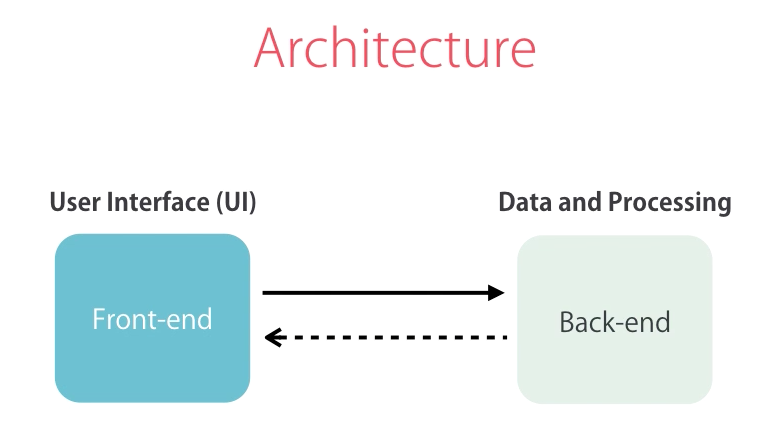
# BAB 8

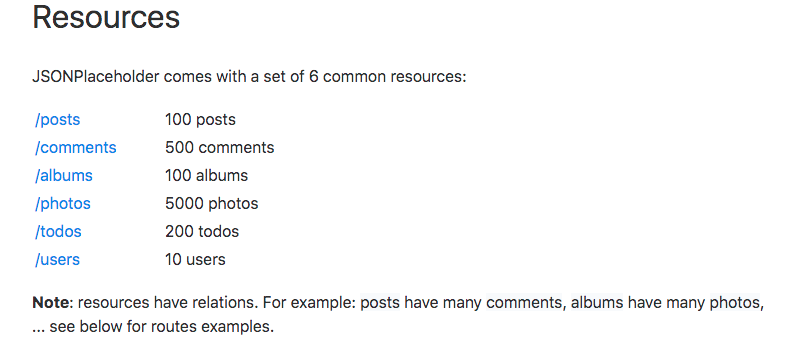
# HTTP SERVICES

**8.1 Jsonplaceholder**

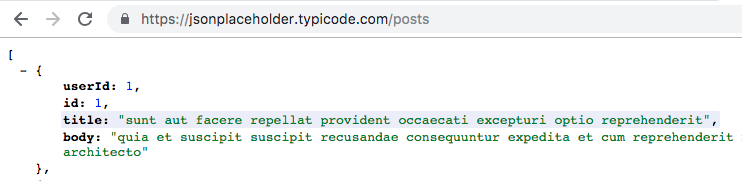
Angular adalah sebuah kerangka kerja dari front-end untuk membuat sebuah aplikasi klien. Namun di sebagian besar aplikasi di dunia memerlukan sebuah layanan HTTP di server untuk mendapatkan atau menyimpan sebuah data. Sebagai pengembang atau developer angular terdakang tidak memerlukan layanan tersebut dikarenakan angular adalah kerangka kerja dari sebuah front-end.



Terkadang juga kita perlu memahami konsep sebuah layanan HTTP yang nantinya digunakan sebagai backend. Untuk lebih jelasnya buka website <https://jsonplaceholder.typicode.com/> dan lihat pada bagian resources.



klik bagian /post maka akan terdapat banyak objek json disini dan di setiap objek terdapat 4 properti yaitu userID, id, title, body.

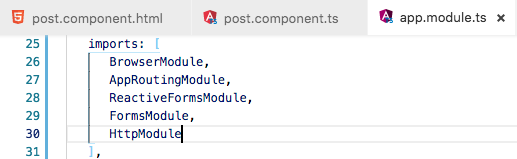


jadi dengan menggunakan <https://jsonplaceholder.typicode.com/posts> kita dapat melakukan pengiriman data ke jsonplaceholder. Ada beberapa macam jenis permintaan di HTTP seperti membuat, memperbaiki dan menghapus sebuah postingan. Tetapi jika kita menggunakan layanan resource pada jsonplaceholder ini sebagai proses layanan HTTP, tidak akan merubah isi dari resource pada jsonplaceholder. Hal ini dikarenakan tidak ada layanan database di balik layanan HTTP ini (jsonplaceholder).

