Building Strongly-typed Forms with AngularJS

Matt Honeycutt

@matthoneycutt | http://trycatchfail.com/strongly-typed-ajs

Before

```
<div class="form-group has-feedback">
    <label class="control-label" for="Name">Name</label>
    <input required ng-model="vm.customer.name"
           class="form-control" name="Name" type="text" placeholder="Full name (ex: John Smith)...">
</div>

<div class="form-group has-feedback">
    <label class="control-label" for="WorkEmail">Work Email</label>
    <input required ng-model="vm.customer.workEmail"
           class="form-control" name="WorkEmail" type="email" placeholder="user@domain.com...">
</div>

<div class="form-group has-feedback">
    <label class="control-label" for="HomeEmail">Home Email</label>
    <input ng-model="vm.customer.homeEmail"
           class="form-control" name="HomeEmail" type="email" placeholder="user@domain.com...">
</div>
```

Before

```
<div class="form-group has-feedback">
    <label class="control-label" for="Name">Name</label>
    <input required ng-model="vm.customer.name"
           class="form-control" name="Name" type="text" placeholder="Full name (ex: John Smith)...">
</div>
<div class="form-group has-feedback">
    <label class="control-label" for="WorkEmail">Work Email</label>
    <input required ng-model="vm.customer.workEmail"
           class="form-control" name="WorkEmail" type="email" placeholder="user@domain.com...">
</div>
<div class="form-group has-feedback">
    <label class="control-label" for="HomeEmail">Home Email</label>
    <input ng-model="vm.customer.homeEmail"
           class="form-control" name="HomeEmail" type="email" placeholder="user@domain.com...">
</div>
```

After

```
@customer.FormGroupFor(x => x.Name)

@customer.FormGroupFor(x => x.WorkEmail)

@customer.FormGroupFor(x => x.HomeEmail)

@customer.FormGroupFor(x => x.WorkPhone)

@customer.FormGroupFor(x => x.HomePhone)

@customer.FormGroupFor(x => x.WorkAddress)

@customer.FormGroupFor(x => x.HomeAddress)
```

Benefits

- ✓ Way (WAY) less markup!
- ✓ Consistency
- ✓ Encapsulation makes changes easier
- ✓ Solution is still flexible!

Overview

Updating our solution

Building a 'form-group' helper

Handling different data types

Magically create better labels

Supporting placeholders

Applying BetterJson and Razor Templates

The FormGroupFor Helper

```
<div class="form-group has-feedback">
    <label class="control-label" for="Name">Name</label>
    <input required ng-model="vm.customer.name"
           class="form-control" name="Name" type="text"
           placeholder="Full name (ex: John Smith)...">
</div>
```

The FormGroupFor Helper

```
<div class="form-group has-feedback"
     ng-class="{
        'has-success': (vm.form.Name.$touched || vm.form.$submitted) && vm.form.Name.$valid,
        'has-error': (vm.form.Name.$touched || vm.form.$submitted) && vm.form.Name.$invalid
        }">
    <label class="control-label" for="Name">Name</label>
    <input required ng-model="vm.customer.name"
           class="form-control" name="Name" type="text" placeholder="Full name (ex: John Smith)...">
    <span ng-show="(vm.form.Name.$touched || vm.form.$submitted) && vm.form.Name.$valid"
          class="fa fa-2x fa-check-square form-control-feedback" aria-hidden="true"></span>
    <span ng-show="(vm.form.Name.$touched || vm.form.$submitted) && vm.form.Name.$invalid"
          class="fa fa-2x fa-exclamation-triangle form-control-feedback" aria-hidden="true"></span>
</div>
```

The FormGroupFor Helper

```
<div class="form-group has-feedback"
     ng-class="{
        'has-success': (vm.form.Name.$touched || vm.form.$submitted) && vm.form.Name.$valid,
        'has-error': (vm.form.Name.$touched || vm.form.$submitted) && vm.form.Name.$invalid
        }">
    <label class="control-label" for="Name">Name</label>
    <input required ng-model="vm.customer.name"
           class="form-control" name="Name" type="text" placeholder="Full name (ex: John Smith)...">
    <span ng-show="(vm.form.Name.$touched || vm.form.$submitted) && vm.form.Name.$valid"
          class="fa fa-2x fa-check-square form-control-feedback" aria-hidden="true"></span>
    <span ng-show="(vm.form.Name.$touched || vm.form.$submitted) && vm.form.Name.$invalid"
          class="fa fa-2x fa-exclamation-triangle form-control-feedback" aria-hidden="true"></span>
</div>

<div class="form-group has-feedback"
     ng-class="{
        'has-success': (vm.form.WorkEmail.$touched || vm.form.$submitted) && vm.form.WorkEmail.$valid,
        'has-error': (vm.form.WorkEmail.$touched || vm.form.$submitted) && vm.form.WorkEmail.$invalid
        }">
    <label class="control-label" for="WorkEmail">Work Email</label>
    <input required ng-model="vm.customer.workEmail"
           class="form-control" name="WorkEmail" type="email" placeholder="user@domain.com...">
    <span ng-show="(vm.form.WorkEmail.$touched || vm.form.$submitted) && vm.form.WorkEmail.$valid"
          class="fa fa-2x fa-check-square form-control-feedback" aria-hidden="true"></span>
    <span ng-show="(vm.form.WorkEmail.$touched || vm.form.$submitted) && vm.form.WorkEmail.$invalid"
          class="fa fa-2x fa-exclamation-triangle form-control-feedback" aria-hidden="true"></span>
</div>
```

