Création de directives

Quels intérêts ?

- Etendre le HTML et lui attribuer une logique métier
- Factoriser le code, permettre la réutilisation

Directive simple & template

```
var app = angular.module('testDirectivesApp', [])
.controller('Controller', ['$scope', function($scope) {
  $scope.client = {
    nom: 'SMITH',
    prenom: 'John'
  };
}]);

app.directive('myClient', function() {
  return {
    template: 'Nom : {{client.nom}} Prénom : {{client.prenom}}'
  };
});
```

Utilisations possibles :

```
<html data-ng-app="testDirectivesApp" data-ng-controller="Controller">
...
    <my-client></my-client>
    <div my-client></div>
```

Résultat :

Nom : SMITH Prénom : John
Nom : SMITH Prénom : John

templateUrl

Excepté pour les petits templates, il est préférable d'utiliser un fichier template séparé en affectant la propriété **templateUrl** (notamment pour éviter les problèmes de quote/guillemets) :

```
app.directive('myClient', function() {
  return {
    templateUrl: 'my-client.html'
  };
});
```

```
Nom : {{client.nom}} Prénom : {{client.prenom}}
```

La propriété templateUrl peut-être définie par une fonction :

```
app.directive('myClient', function() {
  return {
    templateUrl: function(elem, attr){
      return 'client-'+attr.type+'.html';
    }
  };
});
```

Templates :

```
Nom : {{client.nom}} Prénom : {{client.prenom}}
```

```
Adresse: {{client.adresse}}<br>
CP: {{client.cp}} {{client.ville}}
```

Utilisation :

```
<html data-ng-app="testDirectivesApp" data-ng-controller="Controller">
...
    <div myclient type="nom"></div>
    <div my-client type="adresse"></div>
```

restrict

La propriété **restrict** permet de définir le type de directive à créer ; restrict peut prendre les valeurs suivantes :

- éléments - **E**
- attributs - **A**
- classe Css - **C**
- commentaire - **M**
- ou une combinaison de ces valeurs : **EA** par exemple, pour élément et attribut

Si la directive est définie sur 1 élément :

```
app.directive('myClient', function() {
  return {
    restrict: 'E',
    templateUrl: function(elem, attr){
```

```
      return 'client-'+attr.type+'.html';
    }
  };
});
```

Seule la première ligne sera compilée par Angular, la seconde sera ignorée :

```
<html data-ng-app="testDirectivesApp" data-ng-controller="Controller">
...
    <my-client type="nom"></my-client>
    <div my-client type="adresse"></div>
```

scope

Le scope défini la portée d'une directive. Sans précision, le scope d'une directive est le scope de son contrôleur parent. Ce qui explique que notre directive ait pu accéder à la variable **client** de **Controller**.

Isolation

La directive précédemment créée a un défaut : elle dépend du contrôleur dans laquelle elle a été définie, et si nous souhaitons afficher un autre client, il faut définir un autre contrôleur...
La définition de la variable scope de la directive permet de résoudre ce problème :

scope:{} Défini à {}, la directive a un scope qui lui est propre, différent de celui du controller auquel elle appartient :

```
app.directive('myClient', function() {
  return {
    restrict: 'AE',
    scope:{},
    templateUrl: function(elem, attr){
      return 'client-'+attr.type+'.html';
    }
  };
});
```

Avec ce scope isolé, la directive ne peut plus accéder à la variable client définie dans le scope du controller : tester la page tests.html

Nous allons ajouter une propriété permettant d'initialiser la directive, et lui affecter la personne à afficher :

```
<html data-ng-app="testDirectivesApp" data-ng-controller="Controller">
...
    <my-client type="nom" personne="client"></my-client>
    <my-client type="adresse" personne="client"></my-client>
```

```
app.directive('myClient', function() {
    return{
        restrict:'EA',
        scope:{client:"=personne"},
        templateUrl: function(elem, attr){
            return 'client-'+attr.type+'.html';
        }
    }
});
```

L'attribut **personne** “bind” vers la variable **client** utilisée dans le scope de la directive.

Lorsqu'un attribut a le même nom que la variable du scope auquel il est associé, on peut utiliser la notation :

```
scope : {attributeName: "="}
```

Pour résumer (rapidement...) :

Valeurs de scope	**Résultats**
false ou non définie	pas de scope
{}	scope isolé
{attributeName:“=”}	scope isolé, avec passage d'un attribut de type **expression valeur** du même nom
{attributeNameInDirectiveScope:“=attributeName”}	scope isolé, avec passage d'un attribut de type **expression valeur** de nom différent
{attributeName:“@”}	scope isolé, avec passage d'un attribut de type **texte** du même nom
{attributeNameInDirectiveScope:“@attributeName”}	scope isolé, avec passage d'un attribut de type **texte** de nom différent
{attributeName:“&”}	scope isolé, avec passage d'un attribut de type **expression action** du même nom
{attributeNameInDirectiveScope:“&attributeName”}	scope isolé, avec passage d'un attribut de type **expression action** de nom différent
true	scope non isolé

Exemple de passage d'une expression action (ng-click) :

```
    <my-client type="nom" personne="client" ng-click="client.adresse=''"></my-
client>
    <div my-client type="adresse" personne="client"></div>
```

Sur click du nom, l'adresse doit être vidée.