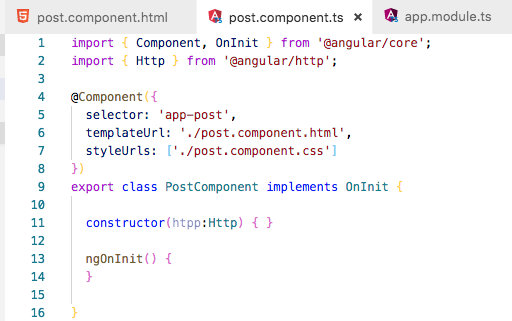
**8.2 Getting Data**

Pada sub bab ini akan mencoba layanan HTTP service untuk mendapatkan data dengan method GET. Untuk mencoba layanan HTTP ini nantinya akan menggunakan data pada jsonplacehoder, untuk lebih jelasnya lakukan tahap-tahap berikut:

* buatlah sebuah component dengan nama post
* buka app.module.ts dan tambahkan HttpModule pada bagian import ( jika terdapat error can’t find module ‘@angular/http’ maka lakukan perintah install http seperti berikut npm install @angular/http@latest

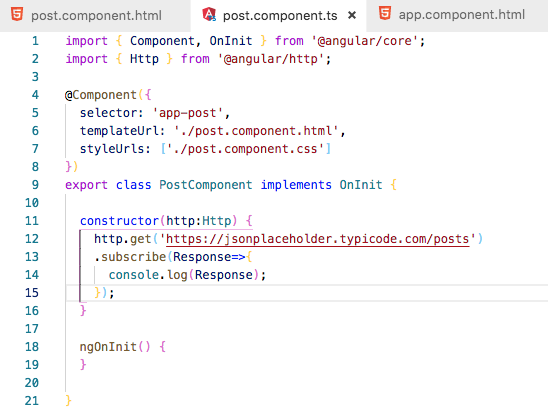


* buka file post.component.ts dan tambahkan parameter http pada constructor seperti berikut



Penjelasan code

* parameter http ini nanti digunakan untuk menyimpan data, mengirim data ke backend
* tambahkan beberapa code constructor pada post.component.ts seperti berikut

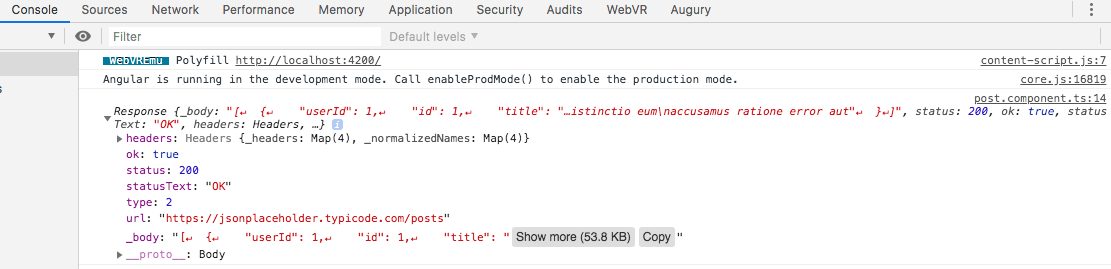


Penjelasan code

* line 12 adalah method get dimana parameternya sebuah string alamat dari jsonplaceholder



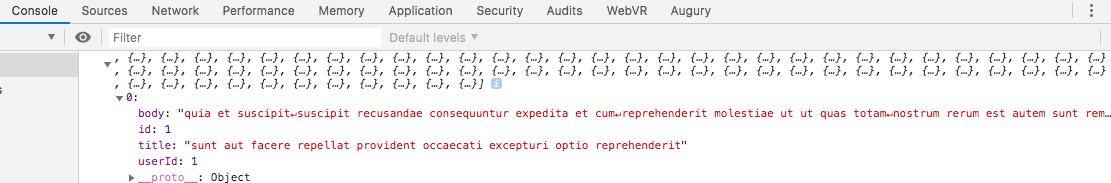
* agar hasil dari method get dapat diamati maka dibutuhkan sebuah method subcribe (line 13) dengan parameter function response
* line 14 untuk menampilkan pada log console
* jalankan server localhost dan amati pada console lognya



