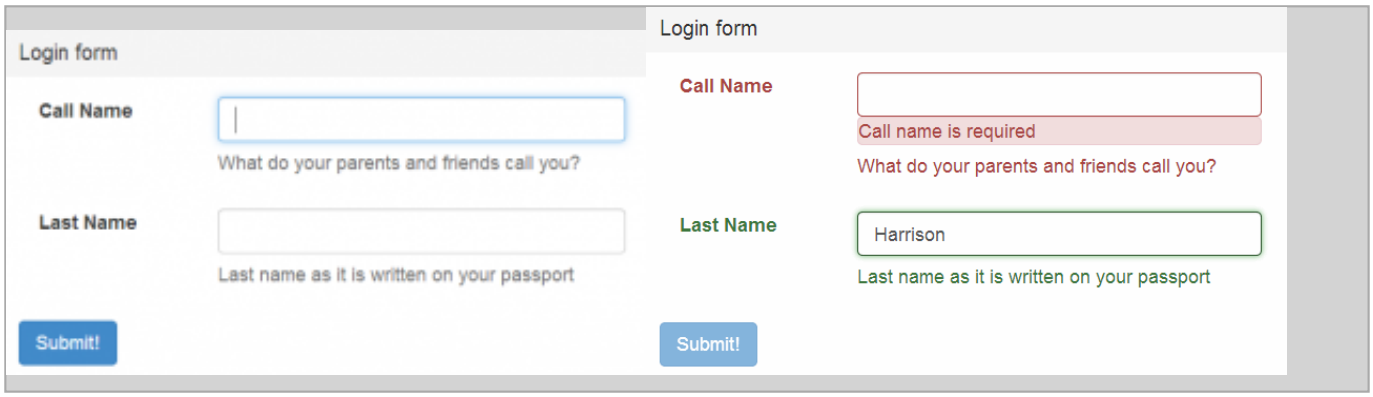
Advanced form control with AngularJS and Bootstrap3

by Benny Bottema - Friday, January 10, 2014

http://www.bennybottema.com/2014/01/10/advanced-form-control-with-angularjs-and-bootstrap3/

Forms in angular are pretty straightforward once you know how to read its state. So let's look at a collection of tricks and put together and complete working Bootstrap 3 form with fabulous form validation.



In this form:

1. No validation while initially filling the form
2. On submit: validate form and keep updating fields as they change
3. Red and green colors indicate the user's progress
4. Once submitted invalid form, the submit button becomes available only once the form is valid
5. Bootstrap 3 form style, including help text and custom tailored alerts for validation messages

What AngularJS gives you

With AngularJS, <form> is actually a built in directive that keeps track of the form's validity status. It has features such as $pristine vs $dirty (did the user interact with the form or not), $valid vs $invalid and so on. Furthermore, it provides specific validations on input-basis. All of this information is available out of the box on the scope using expressions as you would normally. A form's scope is available under the name of the form. So let's take a look at a login form:

```
<!-- disable button until form is valid -->
<form name="loginForm" novalidate="novalidate">
   <button type="submit" ng-
disabled="loginForm.$invalid">log in!</button>
```

```
</form>
```

This form immediately displays validation errors even prior to submitting and the button is disabled while the form is in an invalid state.

Now this approach may satisfy your requirements, but this means you need to start showing form validation errors as soon as the form is visible, because you can't submit the form until it is filled in correctly; not the best way to engage the user:

> "Hey I need some information for you! Hey, I know you didn't fill in anything, but everything is filled in wrong!"

Also notice we put in a **novalidate** to circumvent the browsers built in validation capabilities, since we want to let AngularJS and our own code to take care of that. (I wouldn't assign a value to novalidate normally, but this code prettify plugin goes crazy otherwise).

On submit, validate form before sending request

One thing is missing is a flag that indicates if the form has been submitted. Often what you want to do is only show validation errors once the form has been submitted and then in runtime keep the user updated about his input fixes (still wrong or now it's good). So there's a small trick for that:

```
Submit
```

Now in your form you can use this scope variable to only show validation errors when submitted is true (for example ng-show="submitted && form.email.$error.required"). A better example can be found on jsfiddle.

