

می‌خواهیم یک مثال ساده از دریافت اطلاعات از سرور و نمایش آن در یک View را توسط AngularJS، با هم بررسی کنیم. همانطور که می‌دانید برای نمایش تعدادی از اشیاء در انگولار می‌توان به این صورت نیز عمل کرد:

```
<div ng-init="products=[
  {id:1,name:'product1',price:25000,description:'description of product'},
  {id:2,name:'product2',price:5000,description:'description of product'},
  {id:3,name:'product3',price:5000,description:'description of product'},
  {id:4,name:'product4',price:2000,description:'description of product'},
  {id:5,name:'product5',price:255000,description:'description of product'}
]">
<div>
  <div>
    <table>
      <tr>
        <th>Id</th>
        <th>Product Name</th>
        <th>Price</th>
        <th>Description</th>
      </tr>
      <tr ng-repeat="product in products">
        <td>{{product.id}}</td>
        <td>{{product.name}}</td>
        <td>{{product.price}}</td>
        <td>{{product.description}}</td>
      </tr>
    </table>
  </div>
</div>
</div>
```

در کد فوق توسط ویژگی ng-init می‌توانیم داده‌هایمان را Initialize کنیم و در نهایت توسط ویژگی ng-repeat می‌توانیم داده‌هایمان را در صفحه نمایش دهیم. این ویژگی دقیقاً مانند یک حلقه foreach عمل میکند؛ مثلاً معادل آن در Razor سمت سرور، به این صورت است:

```
@foreach (var product in Model.products)
{
  <td>product.id</td>
  <td>product.name</td>
  <td>product.price</td>
  <td>product.description</td>
}
```

خوب؛ حالا می‌خواهیم این اطلاعات را از سمت سرور بخوانیم و به صورت فوق نمایش دهیم. ابتدا مدل مان را به این صورت تعریف می‌کنیم:

```
namespace AngularAndMvc.Models
{
  public class Product
  {
    public int Id { get; set; }
    public string Name { get; set; }
    public float Price { get; set; }
    public string Description { get; set; }
  }
}
```

سپس در داخل کنترلر زیر اطلاعات را به صورت in memory data تعریف می‌کنیم (جهت سهولت دموی کار) و به view مورد نظر پاس می‌دهیم. البته شما می‌توانید این اطلاعات را از دیتابیس بخوانید؛ روال کار فرقی نمی‌کند:

```
namespace AngularAndMvc.Controllers
```

```
{
    public class ProductController : Controller
    {
        public ActionResult Index()
        {
            return View("Index", "", GetSerializedProduct());
        }

        public string GetSerializedProduct()
        {
            var products = new[]
            {
                new Product{Id=1,Name="product1",Price=4500,Description="description of this product"},
                new Product{Id=2,Name="product2",Price=500,Description="description of this product"},
                new Product{Id=3,Name="product3",Price=400,Description="description of this product"},
                new Product{Id=4,Name="product4",Price=5500,Description="description of this product"},
                new Product{Id=5,Name="product5",Price=66500,Description="description of this product"}
            };
            var settings = new JsonSerializerSettings { ContractResolver=new
            CamelCasePropertyNamesContractResolver()};
            return JsonConvert.SerializeObject(products,Formatting.None,settings);
        }
    }
}
```