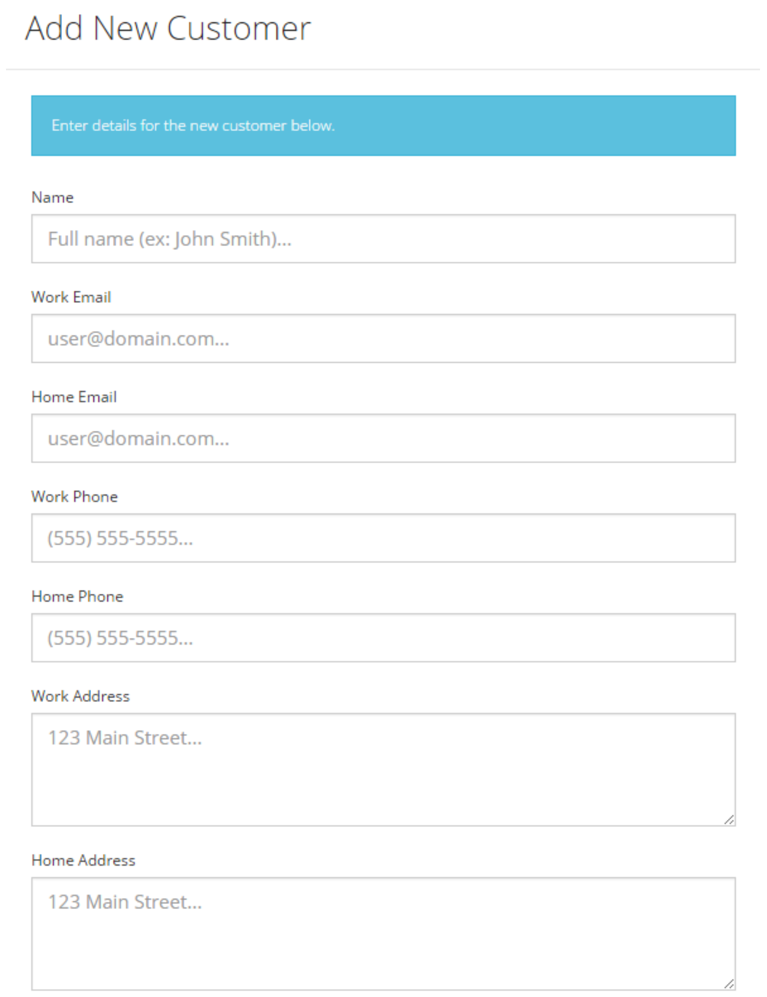
The FormGroupFor Helper

```
<div class="form-group has-feedback"
     ng-class="{
        'has-success': (vm.form.Name.$touched || vm.form.$submitted) && vm.form.Name.$valid,
        'has-error': (vm.form.Name.$touched || vm.form.$submitted) && vm.form.Name.$invalid
        }">
    <label class="control-label" for="Name">Name</label>
    <input required ng- @customer.FormGroupFor(x => x.Name)
          class="form-                                               ame (ex: John Smith)...">
    <span ng-show="(vm.form.Name.$touched || vm.form.$submitted) && vm.form.Name.$valid"
          class="fa fa-2x fa-check-square form-control-feedback" aria-hidden="true"></span>
    <span ng-show="(vm.form.Name.$touched || vm.form.$submitted) && vm.form.Name.$invalid"
          class="fa fa-2x fa-exclamation-triangle form-control-feedback" aria-hidden="true"></span>
</div>

<div class="form-group has-feedback"
     ng-class="{
        'has-success': (vm.form.WorkEmail.$touched || vm.form.$submitted) && vm.form.WorkEmail.$valid,
        'has-error': (vm.form.WorkEmail.$touched || vm.form.$submitted) && vm.form.WorkEmail.$invalid
        }">
    <label class="contr
                        @customer.FormGroupFor(x => x.WorkEmail)
    <input required ng-
          class="form-control" name="WorkEmail" type="email" placeholder="user@domain.com...">
    <span ng-show="(vm.form.WorkEmail.$touched || vm.form.$submitted) && vm.form.WorkEmail.$valid"
          class="fa fa-2x fa-check-square form-control-feedback" aria-hidden="true"></span>
    <span ng-show="(vm.form.WorkEmail.$touched || vm.form.$submitted) && vm.form.WorkEmail.$invalid"
          class="fa fa-2x fa-exclamation-triangle form-control-feedback" aria-hidden="true"></span>
</div>
```

