Form에 대하여

입력 요소 (input, select, textarea)는 사용자에게 데이터를 입력 받기 위한 수단입니다. 양식은 관련 입력 요소를 그룹화하기 위한 입력 요소의 집합입니다.

Angular 폼 입력 요소의 검증 서비스를 제공하기 있어, 사용자가 잘못된 데이터를 입력하면 바로 알아 볼 수 있습니다. 이 기능은 사용자가 어떤 오류인지가 바로 반환되므로, 보다 쾌적한 사용자 경험을 제공합니다. 쾌적한 사용자 경험을 제공하는 역할을 담당하는, 클라이언트 측의 검증은, 쉽게 회피 할 수 있으므로 수신되는 값은 신용 할 수 없다는 것을 잊지 마십시오. 서버 측 유효성 검사는 안전한 애플리케이션 구축을 위한 필수입니다.

1. 간단한 양식

양방향 데이터 바인딩을 인식 시키는데 중요한 지시어가 `ngModel` 입니다. `ngModel` 지시어는 view에서 모델 또한 모델에서 view를 동기화, 양방향 데이터 바인딩을 제공합니다. 또한, 그 동작을 향상시키기 위해 다른 지시어에 대한 API를 제공합니다.

index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Example - example-example31-production</title>
  <script
src="//ajax.googleapis.com/ajax/libs/angularjs/1.3.2/angular.min.js"></script>
<script>
  angular.module('formExample', [])
    .controller('ExampleController', ['$scope', function($scope) {
      $scope.master = {};

      $scope.update = function(user) {
        $scope.master = angular.copy(user);
      };

      $scope.reset = function() {
        $scope.user = angular.copy($scope.master);
      };
```

```
    $scope.reset();
  }]);
</script>
</head>
<body ng-app="formExample">
<div ng-controller="ExampleController">
  <form novalidate class="simple-form">
    Name: <input type="text" ng-model="user.name" /><br />
    E-mail: <input type="email" ng-model="user.email" /><br />
    Gender: <input type="radio" ng-model="user.gender" value="male" />male
    <input type="radio" ng-model="user.gender" value="female" />female<br />
    <button ng-click="reset()">RESET</button>
    <button ng-click="update(user)">SAVE</button>
  </form>

  <pre>form = {{user | json}}</pre>
  <pre>master = {{master | json}}</pre>
</div>
</body>
</html>
```

`novalidate`는 브라우저의 본래의 검증 기능을 비활성화하는 데 사용하는 것에 주의하십시오.

2. CSS 클래스의 사용

입력 요소와 양식의 스타일을 수행 할 수 있도록 ngModel는 아래의 CSS의 클래스를 추가합니다.

- ng-valid
- ng-invalid
- ng-pristine
- ng-dirty

아래의 예에서는 각 폼 입력 요소의 검증 결과를 한눈에 알게 하기 위해 CSS를 사용하고 있습니다. 이 예에서는 `user.name` 과 `user.email`은 필수 요소이고, 입력에 결함이 있는 경우에만 배경이 빨강색이 됩니다. 이렇게 하면 사용자는 모드 값을 입력, 확인 후에 검증 될 때까지 기다릴 필요 없이 입력하면서 즉시 잘못된 입력이 없는지 알 수 있습니다.

3. 폼에의 연결 상태 제어

양식은 FormController 컨트롤러의 인스턴스입니다. 폼의 인스턴스는 필요에 따라 name 속성을 사용하여 scope에 공개 할 수 있습니다. 마찬가지로 입력 요소는 NgModelController의 인스턴스입니다. 컨트롤 인스턴스 뿐만 아니라 name 속성을 사용하여 폼 인스턴스에 게시 할 수 있습니다. 이것은 폼 입력 요소 모두 내부 상태는, primitive data와 연결을 사용하여 view에 결합하는 데 사용할 수 있다는 것을 의미합니다.

따라서 위의 예에서 다음의 기능을 확장 할 수 있습니다.

- 양식에 어떤 변경 있었던 경우에만 리셋 버튼을 사용합니다.
- 폼에 입력 된 값이 정상적인 경우에만 저장 버튼을 사용합니다.
- `user.email`과 `user.agree`에 대한 오류 메시지를 정의합니다.

index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Example - example-example32-production</title>
  <script
src="//ajax.googleapis.com/ajax/libs/angularjs/1.3.2/angular.min.js"></script>
</head>
<body ng-app="formExample">
<div ng-controller="ExampleController">
  <form novalidate class="css-form">
    Name:
      <input type="text" ng-model="user.name" required /><br />
    E-mail: <input type="email" ng-model="user.email" required /><br />
    Gender: <input type="radio" ng-model="user.gender" value="male" />male
    <input type="radio" ng-model="user.gender" value="female" />female<br />
    <button ng-click="reset()">RESET</button>
    <button ng-click="update(user)">SAVE</button>
  </form>
</div>
<style type="text/css">
  .css-form input.ng-invalid.ng-dirty {
    background-color: #FA787E;
```

```
  }
  .css-form input.ng-valid.ng-dirty {
    background-color: #78FA89;
  }
</style>
<script>
  angular.module('formExample', [])
    .controller('ExampleController', ['$scope', function($scope) {
      $scope.master = {};

      $scope.update = function(user) {
        $scope.master = angular.copy(user);
      };

      $scope.reset = function() {
        $scope.user = angular.copy($scope.master);
      };

      $scope.reset();
    }]);
</script>
</body>
</html>
```