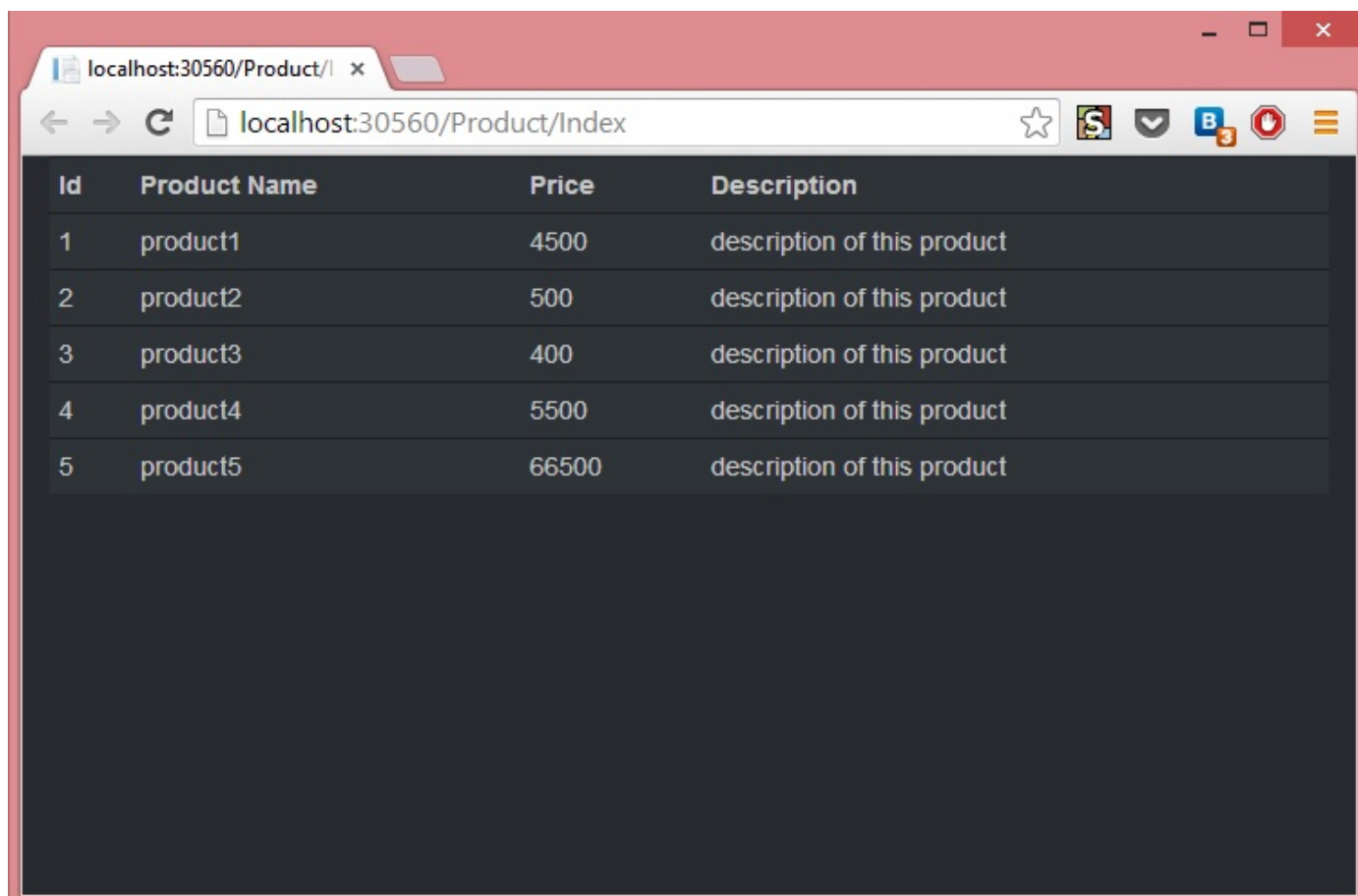
همانطور که در کد بالا مشخص است، اطلاعات را به صورت JSON به View مان پاس داده ایم و برای اینکه ابتدای نام مقادیر بازگشتی به صورت حروف بزرگ نباشند (به صورت خودکار تبدیل به camel case شوند) پارامتر settings را برای متد SerializeObject تعیین کرده ایم:

```
var settings = new JsonSerializerSettings { ContractResolver=new
CamelCasePropertyNamesContractResolver()};
return JsonConvert.SerializeObject(products,Formatting.None,settings);
```

view را نیز به این صورت تغییر می دهیم :

```
@model string
<div ng-init="products = @Model">
    <div>
        <div>
            <table>
                <tr>
                    <th>Id</th>
                    <th>Product Name</th>
                    <th>Price</th>
                    <th>Description</th>
                </tr>
                <tr ng-repeat="product in products">
                    <td>{{product.id}}</td>
                    <td>{{product.name}}</td>
                    <td>{{product.price}}</td>
                    <td>{{product.description}}</td>
                </tr>
            </table>
        </div>
    </div>
</div>
```

تنها تغییری که در کد فوق اعمال شده است، به جای اینکه ویژگی ng-init را به صورت inline مقادیردهی کنیم آن را از کنترلر دریافت کرده ایم. در خروجی هم اطلاعات به این صورت نمایش داده می شوند :



The screenshot shows a web browser window with the address bar displaying 'localhost:30560/Product/Index'. The browser's address bar also shows a tab for 'localhost:30560/Product/'. The main content area of the browser displays a table with the following data:

Id	Product Name	Price	Description
1	product1	4500	description of this product
2	product2	500	description of this product
3	product3	400	description of this product
4	product4	5500	description of this product
5	product5	66500	description of this product

سورس مثال فوق را هم از [اینجا](#) می توانید دریافت کنید.

## نظرات خوانندگان

نویسنده: حمید رضا منصوری  
تاریخ: ۱۱:۱۵ ۱۳۹۳/۰۱/۱۵

با تشکر  
من از این روش در یک view استفاده کردم (در اینجا هدف نمایش لیست کاربران بود) ولی مشکل من این هست که وقتی این view برای کاربر نمایش داده میشه چون انگولار اون رو کش میکنه با افزودن کاربر جدید این لیست تا درخواست مجدد از سرور بروز نمیشه.

میخواستم بدونم آیا با واکنشی اطلاعات توسط \$http مشکلم حل میشه؟  
یا میشه یک view در انگولار کش نشود و به هر با route به اون view مورد نظر از سرور فراخوانی بشه

نویسنده: سیروان عقیقی  
تاریخ: ۱۱:۵۲ ۱۳۹۳/۰۱/۱۵

البته میتونید کش رو سمت سرور با اعمال outputcache بر روی اکشن خودتون غیر فعال کنید:

```
[OutputCache(NoStore = true, Duration = 0, VaryByParam = "None")]  
public ActionResult Index()  
{  
    return View("Index", "", GetSerializedProduct());  
}
```

نویسنده: محسن درپرستی  
تاریخ: ۱۰:۵۳ ۱۳۹۳/۰۱/۱۶

در روش بالا اگر چه اطلاعات از سرور دریافت میشود اما حالتی استاتیک دارد چون لیست محصولات در زمان رندر شدن صفحه تولید و در ng-init قرار میگیرد. و در نتیجه برای بروزکردن حتما باید صفحه مجددا درخواست بشود (رفرش). این روش برای لیست هایی که تغییراتی ندارند یا به ندرت تغییر می کنند مناسب است.  
و بله یک راه برای حل این مشکل استفاده از سرویس http می تواند باشد.