Building the FormGroupFor Helper

Dealing with Different Types of Data

Before

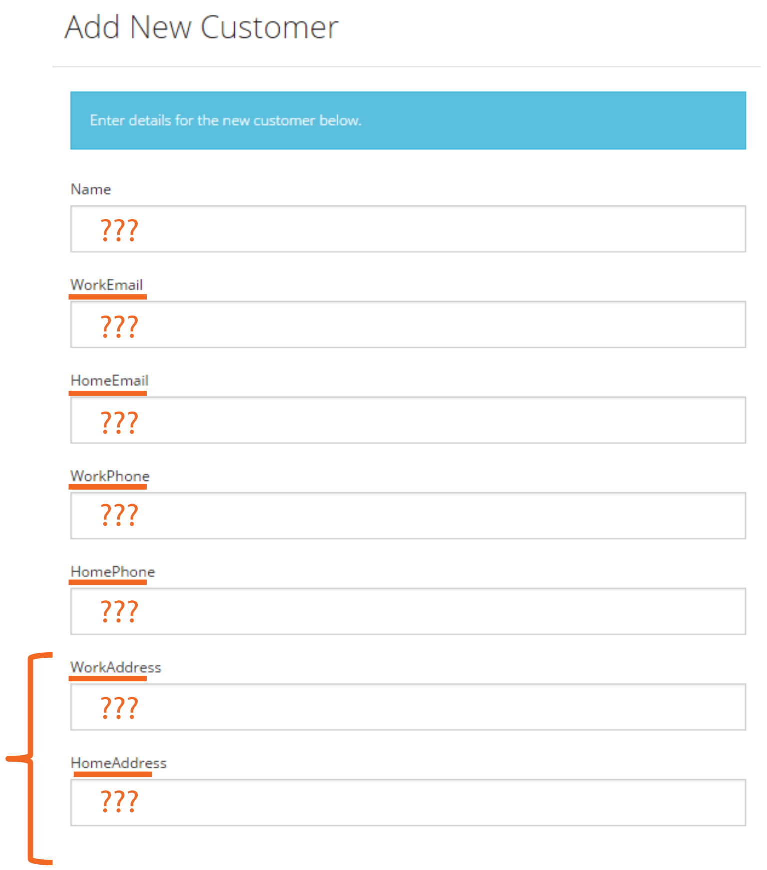


Add New Customer

Enter details for the new customer below.

Name	Full name (ex: John Smith)...
Work Email	user@domain.com...
Home Email	user@domain.com...
Work Phone	(555) 555-5555...
Home Phone	(555) 555-5555...
Work Address	123 Main Street...
Home Address	123 Main Street...

After

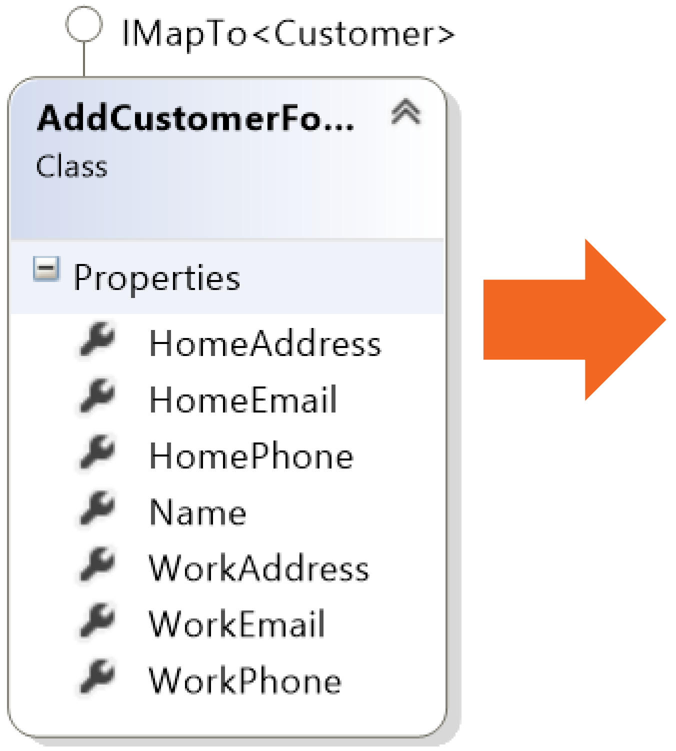


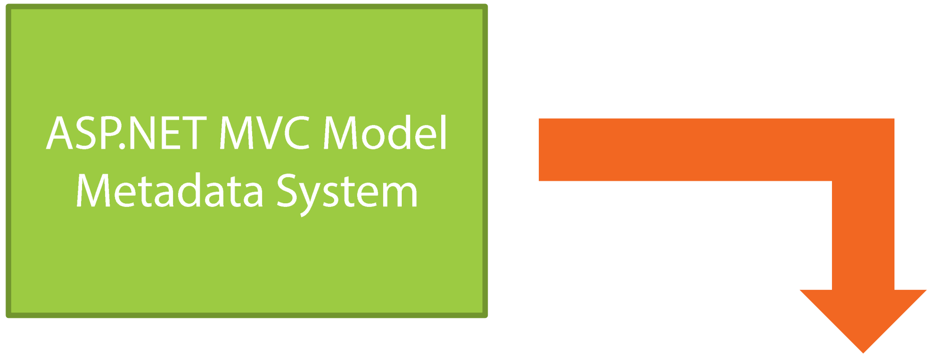
Add New Customer

Enter details for the new customer below.

Name	???
WorkEmail	???
HomeEmail	???
WorkPhone	???
HomePhone	???
WorkAddress	???
HomeAddress	???