However, this does not help us much, because the form is still being submitted since we put in the *novalidate* attribute (the browser is not preventing a submit). So we need to prevent the form from submitting itself and we do that by introducing a directive that cooperates with the built in <form> directive.

- working example on jsFiddle

```
// directive that prevents submit if there are still form errors
```

```
app.directive('validSubmit', [ '$parse', function($parse) {
  return {
   // we need a form controller to be on the same element as this dire
ctive
   // in other words: this directive can only be used on a <form>
   require: 'form',
   // one time action per form
   link: function(scope, element, iAttrs, form) {
    form.$submitted = false;
    // get a hold of the function that handles submission when form is
 valid
    var fn = $parse(iAttrs.validSubmit);

    // register DOM event handler and wire into Angular's lifecycle wi
th scope.$apply
    element.on('submit', function(event) {
     scope.$apply(function() {
      // on submit event, set submitted to true (like the previous tri
ck)
      form.$submitted = true;
      // if form is valid, execute the submission handler function and
 reset form submission state
      if (form.$valid) {
       fn(scope, { $event : event });
       form.$submitted = false;
      }
     });
    });
   }
  };
 }
]);
```

This directive blocks the form from actually submitting until all validations are green. Original credits go to malix (I think), for providing the proof of concept, which I refined for personal use.

Now to use this!

```
<div ng-app="app" ng-controller="MainCtrl">
    <form name="form" valid-submit="sendFormToServer()" novalidate>
        <label for="name" ng-
class="{'error': form.$submitted && form.name.$invalid}">name</label>
        <input id="name" name="name" type="text" required ng-
```

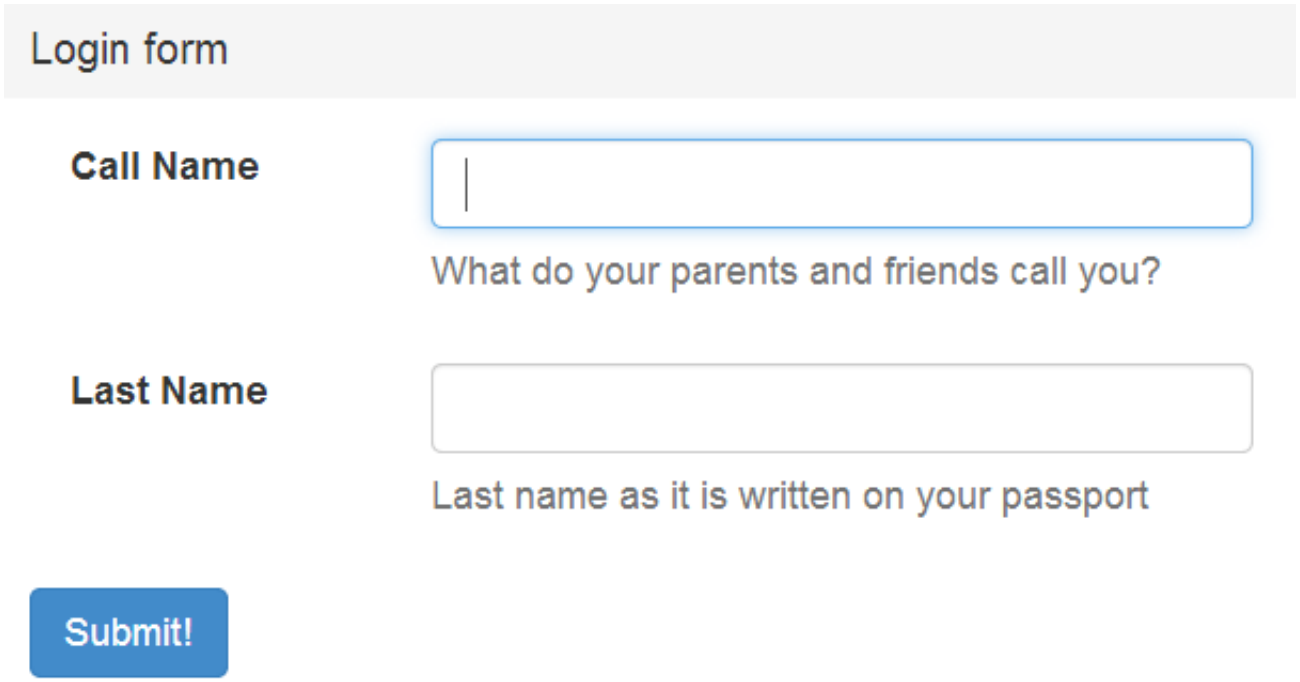
```
model="name"></input>
        <button type="submit">log in!</button>

        <div class="help" ng-
show="form.$submitted && form.name.$invalid">(name is required)</div>
    </form>
</div>
```

