# BEEVA - CURSO DE ANGULAR 2015
## TEORÍA (KATA NIVEL INTERMEDIO)
Fernando Castro / Juan Ferrer

## DIRECTIVAS

## PERSONALIZADAS

Una vez que hemos analizado algunas de las directivas nativas más importantes, vamos a ver ahora cómo podemos crear las nuestras propias, lo que nos permitirá una gran libertad a la hora de definir nuestro código.

Realmente, las directivas personalizadas son uno de los núcleos de Angular. Tanto es así que los propios creadores nos aconsejan desarrollar nuestro código pensando directamente en qué directivas podemos crear, como las reutilizaremos y qué sentido tendrán en cada uno de los documentos que vayamos creando en nuestra aplicación. Serán el centro de atención.

Debemos tener en cuenta que las directivas son la forma más común de incorporar plugins personalizados externos en una aplicación de Angular. Existen múltiples organizaciones y desarrolladores independientes que nos ofrecen multitud de opciones, pero la verdadera potencia del lenguaje será una combinación de lo existente, o ajustando las directivas a las necesidades de nuestra funcionalidad.

### NOMENCLATURA

Para definir adecuadamente una directiva vamos a tener en cuenta las siguientes recomendaciones:

- Asegurarse de **no utilizar un nombre ya existente**
- Seguir las normas de la nomenclatura (**siguiendo el estándar CamelCase** – unir todas las palabras, sin espacios, cada una empezando por mayúscula, excepto la primera)

```
HTML Code
filterDirective
```

En la llamada desde el documento **HTML**:

- Usaremos la palabra *data-* como prefijo (es el prefijo reservado en HTML para atributos personalizados, y nos interesa usarlo en el caso de querer utilizar una herramienta de validación para nuestro documento), separando con guiones cada parte del nombre:

```
HTML Code
<data-filter-directive></data-filter-directive>
```

- Se pueden utilizar también las siguientes nomenclaturas:

```
HTML Code
<filter:directive></filter:directive>
<filter_directive></filter_directive>
```

```
<filter-directive></filter-directive>
<x-filter-directive></x-filter-directive>
```

En el ejemplo vemos que se ha definido en el HTML como una etiqueta, pero tenemos tres opciones más:

- **Como atributo**: `<div data-filter-directive></div>`
- Como clase: `<div class="data-filter-directive "></div>`
- Como comentario: `<!-  directive: data-filter-directive ->`

Para conseguir compatibilidad total con los diferentes navegadores (además de ser la mejor opción para poder utilizar los motores de búsqueda), utilizaremos la segunda opción, definiéndolo como un atributo.

## CONFIGURACIÓN

Veamos ahora con qué propiedades podemos definir la configuración de nuestra directiva (siendo todas opcionales, utilizaremos las que necesitemos en cada momento, y según las decisiones de diseño aplicadas).

- **restrict**

  Con esta propiedad vamos a indicar cómo se llamará a la directiva desde el documento HTML. Las opciones son:

  - **'A': como atributo**
  - **'E'**: como etiqueta
  - **'C'**: como clase
  - **'M'**: como comentario
  - **Combinación** de varias de ellas

- **template / templateUrl**

  El HTML asociado a la vista de la directiva, pudiendo definir la plantilla directamente o en otro fichero, especificando la ruta al mismo. También se puede utilizar una función para poder cargar rutas dinámicas.

```
JavaScript Code
templateUrl: function(elem, attrs){
        switch(attrs.templateType){
                . case 'big': return 'src/templates/big.html'; break;
                . case 'sml': return 'src/templates/small.html'; break;
                . default: return 'src/templates/default.html'; break;
        }
}
```

- **scope**

  Siendo uno de  los atributos más importantes, nos define el espacio de variables a utilizar en nuestra directiva, que se relacionará con el entorno de su padre. Los valores que puede tomar son los siguientes:

- **false**: se utilizará el scope del padre
- **true**: se crea un nuevo scope, pero heredando del scope del padre
- **{…}**: se crea un nuevo scope, pero cerrado y heredando únicamente del $rootScope. Aún así, se puede compartir información con el exterior a través de atributos que definiremos entre llaves, y que pasaremos en la definición de la directiva en el HTML. Los tipos de variables son:
  - @ : string
  - = : variables que se conectan de manera bidireccional (? obligatoriedad)
  - & : funciones

HTML Code
```
<div ng-controller="Controller">
    <my-customer same-name ="naomi"
                    op-focus=exec_focus()"
                    variable="prueba"></my-customer>
      </div>
```

JavaScript Code
```
scope: {
        sameName: '@',
        opFocus: '&',
        variable: '='
}
```

El scope que se acaba de definir estará disponible tanto en la vista como en el controlador de la directiva.

