AngularJS Service에 대하여

Angular Service란?

Angular 서비스는 단일 객체 또는 Web 어플리케이션 일반적인 특정 작업을 수행하는 함수입니다. Angular는 서버에 요청을 보내는 브라우저의 XMLHttpRequest 개체에 액세스하는 $http 서비스와 같은 몇 가지 기본 서비스가 있습니다. 다른 Angular 변수와 식별자와 마찬가지로 기본 제공 서비스는 항상 `$`로 시작됩니다(위의 $http처럼). 또한 사용자 정의 서비스를 만들 수 있습니다.

서비스 사용에 대해

Angular 서비스를 사용하려면 개발자가 서비스에 의존하는 컴포넌트 (컨트롤러, 서비스, 필터, 지시어)에 대한 종속성을 식별(결정)하고, 그 외의 부분에 대해서는 Angular의 의존성 주입 (DI)의 서브 시스템에 맡기고 있습니다. Angular 주입 서브 시스템은 서비스의 인스턴스화, 의존성의 해결, 요청 된 것으로서의 구성 요소에 대한 의존성의 공급을 담당합니다.

Angular는 "생성자 (constructor)" 주입을 사용하여 의존성을 주입합니다. 의존성은 구성 요소의 팩토리 / 생성자 함수에 전달됩니다. 이는 JavaScript가 동적 타입 언어이기 때문에, Angular의 존성 주입 서브 시스템은 서비스 종속성을 식별하기 위한 정적 형 지정을 사용할 수 없기 때문입니다. 따라서 구성 요소는 주입 어노테이션 메서드를 사용하여 명확하게 그 의존성을 정의해야 합니다. 다음은 $inject 속성에 의해 공급 된 예입니다.

```
var MyController = function($location){...};
MyController.$inject = ['$location'];
myModule.controller('MyController', MyController);
```

또한, 아래는 "인라인 (inline)"주입 어노테이션에 의해 공급 된 예입니다.

```
var myService= function($http){...};
myModule.factory('myService', ['$http', myService]);
```

서비스의 정의

프로그램 개발자는 Angular 모듈 이름 등록 및 서비스 팩토리 함수에 의해 자유롭게 자신의 서비스를 정의 할 수 있습니다.

서비스 팩토리 함수의 목적은 프로그램의 일부가 될 서비스를 표현하는 단일 객체 또는 함수를 작성하기 위함입니다. 이 객체 또는 함수는 그 후 어떠한 구성 요소 (컨트롤러, 서비스, 필터, 지시문)에 서비스에 대한 의존성을 결정하기 위해 주입됩니다.

Angular 팩토리 함수는 지연 실행됩니다. 의존성을 충족시키기 위해 필요할 경우에만 실행되며, 이것은 각 서비스마다 한 번만 실행됩니다. 서비스에 의존하는 모든 것은 서비스 팩토리에서 생성 된 단일 인스턴스의 참조를 가져옵니다.

서비스의 의존성 관리

설명

Angular는 서비스 인스턴스의 구축을 위해 필요한 종속성으로 다른 서비스를 선언 할 수 있습니다.

의존성을 선언하려면 팩토리 함수의 표기 안에 지정하여, $inject 속성에 식별하는 문자열의 배열로 설정하거나, 배열 어노테이션 중 하나를 사용하여 함수에 어노테이션을 설정합니다. 선택적으로 $inject 속성 선언은 생략 할 수 있습니다. ("Inferring $inject" 를 참조하십시오. 그러나 이것은 현재 실험용 기능이기 때문에 주의하십시오.)

배열 주석의 사용 예

```
function myModuleCfgFn($provide){
   $provide.factory('myService', ['dep1', 'dep2', function(dep1, dep2){}]);
```

$inject 속성의 사용 예

```
function myModuleCfgFn($provide){
   var myServiceFactory = function(dep1, def2){};
   myServiceFactory.$indect = ['dep1', 'dep2'];
   $provide.factory('myService', myServiceFactory);
}
```

DI 추정의 사용 예 (압축화(mimify)를 할 경우에는 사용하지 마십시오)

```
function myModuleCfgFn($provide){
   $provide.factory('myService', function(dep1, dep2){});
}
```

다음은 두 개의 서비스의 예로, 하나는 다른 1개의 서비스에 의존하고 있으며, 각 두 서비스는 Angular 프레임 워크가 제공하는 다른 서비스에 의존하고 있습니다.

