

Servicios propios

Servicios propios

- Como en los controladores y directivas se registran en los módulos
- Para registrar un servicios utilizamos la API `module.factory`

```
myModule.factory('greetingService', function() {...});
```

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

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

Servicios propios

```
/* Creating a new service for our 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

Ejercicios

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

Propagación de eventos

- 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

Escuchando: \$on

- `$on(name, listener) { ... }` donde:
 - `name` es el nombre del evento que se esta escuchando
 - `listener` es la función que se ejecuta al recibirse el evento
`function(event, args...)`

Escuchando: \$on

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

Propagando: \$broadcast

- El evento puede ser transmitido a todos los \$scopes hijos (y sus hijos)
- El ciclo de vida de eventos se inicia en el ámbito en el que se llamaba **\$broadcast**
- Todos los oyentes que escuchan el nombre del evento en este ámbito, reciben una notificación

Propagando: \$broadcast

- `$broadcast(eventName, result) { ... }` donde:
 - `eventName` es el nombre del evento que se propaga
 - `result` es el resultado devuelto

Propagando: \$broadcast

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

Atacando API: \$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: \$http

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

Ejercicios

- A partir de "gna-api.html" y utilizando **\$on**, **\$broadcast** y **\$http**, crea una app Angular nueva que en vez de obtener los números aleatorios de un servicio angular, los obtenga utilizando una API externa