Model Metadata





AddCustomerForm's Model Metadata

Name	Display Name	Data Type	Required?	…
Name	Full Name	String	Y	…
HomeAddress		MultilineText	N	…
…	…	…	…	…

Possible Uses

Custom conventions

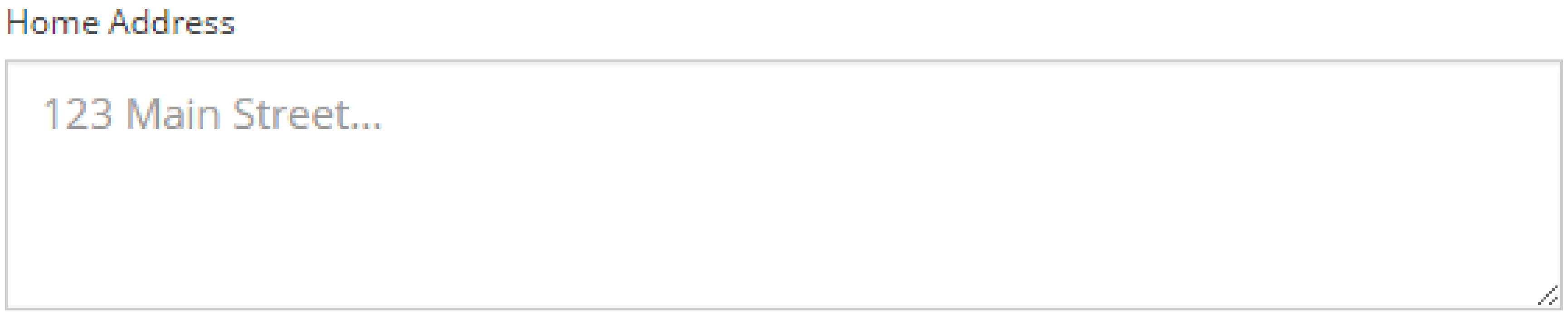
Better defaults

Additional data annotations

Validation

The DataType Data Annotation

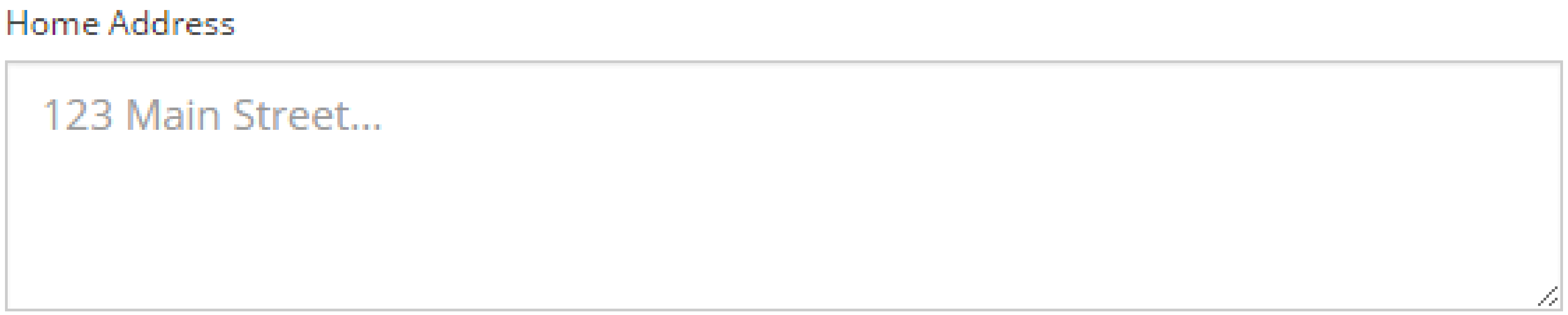
```
public string HomeAddress { get; set; }
```


Home Address

123 Main Street...

The DataType Data Annotation

```
[DataType(DataType.MultilineText)]
public string HomeAddress { get; set; }
```

