URL Parameters, Query Parameters, httpparams in Angular HttpClient

12 Comments / 6 minutes of reading / March 10, 2020

← Http Post Example Http Headers →

This post is a guide on how to Pass the URL Parameters or Query Parameters along with the HTTP Request using the `HttpClient` in Angular. We will be using `HttpParams` to add the URL Parameter, which is then used by the `GET`, `POST`, `PUT` & `PATCH` etc methods to send an HTTP request to the back end API. The URL Parameters also are known by the name Query strings, Query Params, Get Params, etc.

Refer to our tutorial on Angular HTTP get example & Angular HTTP Post Example

The `HttpParms` were known as `URLSearchParams` until Angular 4

Applies to: Angular 5 to the latest edition i.e. Angular 8, Angular 9. Angular 10, Angular 11

Table of Contents

The URL Parameters

In the Angular HttpClient GET Example article, we created a GitHubService. The Service issued GET Request to GitHub API Endpoint to retrieve the List of Repositories belonging to a particular User.

The GitHub API also has a set of parameters, which allows us to specify how we want to sort, which page to retrieve, No of Entries per page and type of the Repository to retrieve, etc.

For Example

```

https://api.github.com/users/tekTutorialsHub/repos?sort=description&page=2

```

The Above query will return the result sorted on the description and retrieves only the second page. The string `sort=description&page=2` after the question mark is called **URL**

Parameter or **Query strings /Query Parameters**. The Question mark is used as a separator. The URL Parameters are also known as the **GET params**.

HttpParams()

We add the URL parameters using the helper class `HttpParams`. The `HttpParams` is passed as one of the arguments to HttpClient.get method.

To use `HttpParams`, you need to import it first as shown below.

```

import { HttpClient,HttpParams } from '@angular/common/http';

```

Then create an instance of the `HttpParams` class.

```

const params = new HttpParams()
  .set('page', PageNo)
  .set('sort', SortOn);

```

And then call the httpClient.get method passing the params as the argument.

```

return this.httpClient.get<repos[]>(this.baseURL + 'users/' + userName + '/repos',{para

```

The following are the list of method available in HttpParams class

HttpParams.set

```

set(param: string, value: string): HttpParams

```

Construct a new body with a new value for the given parameter name. If the parameter already exists then it is replaced else it is added.

```

params = new HttpParams()
    .set('page', '2')
    .set('page', '3')
    .set('sort', 'name');

console.log(params.toString()); //Returns page=3&sort=name

```

HTTPParams is immutable

The HttpParams object is immutable. Every time you call a set method on Params object, it will create and return a new instance of the Params .

For Example

The following code does not work

```

let params = new HttpParams();
```

```

params.set('page', PageNo);
params.set('sort', SortOn);

```

Each call to `set` method does not add the options to the existing `HttpParams` instance, but creates a new instance from the existing instance and returns it.

To work around, you can use the code as follows

```

Let params = new HttpParams()
    .set('page', PageNo)
    .set('sort', SortOn);

```

Or you can try this

```

let params = new HttpParams();

params=params.set('page', PageNo);
params=params.set('sort', SortOn);

```

HttpParams.append

```

append(param: string, value: string): HttpParams

```

Construct a new body with an appended value for the given parameter name. Always appends the value irrespective of whether the parameter exists. The page parameter is appended twice in the following example.

```

params = new HttpParams()
    .set('page', '2')
    .append('page', '3')
    .set('sort', 'name');

console.log(params.toString()); //Returns page=2&page=3&sort=name

```

The URL Is constructed as `page=2&page=3&sort=name`

You can also use the append method similar to the `Set` method

```

let params = new HttpParams();

params=params.append('page', PageNo);
params=params.append('sort', SortOn);

```

HttpParams.has

```

has(param: string): boolean

```

Returns true if the given parameter name already exists in the HttpParams

```

params = new HttpParams()
    .set('sort', 'name');

if (!params.has('page')) {
    params = params.set('page', PageNo)
}

```

HttpParams.get

```

get(param: string): string | null

```

Get the first value for the given parameter name, or null if it's not present.

```

params = new HttpParams()
    .set('page', '2')
    .append('page', '3')
    .set('sort', 'name');

console.log(params.get('page')); // Returns 2 The First occurance of Page

```

HttpParams.getAll

```

getAll(param: string): string[] | null

```

Get all values as for the given parameter name, or null if it's not present.

```

params = new HttpParams()
    .set('page', '2')
    .append('page', '3')
    .set('sort', 'name');

console.log(params.getAll('page')); // Returns ["2", "3"] All occurance of Page

```

HttpParams.keys

```

keys(): string[]

```

Get all the parameter names for this body.

```

let params = new HttpParams()
    .set('page', '2')
    .set('sort', 'name');

console.log(params.keys()); //Returns ["page", "sort"]

```

HttpParams.delete

```

delete(param: string, value?: string): HttpParams

```

Deletes the parameter and returns the new parameter collection. You can delete using the parameter name or by using the name & value. If no argument is specified, then all parameters are deleted.

```

params = new HttpParams()
    .set('page', '2')
    .Append('page', '3')
    .set('sort', 'name');

params = params.delete('page', '3'); //Deletes the parameter page with value 3

params = params.delete('page'); //Delete the all the parameter of page

params = params.delete(''); //Delete all parameters

