İÇİNDEKİLER

ANGULAR JS

Google tarafından javascript dili üzerine MVC/MVVM yapısında geliştirilen, Single Page Application yaklaşımını benimseyen ve client side çalışan javascript frameworküdür.

Günümüz web yazılım geliştirme mantalitesi artık server dan olabildiğince uzaklaşıp tüm işlemleri kullanıcı(Client) tarafında yani bilgisayarımızda kullandığımız browserlar üzerinde yaptırmak istemektedir. Bunu sağlayan yapı javascript olduğu için hem daha hızlı hem de daha kolay javascript yazabilmek için farklı frameworkler vardır.

AngularJS ise MVC altyapısı ile hazırlanmış ve günümüzde popülaritesi gittikçe artan ve kuvvetli bir javascript frameworküdür. Temel yapısı module, directive, controller, service ve filterlardan oluşmaktadır.

JQuery ile karşılaştırmak gerekir ise ikisinin de temelde çalışma yapısı javascript üzerinedir. JQuery bugüne kadar daha çok sayfa üzerindeki html elementlere ulaşma ve bilgilerini style bilgilerini düzenlemeyi ve aynı zamanda JSON gibi veriyi nesneye dayalı bir şekilde yapıları taşımaya yarayan özellikleri kullanmamızı sağlar.

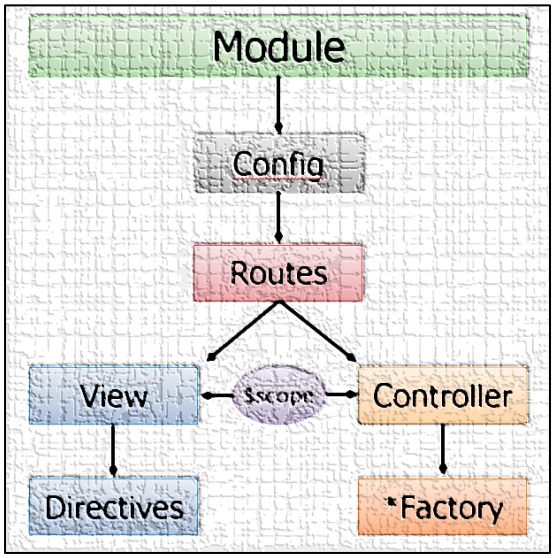
AngularJS JQuery ile aynı özelliklere ve daha fazlasına sahip olmasına rağmen hem JQuery den daha performanslı hem de kolay yazılabilir olması avantajları arasındadır.

AngularJS Çalışma Yapısı ve Çalışma Ortamı

AngularJS diğer javascript frameworkleri gibi web tarayıcı üzerinde ve web sayfaları ile çalışır. AngularJS frameworkünün çalışması için web sayfalarda ng-app directive'i olmalıdır.

AngularJS bu directive ile başlar ve sayfa içerisinde bulanan diğer directiveleride harekete geçirir. Ng-app kullanırken iki ayrım vardır eğer <script> </script> bloğu içerisinde bir module oluşturulduysa bu module tetiklenmesi için ng-app de kendi adını arayacaktır. Module oluştururken hangi isim ile çağıracağımızı belirlenir ve ng-app e de bu adı yazarak module tetiklerken diğer directivelerde çalışmaya başlar.

Genel yapısı resimde göründüğü üzere AngularJS frameworkü mvc yapısını kullandığı için her şey module den başlar module sayfada kullanılan tüm yapılardan sorumludur. Module içerisinde controller, directive ve servisler oluşturulur ve sayfa ile bağlantıyı controller kurar bu işlemi yapan köprümüze $scope nesnesi diyoruz.



$scope controller içerisinde oluşturulan dinamik değişkenler ile view(sayfa)'a değerleri gönderir.

AngularJS ve JQuery Karşılaştırması

Temel olarak aynı dil üzerinden ortaya çıkan ve ortaya çıkış amaçları farklı olan iki farklı framework söz konusu olunca doğal olarak karşılaştırmak isteriz. AngularJS ve JQuery özelliklerini aşağıdaki tabloda bulabilirsiniz. Bu tablo proje geliştirdiğimizde karar vermemize yardımcı olacaktır.

Özellikler	AngularJS	JQuery
Two Away Binding	✔	✖
Dependency Injection	✔	✖
Extension to HTML	✔	✖
Effects and Animations	✔	✔
Support for MVC	✔	✖
AJAX/JSONP	✔	✔
Form Validation	✔	✖

AngularJS Expression

AngularJS sayfalarda verileri göstermek için 'two a way binding' yapısını yani 'iki yönlü bağlamayı' kullanır. Bu kullanım verileri daha hızlı işlemeye yarar ve veri ile model arasında ki köprü kopana dek veri iletimi sürekli sağlanır.

AngularJs Expression ise "{{ }}" süslü parantezlerdir. Verileri temsil eden modelleri ya da verini kendini yazarak DOM üzerine veriyi ekleriz.

index.html

```
<div>
 <label>Değer</label>
   <div>
    <label>{{5+5}}</label>
   </div>
</div>
```

index.html

```
10
```

index.html

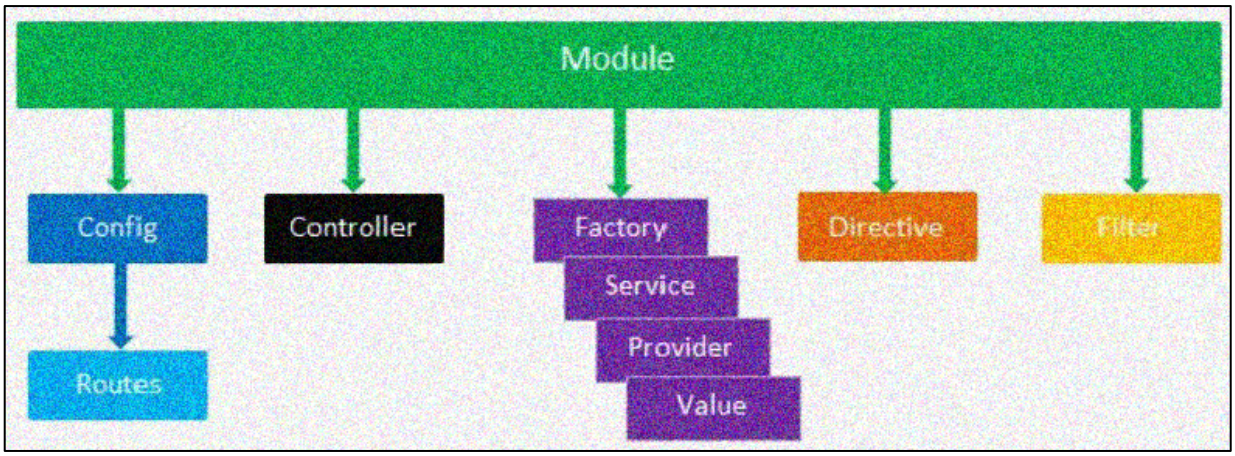
```
<input type="text" ng-model="txtKurumAdi" />
<p>
     {{ txtKurumAdi }}
</p>
```

