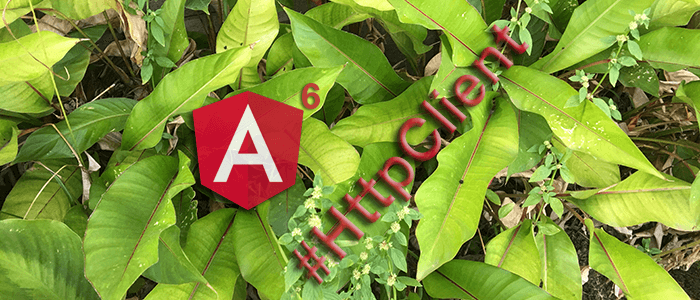
Angular 6 HttpClient: Consume RESTful API Example

by Didin J. on Aug 30, 2018

A comprehensive step by step Angular 6 HttpClient tutorial on consuming RESTful API from the remote server

Simple and comprehensive tutorial of the Angular 6 HttpClient that use for consuming RESTful API from the remote server. Angular 6 HttpClient is one of the fundamentals feature of the Angular 6. We have written a [tutorial](https://www.djamware.com/post/59924f9080aca768e4d2b12e/ionic-3-consuming-rest-api-using-new-angular-43-httpclient) about HttpClient when it first shows up in Angular 4.3. There are not many changes from the previous version, but we need to create an easy step-by-step tutorial for Angular 6 version.

In this tutorial, we will show you how to access external or remote RESTful API using this Angular 6 HttpClient. For now, we will use existing Node.js, Express.js, and MongoDB RESTful API from our GitHub [repository](https://github.com/didinj/NodeRestApi) or if you have PostgreSQL installed in your machine, you can get this [RESTful API application](https://github.com/didinj/node-express-postgresql-sequelize). After this, the HttpClient feature will show you in an easy way and a little bit different from the original Angular 6 docs.

**Table of Contents:**

* [Install Angular CLI and Create Angular 6 HttpClient Web Application](https://www.djamware.com/post/5b87894280aca74669894414/angular-6-httpclient-consume-restful-api-example#ch1)
* [Setup and Configure Angular 6 HttpClient](https://www.djamware.com/post/5b87894280aca74669894414/angular-6-httpclient-consume-restful-api-example#ch2)
* [Create Angular 6 Service for consuming RESTful API using Angular 6 HttpClient](https://www.djamware.com/post/5b87894280aca74669894414/angular-6-httpclient-consume-restful-api-example#ch3)
* [Accessing Angular 6 HttpClient Service from Component](https://www.djamware.com/post/5b87894280aca74669894414/angular-6-httpclient-consume-restful-api-example#ch4)
* [Run and Test Angular 6 HttpClient Web Application](https://www.djamware.com/post/5b87894280aca74669894414/angular-6-httpclient-consume-restful-api-example#ch5)

The following tools, frameworks, and modules are required to complete this Angular 6 tutorial:

* [Node.js](https://nodejs.org/) (Recommended version)
* [Angular 6](https://angular.io/)
* [Angular-CLI](https://github.com/angular/angular-cli)
* Terminal (Mac/Linux) or Node Command Line (Windows)
* IDE or Text Editor

We assume that you have downloaded and installed Node.js environment. Now, let's check the above requirement by open the terminal or Node command line then go to your projects folder. Type this command to check the latest Node and NPM version.

node -v

v8.11.1

npm -v

6.0.0

That's our Node and NPM versions, let's move to the main steps of the Angular 6 and Angular Material tutorial.

**1. Install Angular CLI and Create Angular 6 HttpClient Web Application**

To install or upgrade the latest Angular 6 CLI, type this command in the terminal or Node command line.

sudo npm install -g @angular/cli

If you use windows, it might be not necessary to add `sudo`. Next, create a new Angular 6 Web Application using this Angular CLI command.

ng new angular6-httpclient

Next, go to the newly created Angular 6 project folder.

cd ./angular6-httpclient

Now, you run the new Angular 6 web application using your own host and port.

ng serve --host 0.0.0.0 --port 3000

Open your browser then go to `localhost:3000` you will see this Angular 6 page.



**2. Setup and Configure Angular 6 HttpClient**

In the created Angular 6 application, we have set up and configure Angular 6 HttpClient. The modules already included with Angular 6, we just register the module for use within Angular 6 application. Open and edit `src/app/app.module.ts` then add this import.

