# Form Builder

## Overview

This component provides the functionality for form auto-generating according to the View Model declaration. By specifying attributes for the properties of the view model, you can tell the Form Builder what control type to render.

This component uses Twitter Bootstrap html layout and classes.

## ViewModel Attributes

The main attribute that indicates that the property needs to be rendered as a editable control is EditControl. It has following parameters:

* ElementType – an enum specifying the control type to be rendered. Potential values are:
  + Text (for a simple text box)
  + Hidden (for a hidden field)
  + TextArea (for a text area)
  + Password (for a password field, like for a registration form)
  + WholeNumber (in read-only mode, the data is formatted as a whole number)
  + FloatingPointNumber (in read-only mode, the data is formatted as a floating-point number)
  + DateTime (in read-only mode, the data is formatted as a DateTime, in edit mode it is rendered as a text box with class „datepicker”- you can write your custom javascript to add calendar controls to all inputs with class „datepicker”)
  + Time (in read-only mode, the data is formatted as a Time)
  + CheckBox (for a check box)
  + Enum (for a drop down with an enum as a selection type)
  + List
  + ListBox
  + Guid
* Cols – if the ElementType is TextArea, then this property is translated into „cols” attribute of text area (default = 60)
* Rows – if the ElementType is TextArea, then this property is translated into „rows” attribute of text area (default = 6)
* MaxLength – if the ElementType is Text, then this property indicates the maximal length of the text box.

Foundation FormBuilder also supports native .NET attributes like [Required] and [Display(Name = "Name")]. The Required attribute will instruct the FormBuilder to add client-side validation attributes (if the jQuery Unobtrusive validation is enabled by adding a script reference on the UI level). The Display attribute with its parameters allow you to specify some appearance parameters, like order in which the control will appear, group name, the control label, etc..

So, to define a control that will appear as a TextBox, your view model property could look like this:

[Display(GroupName = "Contact Info", Order = 2, Name = "Email")]

[EditControl(ElementType = ElementType.Text)]

[Required]

public string Email { get; set; }

## UI (View) Level

The point of this component is to simplify the creation of forms so that for the most part you won’t need to explicitly implement your forms, but instead use the following structure:

@using (var form = Html.Bootstrap().BeginForm(BootstrapFormType.Horizontal, new {@class="well", id="autoForm" }))

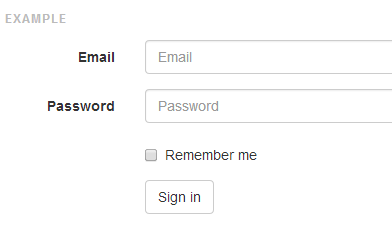
{

@Html.DynamicForm()

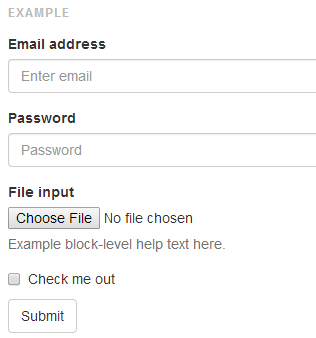
}

This will take your view model, analyze its properties and their attributes and accordingly generate a form and edit controls. By specifying BootstrapFormType Enum value, you instruct the Foundation to apply specific styling rules (class) to the generated form. There are three possible values:

1. **BootstrapFormType.Horizontal:**



1. **BootstrapFormType.Vertical:** (this is the default style)



1. **BootstrapFormType.Inline:**