İndex.html

BilgeAdam

```
BilgeAdam
```

AngularJS Moduler Yapısı

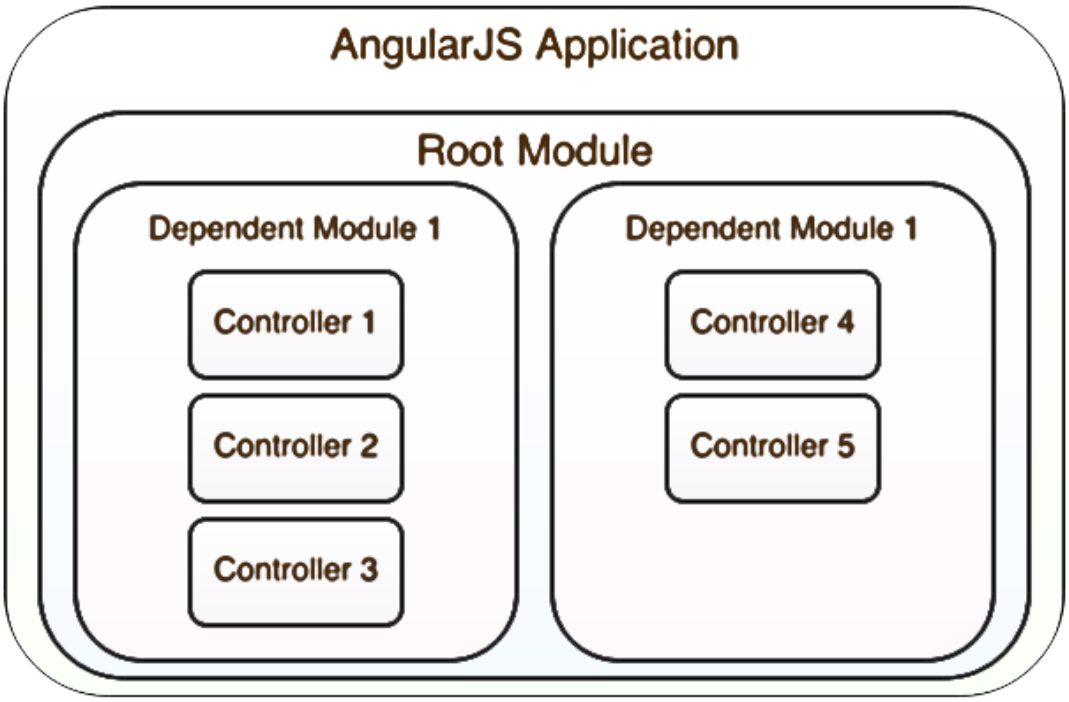


Modüller; controller, değişkenler, servisler, direktifler vb. yapıları içinde barındıran ve yöneten ana kod kısımlarıdır. Bağımlılıkları (Dependency) tanımlar ve yönetir, tekrar tekrar kullanılır. Tanımlanırken dikkat edilmesi gereken bir husus ise isim verirken ilk harf küçük sonraki kelimelerin ilk harfleri büyük yani lowerCamelCase olacaktır

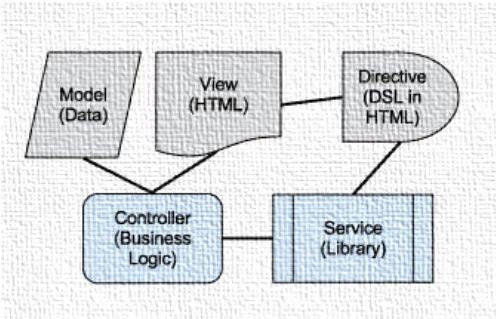
index.html

```
<script>
  var app=angular.module('modulAdi',[]);
      app.config(YapilandirmaAyarlari);
      app.run(CalismaKodlari);
</script>
```

AngularJS module Single Page Applicationlar için önemlidir. Tek sayfada uygulama yapabilmek için projenizi bir merkezden yönetmeniz gerekir ve module tam olarak bu işi yapar. Oluşturulan module içerisinde barındırdığı diğer modüleri kontrol edebilir ve onların çalışmasını sağlayabilir.



AngularJS Controller



Controllerlar AngularJS içerisinde önemli bir yere sahiptir. Controllerlar için view ile bağlantı kurar diyebiliriz fakat bu tanım sade ve biraz basit olacaktır.

Controllerlar sayfaları veritabanı ile haberleştirme işlemini yaparken aynı zamanda iç içe controller çalıştırma özelliği ile diğer controllerları tek bir çatı altında toplayabilir.

Asp.Net MVC kullanıcıları için tanıdık gelen bu isim sizi yanılgıya düşürebilir.

Tanımda söylediğim gibi sayfa ile veri arasındaki bağlantıyı kurar fakat burada Asp.Net MVC den ayrılan özelliği MVC Controllerlar tek başına çalışabilirken, AngularJS Contollerlar bir module ile ayağa kalkar ve module üzerinden çalışır. Anlamak için bir yapıya benzetilecek ise Asp.Net MVC de controller içerisinde kullanılan Actionlar daha anlamlı bir örnek olacaktır.

İsimlendirirken dikkat etmemiz gereken durum ise isimde geçen tüm kelimelerin ilk harfi büyük olacaktır.

index.html

```
<script>
  var app=angular.module('modulAdi',[]);
      app.controller('Selamla',SelamVer);
      SelamVer.$inject=['$scope']
      function SelamVer($scope) {
             $scope.Selam='Hello World'
      }
</script>
```

index.html	index.html
   {{SelamVer}} 	Hello World

Controllerlarda inject yöntemi ile bağımlılık vermenin dışında daha temiz bir kod yazımı için VM mantığını kullanabilirsiniz.

Vm olarak tanımlanan değişkene controllerın kendisi yani this değeri verilerek gerçekleştirilen işlemde $scope bağımlılığı ortadan kalkarken html içerisinde controller'ı tanımlanan değişkene cast etmemiz gerekmektedir.

index.html

```
<script>
  var app=angular.module('modulAdi',[]);
      app.controller('Selamla',SelamVer);
      function SelamVer($scope)
      {
           var vm =this;
           vm.Selam='Hello World'
           return vm;
      }
</script>
```

index.html	index.html
   {{vm.SelamVer}} 	Hello World

AngularJS Directives

Direktif HTML'ye yeni tagler öğretmenin bir yoludur. Html static yapıda bir işaretleme dilidir, içerisinde tagler ile yapılar oluşturulur ve bu taglerin iç özelliklerine de attribute denir. AngularJS directive ise ng ile kullanılan birçok özellik ve custom directive olmak üzere 2 ye ayrılır.