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

Add that module in `@NgModule` imports.

imports: [

  BrowserModule,

  HttpClientModule

],

That it's, now you can use the Angular 6 HttpClient in your application.

**3. Create Angular 6 Service for consuming RESTful API using Angular 6 HttpClient**

We will put HttpClient access to the separate Angular 6 service. For that, type this command to generate an Angular 6 service after stopping the running Angular 6 application by press `CTRL+C`.

ng generate service rest

Next, open and edit `src/app/rest.service.ts` then add or replace these imports.

import { Injectable } from '@angular/core';

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

import { Observable, of } from 'rxjs';

import { map, catchError, tap } from 'rxjs/operators';

Inject the HttpClient module to the constructor params.

constructor(private http: HttpClient) { }

Declare the variable for RESTful API endpoint and HTTP headers.

const endpoint = '<http://localhost:3000/api/v1/>';

const httpOptions = {

  headers: new HttpHeaders({

    'Content-Type':  'application/json'

  })

};

Because we are not using a type checker, so the response should be extracted. Add this function after the constructor for extract it.

private extractData(res: Response) {

  let body = res;

  return body || { };

}

Add the functions for all RESTful API CRUD consumption.

getProducts(): Observable<any> {

  return this.http.get(endpoint + 'products').pipe(

    map(this.extractData));

}

getProduct(id): Observable<any> {

  return this.http.get(endpoint + 'products/' + id).pipe(

    map(this.extractData));

}

addProduct (product): Observable<any> {

  console.log(product);

  return this.http.post<any>(endpoint + 'products', JSON.stringify(product), httpOptions).pipe(

    tap((product) => console.log(`added product w/ id=${product.id}`)),

    catchError(this.handleError<any>('addProduct'))

  );

}

updateProduct (id, product): Observable<any> {

  return this.http.put(endpoint + 'products/' + id, JSON.stringify(product), httpOptions).pipe(

    tap(\_ => console.log(`updated product id=${id}`)),

    catchError(this.handleError<any>('updateProduct'))

  );

}

deleteProduct (id): Observable<any> {

  return this.http.delete<any>(endpoint + 'products/' + id, httpOptions).pipe(

    tap(\_ => console.log(`deleted product id=${id}`)),

    catchError(this.handleError<any>('deleteProduct'))

  );

}

Finally, add the function for handle an error.

private handleError<T> (operation = 'operation', result?: T) {

  return (error: any): Observable<T> => {

    // TODO: send the error to remote logging infrastructure

    console.error(error); // log to console instead

    // TODO: better job of transforming error for user consumption

    console.log(`${operation} failed: ${error.message}`);

    // Let the app keep running by returning an empty result.

    return of(result as T);

  };

}

Simply as that, the main function of the Angular 6 HttpClient. Next, we add the component for a simple access to the Angular 6 service.

**4. Accessing Angular 6 HttpClient Service from Component**

Now, we have to access all RESTful API call that using Angular 6 HttpClient from the component. For that, add or generate all required component using this commands.

ng generate component product

ng generate component product-add

ng generate component product-detail

ng generate component product-edit

Open and edit `src/app/app.module.ts` then add or modify this imports.

import { RouterModule, Routes } from '@angular/router';

import { FormsModule } from '@angular/forms';

Component modules already imported while generated the components. Next, add this constant variable for the routes before the `@NgModule`.

const appRoutes: Routes = [

  {

    path: 'products',

    component: ProductComponent,

    data: { title: 'Product List' }

  },

  {

    path: 'product-details/:id',

    component: ProductDetailComponent,

    data: { title: 'Product Details' }

  },

  {

    path: 'product-add',

    component: ProductAddComponent,

    data: { title: 'Product Add' }

  },

  {

    path: 'product-edit/:id',

    component: ProductEditComponent,

    data: { title: 'Product Edit' }

  },

  { path: '',

    redirectTo: '/products',

    pathMatch: 'full'

  }

];

Modify `@NgModule` imports to be like this.

imports: [

  RouterModule.forRoot(appRoutes),

  FormsModule,

  BrowserModule,

  HttpClientModule

],

Next, open and edit `src/app/app.component.html` then replace all HTML tags with this.