En l'état actuel du code, la directive ng-click n'est pas opérationnelle, il faut la passer à notre directive :

```
app.directive('myClient', function() {
    return{
        restrict:'EA',
        scope:{client:"=personne",onClick:"&ngClick"},
        templateUrl: function(elem, attr){
            return 'client-'+attr.type+'.html';
```

```
        }
    }
});
```

et la déclencher à nouveau dans le template :

```
<p ng-click="onClick">Nom : {{client.nom}} Prénom : {{client.prenom}}</p>
```

Link/compile

Lorsqu'il analyse une directive appelée dans une page et la traduit en un ensemble d'éléments DOM, AngularJS compile la directive en faisant appel aux fonctions suivantes, dans un ordre déterminé :

1. compile
2. controller
3. pre-link
4. post-link

Si une directive doit modifier ou créer le DOM d'un template défini dans une directive, elle doit donc définir certaines de ces fonctions.

fonction compile

```
compile : function(tElement, tAttrs){

}
```

Utilisée pour la manipulation du DOM (manipulation de tElement = template element), elle permet des manipulations qui s'appliqueront à tous les éléments DOM clonés du template associé à la directive. S'il est nécessaire de définir également la fonction link (ou pre ou post link), et que la fonction compile est définie, la fonction compile doit renvoyer le(s) fonction(s) link, étant donné que link est ignoré si compile existe.

fonction controller

```
controller : function($scope, $element, $attrs, $transclude, otherInjectables){

}
```

Elle doit être utilisée pour permettre l'interaction avec d'autres directives.

fonction link

```
link : function(scope, iElement, iAttrs, controller){

}
```

Elle est habituellement utilisée pour enregistrer des listeners DOM (i.e., $watch expressions sur le scope) ou pour mettre à jour le DOM (i.e., manipulation sur iElement = instance individuelle de element). Elle est exécutée après que le template ait été cloné – voir <li ng-repeat...>, ou la fonction link est exécutée après que chaque <li> template (tElement) ait été cloné (en iElement)–. **$watch** permet à la directive d'être notifiée du changement de l'une des propriétés du scope (un scope est associé à chaque instance).

Exemple

Soit la directive SoldeDir, dont les fonctionnalités sont les suivantes :

- Affichage d'une valeur numérique (attribut montant), éventuellement en gras (attribut bold)
- Si le montant est positif, il est affiché en <fc #008000>**vert**</fc>
- S'il est négatif, en <fc #FF0000>**rouge**</fc>, précédé de la mention **Montant négatif**

```
app.directive('soldeDir', function() {
    return {
        restrict : 'EA',
        scope:{montant:"="},
        compile : function(tElem, tAttrs) {
            if (tAttrs.bold == "true")
                tElem.css("font-weight","bold");
            else
                tElem.css("font-weight","normal");
            var linkFunction = function($scope, tElem, tAttrs) {
                var update=function(){
                    if($scope.montant<0){
                        tElem.html("Solde négatif : " + $scope.montant);
                        tElem.css("color", "red");
                    }else{
                        tElem.html($scope.montant);
                        tElem.css("color", "green");
                    }
                }
                update();
            }
            return linkFunction;
        }
    }
});
```

```
    <div ng-init="valeur=-5">
    <input type="text" ng-model="valeur">
    <solde-dir bold="true" montant="valeur"></solde-dir>
```

-5 **Montant négatif : -5**

20 **Montant négatif : -5**

La directive fonctionne pour l'instant à l'initialisation, mais ne reprend pas les éventuelles modifications du

montant.
Pour remédier à ce problème, il faut ajouter dans la fonction link un appel au service **$watch**, pour observer et reporter les modifications du montant :

```
app.directive('soldeDir', function() {
    return {
        restrict : 'EA',
        scope:{montant:"="},
        compile : function(tElem, tAttrs) {
            if (tAttrs.bold == "true")
                tElem.css("font-weight","bold");
            else
                tElem.css("font-weight","normal");
            var linkFunction = function($scope, tElem, tAttrs) {
                var update=function(){
                    if($scope.montant<0){
                        tElem.html("Solde négatif : " + $scope.montant);
                        tElem.css("color", "red");
                    }else{
                        tElem.html($scope.montant);
                        tElem.css("color", "green");
                    }
                }
                update();
            }
                $scope.$watch('montant', function(oldVal, newVal) {
                    update();
                });
            return linkFunction;
        }
    }
});
```

Pour observer les changements d'un attribut de type texte (@) contenant une expression, on utilisera la méthode **$observe** sur l'attribut :

Soit la directive :

```
<dir attrib="test : {{valeur}}"></dir>
```

```
app.directive('dir', function() {
    return {
        restrict : 'E',
        scope:{attrib:"@"},
        link: function($scope, tElm, tAttrs,ctrl,transclude) {
            tAttrs.$observe('attrib', function() {
                $scope.attrib = $scope.$eval(tAttrs.attrib);
            });
        }
    };
```

```
});
```

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/slam4/richclient/angularjs/directives-creation

Last update: **2019/08/31 14:21**