

Supporting Angular Validation with Data Annotations



Matt Honeycutt

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

Where We Are



Built the foundation

HTML helpers that emit AngularJS

Removed a *lot* of magic strings

Eliminated noisy markup

Leveraging *some* model metadata

Overview



Overview of AngularJS validation

Validation indicator directives

ASP.NET MVC validation

Taking C# validation client-side

AngularJS Validation



DANGER: INFORMATION OVERLOAD!



Year of Moo's "Taming Forms in AngularJS 1.3"

<http://goo.gl/NqHb8Y>

Forms

```
<form novalidate
  ng-submit="vm.add()"
  name="vm.form">
  <!-- Snip! -->

  <div class="alert alert-danger" ng-show="vm.errorMessage != null">
    {{vm.errorMessage}}
  </div>

  <div class="form-group has-feedback">
    <label class="control-label" for="Title">Title</label>
    <input required ng-model="vm.opportunity.title"
      class="form-control" name="Title" type="text" placeholder="Title...">
  </div>

  <div class="form-group has-feedback">
    <label class="control-label" for="Description">Description</label>
    <textarea ng-model="vm.opportunity.description" rows="6"
      class="form-control" name="Description" placeholder="Description..."></textarea>
  </div>

  <!-- Snip! -->

</form>
```

← **Form controller (vm.form)**

FormController

- **\$pristine** – true if the form has not been interacted with
- **\$dirty** – true if the form has been interacted with
- **\$valid** – true if the entire form is valid
- **\$invalid** – true if any part of the form is invalid
- **\$submitted** – true if the user attempted to submit the form

Forms

```
<form novalidate
  ng-submit="vm.add()"
  name="vm.form">

<!-- Snip! -->

<div class="alert alert-danger" ng-show="vm.errorMessage != null">
  {{vm.errorMessage}}
</div>

<div class="form-group has-feedback">
  <label class="control-label" for="Title">Title</label>
  <input required ng-model="vm.opportunity.title"
    class="form-control" name="Title" type="text" placeholder="Title...">
</div>

<div class="form-group has-feedback">
  <label class="control-label" for="Description">Description</label>
  <textarea ng-model="vm.opportunity.description" rows="6"
    class="form-control" name="Description" placeholder="Description..."></textarea>
</div>

<!-- Snip! -->

</form>
```

Model Controller (NgModelController)

- Typical form properties: `$pristine`, `$dirty`, `$valid`, `$invalid`, or `$touched`
- `$touched` – True if the control has lost focus

← Model controller
(vm.form.Title)

← Model controller
(vm.form.Description)

HTML5 Input Types

Input	Valid	Invalid
<code><input type="email"></code>	some@user.com	someuser
<code><input type="url"></code>	http://google.com	google.com
<code><input type="number"></code>	1234	abc
<code><input type="date"></code>	01/01/2015	January 01, 2015
<code><input type="time"></code>	09:00 PM	9 am
More...		

HTML5 Validation Attributes

Attribute	Valid	Invalid
required	Any value	
minlength="5"	abcde	abcd
maxlength="5"	abcde	abcdef
pattern="\d+"	12345	12abc
min="100"	101	99
max="100"	99	101

Validation in Heroic CRM

Full Name

Untouched:



Full Name

Invalid:

Full Name

Valid:

Validation in Heroic CRM

Untouched:	<div>Full Name</div> <input type="text" value="Full Name (ex: John Doe)"/>
Invalid:	<div>Full Name</div> <div><input type="text" value="Full Name (ex: John Doe)"/> </div>
Valid:	<div>Full Name</div> <div><input type="text" value="John Doe"/> </div>

Implementing the Validation

```
<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>
```

Implementing the Validation

```
var formGroup = new HtmlTag("div")
    .AddClasses("form-group", "has-feedback")
    .Attr("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" +
        "}");
```

Implementing the Validation

On the client-side...

```
window.app.directive('formGroupValidation', formGroupValidation);

function formGroupValidation() {
  return {
    //This is what we'll implement!
  }
}
```

In FormGroupFor...

```
var formGroup = new HtmlTag("div")
  .AddClasses("form-group", "has-feedback")
  .Attr("form-group-validation", name);
```

Implementing the Validation

form-group-validation

Full Name

John Doe



← input-validation-icons

Implementing the Validation

form-group-validation

Full Name

John Doe



input-validation-icons

Building The FromGroupValidation Directive



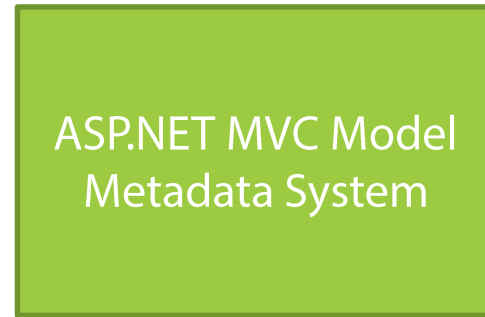
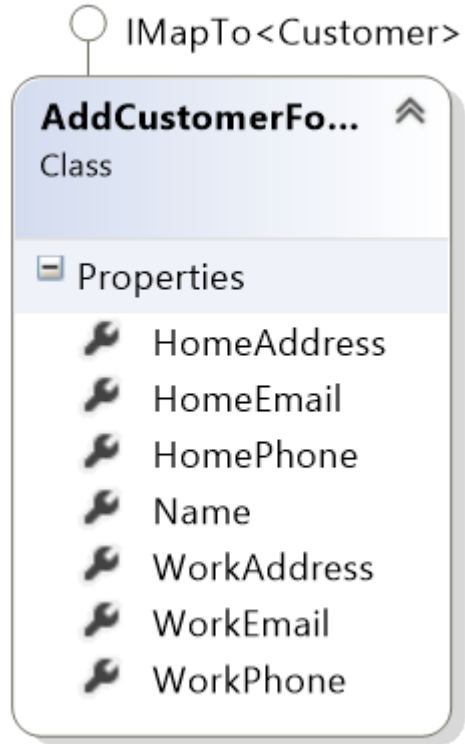
Validation Attributes

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

Validation Attributes

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.

Model Metadata



AddCustomerForm's Model Metadata

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

From Model Metadata to HTML5 Attributes

AddCustomerForm's Model Metadata

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



FormGroupFor



<input type="email" required name="Email">

Adding Validation to FormGroupFor



What We've Accomplished

```
<form novalidate
      name="vm.form"
      ng-submit="vm.add()">

<!-- Snip -->

@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)

<!-- Snip -->

</form>
```

What We've Accomplished

```
<form novalidate
      name="vm.form"
      ng-submit="vm.form.$valid && vm.add()">

<!-- Snip -->

@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)

<!-- Snip -->

</form>
```

What We've Accomplished

form-group-validation

Client-side...

Work Email

Work Email...



← input-validation-icons

Server-side...

```
[Required, DataType(DataType.EmailAddress)]  
public string WorkEmail { get; set; }
```


What We've Accomplished

```
[Required, DataType(DataType.EmailAddress)]  
public string WorkEmail { get; set; }
```



Work Email


Work Email...



But Work Remains....

```
<form novalidate
      name="vm.form"
      ng-submit="vm.form.$valid && vm.add()">
```

```
<!-- Snip -->
```



```
@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)
```

```
<!-- Snip -->
```

```
</form>
```

Up Next...

Angular-powered
templating!