<router-outlet></router-outlet>

Next, open and edit `src/app/product/product.component.ts` then replace all codes with this.

import { Component, OnInit } from '@angular/core';

import { RestService } from '../rest.service';

import { ActivatedRoute, Router } from '@angular/router';

@Component({

  selector: 'app-product',

  templateUrl: './product.component.html',

  styleUrls: ['./product.component.css']

})

export class ProductComponent implements OnInit {

  products:any = [];

  constructor(public rest:RestService, private route: ActivatedRoute, private router: Router) { }

  ngOnInit() {

    this.getProducts();

  }

  getProducts() {

    this.products = [];

    this.rest.getProducts().subscribe((data: {}) => {

      console.log(data);

      this.products = data;

    });

  }

  add() {

    this.router.navigate(['/product-add']);

  }

  delete(id) {

    this.rest.deleteProduct(id)

      .subscribe(res => {

          this.getProducts();

        }, (err) => {

          console.log(err);

        }

      );

  }

}

Open and edit `src/app/product/product.component.html` then replace all HTML tags with this.

<h2>Product List</h2>

<div>

  <button (click)="add()">

    Add

  </button>

</div>

<ul class="products">

  <li \*ngFor="let p of products; let i=index;">

    <a routerLink="/product-details/{{p.\_id}}">

      <span class="badge">{{i+1}}</span> {{p.prod\_name}}

    </a>

    <button class="delete" title="delete product"

      (click)="delete(p.\_id)">x</button>

  </li>

</ul>

Open and edit `src/app/product/product.component.css` then replace all CSS syntax with this.

/\* Products Component's private CSS styles \*/

.products {

  margin: 0 0 2em 0;

  list-style-type: none;

  padding: 0;

  width: 15em;

}

.products li {

  position: relative;

  cursor: pointer;

  background-color: #EEE;

  margin: .5em;

  padding: .3em 0;

  height: 1.6em;

  border-radius: 4px;

}

.products li:hover {

  color: #607D8B;

  background-color: #DDD;

  left: .1em;

}

.products a {

  color: #888;

  text-decoration: none;

  position: relative;

  display: block;

  width: 250px;

}

.products a:hover {

  color:#607D8B;

}

.products .badge {

  display: inline-block;

  font-size: small;

  color: white;

  padding: 0.8em 0.7em 0 0.7em;

  background-color: #607D8B;

  line-height: 1em;

  position: relative;

  left: -1px;

  top: -4px;

  height: 1.8em;

  min-width: 16px;

  text-align: right;

  margin-right: .8em;

  border-radius: 4px 0 0 4px;

}

button {

  background-color: #eee;

  border: none;

  padding: 5px 10px;

  border-radius: 4px;

  cursor: pointer;

  cursor: hand;

  font-family: Arial;

}

button:hover {

  background-color: #cfd8dc;

}

button.delete {

  position: relative;

  left: 194px;

  top: -32px;

  background-color: gray !important;

  color: white;

}

And here the rest of the component codes.

**`src/app/product-add/product-add.component.ts`**

import { Component, OnInit, Input } from '@angular/core';

import { RestService } from '../rest.service';

import { ActivatedRoute, Router } from '@angular/router';

@Component({

  selector: 'app-product-add',

  templateUrl: './product-add.component.html',

  styleUrls: ['./product-add.component.css']

})

export class ProductAddComponent implements OnInit {

  @Input() productData = { prod\_name:'', prod\_desc: '', prod\_price: 0 };

  constructor(public rest:RestService, private route: ActivatedRoute, private router: Router) { }

  ngOnInit() {

  }

  addProduct() {

    this.rest.addProduct(this.productData).subscribe((result) => {

      this.router.navigate(['/product-details/'+result.\_id]);

    }, (err) => {

      console.log(err);

    });

  }

}

**`src/app/product-add/product-add.component.html`**

<div>

  <h2>Product Add</h2>

  <div>

    <label>Product Name:

      <input [(ngModel)]="productData.prod\_name" placeholder="Product Name"/>

    </label><br>

    <label>Product Desc:

      <input [(ngModel)]="productData.prod\_desc" placeholder="Product Description"/>

    </label><br>

    <label>Product Price:

      <input [(ngModel)]="productData.prod\_price" placeholder="Product Price"/>

    </label><br>

  </div>

  <button (click)="addProduct()">Save</button>

</div>

**`src/app/product-detail/product-detail.component.ts`**

import { Component, OnInit } from '@angular/core';

import { RestService } from '../rest.service';

import { ActivatedRoute, Router } from '@angular/router';