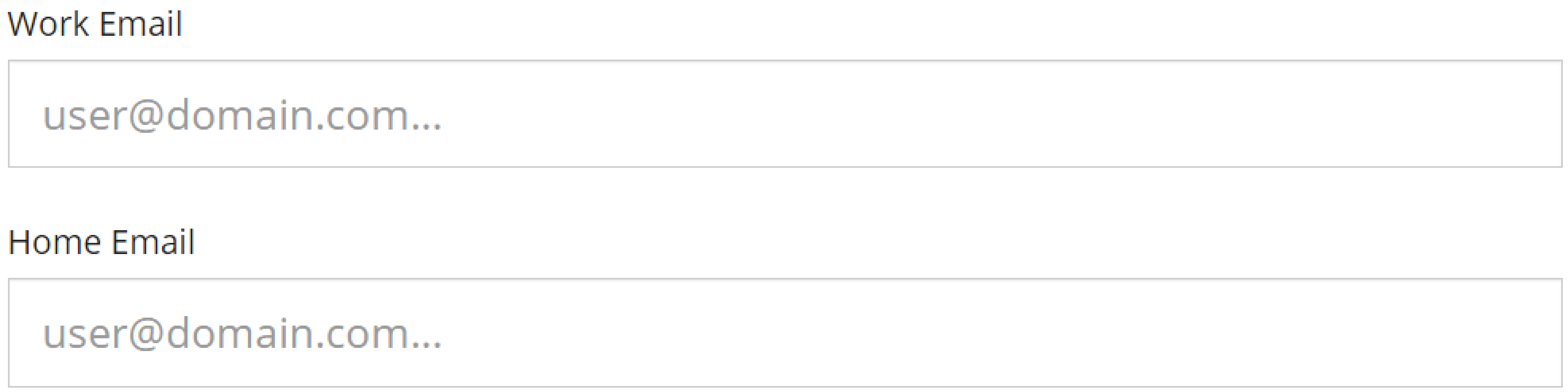

Home Address

123 Main Street...

Adding Support for MultilineText Properties

Generating Better Labels

Before

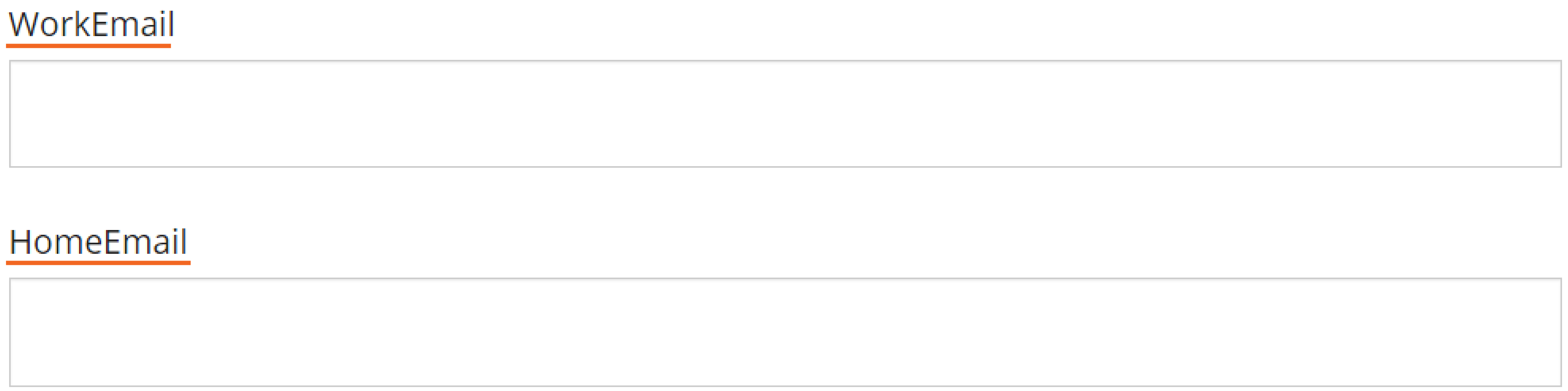

Work Email

user@domain.com...

Home Email

user@domain.com...

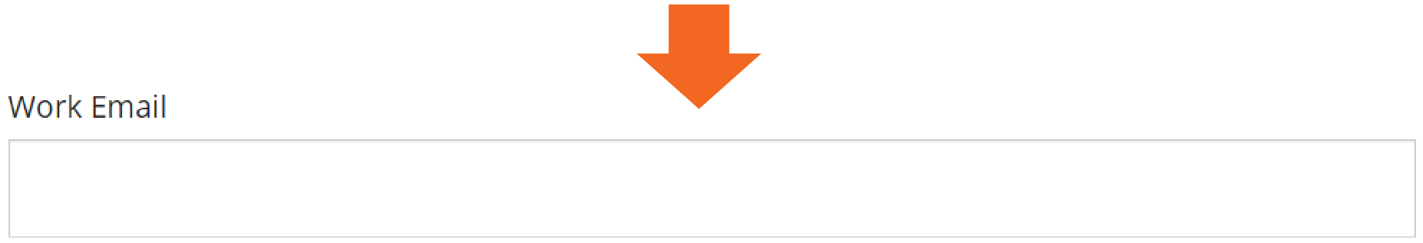
After

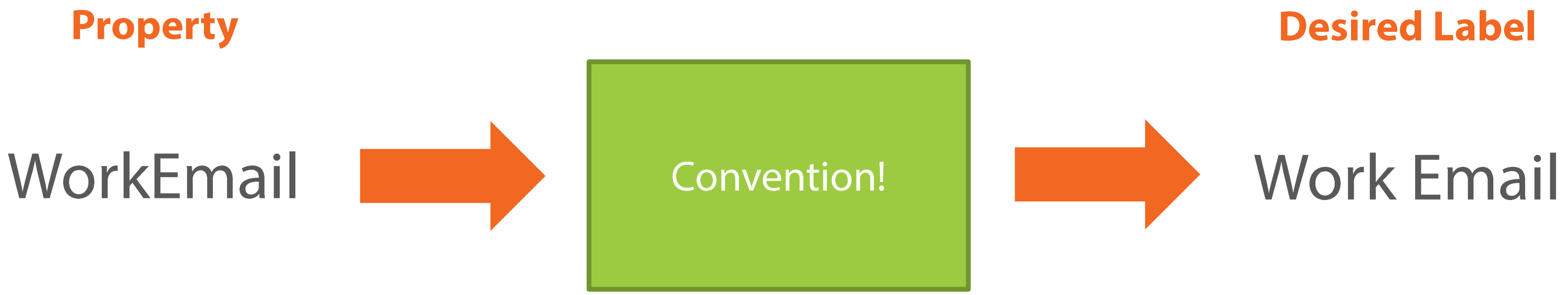

WorkEmail

HomeEmail

Generating Better Labels

```
@customer.FormGroupFor(x => x.WorkEmail, "Work Email")
```