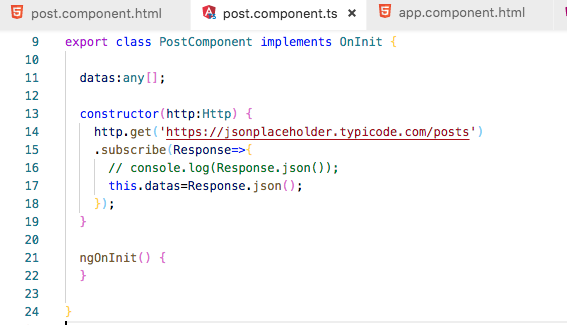
* kita dapat merubah isinya dalam bentuk objek dengan cara menambahkan code berikut pada console.log

../../../../../Users/dimaswahyu/Desktop/Screen%20Shot%202019-04-13%

* amati hasilnya pada console.log

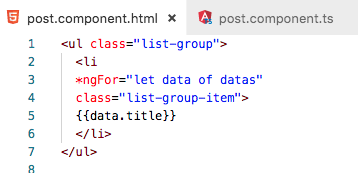


* jika ingin menampilkan data maka tambahkan code berikut pada post.component.ts

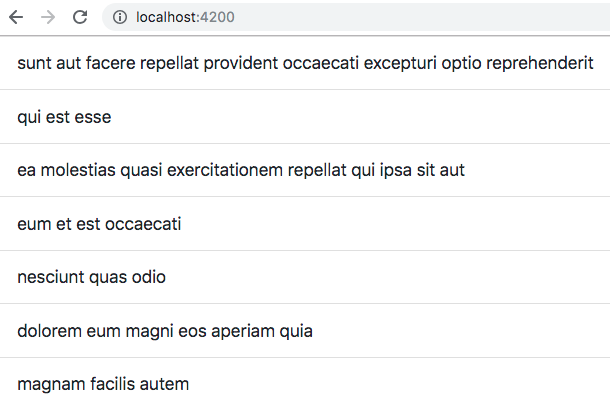


Penjelasan code:

* line 11 membuat property baru dengan tipe ada any yang disimpen array
* line 17 merubah data menjadi json
* setelah itu tambahkan juga pada post.component.html



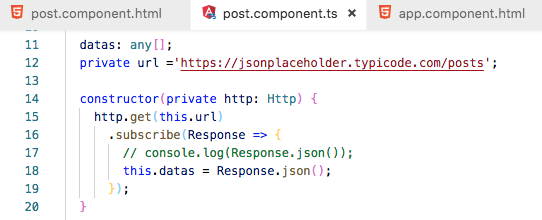
* sehingga jika dijalankan hasilnya seperti berikut



**8.3 Creating Data**

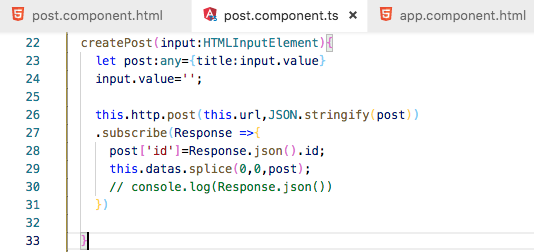
Pada sub bab ini menjelaskan bagaimana menambahkan data pada layanan HTTP. Untuk lebih jelasnya ikuti langkah berikut

* buka file post.component.html dan tambahkan sebuah property url untuk menampung url dari Jsonplaceholder seperti berikut berikut