Note that form validations are ignored if you don't actually bind on a scope with ng-model!

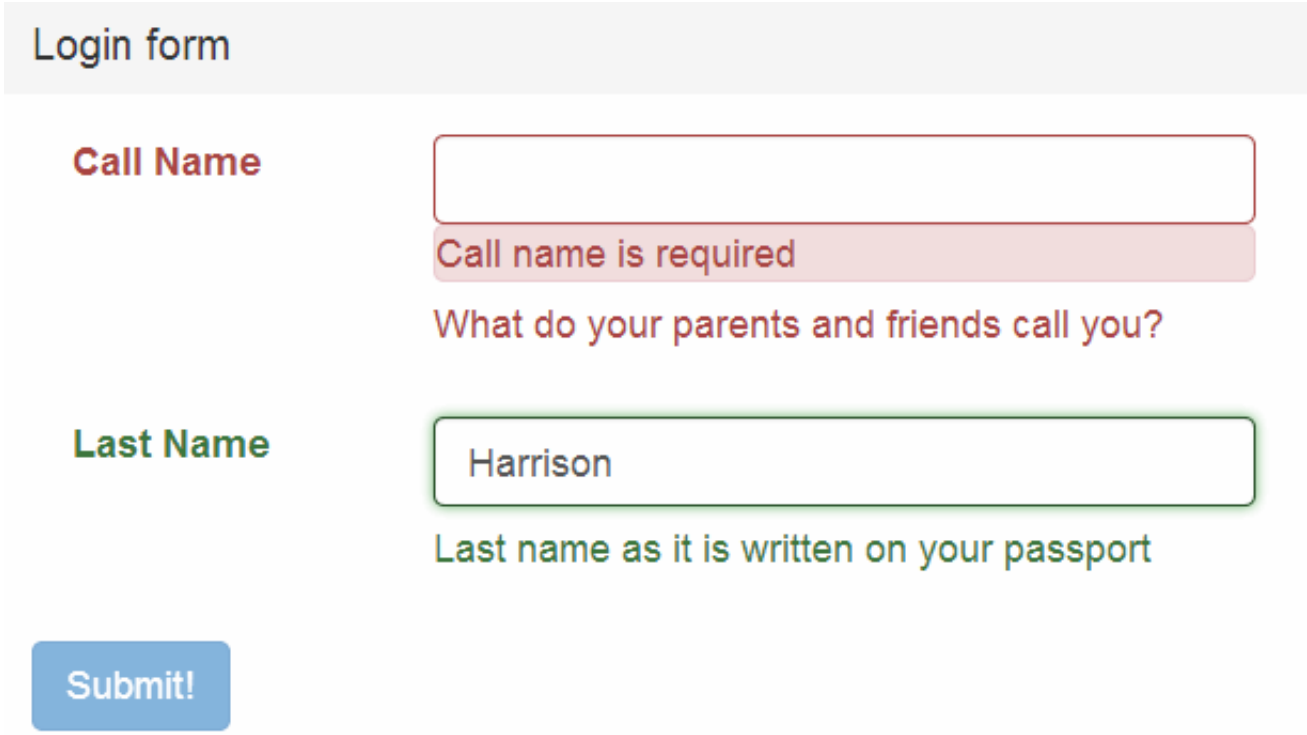
So this is basically all you need, but we can go further. We can make green what was filled in incorrectly and corrected by the user. Finally we can put some Bootstrap 3 sauce on top of this!