```

HttpParams.toString

```

toString(): string

```

Serialize the body to an encoded string, where key-value pairs (separated by =) are separated by &s.

```

params = new HttpParams()
    .set('page', '2')
    .Append('page', '3')
    .set('sort', 'name');

console.log(params.toString()); //Returns page=2&page=3&sort=name

```

Http Parameters from a string

Another way to pass the value is to use the `fromString` shortcut

```

```

```
let params = new HttpParams({fromString: 'page=' + PageNo + '&sort=' + SortOn});

```

Http Parameters from an object

```

let params = new HttpParams({ fromObject: { page: PageNo, sort: SortOn } });

```

Without params

You can also add the parameters directly to the URL, without going through the HttpParams as shown below.

```

//You an also do it this way.
return this.httpClient.get<repos[]>(this.baseURL + 'users/' + userName + '/repos?'+'pag

```

Angular Httpparams Example

We are updating the project, which was created in the tutorial Angular Http GET Example.

app.module

Import the httpClientModule from the @angular/common/http package.

```

import {HttpClientModule} from '@angular/common/http';

```

Declare it in Imports metadata array in app.module.ts

```

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    FormsModule
  ],
  providers: [GitHubService],
  bootstrap: [AppComponent]
})

```

Passing the URL Parameters

Open the `github.service.ts`.

Import the `HttpClient` & `HttpParams` from the `@angular/common/http`. We also require the Observable module from RxJs

```

import { Injectable } from '@angular/core';
import { HttpClient, HttpParams } from '@angular/common/http';
import { repos} from './repos';
import { Observable } from 'rxjs';

```

Inject `HttpClient` in the Constructor

```

constructor(private httpClient:HttpClient){
}

```

In the GetRepos method create the params object

```

const params = new HttpParams()
    .set('page', PageNo)
    .set('sort', SortOn);

```

And use the params when calling the httpClient.get method as shown below

```

return this.httpClient.get<repos[]>(this.baseURL + 'users/' + userName + '/repos',{para

```

The complete `github.service.ts`

```

import { Injectable } from '@angular/core';
import { HttpClient, HttpParams } from '@angular/common/http';
import { Observable} from 'rxjs/Rx';

import { repos} from './repos';


@Injectable()
export class GitHubService {

  baseURL= "https://api.github.com/";

  constructor(private httpClient: HttpClient){
  }

  getRepos(userName: string, PageNo: string, SortOn: string): Observable<repos[]> {


      let params = new HttpParams()
            .set('page', PageNo)
            .set('sort', SortOn);

      console.log(params.toString());

      return this.httpClient.get<repos[]>(this.baseURL + 'users/' + userName + '/repos',
  }

}

```

App.component.ts

```

import { Component } from '@angular/core';
import { Observable} from 'rxjs/Rx';

import { GitHubService } from './github.service';

import { repos} from './repos';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
})


export class AppComponent
{

    userName = 'tektutorialshub';
    pageNo  = '1';
    sortOn = 'description';

    repos: repos[];

    loading = false;
    errorMessage = '';

    constructor(private githubService: GitHubService) {
    }

  public getRepos() {
    this.loading = true;
    this.errorMessage = '';
    this.githubService.getRepos(this.userName,this.pageNo,this.sortOn)
      .subscribe((response) => {this.repos = response;},
                (error) => {
                  this.errorMessage = error.message; this.loading = false;
                 },
                 () => {this.loading = false;})

  }

}

```

App.component.html

```

<div class="container">

  <h1 class="heading"><strong>HTTP </strong>Demo</h1>


  <div class="form-group">
    <label for="userName">User Name</label>
    <input type="text" class="form-control" name="userName" [(ngModel)]="userName'

    <label for="pageNo">Page No</label>
    <input type="text" class="form-control" name="pageNo" [(ngModel)]="pageNo">

    <label for="sortOn">Sorted On</label>
    <input type="text" class="form-control" name="sortOn" [(ngModel)]="sortOn">
  </div>

  <div class="form-group">
     <button type="button" (click)="getRepos()">Get Repos</button>
  </div>

  <div *ngIf="loading">loading...</div>

  <div *ngIf="errorMessage" class="alert alert-warning">
    <strong>Warning!</strong> {{errorMessage}}
  </div>

  <div class='table-responsive'>
```

```
        <table class='table'>
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Name</th>
                    <th>HTML Url</th>
                    <th>description</th>
                </tr>
            </thead>
            <tbody>
                <tr *ngFor="let repo of repos;">
                    <td>{{repo.id}}</td>
                    <td>{{repo.name}}</td>
                    <td>{{repo.html_url}}</td>
                    <td>{{repo.description}}</td>
                </tr>
            </tbody>
        </table>
    </div>



  <pre>{{repos | json}}</pre>

</div>

```

Summary

We learned how to pass Get Parameters or request parameters when we invoke the HTTP get Request using httpClient.get method

References

- HttpClient
- HttpParams

← Http Post Example Http Headers →