@Component({

  selector: 'app-product-detail',

  templateUrl: './product-detail.component.html',

  styleUrls: ['./product-detail.component.css']

})

export class ProductDetailComponent implements OnInit {

  product:any;

  constructor(public rest:RestService, private route: ActivatedRoute, private router: Router) { }

  ngOnInit() {

    this.rest.getProduct(this.route.snapshot.params['id']).subscribe((data: {}) => {

      console.log(data);

      this.product = data;

    });

  }

}

**`src/app/product-detail/product-detail.component.html`**

<div \*ngIf="product" class="products">

  <h2>{{product.prod\_name | uppercase}} Details</h2>

  <div><span>Description: </span>{{product.prod\_desc}}</div>

  <div><span>Price: </span>{{product.prod\_price}}</div>

  <div><span>Update Date: </span>{{product.updated\_at | date}}</div>

  <div>

    <button routerLink="/product-edit/{{product.\_id}}">

      Edit

    </button>

  </div>

</div>

**`src/app/product-edit/product-edit.component.ts`**

import { Component, OnInit, Input } from '@angular/core';

import { RestService } from '../rest.service';

import { ActivatedRoute, Router } from '@angular/router';

@Component({

  selector: 'app-product-edit',

  templateUrl: './product-edit.component.html',

  styleUrls: ['./product-edit.component.css']

})

export class ProductEditComponent implements OnInit {

  @Input() productData:any = { prod\_name: '', prod\_desc: '', prod\_price:0 };

  constructor(public rest:RestService, private route: ActivatedRoute, private router: Router) { }

  ngOnInit() {

    this.rest.getProduct(this.route.snapshot.params['id']).subscribe((data: {}) => {

      console.log(data);

      this.productData = data;

    });

  }

  updateProduct() {

    this.rest.updateProduct(this.route.snapshot.params['id'], this.productData).subscribe((result) => {

      this.router.navigate(['/product-details/'+result.\_id]);

    }, (err) => {

      console.log(err);

    });

  }

}

**`src/app/product-edit/product-edit.component.html`**

<div>

  <h2>Product Edit</h2>

  <div>

    <label>Product Name:

      <input [(ngModel)]="productData.prod\_name" placeholder="Product Name"/>

    </label><br>

    <label>Product Desc:

      <input [(ngModel)]="productData.prod\_desc" placeholder="Product Description"/>

    </label><br>

    <label>Product Price:

      <input [(ngModel)]="productData.prod\_price" placeholder="Product Price"/>

    </label><br>

  </div>

  <button (click)="updateProduct()">Update</button>

</div>

**5. Run and Test Angular 6 HttpClient Web Application**

Before running the Angular 6 HttpClient application, run the RESTful API server first. We use our [Express.js/MongoDB](http://express.js/MongoDB) RESTful application, so open the other tabs to run MongoDB server and Express.js server.

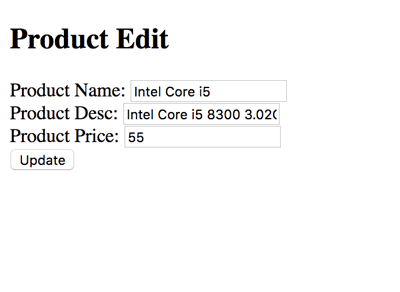
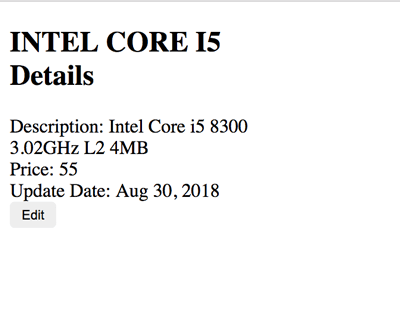
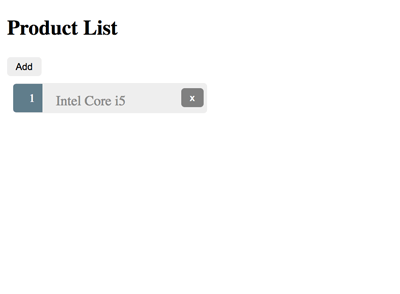
mongod

nodemon

Next, type this command in the current Terminal tabs to run the Angular 6 HttpClient application.

ng serve

You should enable CORS to make RESTful API server accessible from different server PORT. And here our Angular 6 HttpClient Web application looks like.

