/GridSystem/*.

Most html pages in the Timer template consist of sections (<section> html tags) therefore create a custom grid widget "*Section.html*" and place it in *<root>/ResourcePackages/BootstrapTimer/GridSystem/*.

In some cases a <div> html tag with CSS class *row* is used to wrap a section. Therefore create a grid widget called *row* (*<root>/ResourcePackages/BootstrapTimer/GridSystem/row.html*).   
Tip: It is important to put "*sf\_colsIn*" class to the <div> html tag in order to create a placeholder in Sitefinity UI.

7. Open *default.cshtml* *(<root>/ResourcePackages/BootstrapTimer/Mvc/Views/Layouts/*) and add all references to \*.css and \*.js files, following this article — <http://docs.sitefinity.com/feather-refer-to-resources-inside-views#take-advantage-of-the-feather-routing-when-registering-resources> and using the @Html.Script and @Html.StyleSheet helper methods.

# 8. Place the favicon.ico file in the root of the project (under *<root>*) and after application restart, the site will start using it.

# **Page templates setup**

# 9. The parts of the Timer template which are used on every page like header and footer are placed in *default.cshtml* under *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/Layouts/* as well as placeholders for the widgets that will be added in Sitefinity UI.

# *default.cshtml* is the base of the default page template in Sitefinity under *Design > Page Templates.*

# 10. Add the reusable parts of Timer template to the default page template in Sitefinity under *Design > Page Templates* to be used by all pages.

Open the default page template (Sitefinity under *Design > Page Templates*) and add *container* grid widget to header and footer placeholders.

The footer consists of Bootstrap columns so put *grid-8+4* grid widget in it. The rest of the footer is html code, which can be placed in a *Content block* widgets (it's easier to place the HTML directly in the HTML view of the *Content block* widget).

# 11. Next thing to do is choose which Bootstrap navigation to use. The most suitable one turns out to be *NavigationView.Horizontal.cshtml* (from Bootstrap package).

# Currently the widgets get their views from *<root>/Mvc/Views*, where all default Bootstrap widget templates have been moved. In order to modify *NavigationView.Horizontal.cshtml* make a new folder inside BootstrapTimer package under *<root>/ResourcePackages/BootstrapTimer/MVC/Views* and call it *Navigation* (*<root>/ResourcePackages/BootstrapTimer/MVC/Views/ Navigation/*). Then copy the default *NavigationView.Horizontal.cshtml* view to this folder and modify its html as per Timer template navigation template. Tip: Make sure to remove the jQuery reference from the Navigation template since it is already loaded in the Layout file.

12. All pages contain section elements (<section>) which a custom grid widget has already been created for in step 6. Most of the sections in Timer template have IDs assigned to them. These IDs are used in the CSS to define CSS selectors. Because multiple instances of the same grid widget are used on the same page there cannot be unique ID attributes for every dragged grid widget. This is why the IDs from Timer template have to be replaced with CSS classes. “*sf-*“ prefix is prepended to each of these CSS classes so they can be distinguished from the native Timer template CSS styles. After that these classes can be easily applied to certain grid widgets dragged on the page through Sitefinity UI to alter their looks. For example, if Timer template contains a section element with id="hero-area" (<section id=”hero-area”>) the ID will be replaced with CSS class "sf-hero-area" (<section class=”sf-hero-area”>).

# 13. There are repeated section elements among the pages. The repeated sections have to be placed in the templates in order to be used by all pages.

# All pages have a "*call-to-action*" section so a "*call-to-action*" section is added to the default page template in Sitefinity under *Design > Page Templates*.

# 7 pages have a "*global-page-header*" section therefore another template based on the default one is created. It is called *default-with-global-header* and the "*global-page-header*" section is placed in it. The "*global-page-header*" consists of a wrapping <div> with CSS class, a <h2> heading and a breadcrumb. Drag a *grid-12* grid widget and set the CSS class needed in the widget designer in its Classes property. For the heading drop a *Content block* widget in the *grid-12* grid widget and make it editable on pages from its actions menu. In the template the common markup is specified and on each page the particular *Content block* is edited to write the particular page name. For the breadcrumb use a Feather breadcrumb with a different template as done for other widgets before.

# Now that the common elements are all setup let’s proceed with individual pages.

**Page creation and setup**

14. **Home**

a) The home page contains a header with information about the designer. To create this header use a *Content block* widget and paste the required html in it.