- **compile / link**

Son las funciones que utilizaremos para inicializar la directiva, además de incluir toda la funcionalidad asociada.

compile se ejecuta una sola vez por directiva, al iniciarse la aplicación de Angular. Se crearán funciones de inicialización que utilizarán las instancias futuras. Recibe como parámetros las referencias a todos los elementos que instancia la directiva y sus atributos.

JavaScript Code
```
compile: function(elemArr, attrArr, transclude){…}
```

link se ejecuta una vez por cada instancia de la directiva. Recibe como parámetros el elemento que está instanciando la directiva, sus parámetros y el scope.

JavaScript Code
```
link: function(scope, elem, attrs, transclude){…}
```

En el caso de utilizar directivas anidadas, se pueden definir estas funciones con dos métodos: **pre** y **post**. Estas son todas las opciones disponibles, fusionando ambos funcionalidades:

- compile que devuelve un objeto con los métodos de link
- compile que devuelve el método post de link
- link con sus dos métodos
- el método post de link

## CONFIGURACIÓN AVANZADA

Una vez vistas las propiedades más generales, vamos a ir un paso más allá examinando las opciones más avanzadas que nos permiten las directicas personalizadas

- **transclude**

Aunque el propio nombre de transclusión puede llevar a pensar en una propiedad muy compleja, es en realidad más sencillo de lo que podría parecer: es el método por el cual una directiva muestra su contenido incluido en el documento HTML que fue reemplazado por su propia plantilla.

HTML Code
```
<div foo>
        Contenido
</div>
```

JavaScript Code
```
.directive("foo", function() {
        return {
                template: "<div>plantilla</div>"
        }
})
```

Por un lado, indicaremos una nueva propiedad en la configuración de la directiva, transclude: true.

Varios métodos para poder usar la propiedad:

- **ng-transclude**
  - usaremos la propiedad ng-transclude en la parte de la plantilla donde queramos incluir el contenido del HTML

JavaScript Code
```
.directive("foo", function() {
        return {
                transclude: true,
                template: "<div>plantilla</div>
                        <div ng-transclude></div>"
        };
})
```

- **Función transclude**

  - usaremos la función que vendrá como propiedad dentro de link, de tal forma que podamos añadir el código en el DOM

```
JavaScript Code
.directive("foo", function() {
        return {
                transclude: true,
                template: "<div>plantilla</div>",

                link: function(scope, element, attrs,  transclude) {

                        transclude(function(clone) {
                                element.append(clone);
                        });
                }
        };
})
```

- **require (herencia de directivas)**

Con esta propiedad vamos a recibir instancias de otras directivas, pudiendo utilizar los métodos y variables de sus respectivos controladores. Nos llegará en forma de string (en el caso de obtener una), o de un array de strings (si vamos a utilizar varias).

```
JavaScript Code
.directive('myPane', function() {
        return {
                require: '^myTabs',
                restrict: 'E',
                transclude: true,
                scope: {
                        title: '@'
                },
                 link: function(scope, element, attrs, tabsCtrl) {
                        tabsCtrl.addPane(scope);
                },
                templateUrl: 'my-pane.html'
        };
});
```

En este caso, podemos definir dentro la directiva 'componente' un controller donde definir sus funciones y variables:

```
JavaScript Code
.directive('myTabs', function() {

        return { restrict: 'E',
                  transclude: true,
                  scope: {},
                  controller: ['$scope', function($scope) {

                        var panes = $scope.panes = [];

                        $scope.select = function(pane) {
```

```
                angular.forEach(panes, function(pane) {
                        pane.selected = false;
                });
                 pane.selected = true;
        };

        this.addPane = function(pane) {
                if (panes.length === 0) {
                        $scope.select(pane);
                }
                panes.push(pane);
        };
    }],
     templateUrl: 'my-tabs.html'
  };
})
```

Para requerir otra directiva podremos utilizar las siguientes opciones:

- **sin prefijo**: directiva obligatoria en el mismo elemento donde se encuentra la que estamos definiendo
- **?**: optativa en el mismo elemento
- **^**: obligatoria en algún elemento padre
- **?^**: optativa en algún elemento padre

## BIBLIOGRAFÍA

Se ha recurrido al abundante material disponible en la red para explicar de la mejor manera posible los diferentes conceptos que aparecen en esta documentación:


Directiva custom ng-else: https://github.com/zachsnow/ng-elif
Directivas personalizadas: http:// prendedor.es/creacion-directivas-personalizadas-angularjs/
Transclusión: https://www.accelebrate.com/blog/angularjs-transclusion-part-1/
https://www.accelebrate.com/blog/transclusion-angularjs-part-2-2/