Table of Contents

Services

Sharing code tussen controllers door middel van Services

Om business logica te delen tussen controllers kan je gebruik maken van Services. Door dependency injection kan je een service in je controllers gebruiken.

Verder is het beter om bijvoorbeeld de $http service niet in de controller te implementeren, maar in een service. Op die manier laat je hergebruik van code toe. Buiten data-binding laat AngularJS ook toe om onze code in logische secties onder te verdelen.

In Angular worden controllers aangemaakt, maar ook verwijderd als ze verschijnen en verdwijnen van een pagina. Dit in tegenstelling tot services. Deze worden éénmaal aangemaakt (wanneer je ze nodig hebt), maar als andere componenten deze nodig hebben zal Angular dezelfde instantie van de service hergebruiken. Je "dependency inject" de service zoals bijvoorbeeld de $scope service.

Een service is een object met functies. Deze functies kunnen aangeroepen worden door controller, directives, filters,.. Dus de business logica om een HTTP call te doen (om data van een server te halen) kan ook in de service geimplementeerd worden.

AngularJS heeft ook enkele interne services, zoals $http, $route, $window, $location. Als een controller deze wil gebruiken moeten ze geinjecteerd worden in de controller:

module.controller("testCtrl",function($http){}); of module.controller("testCtrl,function($window){});

Requesting JSON data met AJAX

Dit is mogelijk door gebruik te maken van de $http service om data te fetchen en te bewaren in de scope. De controller heeft een dependency naar de $scope en $http module. Een http get request wordt naar data/post.json gestuurd en krijgt een $promise object met een success en error methode terug. Bij een success wordt de JSON data aan de $scope.posts variabele.

De $http service supporteert get,head,post,put, delete en jsonp.

De $http service voegt automatisch HTTP headers toe, maar je kon deze zelf aanbrengen: $http.defaults.headers.common["X-custom-Headers"] = "Tom"

```
app.controller("PostCtrl",function($scope,$http)
    {
        $http.get("data/post.json").success(function(data)
        {
            $scope.posts = data;
        }).error(function(data)
        {
            //log error
        });
    });
```

Introductie

In AngularJS zijn services singleton objecten of functies die bepaalde taken uitvoeren. De controller is verantwoordelijk voor het binden van de data aan de view (door middel van $scope), en zou geen logica mogen bevatten om data op te halen of te manipuleren.

Voor het ophalen of manipuleren gebruiken we services. Services bevatten functionaliteit die kan opgeroepen worden vanuit controllers, directives, filters. Om code te delen tussen verschillende controllers kan je dus ook beroep doen op deze services.

Angular heeft verschillende interne services: $http, $route, $window, $location..

Eigen service ontwikkelen

Er zijn 2 mogelijkheden om een service aan te maken:

```
var app = angular.module("myapp",[]);
app.service("testservice",function(){
    this.data = [1,2,3,4,5];
});
```

of via een factory methode:

app.factory("testservice,function(){ var obj = {}; obj.data = [1,2,3,4,5]; return obj; });

Een voorbeeld

```
<script>

    var app = angular.module("myapp",[]);

    app.factory("myservice",function(){
        var obj = {};
        obj.users = ["tom","mieke","hannes","arno"];

        return obj;
    });


    app.controller("myctrl",function($scope, myservice){

        $scope.users = myservice.users;

    });

    app.controller("myctrl2",function($scope,myservice){
        $scope.getdata = function(){
            $scope.data = myservice.users;
        };

    });
</script>

    </head>

    <body ng-app="myapp">
        <div ng-controller="myctrl">

        </div>
```

```
        <div ng-controller="myctrl2">
            <ul>
                <li ng-repeat="x in data"></li>
            </ul>
            <button ng-click="getdata()" value="getdata">
                get data
            </button>
        </div>

    </body>
```

De service wordt ingeladen door dependency injection.

Oefening

Maak een service aan die 2 getallen kan optellen, aftrekken, vermenigvuldigen en delen. <

De factory methode maakt een singleton testService die 2 functies bevat. De controllers krijgen de UserService door deze te injecteren in de controller's functie als parameter.

De $http service die je regelmatig in een controller ziet opduiken, zou beter zijn om deze in een custom service te ontwikkelen, om zo meer beheerbaardere code te schrijven.

http://viralpatel.net/blogs/angularjs-service-factory-tutorial/

Eigen services aanmaken

De factory methode maakt een singleton UserService die 2 functies bevat. De controllers krijgen de UserService door deze te injecteren in de controller's functie als parameter.