There's a button “View works” which scrolls to the “Latest Works” section. In Timer template this is done with a javascript method scrollingTo() which takes the value of the data-section attribute and tries to get an element which matches the selector. In the original template this selector is the ID of the element. Since Sitefinity *grid widgets* don't have IDs and all IDs have been replaced with CSS classes, use .sf-works for data-section attribute of the “View works” button.

b) Next comes the “About me” information, which consists of some text and an image. To accomplish this part of the design drop a *section* grid widget and in the widget designer set *sf-about* in the *section* property. After that place a *container* grid widget and then place another *grid-6+6* grid widget in it. The two columns of *grid-6+6* grid widget have *col-sm-6* CSS classes so click *Edit*  and add them to *Column 1* and *Column 2* properties. The layout is ready so drop two *Content blocks* widgets in each column. The first *Content block* widget contains text and simple markup with classes. The one in the second column contains an image and some markup. Add the image from the *Insert image* tool of the *Content block* widget.

TIP: *Image* widget is not used in this particular case, because some additional markup around the image is needed and the easiest way to add it is directly in the *Content block* widget.

c) For the next section add a *section* grid widget and set the CSS class needed as was done in the previous step. After that drag a *container* grid widget in the *section* grid widget. This section consists of artworks with title, image and details information. In order to add new items and modify existing ones easily create a dynamic module called *Artwork*, with content items "*works*" and with the following fields:

- Title - Type: Short text

- Information - Type: Long text

- Description - Type: Short text

- Image - Related media (images, videos, files)

After creating the module create a few module items in Sitefinity under *Content > Artwork*. Then modify the List and Details templates of the module, to suit the design. For every custom module List and Detail widget templates are created in Sitefinity under *Design > Widget Templates*. The name of the templates are *List.work* (*List.[name-of-content-item]*) and *Detail.work* (*Detail.[name-of-content-item]*). To modify the default List or Detail view or to create new ones, create a folder with the name of the content item *work* (not the content type!) in *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/*  and inside *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/work* add the templates following the naming convention.

To modify the List view add a cshtml file *List.work.cshtml* (because the content items name is *work*) in *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/work*. To create another list or detail template create a .cshtml file and follow the naming convention “List.” or “Detail.” + *[name-of-the-template]* + “.cshtml”. *[name-of-the-template]* will be shown in the UI.

Create *List.WorkList.cshtml* in *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/work* and modify its mark-up. After that drop a *works* widget to the page, open its designer, choose *WorkList* list template and configure it to open single items on *Artworks details* page.

d) For the last section add a *section* grid widget and a *container* grid widget in it and setup the CSS classes. This section consists of two rows with 3 column elements on each row. The elements have *col-md-4 col-lg-4 col-xs-12* classes assigned to them. Again, use the default grid widgets that come with *“-md”* classes and manually add the other resolution classes through the grid widget designer. The rest is achieved with *Content block* widgets in each column element.

15. **Artwork details**

# Create the page and drag a *works* widget. Create a new Detail template like done for the List view and configure the *works* widget to use it.

# 16. **About**

a) Edit the *Content block* widget in the “global-page-header” section to write the page title.

b) Setup the next section with a *section* grid widget and another *container* grid widget in it and configure them as needed. The section has two columns so drag a *grid-6+6* grid widget in the *container* grid widget. The first column contains an image. Use the *Image* widget to upload the image to Sitefinity libraries and add it to the page. The image has an *img-responsive* class. Add the class through the *Image* widget designer.

Tip: All static content (for example text, that shouldn't be part of any module) can be placed directly in a *Content block* widget.

c) Setup the next section with a *section* grid widget and set the CSS classes needed (*about-feature clearfix*). After that drop a Content block widget and add the html for this section through the HTML view if the *Content block* widget editor.

# d) Setup the next section with a *section* grid widget and another *container* grid widget in it and configure them as needed. "Meet the team" section can be handled with a dynamic module. Create a module Team with content items "*teammembers*" and fields:

# - Title (Name) - Type: Short text

# - JobTitle (Job Title) - Type: Short text

# - ShortInfo (Short info) - Type: Long text

# - Facebook - Type: Short text

# - Twitter - Type: Short text

# - LinkedIn - Type: Short text

# - Google (Google+) - Type: Short text

After the module is created create a List template, change its markup and configure it as was done for the Artworks module.