1. Ng-Directive : AngularJs framework yapısında hazırlanan ve html tagleri içerisinde kullanılan arka planda tanımlanan işleri çalıştırmak için hazırlanan hazır directivelerdir.
2. Custom Directiveler: AngularJs framework'ünü kullanarak biz geliştiriciler tarafından hazırlanan ve tag, attribute, class ya da comment olarak kullanılan directivelerdir.

NG- Directive

Ng-app AngularJS framework'ün başlaması için gerekli olan attributedur. AngularJs sayfa render aşamasında ng-app i arar ve ng-app i gördüğü yerde çalışmaya başlar bu işlem sayfanın farklı kısımlarında uygulanabilir.

- **Html tag:** Html tag içerisinde uygulanan ng-app sayfanın tamamında geçerli olma anlamını taşır.
- **Body tag:** Sadece body bloğunda çalışmasını istediğimiz durumlarda yazarız.
- **Custom:** Bu kısımda sayfamızda element bazlı çalıştıracağımız AngularJS yapısını gerçekleştiririz genelde farklı modülleri tek sayfada kullanmak için bu yönteme başvurulur.

ng-app="module_Adı" kullanımı ise tüm framework değilde tanımlı olan bir module uzerinden AngularJS framework u çalıştırılacak anlamına gelir.

ng directiveleri içerisinde sıklıkla kullanılanlara göz atmak gerekirse;

- **ng-model:** Giriş alanının değerini uygulama değişken adıyla ilişkilendirir.
- **ng-bind:** Html taglerinin innerHtml özelliklerine ng-model uzerindeki değeri yazar.
- **ng-init:** Eğer projemizde controllerlardan bir veri almayacaksak ama değer tanımlaması yapılmış bir değişken, dizi ya da nesneye ihtiyacımız var ise ng-init değişken ve değer tanımlaması yapmamız için bize yardımcı olur.
- **ng-show:** Verilen HTML öğesini ngShow özniteliğine verilen ifadeye dayanarak gösterir veya gizler.(true-false)
- **ng-hide:** Verilen ifadeye dayanarak verilen HTML öğesini gösterir veya gizler.(true-false)
- **ng-swicth:** Şartlı olarak bir kapsam ifadesine dayalı olarak şablonunuzda DOM yapısını değiştirmek için kullanılır. Bir anlamda c# da ki switch-case yapısı diyebiliriz.
- **ng-if:** {expression} temelli DOM ağacının bir bölümünü kaldırır veya yeniden oluşturur. Ng-If'e atanan expression yanlış bir değer olarak değerlendirilirse öğe DOM'dan kaldırılır, aksi takdirde öğenin bir klonu DOM'a yeniden yerleştirilir.
- **ng-class:** Static halde tanımlanan classlar yerine dinamik şekilde ng-model üzerinden hazırlayabileceğimiz classlar sunarlar.
- **ng-checked:** içindeki ifade mantıksal ise, işaretli özniteliği eleman üzerinde ayarlar. Bir nevi birden fazla seçimli listede hepsini seç ya da tüm seçimleri kaldır olarak adlandırılabilir.
- **ng-value:** Verilen ifadeyi öğenin değerine bağlar.

Dikkat: Daha detaylı bilgi ve daha fazla directive için https://docs.angularjs.org/api adresine bakabilirsiniz.

ng-model örnek:

index.html

```
Ek Açıklama Girilsin mi?: <input type="checkbox" ng-model="durum" value="" />
<div ng-show="durum">
     <p>
          Ek Açıklama :<input type="text" ng-model="tabloAdi" />
     </p>
</div>
```

Yukarıdaki ng-model örneğimizde sayfamıza ek açıklama girme istediğine göre açılıp kapanan bir text alanı ekledik bu alanın ek açıklama girmek isteyen kullanıcılar tarafından açılması halinde istedikleri açıklamayı girebilecekleri bir text alanı karşılarına gelecektir.

Bu işlemi gerçekleştirmek için AngularJS ng- direktiflerinden ng-model ve n-show kullandık. Ng-model kullanıldığı input'un bize vermiş olduğu değeri karşılar. Aynı zaman da ng-model e verilen değişken üzerine değer alabilen ve ng-model ile verildiği input üzerinde değeri değiştirebilen bir direktiftir.

ng-bind örnek:

index.html

```
    <p>
        <input type="text" ng-model="sayi" />
        <label ng-bind=”sayi”></label>
    </p>
```

ng-init örnek:

index.html

```
<div ng-init=”sayi=5”>
    <p>
         Atanan Değer:<label ng-bind=”sayi”></label>
    </p>
</div>
```

Custom Directive

Html içerisinde sürekli kullandığımız blokları directive olarak oluşturarak hem bunları tekrar yazmaktan hem de aynı işlemleri farklı controllerlar içerisinde oluşturmaktan kurtuluruz.

Çünkü sayfada kullandığımız her bir blok değerleri controllerlardan çekmelidir. Tekrarlanan alanlarda directiveler kurtarıcı olarak karşımıza çıkar. Oluştururken directive'in nasıl hizmet vereceğini belirleriz. Bunlar 4 çeşittir;

1. Element → E
2. Attribute → A
3. Class → C
4. Comment → M

Directive'i oluştururken restiric özelliğine bu 4 kısaltmadan kullanmak istediğimizi yazarak oluşturduğumuz hazır html bloğu ister element ister attribute(öz nitelik), ister class ya da yorum satırı olarak kullanabiliriz.

index.html

```
<div ng-app="myApp" class="row">
    <div class="col-lg-4">
      <bilgeadamelement></bilgeadamelement>
    </div>
</div>
```

```
<script>
        var myapp = angular.module("myApp", []);
        myapp.directive("bilgeadamelement",DirectiveOlustur )

function DirectiveOlustur() {
            return {
                restrict: 'E',
                template:'<h4> Bilge Adam AngularJS</h4>'
            }
        })
</script>
```

Yukarıdaki örneğimizde gördüğünüz gibi element kullanımı olan custom directive oluşturuldu ve sayfa üzerinde nasıl çağırıldığı gösterildi.

Direktif oluştururken önce module oluşturmak gerekir. AngularJS de yapı module üzerinden ayağa kalkar. Eğer direktif içerisinde bağımlılık isteyen bir yapı kullanılacak ise bu modülde mutlaka belirtilmelidir. Sonrasında module üzerinden directive ayağa kaldırılır. Directive oluştururken isim vermek önemlidir kullanırken de önemlidir. Bunun sebebi oluştururken verilen isim CamelCase, kullanılırken verilen isim lowercase olmalıdır.