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

```
/**
 * The `batchLog` service allows for messages to be queued in memory and flushed
 * to the console.log every 50 seconds.
 *
 * @param {*} message Message to be logged.
 */
batchModule.factory('batchLog', ['$interval', '$log', function($interval, $log) {
  var messageQueue = [];

  function log() {
    if (messageQueue.length) {
      $log.log('batchLog messages: ', messageQueue);
      messageQueue = [];
    }
  }

  // start periodic checking
  $interval(log, 50000);

  return function(message) {
    messageQueue.push(message);
  }
}]);

/**
 * `routeTemplateMonitor` monitors each `$route` change and logs the current
 * template via the `batchLog` service.
 */
batchModule.factory('routeTemplateMonitor', ['$route', 'batchLog', '$rootScope',
  function($route, batchLog, $rootScope) {
    $rootScope.$on('$routeChangeSuccess', function() {
      batchLog($route.current ? $route.current.template : null);
    });
  }]);
```

이 샘플 주목할 점은 다음과 같습니다.

- batchLog 서비스는 내장 $timeout 과 $log 서비스에 의존하고 있어 한번에 console.log 메시지를 로그 출력 해줍니다.
- routeTemplateMonitor 서비스는 batchLog 서비스뿐만 아니라 내장 $route 서비스에 의존하고 있습니다.
- 이러한 서비스는 모두 주입 어노테이션 팩토리 함수의 표기와 배열 어노테이션을 사용하여 의존성 선언을 하고 있습니다. 배열의 의존성을 나타내는 문자열의 순서는 중요하며, 이것은 팩토리 함수 내의 인수 이름의 순서와 같지 않으면 안됩니다. 의존관계가 함수의 표기에서 알 수 있지 않으면, ID 와 순서를 나타내는 배열은 인젝터가 주입하는 서비스와 그 순서를 결정하는 데 사용됩니다.

서비스 작성

개요

Angular 몇 가지 유용한 서비스를 제공하여줍니다만, 어떠한 규모의 응용 프로그램에서는, 독자적인 커스텀 서비스를 작성하는 것이 유용하다는 것을 알 수 있다. 이렇게 하려면 모듈의 서비스 팩토리 함수, Module # factory API, 또는 직접 모듈 config 함수 내부에서 $provide 의 API를 통해 등록하는 것부터 시작합니다.

등록 된 서비스의 팩토리 함수의 공급을 필요로하는 의존성 선언은 물론, 명칭(id)를 기준으로 Angular의 DI 시스템(인젝터)를 사용하여 그들을 등록함으로써, 모든 Angular 서비스는 의존성 주입 (DI)의 성립됩니다. 의존성 테스트 중에 mock / stub / dummy로 교체 할 수 있기 때문에 서비스의 고급 테스트를 할 수 있습니다.

서비스의 등록

서비스를 등록하려면, 해당 서비스의 일부분이 되는 모듈을 가질 필요가 있습니다. 그러면 모듈 API 또는 모듈의 구성(config) 함수에서 $provide 서비스 중 하나를 통해 서비스를 등록 할 수 있습니다. 아래의 임시 코드에서 두 가지 방법을 확인하십시오.

angular.Module API를 사용

```
var myModule = angular.module(‘myModule’, []);
myModule.factory(‘serviceId’, function(){
    var shinyMewServiceInstance;
    // factory function body that constructs shinyNewServiceInstance
    return shiyNewServiceInstance;
}
```

$provide 서비스 사용

```
angular.module('myModule', []).config(['$provide', function($provide) {
    $provide.factory('serviceId', function() {
        var shinyNewServiceInstance;
        // factory function body that constructs shinyNewServiceInstance
        return shinyNewServiceInstance;
    });
}]);
```

서비스 인스턴스가 아니라, 호출 될 때 그 인스턴스를 만드는 팩토리 함수가 등록되는 것에 주의하십시오.

의존성

서비스는 의존되는 것은 불가능하지만, 자신의 의존성을 가질 수(지정) 있습니다. 이것은 팩토리 함수의 인수로 지정할 수 있습니다. 자세한 내용은 Angular의 의존성 주입 (DI) 를 참조하여, 배열 어노테이션과 $inject 속성을 사용하여 코드 압축 해도 안전하게 의존성 주입(DI) 어노테이션을 수행하는 방법을 확인해 두어야 합니다.

아래 예제는 매우 간단한 서비스입니다. 이 서비스는 $window 서비스(팩토리 함수 인수로 전달되는)에 의존 한 단순한 함수입니다. 이 서비스는 단순히 모든 통지를 저장 한 후, 3이되면 서비스가 window 경고로 모든 알림을 표시하는 것입니다.

index.html

```
<div id="simple" ng-controller="MyController">
  <p>Let's try this simple notify service, injected into the controller...</p>
  <input ng-init="message='test'" ng-model="message" >
  <button ng-click="callNotify(message);">NOTIFY</button>
  <p>(you have to click 3 times to see an alert)</p>
</div>
```

script.js

```
angular.module('myServiceModule', []).
  controller('MyController', ['$scope','notify', function ($scope, notify) {
    $scope.callNotify = function(msg) {
      notify(msg);
    };
  }]).
 factory('notify', ['$window', function(win) {
    var msgs = [];
    return function(msg) {
      msgs.push(msg);
      if (msgs.length == 3) {
        win.alert(msgs.join("\n"));
        msgs = [];
```

```
    }
  };
}]);
```

protactor.js

```
it('should test service', function() {

expect(element(by.id('simple')).element(by.model('message')).getAttribute('value')
)
      .toEqual('test');
});
```

Angular 서비스의 인스턴스화

Angular의 모든 서비스는 지연되어 인스턴스화됩니다. 이것은 그 서비스의 인스턴스가 필요한 경우 또는 프로그램 구성 요소에 종속되어 필요한 경우에만 만들어지는 것입니다. 즉, Angular는 직접 또는 프로그램에 의해 간접적으로 요구되지 않는 한 서비스를 인스턴스화하지 않는다는 것입니다.

