TEMA 4

Daniel Estévez  @  & 

 & 

Servicios propios

- Libertad de definir servicios a través del servicio "**factory function**"
- El servicio "**factory function"** genera el objeto único o función que representa el servicio al resto de la aplicación

Servicios propios

- El objeto o función que devuelve el servicio se inyecta en cualquiera de los componentes (controlador, servicio, filtro o directiva) que especifica una dependencia en el servicio

Servicios propios

scripts.js

```
var myApp = angular.module('myApp', []);
myApp.directive('greeting', function() {
    return {
        restrict: 'E',
        template:  '<button type="submit" class="btn btn-default" ng-click="
        greeting()"> {{ greeting }} </button>'
        link: function (scope, attrs) {
            scope.greeting = greetingService.greeting(attrs.name);
        };
    });
```

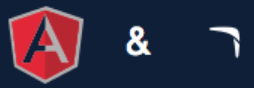
Servicios propios

scripts.js

```
/* Creating a new service for aur app */
myApp.factory('greetingService', function() {
     return {
          greeting: function (name) {
               return 'Hola!!!' + name + 'Waaasssup?'
          }
     };
});
```

Servicios propios

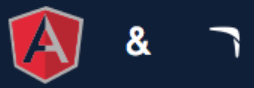
- Los servicios pueden tener sus propias dependencias
- Al igual que declarar las dependencias en un controlador, se pueden declarar dependencias especificando el factory del servicio a instanciar



Servicios propios

Ejercicio

1) Utilizando "gna-servicio.html" crea una app Angular nueva que mejore la que teníamos, delegando la lógica de generar números aleatorios en un servicio



Propagando eventos ($on y $broadcast)

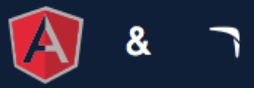
- Scopes pueden propagar eventos en forma similar a los eventos DOM
- El evento puede ser transmitido a los hijos del scope o emitido a los padres
- `$on(name, listener) { ... }` donde:
  - `name` es el nombre del evento que esta escuchando (‘String’)
  - `listener` es la funcion que se ejectua al recivirse el evento `function(event, args...)`

Propagando eventos ($on y $broadcast)

```
$scope.$on("event:newChat",function($event, amigo){
     scope.amigoQueMeHabla = amigo;
});
```

Propagando eventos ($on y $broadcast)

- El evento puede ser transmitido todos los $scopes hijos (y sus hijos), notificando a los que estén escuchando dicho evento
- El ciclo de vida de eventos se inicia en el ámbito en el que se llamaba **$broadcast**.



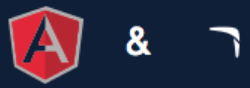
Propagando eventos ($on y $broadcast)

- Todos los oyentes que escuchan por el nombre del evento en este ámbito recibir una notificación.
- `$broadcast(eventName, result) { ... }` donde:
  - `eventName` es el nombre del evento que esta escuchando (‘String’)
  - `result` es el resultado devuelto por el que emite el evento

```
$rootScope.$broadcast("event:newChat", “Dani”);
```

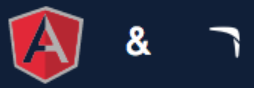
API REST

- REST, **REpresentational State Transfer**, es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP
- REST nos permite crear servicios y aplicaciones que pueden ser usadas por cualquier dispositivo o cliente que entienda HTTP
- Es increíblemente más simple y convencional que otras alternativas que se han usado en los últimos diez años como SOAP y XML-RPC



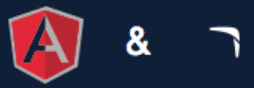
APIs REST

- Para desarrollar APIs REST los aspectos claves que hay que dominar y tener claros son:
  - Métodos HTTP
  - Códigos de estado
  - Aceptación de tipos de contenido



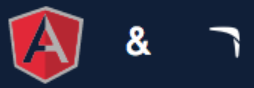
APIs REST

- Ejemplos:
  - GET /facturas Nos permite acceder al listado de facturas
  - POST /facturas Nos permite crear una factura nueva
  - GET /facturas/123 Nos permite acceder al detalle de una factura



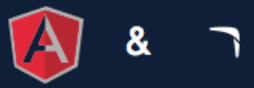
APIs REST

- Ejemplos:
  - PUT /facturas/123 Nos permite editar la factura, sustituyendo la totalidad de la información anterior por la nueva.
  - DELETE /facturas/123 Nos permite eliminar la factura
  - PATCH /facturas/123 Nos permite modificar cierta información de la factura, como el número o la fecha de la misma



APIs REST

- Ejemplos:
  - PUT /facturas/123 Nos permite editar la factura, sustituyendo la totalidad de la información anterior por la nueva.
  - DELETE /facturas/123 Nos permite eliminar la factura
  - PATCH /facturas/123 Nos permite modificar cierta información de la factura, como el número o la fecha de la misma



Atacando API REST externa ($http)

- $http es un servicio del core de Angular que facilita la comunicación con servidores HTTP remotos a través el objeto XMLHttpRequest del navegador o mediante JSON.
- Recibe un solo argumento (un objeto de configuración) y devuelve una promesa con dos métodos específicos: `success` and `error`

Atacando API REST externa ($http)

```
$http({
     url:'https://mail.danimailservice.com/mail/u/0/#inbox',
          method: 'GET'
          }).success(function(inboxEmails) {
               $rootScope.$broadcast("event:loadMails", inbox);
          }).error(function() {
          console.log('Error trying to retrieve the inbox from external
     API service.');
});
```

Atacando API REST Externa ($http)

Ejercicio

2) Utilizando "gna-api.html" crea una app Angular nueva que en vez de obtener los números aleatorios de un servicio angular, los obtenga utilizando una API externa