Let's just cut to the chase:

Final solution in Bootstrap 3



Pristine pre-submit state: no errors, button enabled



Invalid post-submit state: errors/successes shown runtime, submit button disabled

```
<div ng-app="app" ng-controller="MainCtrl">
 <div class="panel-group">
     <div class="panel panel-default">
         <div class="panel-heading">
             <h4 class="panel-title">Login form</h4>
         </div>
         <div class="panel-body">
                <ng-include src="'form.html'"></ng-include>
            </div>
     </div>
 </div>

    <!-- kept seperate from the bootstrap markup to keep this example
clean -->
    <script type="text/ng-template" id="form.html">
        <form name="form" valid-submit="sendForm()" novalidate>
            <!-- call name-->
            <div class="form-group clearfix" ng-class="{
                    'has-
error': form.$submitted && form.callName.$invalid,
                    'has-
success': form.$submitted && form.callName.$valid}">
```

```
                <label class="col-sm-2 control-
label" for="callName">Call Name</label>
                <div class="col-sm-5">
                    <input id="callName" name="callName" class="form-c
ontrol" type="text" ng-
model="person.callName" required autofocus></input>
                    <div class="alert alert-danger" ng-show="form.$sub
mitted && form.callName.$error.required">Call name is required</div>
                    <p class="help-
block">What do your parents and friends call you?</p>
                </div>
            </div>
            <!-- last name-->
            <div class="form-group clearfix" ng-class="{
                    'has-
error': form.$submitted && form.lastName.$invalid,
                    'has-
success': form.$submitted && form.lastName.$valid}">
                <label class="col-sm-2 control-
label" for="lastName">Last Name</label>
                <div class="col-sm-5">
                    <input id="lastName" name="lastName" class="form-
control" type="text" ng-model="person.lastName" required></input>
                    <div class="alert alert-danger" ng-show="form.$sub
mitted && form.lastName.$error.required">required</div>
                    <p class="help-
block">Last name as it is written on your passport</p>
                </div>
            </div>
            <!-- form controls-->
            <div class="form-group">
                <button type="submit" class="btn btn-primary" ng-
disabled="(form.$submitted && form.$invalid)">Submit!</button>
            </div>
        </form>
    </script>
</div>




var app = angular.module('app', []);

// directive that prevents submit if there are still form errors
app.directive("validSubmit", [ "$parse", function($parse) {
  return {
```

```
   // we need a form controller to be on the same element as this dire
ctive
   // in other words: this directive can only be used on a <form>
   require: 'form',
   // one time action per form
   link: function(scope, element, iAttrs, form) {
    form.$submitted = false;
    // get a hold of the function that handles submission when form is
 valid
    var fn = $parse(iAttrs.validSubmit);

    // register DOM event handler and wire into Angular's lifecycle wi
th scope.$apply
    element.on("submit", function(event) {
     scope.$apply(function() {
      // on submit event, set submitted to true (like the previous tri
ck)
      form.$submitted = true;
      // if form is valid, execute the submission handler function and
 reset form submission state
      if (form.$valid) {
       fn(scope, { $event : event });
       form.$submitted = false;
      }
     });
    });
   }
  };
 }
]);

// handle form submission when the form is completely valid
app.controller("MainCtrl", function($scope) {
  $scope.sendForm = function() {
    alert('form valid, sending request...');
  };
});

// focus on the first input when the page loads
window.focus = function(selector) {
 // timeout is needed for Chrome (is a bug in Chrome)
 setTimeout(function(){
  $(!!selector ? selector : "[autofocus]:not(:focus)").first().focus()
;
 }, 1);
};
```

```
.form-group .alert {
  padding: 0px;
  margin-bottom: 0px;
}
```

final solution in jsFiddle