İsimlendirme yaptıktan sonra function ismi belirlenir ve function oluşturularak directive hazır hale getirilir. Directiveler bize template sunduğu için return snytaxı içinde yazılır.

Restrict : Directivein kullanım şeklini belirler ‘E’ element anlamına gelir.

Template : Directivin oluşturacağı görseli belirler ki burada templatei oluşturmak biz yazılımcılara bağlıdır.

Diğer directive oluşturma seçenekleri aşağıdaki gibidir :

Attribute kullanımı :

index.html

```
<div ng-app="myApp" class="row">
    <div class="col-lg-4">
        <div bilgeadamattribute></div>
    </div>
</div>
<script>
        myapp.directive("bilgeadamattribute", function () {
            return {
                restrict: 'A',
                template: '<h4> Bilge Adam AngularJS attribute</h4>'
            }
        })
```

```
</script>
```

Class Kullanımı :

index.html

```
<div ng-app="myApp" class="row">
    <div class="col-lg-4">
         <div class="bilgeadamclass"></div>
    </div>
</div>
<script>
        var myapp = angular.module("myApp", []);
        myapp.directive("bilgeadamclass", function () {
            return {
                restrict: 'C',
                template: '<h4> Bilge Adam AngularJS class</h4>'
            }
        })
</script>
```

TemplateUrl Kullanımı:

index.html

```
<
    <div class="col-lg-4">
         <div bilgeadamurltemplate></div>
    </div>
</div>
<script>

        myapp.directive("bilgeadamurltemplate", function () {
            return {

                templateUrl: 'myDirective'
            }
        })
</script>
```

myDirective.html

```
<h4> Bilge Adam AngularJS TemplateUrl</h4>
```

TemplateUrl kullanımı ise directive içerisinde ki template urli verilen sayfadan gelmektedir. Burada önemli olan template adresini(url) doğru vermektir.

Aklımıza şöyle bir soru gelebilir sayfanın html yapısı mı? Yoksa html içeriği mi? Bu sorunun cevabı sadece render işlemidir. Yani tarayıcıda ne görüyor isek o bize template olarak gelecektir.

AngularJS Filters (AngularJS Filtreleri)

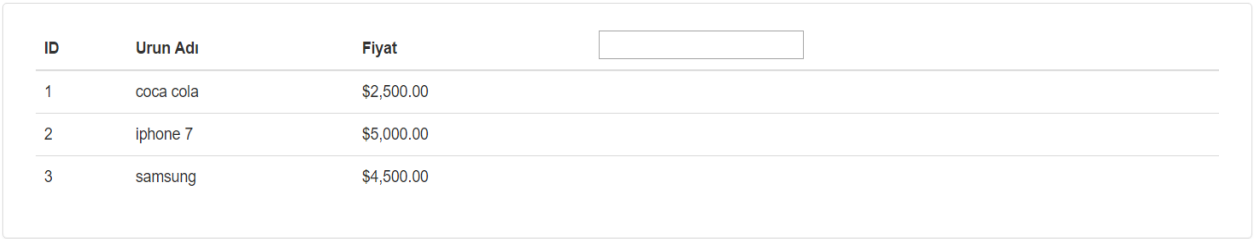
AngularJS Filtreleri sayfada render edilmiş template üzerinde verileri filtrelememizi sağlar. Normalde veriler veri tabanından her zaman düzgün formatta ya da şekilde gelmezler. Ya da verileri veri tabanından sorgulayarak değil sayfa üzerinde ayıklamamız gerekebilir. Tüm Harfleri büyük ya da tüm harfleri küçük kullanmak isteyebiliriz. Filtreler tam olarak bunu tek satır kod yazdırmadan bize sağlıyor.

index.html

```
<div ng-app="myApplication" class="row" ng-controller="KategoriController" style="margin-
top:50px;">
    <div class="panel panel-default">
        <div class="panel-body">
            <div class="col-lg-12">


  <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>ID</th>
                            <th>Urun Adı</th>
                            <th>Fiyat</th>
                            <th><input type="text" ng-model="test" /></th>
                        </tr>
                    </thead>
                    <tbody>
                        <tr ng-repeat="x in kategoriler| filter :test">
                            <td>{{x.ProductID}}</td>
                            <td>{{x.ProductName|lowercase}}</td>
                            <td>{{x.Price |currency}}</td>
                            <td></td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
```

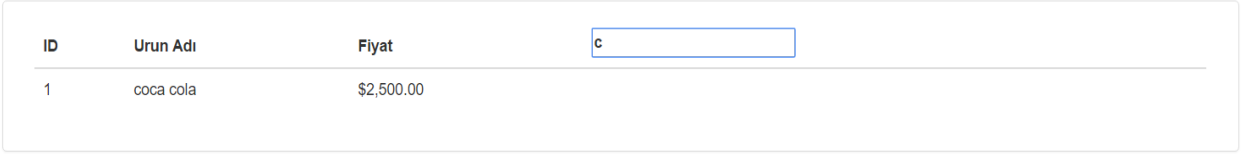
```
    </div>
</div>



<script>
        var myapp = angular.module("myApplication", [])
        myapp.controller("KategoriController", function ($scope) {
            $scope.kategoriler = jsondeger;

        })
        var jsondeger = [{
            ProductID: 1,
            ProductName: "Coca Cola",
            Price: 2.5
        }, {
            ProductID: 2,
            ProductName: "Iphone 7",
            Price: 5000
        }, {
            ProductID: 3,
            ProductName: "Samsung",
            Price: 4500
        }]
</script>
```

Yukarıdaki kod bloğunda currency filter uygulanan fiyat kolonu 2.500 rakamının başına $ işaretini getimiştir.



ID	Urun Adı	Fiyat
1	coca cola	$2,500.00
2	iphone 7	$5,000.00
3	samsung	$4,500.00

İnput içerisinde tanımlanan model ise tr bloğunda filter değeri olarak kullanıldığında dinamik bir search işlemine sahip oluruz ve aranan kelimelere uygun değerler gösterilirken diğer değerler saklanır. Arama alanı temizlendiğinde tüm değerler tekrar gösterilir.



ID	Urun Adı	Fiyat
1	coca cola	$2,500.00

AngularJS Animation

AngularJS ile JQuery de olduğu gibi animasyonlar yapmak mümkün. Biraz farklı bir yol izleminiz gerekse de animasyonları oluşturmak ve kullanmak, hızlı ve kolay bir iş.

