Controller에 대하여 – Controllers

컨트롤러의 이해 – Understanding Controllers

Angular에서 컨트롤러는 Angular Scope 를 인수로 사용하는 JavaScript 생성자 함수입니다. ng-controller 지시어를 통해 컨트롤러가 DOM 요소에 할당되면 Angular는 지정된 컨트롤러의 생성자 함수를 사용하여 새로운 컨트롤러 객체를 인스턴스화 합니다. 새로운 scope 범위는 $scope로 컨트롤러의 생성자 함수에 주입 됨으로써 인수로 사용할 수 있습니다.

컨트롤러는 다음과 같은 목적으로 사용됩니다.

- $ scope 객체의 초기 상태를 설정합니다.
- $ scope 객체의 동작을 추가합니다.

$Scope 객체의 초기 상태 설정 – Setting up the initial state of a $scope object

일반적으로 응용 프로그램을 작성하는 경우 Angular의 $scope의 초기 설정이 필요합니다. 개발자는 $scope 객체에 속성을 할당하여 초기 상태의 설정합니다. 속성은 View와 모델을 포함합니다. (이 모델은 View에 의하여 제공됩니다) 모든 $scope 속성은 컨트롤러가 등록 된 DOM의 부분 템플릿에서 사용할 수 있습니다.

다음은 'Hola!' 문자열을 포함 greeting 속성이 $ scope에 할당 된 매우 간단한 컨트롤러의 생성자 함수의 예입니다.

```
function GreetingCtrl($scope) {
  $scope.greeting = 'Hola!';
}]);
```

DOM에 컨트롤러가 할당되면 greeting 속성은 템플릿에 데이터 바인딩 됩니다.

```
<div ng-controller = 'GreetingCtrl'>
  {{ greeting }}
</div>
```

주의: Angular은 글로벌 공간에 컨트롤러 함수를 작성하는 것을 허용하고 있지만, 이것은 추천하지 않습니다. 실제 응용 프로그램에서는 Angular 모듈의 .controller 메소드를 아래와 같이 사용하십시오.

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

myApp.controller('GreetingController', ['$scope', function($scope) {
   $scope.greeting = 'Hola!';
}]);
```

위는 인 라인 주석 (inline injection annotation) 에 의해 컨트롤러가 $scope에 의존하고 있는지를 명확하게 하여 Angular에서 컨트롤러에 $scope 서비스가 제공됩니다. 자세한 내용은 가이드의 의존성 주입 (DI) 를 참조하십시오.

$scope 객체에 동작 추가 – Adding Behavior to a Scope Object

View에서 이벤트에 대한 반응이나 계산을 실행하려면 scope에 동작을 제공해야 합니다. $scope 객체에 대해 메소드를 이용하여 동작을 scope에 추가합니다. 이러한 메소드는 템플릿 / View에서 호출 됨으로써 사용할 수 있습니다.

아래의 예는 컨트롤러를 사용하여 scope에 숫자를 두 배로 증가시키는 dobule 메소드를 추가하고 있습니다.

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

myApp.controller('DoubleController', ['$scope', function($scope) {
  $scope.double = function(value) { return value * 2; };
}]);
```

DOM에 컨트롤러가 할당되면 double 메소드는 템플릿 Angular 식으로 실행됩니다.

```
<div ng-controller="DoubleController">
Two times <input ng-model="num"> equals {{ double(num) }}
</div>
```

AngularJS의 개념에서 다루고 있지만, scope에 객체가 할당되면 이것이 모델 속성이 됩니다. scope에 할당 된 메소드는 템플릿 / View에서 사용할 수 있게 되며, Angular 식과 ng 이벤트 핸들러 지시문을 통해 실행됩니다. (예: ngClick)

올바른 컨트롤러의 사용 방법 – Using Controllers Correctly

일반적으로 컨트롤러에 많은 것을 할당할 필요는 없습니다. 단일 View에 필요한 비즈니스 로직만을 포함할 필요가 있습니다.

컨트롤러를 슬림 하게 유지하는 가장 좋은 방법은 캡슐화이며, 서비스에 대해 컨트롤러에 의존하지 않도록 해야 하며, 이러한 서비스는 의존성 주입을 통해 컨트롤러에서 사용하도록 합니다. 이 내용은 이 가이드의 의존성 주입 서비스로 다루고 있습니다.