싱클톤으로 서비스

마지막으로 중요한 것은, Angular 서비스 모두가 싱글 톤 프로그램임을 알고 있어야 합니다. 이것은 인젝터마다 주어지는 서비스의 인스턴스는 하나뿐이라는 것을 의미합니다. Angular는 글로벌 상태에 치명적인 알레르기가 있기 때문에, 주어진 서비스의 각 인스턴스 자신이 여러 인젝터를 만드는 것은 가능하지만, 중요하고 결정적인 시험을 제외하고는 사용하지 않는 것이 좋습니다.

서비스의 컨트롤러 삽입

개요

컨트롤러에 의존하는 것으로서 서비스를 사용하는 것은, 다른 서비스에 의존 관계로 서비스를 사용하는 것과 매우 비슷합니다.

JavaScript는 동적 타입 언어이기 때문에 DI는 정적 인 형태 (정적 타입 언어처럼)에서 주입하는 서비스를 읽을 수 없습니다. 따라서 주입되는 서비스 이름의 문자열을 포함하는 배열의 $inject 속성을 사용하여 서비스 이름을 지정할 수 있게 되어 있습니다. 서비스 이름은 Angular에 등록 된 해당 서비스 ID와 일치해야 합니다. 배열의 서비스 순서(서비스 ID 항목의 순서)는 팩토리 함수를 호출 할 때 사용됩니다. 팩토리 함수의 인수의 이름은 자유롭게 지정할 수 있지만, 관습에 의해 서비스 ID에 일치시켜두면 뒤에 기술하는 혜택을 누릴 수 있습니다.

```
function myController($loc, $log){
   this.firstMethod = function(){
      $loc.setHash();
   }
   this.secondMethod = function(){
      $log.info('...');
   }
}

myController.$inject = ['$location', '$log'];
```

index.html

```
<div id="simple" ng-controller="MyController">
  <p>Let's try this simple notify service, injected into the controller...</p>
  <input ng-init="message='test'" ng-model="message" >
  <button ng-click="callNotify(message);">NOTIFY</button>
  <p>(you have to click 3 times to see an alert)</p>
</div>
```

script.js

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

```
 controller('MyController', ['$scope','notify', function ($scope, notify) {
   $scope.callNotify = function(msg) {
     notify(msg);
   };
 }]).
factory('notify', ['$window', function(win) {
   var msgs = [];
   return function(msg) {
     msgs.push(msg);
     if (msgs.length == 3) {
       win.alert(msgs.join("\n"));
       msgs = [];
     }
   };
 }]);
```

End to End Test

```
it('should test service', function() {

expect(element(by.id('simple')).element(by.model('message')).getAttribute('value')
)
      .toEqual('test');
});
```

암시적 의존성 주입(DI)

Angular의 DI의 새로운 기능으로는 인수의 이름에서 의존성을 결정하는 것이 가능합니다. 위의 예제를 이 $window, $scope, 알림 서비스를 암시적 의존성을 사용한 것으로 고쳐 작성해 봅시다.

주의: 이 예제는 소스 코드를 압축(minify)하면 작동하지 않을 수 있습니다. 자세한 내용은 "가이드 개념 설명 – UI 로직 추가: 컨트롤러"의 마지막 단계를 참조하세요.

index.html

```
<div id="simple" ng-controller="MyController">
  <p>Let's try this simple notify service, injected into the controller...</p>
  <input ng-init="message='test'" ng-model="message" >
  <button ng-click="callNotify(message);">NOTIFY</button>
  <p>(you have to click 3 times to see an alert)</p>
</div>
```

script.js

```
angular.module('myServiceModule', []).
  factory('notify', function(win) {
    var msgs = [];
    return function(msg) {
      msgs.push(msg);
      if (msgs.length == 3) {
        win.alert(msgs.join("\n"));
        msgs = [];
      }
    };
  });

  function myController($scope, notify){
     $scope.callNotify = function(msg){
         notify(msg);
     };
  }
```

다만, 코드를 압축화 할 예정이라면, 압축에 의해 변수 명이 변경되므로, $inject 속성을 이용하는 등, 명시적인 의존관계를 지정할 필요가 있습니다.