Module

Module이란?

많은 응용 프로그램은 인스턴스화, 각종 연결, 응용 프로그램을 시작하는 주(main) 메소드를 가집니다. Angular 애플리케이션은 그런 main 메소드를 가지지 않습니다. 대신 모듈 응용 프로그램이 어떻게 기동 할 것인가의 정의를 지정합니다. 여기에는 다음과 같은 장점이 있습니다.

- 프로세스가 더 선언적이고 쉽게 이해할 수 있습니다.
- 단위 테스트에서 모든 모듈을 읽어 들일 필요가 없고, 단위 테스트를 비교적 기술하기 쉬워집니다.
- 추가 모듈은 시나리오 테스트 로드가 가능하며, 설정의 일부를 override하여, 응용 프로그램의 END TO END 테스트 도와줍니다.
- 타사 코드를 재사용 가능한 패키지로 할 수 있습니다.
- 모듈을 임의 / 병렬로 순서대로 읽을 수 있습니다. (모듈 실행의 지연 특성으로 인해)

개요

그러면 "Hello World" 모듈을 움직여 보자.

주의해야 할 중요한 것은 다음과 같습니다.

- Module API 를 참조하고 내용을 파악하고 있어야 합니다.
- `<html ng-app = "myApp">` 이 Angular에 참조 된 것을 기억하십시오. 여기에서 응용 프로그램을 시작할 때 사용되는 모듈이 지정되게 됩니다.

index.html

```
<div ng-app="myApp">
  <div>
    {{ 'World' | greet }}
  </div>
</div>
```

script.js

```
// declare a module
var myAppModule = angular.module('myApp', []);


// configure the module.
// in this example we will create a greeting filter
```

```
myAppModule.filter('greet', function() {
 return function(name) {
    return 'Hello, ' + name + '!';
  };
});
```

권장설정

위의 예제가 간단하지는 않지만, 규모가 있는 애플리케이션에는 대응할 수 없습니다. 대신 아래와 같이 응용 프로그램을 여러 모듈로 분할하는 것이 좋습니다.

- 서비스 선언을 위한, `service` 모듈
- 지시어 선언을 위한, `directive` 모듈
- 필터 선언을 위한, `filter` 모듈
- 그리고 위의 모듈에 의존하여, 초기화 코드를 가진 응용 프로그램 계층 모듈

분할하는 것은, 테스트를 어렵게 하는 초기화 코드를 무시하고 싶다는 요구가 자주 있기 때문입니다. 분리 된 모듈로 배치함으로써 테스트가 쉬워집니다. 또한 테스트 관련 모듈 만 가져 와서 그 테스트에 더 집중할 수 있습니다.

위는 제안에 불과하므로 필요에 따라 좋은 방법으로 처리해 보시기 바랍니다.

index.html

```
<div ng-controller="XmplController">
  {{ greeting }}
</div>
```

script.js

```
angular.module('xmpl.service', [])

  .value('greeter', {
    salutation: 'Hello',
    localize: function(localization) {
      this.salutation = localization.salutation;
    },
    greet: function(name) {
      return this.salutation + ' ' + name + '!';
```

```
    }
  })

  .value('user', {
    load: function(name) {
      this.name = name;
    }
  });

angular.module('xmpl.directive', []);

angular.module('xmpl.filter', []);

angular.module('xmpl', ['xmpl.service', 'xmpl.directive', 'xmpl.filter'])

  .run(function(greeter, user) {
    // This is effectively part of the main method initialization code
    greeter.localize({
      salutation: 'Bonjour'
    });
    user.load('World');
  })

  .controller('XmplController', function($scope, greeter, user){
    $scope.greeting = greeter.greet(user.name);
  });
```

Module의 Load와 의존관계

모듈은 부팅 시 응용 프로그램에 적용되는 설정 및 실행 블록의 집합입니다. 가장 간단한 형태의 모듈은 두 개의 블록으로 구성되어 있습니다.

1. **구성 (config) 블록** - 구성 블록은 provider의 등록 및 설정 단계에서 실행됩니다. provider와 상수 만 설정 블록에 주입하는 것이 가능합니다. 이것은 설정이 완전히 끝나기 전에 서비스 인스턴스 화가 우발적으로 이루어지는 것을 방지합니다.
2. **실행 (run) 블록** - 실행 블록은, 인젝터가 구성한 프로그램의 시작에 사용 된 후 실행됩니다. 인스턴스와 상수 만 실행 블록에 주입 가능합니다. 이것은 응용 프로그램 실행 중에 시스템 설정을 하는 것을 방지합니다.

```
angular.module('myModule', []).
config(function(injectables) { // provider-injector
  // This is an example of config block.
  // You can have as many of these as you want.
  // You can only inject Providers (not instances)
  // into config blocks.
}).
run(function(injectables) { // instance-injector
  // This is an example of a run block.
  // You can have as many of these as you want.
  // You can only inject instances (not Providers)
  // into run blocks
});
```

구성(congig) 블록

모듈은 config 블록과 동등의 편리한 메소드가 존재합니다. 예를 들어,

```
angular.module('myModule', []).
  value('a', 123).
  factory('a', function() { return 123; }).
  directive('directiveName', ...).
  filter('filterName', ...);

// is same as

angular.module('myModule', []).
```

```
config(function($provide, $compileProvider, $filterProvider) {
  $provide.value('a', 123);
  $provide.factory('a', function() { return 123; });
  $compileProvider.directive('directiveName', ...);
  $filterProvider.register('filterName', ...);
});
```

구성 (config) 블록은 등록 된 순서로 적용되기 시작합니다. 한 가지 예외가 상수의 정의는 모든 구성 블록의 선두에 배치됩니다.

실행(run) 블록

실행 블록은 Angular로의 main 메소드의 개념에 가장 가까운 것입니다. 실행 블록의 코드에서 응용 프로그램 실행시킬 필요가 있습니다. 모든 service가 구성된 모든 인젝터가 생성 된 후 실행됩니다. 실행 블록 단위 테스트가 어려운 코드가 포함될 수 있기 때문에, 그 코드는 단위 테스트에서 무시할 수 있도록 관련 모듈에 정의해야 합니다.

의존관계

모듈은 다른 모듈을 목록 화하여 종속성을 표시 할 수 있습니다. 모듈에 종속 경우 필요한 모듈이 필요로 하는 모듈보다 전에 로드 되지 않으면 안 되는 것을 의미합니다. 달리 말하면 필요한 모듈 구성 (config) 블록은 필요로 하는 모듈의 구성 (config0 블록보다 먼저 실행되어야 한다는 것입니다. 동일한 것이 실행 (run) 블록도 그렇습니다. 하나의 모듈이 여러 모듈에서 필요로 되는 경우에도 한 번만 로드 하면 문제 없습니다.

비 동기 load

모듈은 $injector 구성을 관리하는 방법이며, VM(JavaScript 실행 환경) 내에 스크립트 load와는 무관합니다. 스크립트 로드를 처리하는 기존의 프로젝트가 있다고 하면, 그 스크립트에 Angular가 사용될지도 모릅니다. 모듈은 로드 할 때 아무것도 하지 않기 때문에 임의의 순서로 VM (JavaScript 실행 환경)에 로드 할 수 있습니다. 따라서 스크립트 로더는 이 속성을 사용하여 로드 프로세스를 병렬화 할 수 있습니다.