# 17. **Service**

a) “What we love to” and “Our happy clients” sections are similar to section that have already been setup on other pages.

# b) There is another list with artworks on this page. This list contains only featured works. Go to Sitefinity *Content > Artwork* and mark all featured works with tag *Featured*. Drop a *works* widget on the page and narrow the items to be displayed to the ones tagged with *Featured*. In the *List settings* tab set 4 items to be displayed and configure the list template. In *Single item settings* tab configure single items to be opened in *Arwork details* page.

# 18. **Gallery**

# a) This page contains an image gallery with artworks. So create a custom *Gallery* template *List.WorksGallery.cshtml* in *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/ImageGallery* and modify its markup.

# Create a new *Artworks* album in Sitefinity *Content > Images* and configure the *Image gallery* widget to display images from this selected gallery only.

# 19. **Blog full**

# In the *Blog posts* module add a custom field from type *Related images* in *Custom Fields for posts*. This field will be used for the header image. Some blog posts might have related Artworks therefore create another custom field ArtWork from type Related data and data type works. The meta-data business and people will be tags.

a) Create *Blog full* page and drag a *Blog posts* widget.

b) Create a List template under *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/BlogPost/* and modify its markup.

c) Configure the *Blog posts* widget to use BlogPostList template and to open single items in *Blog details* page.

# 20. **Blog details**

a) Drag a *Blog posts* widget.

b) Create a Detail template of the *Blog posts* widget under *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/BlogPost/* and added a template for the related Artworks in it.

Go to Sitefinity *Content > Artwork > Pages where items like this are published* and move *Artwork details* page on top. After that related Artworks of a blog post will be opened in *Artwork details* page.

c) Each blog post detail template contains a social share option. Feather comes with a *Social share* widget that can be enabled through the designer of the widget. To do so open *Edit > Advanced > Modal* and set *EnableSocialShare* to *True*. To change the template of the *Social share* widget create *SocialShare* folder under *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/* and copy *SocialShare.cshtml* from *<root>/Mvc/Views/SocialShare*.   
TIPS: To enable more social sharing options go to Sitefinity *Administration > Settings > Social sharing* and check the ones you need.

d) To enable comments for blogs go to *Administration > CommentsModule > Commentable* *types*. Then create custom comments templates in *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/Comments*.   
Tip: When modifying the template make sure to keep the Sitefinity additions like sorting and comments count. Also, you can enable Captcha from *Administration > CommentsModule > UseSpamProtectionImage.*

# 21. **Blog left sidebar**

# The specific sections for this page are search section, categories section and recent posts section.

# a) For search section use a *Search box* widget and create a new search index for blog posts in Sitefinity *Administration > Search Indexes*. Create a new *Search box* widget template in *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/SearchBox* and change the markup as needed. Configure the *Search box* widget to use the created search index and the SearchBox widget template as well as to display results on *Blog left search* page.

# b) For the categories section use a *Categories* widget and create a custom *CategoriesSidebarList* template (as done with other widgets before)

# c) For recent blog posts section use a *Blog posts* widget and create a new template *BlogPostSimple*. Configure the widget to show only 3 items in total, to sort posts so that last published are on top and to open single item in *Blog details* page.

d) Drag a *Blog posts* widget and configure it to open single items in *Blog details* page as done for the *Blog full* page.

22. **Blog left search results**

a) Create this page as a duplicate of *Blog left sidebar* but change the *Blog posts* widget with a *Search results* widget.

b) Go to Sitefinity *Content > Blogs* and click *Actions > Title and Properties* for *Blog* item. Set *Default page* property to *Blog > Blog details.* After that blog posts from the *Search result* widget will be opened in *Blog details* page like in the other cases.

23. **Contacts**

# The interesting section on this page is the Contact form. The other sections are realized with *section* + *container* grid widgets and *Content block* widgets.

# a) Create a *Contact with me* form *ContactWithMe* in Sitefinity *Content > Forms*.

b) To change the markup of the fields of the form create ParagraphTextField, TextField and SubmitButton folders in *<root>/ResourcePackages/BootstrapTimer/Mvc/Views/* and copy the corresponding templates from *<root>/Mvc/Views/.* After that change the markup as needed.

c) Customize the message which is displayed after the form is submitted in the *Form* widget designer in *Settings* tab.