AngularJS ile animasyon oluşturmak için kullanılan yapılar;

1. Css kodlama
2. Ng-Directiveleri

Olmak üzere ikiye ayrılır.

Aşağıda gördüğünüz örneğimizde oluşturulan listemizde search işlemi yaptığımızda listeden kaldırılan veriler css dosyasında tanımlanan özelliklere göre hareket etmektedir. Alışılanın aksine bu css değerlerini class ile değil ng-animate ile izliyoruz.

Ng-animate tanımlanan css fonksiyonlarını dinler ve içerisinde bulunduğu blokta bu fonksiyonları gerçekleştiren yapılar var ise harekete geçer ve kodları çalıştırır. Class attribute ise dinleme yapmadan render aşamasında tanımlanan css işlemlerini gerçekleştirir. Bu özelliği ile AngularJs sadece sayfalarda verileri dinamik olarak kontrol etmiyor neredeyse erişimi olan her alanda dinamik dinleme ve izleme yaparak olayları daha az kod bloğu ile daha hızlı gerçekleştirmektedir.

index.html

```
<body ng-app="" ng-init="names=['Igor Minar', 'Brad Green', 'Dave Geddes', 'Naomi Black',
'Greg Weber', 'Dean Sofer', 'Wes Alvaro', 'John Scott', 'Daniel Nadasi'];">
    <div class="well" style="margin-top: 30px; width: 200px; overflow: hidden;">
        <form class="form-search">
            <div class="input-append">
                <input type="text" ng-model="search" class="search-query" style="width: 80px">
                <button type="submit" class="btn">Search</button>
            </div>
            <ul class="nav nav-pills nav-stacked">
                <li ng-animate="'deger'" ng-repeat="name in names | filter:search">
                    <a href="#"> {{name}} </a>
                </li>
            </ul>
        </form>
    </div>
</body>
```

index.css

```
.deger-enter,
.deger-leave
{
    -webkit-transition: 800ms cubic-bezier(0.250, 0.250, 0.750, 0.750) all;
    -moz-transition: 800ms cubic-bezier(0.250, 0.250, 0.750, 0.750) all;
    -ms-transition: 800ms cubic-bezier(0.250, 0.250, 0.750, 0.750) all;
```

```
    -o-transition: 800ms cubic-bezier(0.250, 0.250, 0.750, 0.750) all;
    transition: 800ms cubic-bezier(0.250, 0.250, 0.750, 0.750) all;
    position: relative;
    display: block;
}

.deger-enter.deger-enter-active,
.deger-leave {
    opacity: 1;
    top: 0;
    height: 30px;
}

.deger-leave.deger-leave-active,
.deger-enter {
    opacity: 0;
    top: -50px;
    height: 0px;
}
```

Ng-directiveleri ile yapılan animasyonlarda ise kilit unsur model ilişkisidir. Bir checkbox yardımı ile sayfada bulunan divi aç-kapa yapmak istediğimiz zaman ki bunu JQuery de yapmak isteseydik eğer önce checkbox kontrolünü yakalayıp sonra onun değerine göre div etiketini yakalatıp show özelliğini hide ya da tam tersi bir işlem yapacaktık.

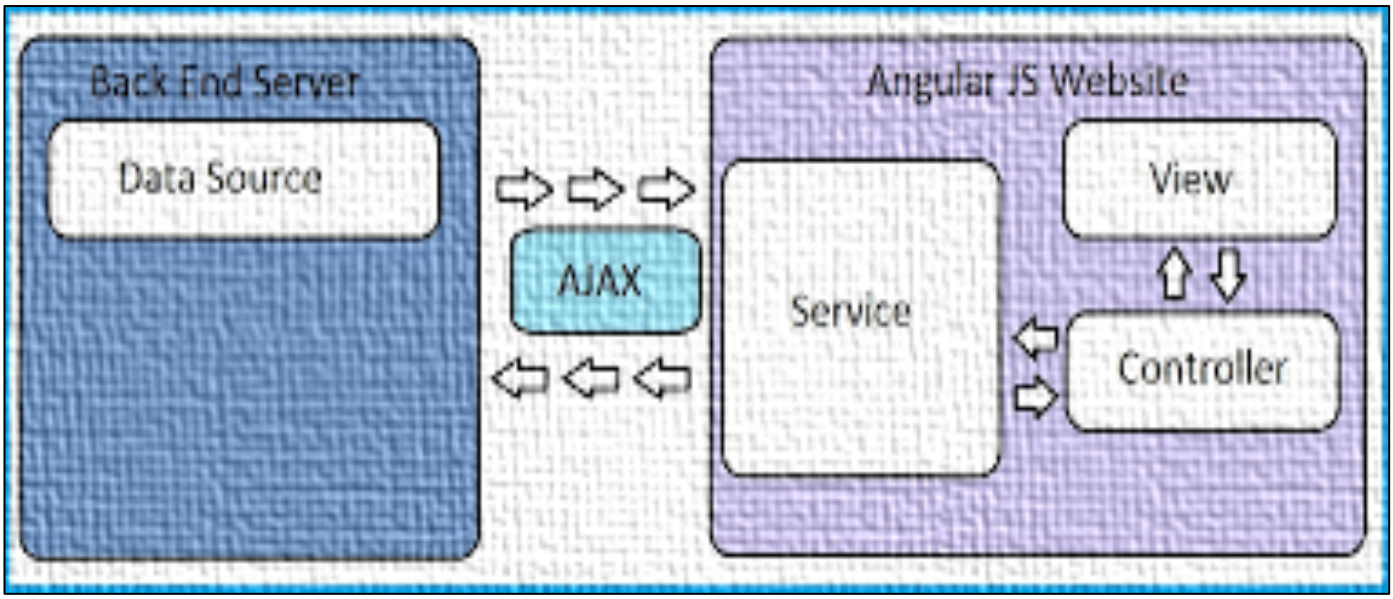
AngularJS ise 'two a way binding'i sadece model-bind ilişkisini için değil üzerine değer alabilen tüm directiveler için geçerli kıldığından checkbox'ın geriye döndüğü değeri yani true ya da false'u yakaladığımızda ve bunu aç-kapa yapmak istediğimiz bir tag'in ng-show veya ng-hide özelliğine tanımlamamız yeterlidir. Bir model eklemek bu işi çözmek için yeterlidir.

index.html

```
<head>
    <title></title>
    <script src="Scripts/angular.js"></script>
    <style>
        .content-area {
  border: 1px solid black;
  margin-top: 10px;
  padding: 10px;
}

.sample-show-hide {
  transition: all linear 0.5s;
```