Do not use Controllers for:

DOM 조작에 관한 일

> 컨트롤러는 비즈니스 로직만을 포함한다. DOM 조작 (애플리케이션의 프레젠테이션 로직)는 테스트하기 어려운 것으로 알려져 있습니다. 프레젠테이션 로직을 컨트롤러에 두는 것은 비즈니스 로직의 테스트에 심각한 영향을 주는 것을 의미합니다. Angular는 자동으로 이루어지는 DOM 조작을 위해 데이터 바인딩 기능을 제공합니다. 만약 수동으로 DOM 조작의 실행이 필요한 경우 directive 안에서 프레젠테이션 로직을 캡슐화합니다.

입력 형식

> 대신 Angular의 Form 컨트롤러를 사용하십시오.

필터 출력

> 대신 Angular 필터 를 사용하십시오.

컨트롤러 상태 코드 공유

> 대신 Angular 서비스 를 사용하십시오.

다른 구성 요소의 라이프 사이클 관리

> (예: 서비스 인스턴스의 작성 등)

간단한 컨트롤러의 예제

Angular에서 컨트롤러의 구성 요소가 어떻게 작동하는지 실례로 알기 쉽게 이해하기 위해 다음의 요구 사항을 충족 시키는 작은 응용 프로그램을 만들어 봅시다.

- 2 개의 버튼과 하나의 메시지를 가지는 템플릿
- spice 라고 명명 된 문자열의 모델
- spice 값을 설정하는 두 가지 기능을 가진 컨트롤러

템플릿의 메시지는 spice 모델에 연결되어 있어, 기본적으로 "very"문자열이 설정됩니다. 버튼을 클릭하면 spice 모델에 chili (고추) 또는 jalapeño (멕시코고추)가 설정되도록 연결되어 메시지는 데이터 바인딩에 의해 자동으로 업데이트됩니다.

app.js

```
var myApp = angular.module('spicyApp1', []);

myApp.controller('SpicyController', ['$scope', function($scope) {
    $scope.spice = 'very';

    $scope.chiliSpicy = function() {
        $scope.spice = 'chili';
    };

    $scope.jalapenoSpicy = function() {
        $scope.spice = 'jalapeño';
    };
}]);
```

index.html

```
<div ng-controller="SpicyController">
 <button ng-click="chiliSpicy()">Chili</button>
 <button ng-click="jalapenoSpicy()">Jalapeño</button>
 <p>The food is {{spice}} spicy!</p>
</div>
```

이 예에서 다음 사항을 확인하십시오.

- ng-controller directive 는 템플릿 (암묵적으로) scope 를 만드는 데 사용되며, scope 는 SpicyCtrl 컨트롤러에 의해 관리됩니다.
- SpicyCtrl 는 단순한 플레인 JavaScript 함수입니다. 관습에 따라 함수 이름의 첫 글자를 대문자로, 마지막에 "Ctrl"또는 "Controller"를 붙입니다.
- $scope 에 속성을 할당하여 모델을 만들거나 업데이트됩니다.
- 컨트롤러 메서드는 직접 scope 를 할당하여 만들 수 있습니다. (chiliSpicy 메서드 참조)
- 컨트롤러의 메서드와 속성은 템플릿에서 사용할 수 있습니다. (요소와 그 자식 요소에서)

Spicy에서 인수를 가져오는 방법 – Spicy Arguments Example

또한 컨트롤러의 메서드는 인수를 받을 수 있습니다. 위의 예제를 조금 변경 한 다음 데모를 확인하십시오.

index.html

```
<div ng-controller="SpicyController">
      <input ng-model="customSpice">
      <button ng-click="spicy('chili')">Chili</button>
      <button ng-click="spicy(customSpice)">Custom spice</button>
      <p>The food is {{spice}} spicy!</p>
</div>
```

app.js

```
var myApp = angular.module('spicyApp2', []);

myApp.controller('SpicyController', ['$scope', function($scope) {
    $scope.customSpice = "wasabi";
    $scope.spice = 'very';

    $scope.spicy = function(spice) {
        $scope.spice = spice;
    };
}]);
```

여기에서 SpicyCtrl 컨트롤러는 spice라는 하나의 인수를 spicy라는 하나의 메소드를 정의하고 있는 것에 주목하십시오. 이 템플릿은 컨트롤러의 메소드를 참조하여, 첫 번째 버튼과의 연결에 의해 'chili'라는 문자열을 전달하고, 두 번째 버튼은 spice 모델 속성 (input 입력란과 연결되어 있다)의 값을 전달합니다.