Humanizer

```
PM> Install-Package Humanizer
```

Truncate text:

```
"Long text to truncate".Truncate(10) => "Long text…"
```

Humanize enums:

```
EnumUnderTest.MemberName.Humanize() => "Member name"
```

Date and times:

```
DateTimeOffset.AddHours(1).Humanize() => "an hour from now"
```

Humanizer

```
PM> Install-Package Humanizer
```

Humanize camel case:

```
“camelCaseInputStringIsTurnedIntoSentence".Humanize(
) => “Camel case input string is turned into
sentence"
```

Humanize Pascal case:

```
"PascalCaseInputStringIsTurnedIntoSentence".Humanize() =>
"Pascal case input string is turned into sentence"
```

Humanizer

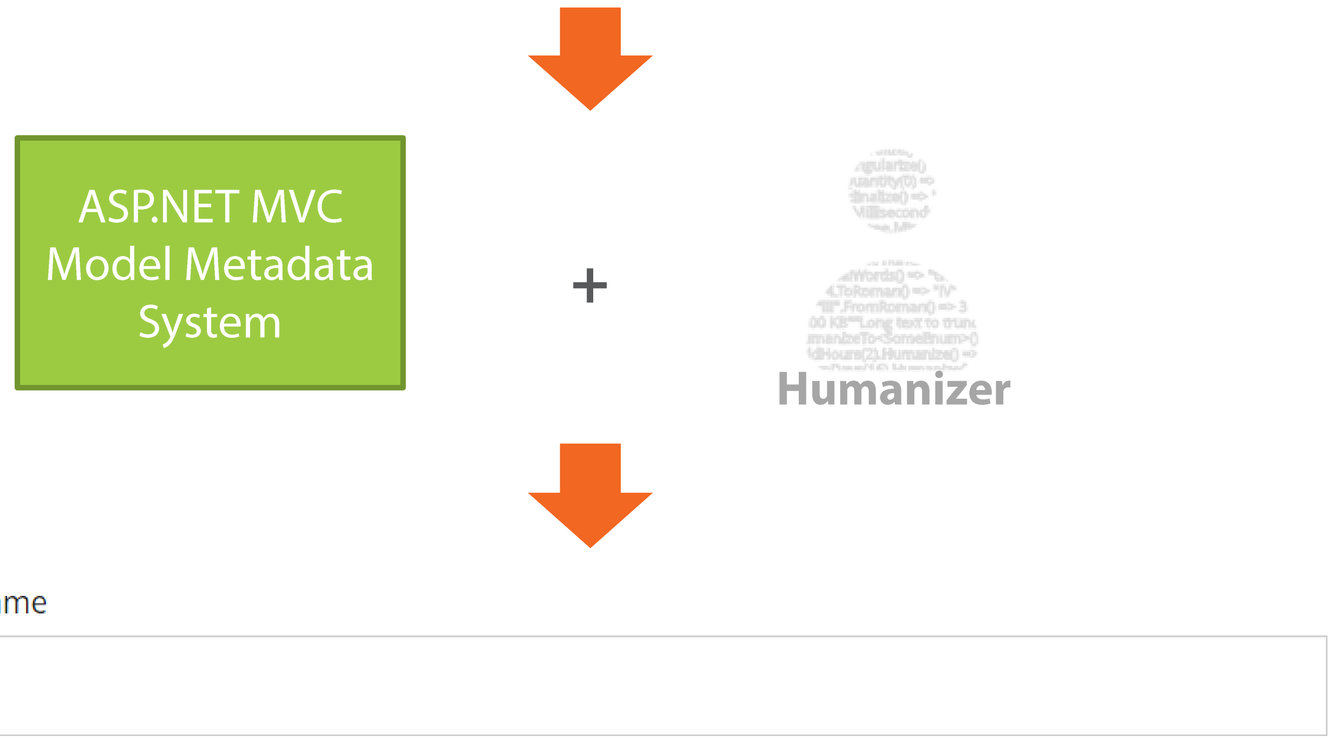
```
@customer.FormGroupFor(x => x.WorkEmail)
```

Humanizer

Work Email

Humanizer

```
[Display(Name = "Full Name")]
public string Name { get; set; }
@customer.FormGroupFor(x => x.Name)
```



Adding Humanizer to Our FormGroup Helper

Adding Placeholders to Our Forms

Before

Work Email

user@domain.com...

Home Email

user@domain.com...

After

Work Email

???

