Welcome to uBootstrap, a Starter Kit built using Html5 + CC3 + .less. Enjoy!

uBootstrap it's a Starter Kit for Umbraco based onBootstrap from Twitter and will help you to quickly setup a base multilingual website using html5, css3, .less, html5 boilerplate and modernizr.

Learn about the functionalities available in this package

This starter kit is based on Bootstrap from Twitter and uses html5, css3, and .less. It is a multilingual package (English and Spanish) and uses the built-in Umbraco dictionary to localize labels used within the site

Other functionalities you might want to know

- Atom feed for news list
- Default sitemap xml for search engines
- Default robots.txt
- Because we are humans
- Pretty icons
- Resized images with ImageGen
- Built-in Examine search
- Friendly not found page

Less is more

Write regular CSS with your .NET apps, then add a few variables, mixins and nested rules.

This package already contains the dotless.Core.dll and has set the required handler in the web.config. You can also manage the .less files form Settings > Less Files.

List of .less files

- bootstrap.less
- forms.less
- mixins.less
- patterns.less
- reset.less
- scaffolding.less
- tables.less (not used by default, look at theme.less)
- theme.less
- type.less
- variables.less

Don't like the blue theme. How can I change it?

Open the variables.less file and change the **@linkColor**. You can also change the **@basefont** and **@baseline**

```
/* Accent Colors */

@blue:              #049CDB;

@blueDark:          #0064CD;

@green:             #46a546;

@red:               #9d261d;

@yellow:            #ffc40d;

@orange:            #f89406;

@pink:              #c3325f;

@purple:            #7a43b6;



/* Links */

@linkColor:         @blue;

@linkColorHover:    darken(@linkColor, 10);



/* Baseline grid */

@basefont:          14px;

@baseline:          20px;
```

I already change the @linkColor but I can't see the changes. Why?

All stylesheets and javascripts are compressed using ClientDependency, so simply change the version number in config/ClientDependency.config

Before installing uBootstrap

In order for this package to work properly, you need to install the famfamfam icons and ImageGen

- Pretty icons
- ImageGen

Please notice that at the moment this package has only been tested on **Umbraco Version 4.7.1** running on**IIS7.x** on **Integrated Mode**

Testing locally

In order for both English and Spanish websites to run, you must add bootstrap.local and es.bootstrap.local to the host header on IIS and in you host file, You can also visit this demo site if you want to have a look.

Stylesheets and javascripts are compressed with ClientDependency

The stylesheets and scripts are compressed by default using the built-in ClientDependency framework.

Examples

There are macro named CdLoader which will help us to load the css and js wherever we want. For example, see the code in the Public.master template:

```
<umbraco:Macro Alias="CdLoader" CssPaths="/less/theme.less" runat="server" />
```

Or the scripts loaded at the bottom of the page in Public_Newslist.master

```
<umbraco:Macro Alias="CdLoader" JsPaths="analytics.js|bootstrap-
twipsy.js|twitter.js|facebook.js|site.js" IncludeDefaultPaths="true"
runat="server" />
```

No runat server in the master template. Why?

One of the things you'll notice is that the Public.master template doesn't have a *head runat="server"* or *form runat="server"*, therefore, canvas is not available.

Why?

I never liked the idea of putting the whole body html in a big form tag. I think it's semantically incorrect and I also dislike the what asp.net makes by adding a viewstate, hidden inputs, naming controls, etc.

So, how do you handle a form?

You'll notice the ContactForm.cshtml uses the AntiforgeryToken located in the MainHelper class

```
public static HtmlString GetAntiForgeryHtml(this HttpContextBase context)

{

    return AntiForgery.GetHtml(context, String.Empty, String.Empty,
String.Empty);

}



public static bool IsValidAntiForgery(this HttpContextBase context)

{

    try

    {

        AntiForgery.Validate(context, String.Empty);

        return true;

    }

    catch

    {

        return false;

    }

}
```

This way, we prevent Cross-Site Request Forgery (CSRF). We also use the ModelState to validate the required fields, and dynamically map the from post using this helper

```
public static dynamic ToDynamic(this NameValueCollection collection)

{
```

```
    return new DynamicRequest(collection);

}



private class DynamicRequest : DynamicObject

{

    private readonly NameValueCollection nvc;



    public DynamicRequest(NameValueCollection collection)

    {

        nvc = collection;

    }



    public override bool TryGetMember(GetMemberBinder binder, out object
result)

    {

        result = nvc[binder.Name];

        return true;

    }

}
```

So as you can see, we can create a whole contact form without wrapping it in a *runat="server"*