4. 검증의 정의

Angular는 HTML5의 대부분에서 일반적인 input 형 (text, number, url, email, radio, checkbox)을 위한, directive을 위한 검증 (required, pattern, minlength, maxlength, min, max)의 기본 기능을 제공합니다.

`ngModel` 컨트롤러에 커스텀 검증 기능을 추가 한 개발자 자신의 지시어를 정의함으로써 개발자 자신의 검증을 정의 할 수 있습니다. 컨트롤러를 유지하기 위해 지시어는 아래의 예와 같이 종속성을 지정합니다. 검증은 2 곳에서 할 수 있습니다.

모델에서 **view** 에 업데이트

연결된 모델이 변경 될 때마다, NgModelController#$formatters 배열의 모든 기능이 파이프라인 화 되어, 이 기능을 NgModelController#$setValidity를 통해 값의 포맷 기회와 폼 입력 요소의 검증 상태를 변경할 수 있는 기회가 주어집니다.

view 에서 모델로 업데이트

마찬가지로 사용자가 입력 요소에 입력 · 선택을 할 때마다, NgModelController#$setViewValue를 호출합니다. 이것은 NgModelController#$parsers 배열 내의 함수를 파이프 라인화 하여 NgModelController#$parsers 통해, 이러한 각 기능의 값을 변환 할 수 있는 기회와 폼 입력 요소의 검증 상태를 변경할 수 있는 기회가 주어집니다.

예를 들어, 다음 두 지시어가 있다고 합시다.

- 먼저 하나는 입력 된 것이 정당한 정수를 판정합니다. 예를 들어, 1.23는 소수이므로 틀립니다. 배열에 push 대신 unshift를 사용하는 것에 주의하십시오. 이것은 첫 번째 parser의 함수로, 숫자로 변환하기 전에 검증하기 위한 함수를 실행되어 문자열의 제어를 실시하고 있기 때문입니다.
- 두 번째 문은 부동 소수점 숫자입니다. 1.2과 1,2를 두 개를, 정당한 부동 소수점 숫자 인 1.2의 수치로 분석합니다. 여기서 1,2 같은 숫자를 부정한 값으로 하여, 사용자의 입력을 허용하지 않습니다. 브라우저에서 HTML5의 number의 input 형식을 사용하지 않은 점에 유의하십시오.

index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Example - example-example98-production</title>
```

```
  <script
src="//ajax.googleapis.com/ajax/libs/angularjs/1.2.25/angular.min.js"></script>
  <script src="script.js"></script>
</head>
<body ng-app="form-example1">
<form name="form" class="css-form" novalidate>
  <div>
    Size (integer 0 - 10):
    <input type="number" ng-model="size" name="size"
           min="0" max="10" integer />{{size}}<br />
    <span ng-show="form.size.$error.integer">This is not valid integer!</span>
    <span ng-show="form.size.$error.min || form.size.$error.max">
      The value must be in range 0 to 10!</span>
  </div>

  <div>
    Length (float):
    <input type="text" ng-model="length" name="length" smart-float />
    {{length}}<br />
    <span ng-show="form.length.$error.float">
      This is not a valid float number!</span>
  </div>
</form>
</body>
</html>
```

script.js

```
var app = angular.module('form-example1', []);

var INTEGER_REGEXP = /^\-?\d+$/;
app.directive('integer', function() {
  return {
    require: 'ngModel',
    link: function(scope, elm, attrs, ctrl) {
      ctrl.$validators.integer = function(modelValue, viewValue) {
```

```
        if (ctrl.$isEmpty(modelValue)) {
          // consider empty models to be valid
          return true;
        }

        if (INTEGER_REGEXP.test(viewValue)) {
          // it is valid
          return true;
        }

        // it is invalid
        return false;
      };
    }
  };
});

app.directive('username', function($q, $timeout) {
  return {
    require: 'ngModel',
    link: function(scope, elm, attrs, ctrl) {
    var usernames = ['Jim', 'John', 'Jill', 'Jackie'];

      ctrl.$asyncValidators.username = function(modelValue, viewValue) {

        if (ctrl.$isEmpty(modelValue)) {
          // consider empty model valid
          return $q.when();
        }

        var def = $q.defer();

        $timeout(function() {
          // Mock a delayed response
          if (usernames.indexOf(modelValue) === -1) {
```

```
            // The username is available
            def.resolve();
          } else {
            def.reject();
          }
        }, 2000);

        return def.promise;
      };
    }
  };
});
```