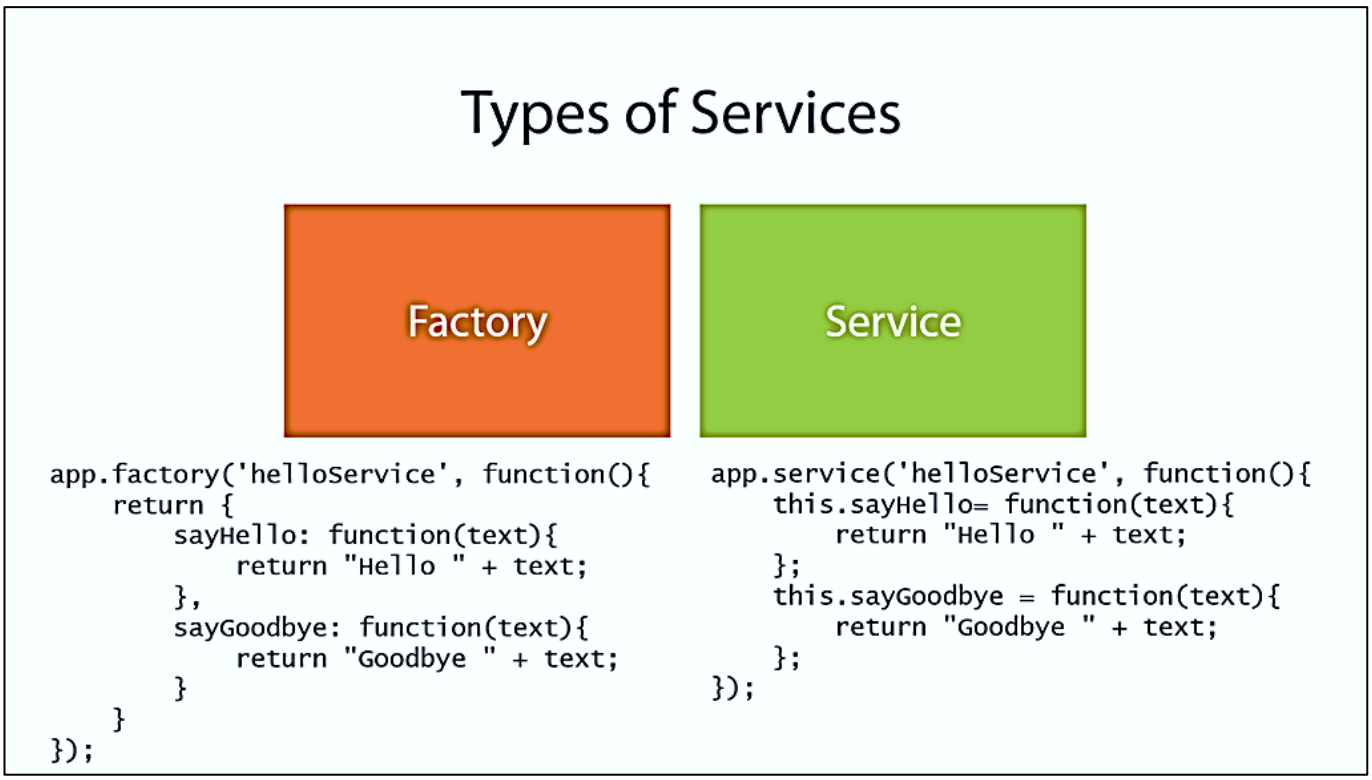
```
}
.sample-show-hide.ng-hide {
  opacity: 0;
}
    </style>
</head>
<body ng-app="">
    <div ng-init="checked = true">
        <label>
            <input type="checkbox" ng-model="checked" />
            Is visible
        </label>
        <div class="content-area sample-show-hide" ng-show="checked">
            Content...
        </div>
    </div>
</body>
```

AngularJS Service



AngularJS de servisler 2 ye ayrılır bunlar;

1. Factory servisler
2. Services servisler



Factory servisler: Projemize dışarıdan veri almak istediğimizde kullanmamız önerilen servislerdir. Temelde verinin nereden geldiği önemli değildir. Başka bir proje ya da bir servis veri kaynağı olabilir. Önemli olan verinin geldiği tip burada Json kullanmak önemlidir.

Services servisler: Projemizde bulunan diğer controllerlar aracılığı ile veri çekmek için kullanılan yapılardır.

İki servis yapısı da içinde $http directive'ini kullanır. $http servislerin istenilen URL yada data kaynağına gitmesini sağlayan framework directivedir.

$http ile istek yapılan adres den gelen cevaplar izlenir bu durumlar işlem tamamlanması ya da hata çıkması gibi cevaplardır. İstek yapmak için $http.get(adres) metodu kullanılır. Eğer işlemimiz hatasız bir şekilde çalıştıysa $http.get(adres).then(cevap) ile cevap karşılanır ve ya $http.get(adres).then(cevap).error(hata) ile hata dinlenebilir.

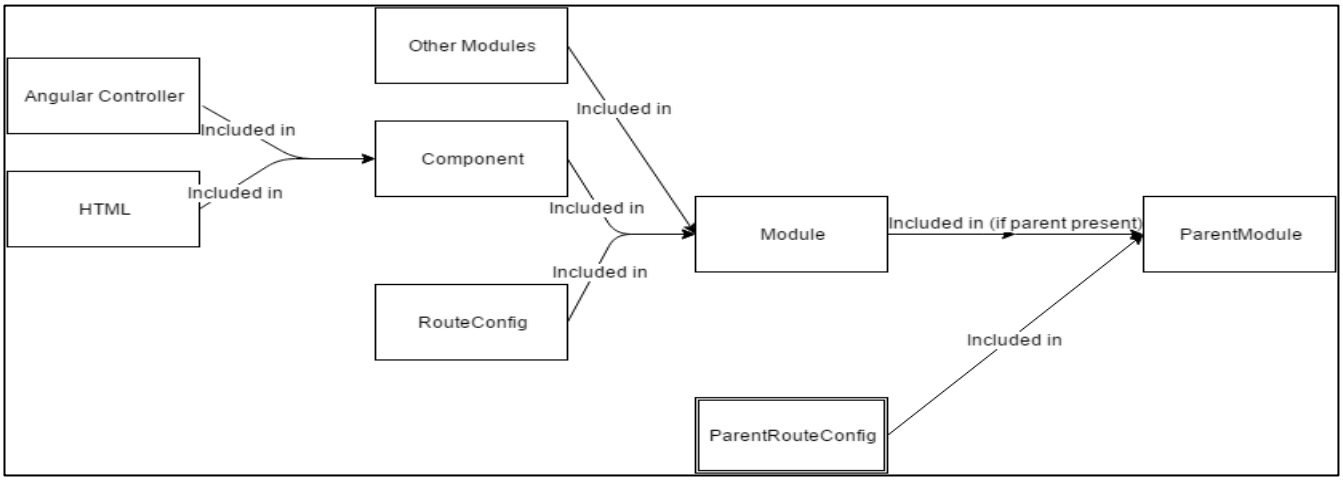
Her iki yöntem de Ajax işlemleriniz için kullanılabilir yalnız geliştiriciler web api gibi restfull çalışan servisler de Factory servisleri önermektedir.