Penjelasan code

* pada line 12 adalah sebuah property yang digunakan untuk menampung nilai string berupa alamat url dari Jsonplaceholder. Hal ini dilakukan agar mempermudah penulisan
* pada line 15 parameter get yang awalnya berupa alamat url dari jsonplaceholder diganti dengan this.url
* pada line 14 tambahkan private pada parameter http
* buatlah method dengan nama createPost apa post.component.ts dan tambahkan code seperti berikut ( sebelum membuat method createPost, pastikan terdapat privilege private pada parameter http -> line 14)



Penjelasan code

* line 22 adalah sebuah method dengan nama createPost dengan parameter input dan berupa HTMLInputElement
* line 23 sebuah object **post** dengan tipe data any dan dikarenakan nanti pada line 26 terdapat sebuah parameter (parameter kedua => JSON.stringify(post)) yang dimana data harus diterima berupa objek maka dibuatlah sebuah object dengan nama **post**. Dimana object post ini harus mempunyai property title dan value dari title tersebut
* line 24 mengset input.value dengan nilai string kosong
* line 26 adalah method post dengan 2 parameter yaitu parameter pertama adalah url dengan tipe string dan parameter kedua adalah body dimana data yang harus diterima berupa objek json yang dikonversi menjadi string.
* line 27 agar hasilnya dapat diamati maka perlu ditambahkan subscribe dengan method respone seperti pada gambar berikut. Pada saat kursor kita arahkan ke subscribe maka akan muncul keterangan seperti berikut.



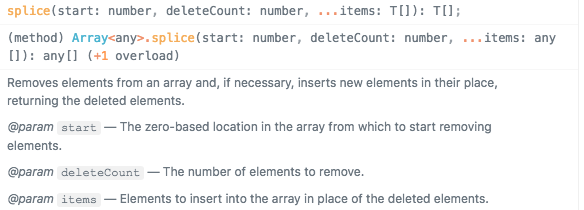
* line 28 pada saat melakukan console.log(Response.json()) server akan mengembalikan sebuah id seperti berikut
  + - * rubah code pada subscribe seperti berikut



* + - * tambahkan data dan lihat pada console log



* + - * maka terlihat bahwa server mengembalikan hanya sebuah id.
      * maka dari itu tambahkan sebuah property post [‘id’] yang merupakan parameter response.json().id agar dapat melihat hasilnya pada tampilan.
* line 29 untuk menampilkan hasilnya maka dapat menggunakan method splice.



Method splice mempunyai 3 paramater

* + - * parameter pertama adalah posisi awal
      * parameter kedua adalah jumlah objek yang ingin anda hapus
      * parameter ketiga adalah objek yang akan ditambahkan di posisi awal dan di sini ditambahkan object post pada indeks ke nol
* hasilnya seperti berikut ( isikan tambahkan data pada inputan dan setelah itu tekan enter maka data akan masuk dalam list). Untuk penambahan data ini bersifat sementara jadi jika kita refresh data akan kembali ke keadaan semula

|  |
| --- |
| ../../../../../Users/dimaswahyu/Desktop/Screen%20Shot%202019-04-13 |
| ../../../../../Users/dimaswahyu/Desktop/Screen%20Shot%202019-04-13 |

**8.4 Updating Data**

Pada sub bab ini kita tidak dapat merubah isi data di Jsonplaceholder dikarenakan untuk merubah data dengan pada situs jsonPlaceholder maka secara tidak langsung harus merubah data pada server Jsonplaceholder. Sehingga pada sub bab ini hanya menggunaka console log untuk melihat proses untuk merubah data. Untuk merubah data terdapat 2 cara yaitu

* method put
* method patch
  + jika hanya merubah atau memodifikasi salah data atau property <https://stackoverflow.com/questions/31089221/what-is-the-difference-between-put-post-and-patch>

Untuk lebih jelasnya ikuti langkah berikut :

* buka file post.component.html dan tambahkan code berikut ( line 4-6)



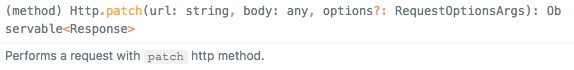
Penjelasan code

* line 4 – line 6 adalah tag button dimana terdapat event click yang mana event click ini melakukan proses pada method updatePost(data) dengan parameter data
* buka file post.component.ts dan tambahkan code berikut