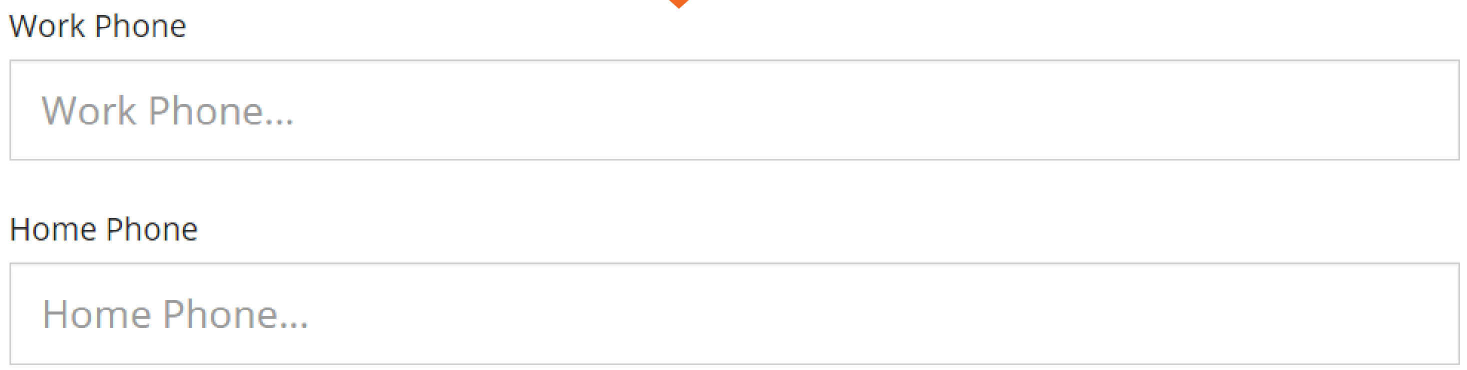
Home Email

???

Reasonable Defaults

```
@customer.FormGroupFor(x => x.WorkPhone)

@customer.FormGroupFor(x => x.HomePhone)
```

Humanizer



Work Phone

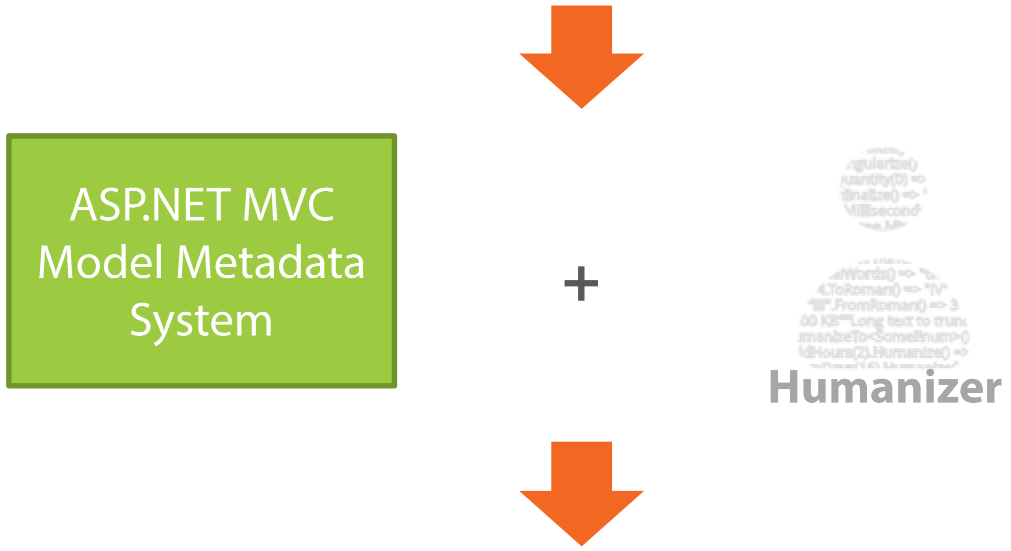
Work Phone...

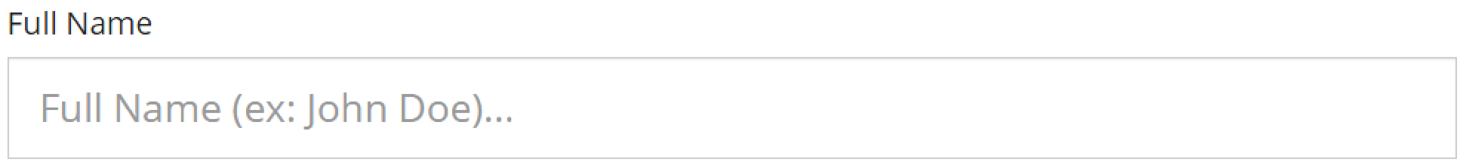
Home Phone

Home Phone...

Reasonable Defaults

```
[Display(Name = "Full Name",
    Prompt = "Full Name (ex: John Doe)...")]
@customer.FormGroupFor(x => x.Name)
```




Full Name

Full Name (ex: John Doe)...