index.html

```
<script>
        BlogApp.controller('KategoriyeGoreMakale', function ($scope, kategoriyegoregetir) {
            kategoriyegoregetir.list(function (x) {
                $scope.Articles = x;
            })
        })
        BlogApp.factory('kategoriyegoregetir', function ($http, $routeParams) {
            return {
                list: function (x) {
                    $http.get("/Home/CategoryArticles/" + $routeParams.makaleid).success(x);
                }
```

```
            };
        });
</script>
```

AngularJS Route



AngularJS Route uygulamamızda ki viewler ile kullanıcının gitmek istediği sayfalar arasında ilişki kurarak istenilen sayfaların yönlendirilmesini gerçekleştirir. Bunu yaparken işlem Asynchronous(Asenkron) gerçekleşir.

Angular Route AngularJS içinde gömülü halde bulunmadığı için mutlaka inject (yerleştirmek) edilmelidir bu işlem aşağıdaki gibi yapılmaktadır.

İndex.html

```
<script>
var app = angular.module('myApp', ['ngRoute']);
</script>
```

Yukarıda gördüğümüz işlem AngularJs de Dependency Injection dediğimiz yöntemdir. Module Route yapısını kullanabilmek için ngRoute u injecte etmektedir.

Bu işlem tamamlandıktan sonra angularjs module yapısının içinde route ayarları tanımlanır. Bu işlem module yapısı içinde Config bloğunda gerçekleşir.

$routeProvider yönlendirmeleri yapdığımız provider yapısıdır. $locationProvider ise sayfalara nasıl yönlendirileceğimizi belirler ve kısaca örneklemek gerekirse angularda route yönlendirmesinde urlde # işareti görünür bu işareti kaldırmak istersek $locationProvideri kullanırız.

Bu iki provideri önce inject etmemiz gerekmektedir bu işlem aşağıdaki gibi olur.

index.html

```
<script>
var app = angular.module('myApp', ['ngRoute']);
    app.confing(RouteOlustur);

    RouteOlustur.$inject = ["$routeProvider", "$locationProvider"];
</script>
```

Artık sıra yönlendirme işlemlerini gerçekleştirmeye geldi bu işlemler RouteOlustur functionı içinde gerçekleştirilir.

index.html

```
<script>
var app = angular.module('myApp', ['ngRoute']);
    app.confing(RouteOlustur);

    RouteOlustur.$inject = ["$routeProvider", "$locationProvider"];
function RouteOlustur($routeProvider,$locationProvider) {

        $routeProvider
            .when("/", {
                templateUrl: "intro.html",
                controller: "IntroController"
            })
            .when("/about", {
                templateUrl: "about.html",
                controller: "AboutController"
            })
            .otherwise({
               redirectTo: "/"
           })

        $locationProvider.html5Mode({
            enabled: true,
            requireBase: false
        });

    }
</script>
```

Yukarıda bulunan örnekte RouteOlustur functionı içinde $routeProvider ın when metodu ile yönlendirme ayarları yapılır.

When metodunun ilk parametresi yapılan isteğin urlini temsil eder. İkinci parametresi ise urlin nereye yönlendirileceğini ve sayfanın gerekliliklerini belirlediğimiz alandır.

templateUrl: Yönlendirme yapılacak olan sayfa yolu.

Controller: Yönlendirme yapılan sayfada tanımlı bulunan controllerın yönlendirme işleminde çalışması için kullanılır.

Eğer kullanıcı sayfada yanlış bir yönlendirme yaptıysa bu sefer de otherwise seçeneğine yönlendirilir ve bu aşamada genelde yol olarak başlangıç sayfamız verilir.

$locationProvider.html5Mode metodu parametre olarak true değerini aldığında yönlendirme yapılan urlde # işaretinin olmamasını sağlar ve yönlendirme daha başarılı gerçekleştirilir.

AngularJS CRUD

AngularJS de Create Read Update Delete işlemlerini gerçekleştirirken $http servisini kullanırız. Bu servis üzerinden kullanılan komutlar ile ekleme silme güncelleme ve listeleme işlemleri gerçekleştirilir.

Bu komutlar ekleme işlemleri için post, update işlemleri için put, silme işlemleri için delete ve listeleme işlemleri için get olarak adlandırabiliriz.

Post a bir parantez açmak gerekirse ;

Post aslında post requestden gelir ve amacı elde edilen verinin başka bir yöne doğru yönlendirilmesidir. İnsert işleminde ise bize daha önce gelen sayfa üzerinde bilgileri doldurarak veriyi tekrar server a gönderdiğimiz için bu işlemi ekleme mekanizması içinde kullanırız.

Şimdi kısa örneklerle nasıl kullanıldıkları hakkında örnekler yapalım. Örneğimize geçmeden önce AngularJs server side bir dil olmadığı için veri tabanı kullanımı Rest servisler ile karşılarız. Bu örneklerimizde kullandığımız servis Northwind veri tabanında bulunan Categories tablosunun crud işlemlerini içeren bir servistir.

index.html

```
<div  ng-controller="InsertController as vm">
 <form novalidate class="simple-form">

   <label>CartegoryName: <input type="text" ng-model="vm.category.CategoryName"  /></label><br
/>

  <label>Description: <input type="text" ng-model="vm.category.Description" /></label><br />

  <br />

  <input type="button" ng-click="vm.reset()" value="Temizle" class="btn btn-warning" />

  <input type="button" ng-click="vm.insert(vm.category)" value="Kaydet" class="btn btn-
primary" />

 </form>

</div>
```

index.html içerisinde oluşturduğumuz form bize veri tabanına kategori göndermek için tasarlanmıştır. Bu işlemde angularjs factory servis kullanılarak veri tabanı ile haberleşecektir. Form üzerinde bulunan inputlar contorollerdan gelen category nesnesi üzerinden değerleri servise gönderecektir. Butonlardan ng-click =”vm.reset()” metodu form üzerindeki nesneleri temizler. Diğer buton ise servisi ayağa kaldıran fiction’ı tetikler.