That it's, an example of Angular 6 HttpClient RESTful API consumption. You can get the full source code on our [GitHub](https://github.com/didinj/angular6-httpclient-example.git).

That just the basic. If you need more deep learning about MEAN Stack, Angular, and Node.js, you can find the following books:

* [Angular 4 Projects](http://www.anrdoezrs.net/click-8263647-12169838?url=https%3A%2F%2Fwww.apress.com%2Fus%2Fbook%2F9781484232781%3Futm_medium%3Daffiliate%26utm_source%3Dcommission_junction%26utm_campaign%3D3_nsn6445_product_PID%25zp%26utm_content%3Dus_10092017&cjsku=9781484232798)http://www.awltovhc.com/image-8263647-12169838
* [Pro MEAN Stack Development](http://www.dpbolvw.net/click-8263647-12169838?url=https%3A%2F%2Fwww.apress.com%2Fus%2Fbook%2F9781484220436%3Futm_medium%3Daffiliate%26utm_source%3Dcommission_junction%26utm_campaign%3D3_nsn6445_product_PID%25zp%26utm_content%3Dus_10092017&cjsku=9781484220443)http://www.ftjcfx.com/image-8263647-12169838
* [Practical Node.js](http://www.jdoqocy.com/click-8263647-12752006?url=https%3A%2F%2Fwww.apress.com%2Fde%2Fbook%2F9781430265955%3Futm_medium%3Daffiliate%26utm_source%3Dcommission_junction%26utm_campaign%3D3_nsn6445_product_PID%25zp%26utm_content%3Dde_10092017&cjsku=9781430265962)http://www.ftjcfx.com/image-8263647-12752006
* [Pro Express.js](http://www.kqzyfj.com/click-8263647-12169838?url=https%3A%2F%2Fwww.apress.com%2Fus%2Fbook%2F9781484200384%3Futm_medium%3Daffiliate%26utm_source%3Dcommission_junction%26utm_campaign%3D3_nsn6445_product_PID%25zp%26utm_content%3Dus_10092017&cjsku=9781484200377)http://www.ftjcfx.com/image-8263647-12169838

For more detailed on MEAN stack and Node.js, you can take the following course:

* [Angular (Angular 2+) & NodeJS - The MEAN Stack Guide](https://click.linksynergy.com/link?id=6nYo96*QrJE&offerid=358574.833442&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fangular-2-and-nodejs-the-practical-guide%2F)https://ad.linksynergy.com/fs-bin/show?id=6nYo96*QrJE&bids=358574.833442&type=2&subid=0
* [Start Building Web Apps And Services With Node. js + Express](https://click.linksynergy.com/link?id=6nYo96*QrJE&offerid=358574.150396&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fthe-ultimate-guide-to-nodejs-express%2F)https://ad.linksynergy.com/fs-bin/show?id=6nYo96*QrJE&bids=358574.150396&type=2&subid=0
* [Build a REST API with node. js, ExpressJS, and MongoDB](https://click.linksynergy.com/link?id=6nYo96*QrJE&offerid=358574.654736&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fnodejs-api%2F)https://ad.linksynergy.com/fs-bin/show?id=6nYo96*QrJE&bids=358574.654736&type=2&subid=0
* [Angular 5 Bootcamp FastTrack](https://click.linksynergy.com/link?id=6nYo96*QrJE&offerid=358574.1461406&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fangular-bootcamp-fasttrack%2F)https://ad.linksynergy.com/fs-bin/show?id=6nYo96*QrJE&bids=358574.1461406&type=2&subid=0

Thanks!

Follow

[Previous Article](https://www.djamware.com/post/5b6a681f80aca76a2cbd98fb/mongodb-express-vuejs-2-nodejs-mevn-and-socketio-chat-app)

[MongoDB, Express, Vue.js 2, Node.js (MEVN) and SocketIO Chat App](https://www.djamware.com/post/5b6a681f80aca76a2cbd98fb/mongodb-express-vuejs-2-nodejs-mevn-and-socketio-chat-app)

[Next Article](https://www.djamware.com/post/5bb1f05280aca74669894417/node-express-sequelize-and-postgresql-association-example)

[Node, Express, Sequelize, and PostgreSQL Association Example](https://www.djamware.com/post/5bb1f05280aca74669894417/node-express-sequelize-and-postgresql-association-example)