Adding Support for Placeholders

What We've Accomplished

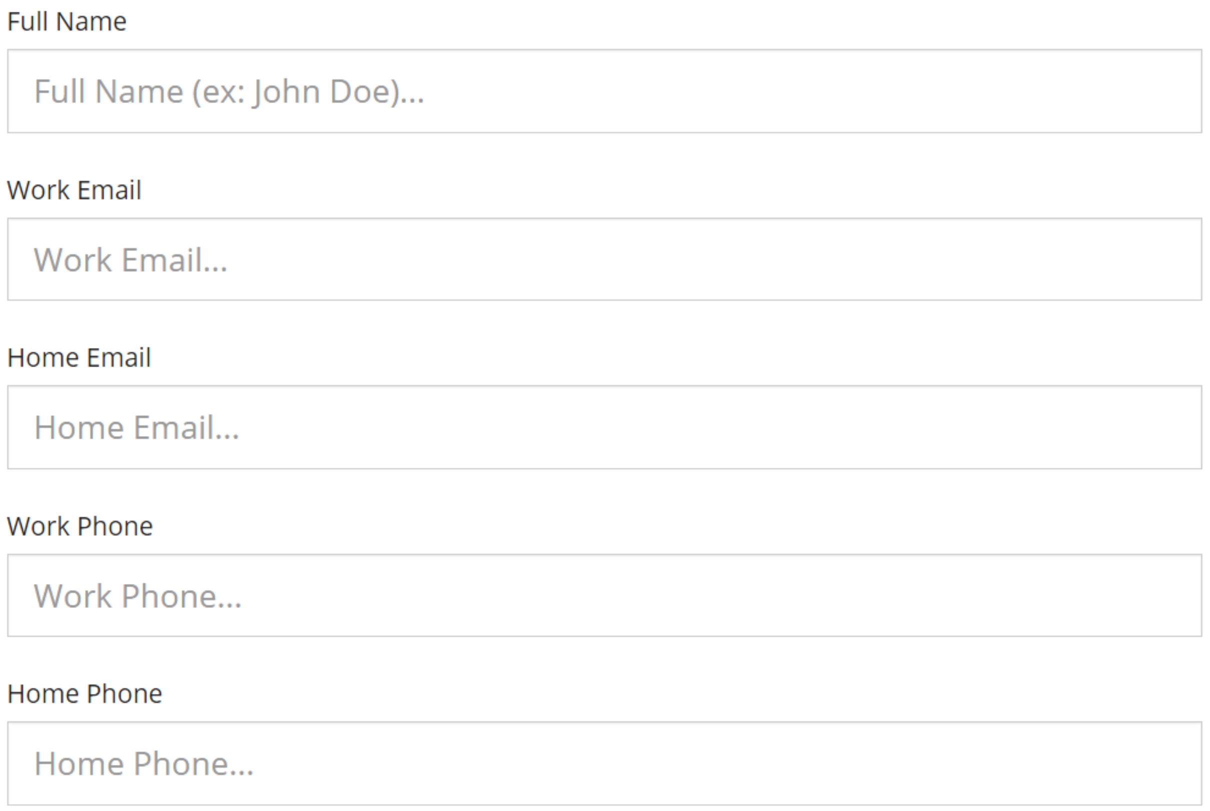
```
<!-- More above… -->
<div class="form-group has-feedback">
    <label class="control-label" for="Name">Name</label>
    <input required ng-model="vm.customer.name"
           class="form-control" name="Name" type="text" placeholder="Full name (ex: John Smith)...">
</div>

<div class="form-group has-feedback">
    <label class="control-label" for="WorkEmail">Work Email</label>
    <input required ng-model="vm.customer.workEmail"
           class="form-control" name="WorkEmail" type="email" placeholder="user@domain.com...">
</div>

<div class="form-group has-feedback">
    <label class="control-label" for="HomeEmail">Home Email</label>
    <input ng-model="vm.customer.homeEmail"
           class="form-control" name="HomeEmail" type="email" placeholder="user@domain.com...">
</div>
<!– And below!… -->
```

What We've Accomplished

```
@customer.FormGroupFor(x => x.Name)

@customer.FormGroupFor(x => x.WorkEmail)

@customer.FormGroupFor(x => x.HomeEmail)

@customer.FormGroupFor(x => x.WorkPhone)

@customer.FormGroupFor(x => x.HomePhone)

@customer.FormGroupFor(x => x.WorkAddress)

@customer.FormGroupFor(x => x.HomeAddress)
```

- ✓ Code reduction:
  - ✓ 28 *fewer* lines of code
  - ✓ 21 *fewer* tags
  - ✓ 35 *fewer* attributes
- ✓ No more CSS classes
- ✓ Still AngularJS-powered!

What We've Accomplished



Full Name

Full Name (ex: John Doe)...

Work Email

Work Email...

Home Email

Home Email...

Work Phone

Work Phone...

Home Phone

Home Phone...

- ✓ Code reduction:
  - ✓ 28 *fewer* lines of code
  - ✓ 21 *fewer* tags
  - ✓ 35 *fewer* attributes
- ✓ No more CSS classes
- ✓ Still AngularJS-powered!
- ✓ Intelligent defaults

There's More Metadata Where That Came From…

Validation Attribute	Description
CustomValidationAttribute	Uses a custom method for validation.
DataTypeAttribute	Specifies a particular type of data, such as e-mail address or phone number.
EnumDataTypeAttribute	Ensures that the value exists in an enumeration.
RangeAttribute	Designates minimum and maximum constraints.
RegularExpressionAttribute	Uses a regular expression to determine valid values.
RequiredAttribute	Specifies that a value must be provided.
StringLengthAttribute	Designates maximum and minimum number of characters.
ValidationAttribute	Serves as base class for validation attributes.

Up Next…

Validation!