Bu örneğimizin script kısmında ise controller ve http servis üzerinden veri tabanına veri gönderimi vardır.

index.html

```
<script>
var app= angular.module('myapp', []);


 app.factory("KategoriGonderFactory", KategoriGonder);

 KategoriGonder.$inject = ['$http'];

 function KategoriGonder ($http) {
         var Post = {};
         Post = $http.post("http://localhost:10459/api/Categories", category);
         return Post;
     }
```

```
     app.controller('InsertController', InsertController);

     InsertController.$inject = ['$http', 'KategoriGetirFactory'];

     function InsertController($http, KategoriGetirFactory) {

        var vm = this;
        vm.category = {};

    vm.insert = function (category) {

      KategoriGetirFactory.then(Success);

    };
    function Success(response) {
        alert("Ekleme İşlemi Gerçekleştirilmiştir!");
    }

    vm.reset = function () {
        vm.category = {};
    };
return vm;
}
</script>
```

Güncelleme işlemi ise put ile kullanır ve ama bu işlemde önce bir kategori seçilmelidir. Bu işlem için http.get kullanılacaktır.

Aşağıdaki örnekte hem get hem de put işlemi birlikte yapılacaktır.

index.html

```
<body>
    <div  ng-controller="InsertController as vm">
        <div >
            <div class="col-md-6">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>
                                CategoryID
                            </th>
                            <th>
                                CategoryName
                            </th>
                            <th>
                                Description
                            </th>
                            <th>
                                Duzenle
                            </th>
                        </tr>
                    </thead>
                    <tbody>
                        <tr ng-repeat="category in vm.Categories">
                            <td>
                                {{category.CategoryID}}
                            </td>
                            <td>
                                {{category.CategoryName}}
```

```
                            </td>
                            <td>
                                {{category.CategoryName}}
                            </td>
                            <td>

      <input type="button"  value="Guncelle" class="btn btn-success" ng-
click="vm.select(category.CategoryID)" />
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>


        <div>


  <form novalidate class="simple-form">

                    <input type="hidden" name="name" ng-model="vm.category.CategoryID"  />
                    <label>CategoryName: <input type="text" ng-model="vm.category.CategoryName"
/></label><br />
                    <label>Description: <input type="text" ng-model="vm.category.Description"
/></label><br />
                    <br />
                    <input type="button" ng-click="vm.reset()" value="Temizle" class="btn btn-
warning" />
                    <input type="button" ng-click="vm.update(vm.category)" value="Güncelle"
class="btn btn-success" />


            </form>

        </div>
    </div>
</body>
```

Güncelleme işlemlerinde önce güncellenecek olan kategori seçilir ve inputlara değerler atanır. Burada dikkat edilmesi gereken durum bir kategoriyi güncellemek için onun kim olduğunu veri tabanına söyleyebilmek adına hidden bilgi içinde ID değeri saklanır ve ng-model kullanılan inputlar üzerinde ki bilgiler servis aracılığı ile veri tabanına iletilir.

Bu işlemler aşağıda bulunan kod blokları ile gerçekleştirilecektir.Unutmayın önce güncellemek istenen değer veri tabanından alınır sonra üzerinde tekrar değişiklik yapılarak veri tabanına gönderilir.

index.html

```
<script>
    var app= angular.module('myapp', []);

     app.factory("KategoriGetirFactory", KategoriGetir);
     KategoriGetir.$inject = ['$http'];
     function KategoriGetir ($http) {
         var Get = {};
         Get =$http.get("http://localhost:10459/api/Categories")
         return Get;
     }

     app.controller('InsertController', InsertController);
```

```
    InsertController.$inject = ['$http', 'KategoriGetirFactory'];

    function InsertController($http, KategoriGetirFactory) {

        var vm = this;
        vm.category = {};
        vm.Categories = {};

        KargoGetirFactory.then(
             function (response) {

                 vm.Categories = response.data;

             });



    vm.reset = function () {
        vm.category = {};
    };
    vm.reset();

    vm.update = function (category) {
        $http.put("http://localhost:10459/api/Categories", category).then(SuccessUpdate);
        function SuccessUpdate(response) {

            vm.Categories = response.data;
            alert("Update İşlemi Gerçekleştirilmiştir!");
        }
    }

    vm.select = function (CategoryID) {
         $http.get("http://localhost:10459/api/Categories/" + CategoryID).then(SuccessGetId);

        function SuccessGetId(response) {
            vm.category = response.data;
        }
    }
    return vm;
    }

</script>
```

Yukarıdaki örneğimizde tabloda Güncelle butonuna basıldığında güncellemek istenilen değer script içerisinde ki vm.select functionı ile veritabanından alınarak inputlara gönderilir. Sonrasında ise değiştirilen değerler form içinde bulunan güncelle butonu ile script kodları içerisinde ki update functionına iletilir.burada önemli olan değeri nesne halinde göndermektir. Bunun nedeni api bizden class yani nesne beklemektedir. Değerler nasıl birbirleri ile haberleşiyor diye merak ediyorsanız burada devreye JSON girmektedir.Json olarak gönderdiğimiz değer serialize edilerek class a dönüşür ve bizde c# içerisinde istediğimiz yerde kullanabiliriz.

Son olarak delete işlemini kullanırsak burada yine aynı işlemler gerçekleşir ve bu işlemlerde delete functionına değeri temsil eden id göndermek yeterlidir.Kullanımı ise;

index.html

```
<script>
    var app= angular.module('myapp', []);

     app.factory("KategoriGetirFactory", KategoriGetir);
     KategoriGetir.$inject = ['$http'];
     function KategoriGetir ($http) {
         var Get = {};
         Get =$http.get("http://localhost:10459/api/Categories")
         return Get;
     }

     app.controller('InsertController', InsertController);

     InsertController.$inject = ['$http', 'KategoriGetirFactory'];

     function InsertController($http, KategoriGetirFactory) {

        var vm = this;
        vm.category = {};
        vm.Categories = {};

        KargoGetirFactory.then(
             function (response) {

                 vm.Categories = response.data;

             });

vm.delete = function (CategoryID) {
        $http.delete("http://localhost:10459/api/Categories/" +
CategoryID).then(SuccessDelete);

    function SuccessDelete(response) {
        vm.Categories = response.data;
        alert("Silme İşlemi Gerçekleştirilmitir!")
    }
    };
    return vm;
    }

</script>
```

AngularJS CRUD işlemleri için gerekli olan bir api yada herhangi bir Rest servistir. Eğer servis üzerinde bulunan kodlar hatasız ise angularjs sadece adreslerini kullanarak ve doğru parametre vererek işlemlerini gerçekleştirmektedir. Burada $http servisinin özelliklerini kullanmak önemlidir. Eğer biz $http.delete() yerine $http.post() kullanırsak evet değer gidecektir ama api bu değeri post işlemi olarak algılar ve silme işlemini yapmaz. Sonuç olarak $http.methodadi() bizim yapmak istediğimiz işleri simgeler ve servis adreslerini buna göre yönlendirir.