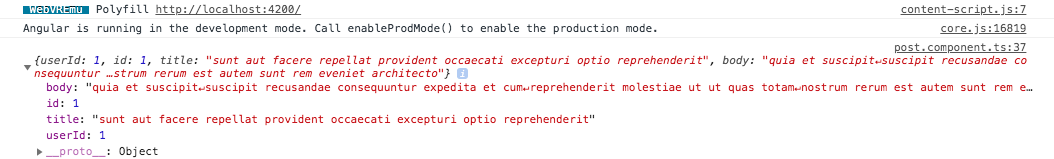


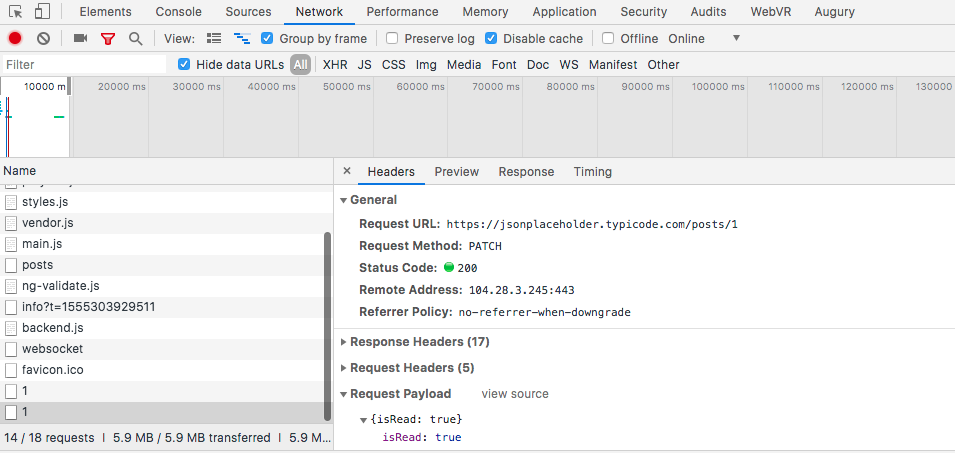
Penjelasan code

* line 34 adalah nama method updatePost dengan parameter data disesuaikan dengan post.component.html line 4
* line 35 adalah metode patch dengan parameter seperti pada gambar berikut



* line 36 adalah method Response
* line 37 adalah untuk menampilkan hasilnya pada console
* jalankan pada localhost dan amati pada console nya, maka muncul data yang akan kita update (id=1)

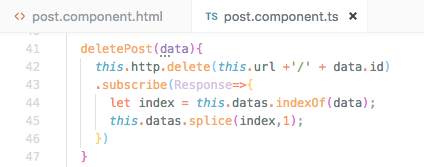




Jika kita klik 1 pada network dan amati pada header, pada header terdapat informasi tentang request methodnya adalah PATCH dan Request Payload yang berisi isRead:true

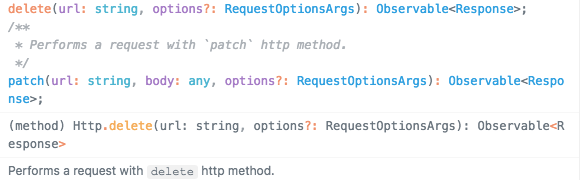
Pada sub bab ini menjelaskan penggunakan services delete dengan menggunakan data Jsonplaceholder. Untuk lebih jelas ikuti langkah berikut