$scope 상속의 예 – Scope Inheritance Example

DOM 계층이 다른 계층에 컨트롤러가 할당되는 것은 자주 있는 일입니다. ng-controller 지시어는 새로운 하위 범위를 만들기 위해 각각으로부터 상속 한 범위의 계층 구조를 취득합니다. 각 컨트롤러가 받는 $scope는 위 계층의 컨트롤러에 의해 정의 된 속성과 메서드에 액세스 할 수 있습니다. 범위의 상속에 대해 더 자세히 알고 싶다면, Understanding Scopes를 참조하십시오.

index.html

```
<div class="spicy">
  <div ng-controller="MainController">
    <p>Good {{timeOfDay}}, {{name}}!</p>
    <div ng-controller="ChildController">
      <p>Good {{timeOfDay}}, {{name}}!</p>
      <div ng-controller="GrandChildController">
        <p>Good {{timeOfDay}}, {{name}}!</p>
      </div>
    </div>
  </div>
</div>
```

app.css

```
div.spicy div {
  padding: 10px;
  border: solid 2px blue;
}
```

app.js

```
var myApp = angular.module('scopeInheritance', []);
myApp.controller('MainController', ['$scope', function($scope) {
  $scope.timeOfDay = 'morning';
  $scope.name = 'Nikki';
}]);
myApp.controller('ChildController', ['$scope', function($scope) {
  $scope.name = 'Mattie';
}]);
myApp.controller('GrandChildController', ['$scope', function($scope) {
  $scope.timeOfDay = 'evening';
  $scope.name = 'Gingerbread Baby';
}]);
```

3 ng-controller directive 가 템플릿에 중첩되어 있는 것에 주목하십시오. 결과적으로 view 에 4 개의 scope 가 만들어집니다.

- 루트 Scope
- timeOfDay 와 name 속성을 가진 MainCtrl scope 입니다.
- ChildCtrl scope 는 timeOfDay 속성을 상속하지만 name 속성은 덮어 씁니다.
- GrandChildCtrl scope 는 MainCtrl 에서 정의 된 timeOfDay 속성 ChildCtrl 에 정의 된 name 속성을 모두 덮어 씁니다.

메소드의 상속도 속성과 동일합니다. 위의 예라면 모든 속성을 문자열 값을 반환하는 메서드로 대체 할 수 있습니다.

컨트롤러 테스트 – Testing Controllers

컨트롤러를 테스트하는 방법은 여러 가지가 있지만, 가장 좋은 방법의 다음과 같이 $rootScope 와 $controller의 주입을 포함하는 것을 들 수 있습니다.

컨트롤러 정의 – Controller Definition

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

myApp.controller('MyController', function($scope) {
  $scope.spices = [{"name":"pasilla", "spiciness":"mild"},
                {"name":"jalapeno", "spiciness":"hot hot hot!"},
                {"name":"habanero", "spiciness":"LAVA HOT!!"}];
  $scope.spice = "habanero";
});
```

컨트롤러 테스트 – Controller Test

```
describe('myController function', function() {

describe('myController', function() {
  var $scope;
  beforeEach(module('myApp'));

  beforeEach(inject(function($rootScope, $controller) {
```

```
    $scope = $rootScope.$new();
    $controller('MyController', {$scope: $scope});
  }));

  it('should create "spices" model with 3 spices', function() {
    expect($scope.spices.length).toBe(3);
  });

  it('should set the default value of spice', function() {
    expect($scope.spice).toBe('habanero');
  });
});
});
```

만약 중첩 된 컨트롤러의 테스트를 필요로 하는 경우 기존의 DOM에서의 테스트에서 동일한 scope 계층 구조를 만들 필요가 있습니다.

```
describe('state', function() {
var mainScope, childScope, grandChildScope;

beforeEach(module('myApp'));

beforeEach(inject(function($rootScope, $controller) {
    mainScope = $rootScope.$new();
    $controller('MainController', {$scope: mainScope});
    childScope = mainScope.$new();
    $controller('ChildController', {$scope: childScope});
    grandChildScope = childScope.$new();
    $controller('GrandChildController', {$scope: grandChildScope});
}));

it('should have over and selected', function() {
    expect(mainScope.timeOfDay).toBe('morning');
    expect(mainScope.name).toBe('Nikki');
    expect(childScope.timeOfDay).toBe('morning');
```

```
    expect(childScope.name).toBe('Mattie');
    expect(grandChildScope.timeOfDay).toBe('evening');
    expect(grandChildScope.name).toBe('Gingerbreak Baby');
});
});
```