* buka file post.component.ts dan buatlah method deletePost

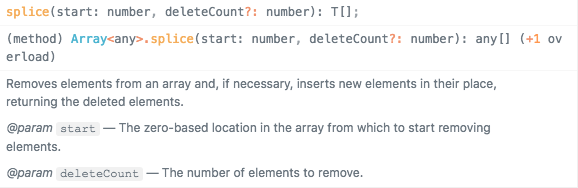


Penjelasan code

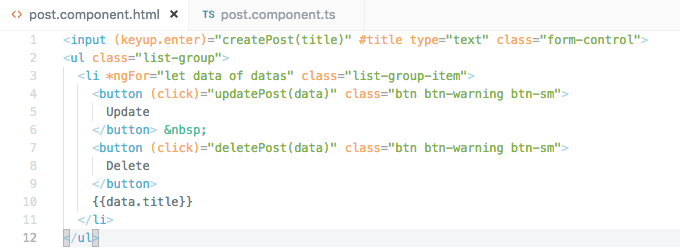
* line 41 adalah method dengan nama deletePost dengan parameter data
* line 42 method delete untuk menghapus data dengan parameter seperti pada gambar berikut



* line 43 method response untuk mengamati hasil dari proses delete
* line 44 sebuah property dengan nama index yang menampung nilai dari array datas dengan parameter data
* line 45 method splice untuk menampilkan data dengan parameter seperti pada gambar berikut



* buka file post.component.html dan tambahkan code seperti pada gambar berikut ( line 7- 9)



* jalankan localhost dan hasilnya seperti berikut (delete bagian awal)

|  |
| --- |
| ../../../../../Users/dimaswahyu/Desktop/Screen%20Shot%202019-04-15 |
| ../../../../../Users/dimaswahyu/Desktop/Screen%20Shot%202019-04-15 |

**8.5 OnInit Interface**

OnInit pada post.component.ts merupakan fungsi yang dieksekusi ketika komponen post pertama kali dibuat atau dieksekusi. OnInit merupakan salah satu fase pada angular lifecyle. Setiap komponen memiliki lifecycle yang sangat berguna saat melakukan pembuatan aplikasi. Komponen memiliki lifecycle yang diatur oleh angular. Angular membuat sebuah komponen, melakukan rendering, melakukan check terhadap terjadinya pergantian(*change*) data pada class, dan terahir akan menghancurkan komponen tersebut dari DOM. Berikut ini adalah penjelasan mengenai *8 lifecycle hooks* pada angular. Setiap hooks berjalan secara *sekuensial* yang berarti menunggu hooks sebelumnya selesai dijalankan/eksekusi.

|  |  |
| --- | --- |
| Hooks | Penjelasan |
| ngOnChanges() | Dipanggil sebelum dilakukan ngOnInit() dan ketika terjadinnya pergantian data pada *input properties*. |
| ngOnInit() | Dipanggil saat pertama kali melakukan inisiasi pada komponen atau directive. NgOnInit() akan dipanggil setelah ngOnChanges() |
| ngDoCheck() | Dipanggil setelah ngOnInit(). Hooks ini mengindentifikasi perubahan yang tidak dapat dihandle oleh angular. |
| ngAfterContentInit() | Merupakan hooks yang mengakomdasi event saat terdapat konten external(selain dari angular) masuk pada component view. Hook ini hanya terdapat pada komponen. |
| ngAfterContentChecked() | Angular akan melakukan *check*(perubaan) pada external component. Hook ini hanya terdapat pada komponen. |
| ngAfterViewInit() | Hook yang dipanggil ketika menginisiasi *component views* dan *child views.* Hook ini hanya terdapat pada komponen |
| ngAfterViewChecked() | Hook yang dipanggil ketika angular telah melakukan check pada *component views* dan *child views*. Hook ini hanya terdapat pada komponen |
| ngOnDestroy() | Hook yang dipanggil sebelum angular menhancurkan(*destroy*) komponen atau directive. |

Sebagian besar aplikasi angular gunakan ngOnInit untuk semua inisialisasi / deklarasi dan menghindari hal-hal untuk bekerja di konstruktor. Konstruktor seharusnya hanya digunakan untuk menginisialisasi anggota kelas tetapi tidak harus melakukan "kerja" yang sebenarnya. Oleh sebab itu code pada post.component.ts dapat diganti menjadi seperti berikut



penjelasan code jadi jika anda ingin memerlukan inisialisasi maka menggunakan metode ngOnInit dan tambahkan this pada line 19 dikarenakan parameter http tidak ada di ngOnInit()