# ***Doraemon Explorer Hub***

***TASK 1-*** To create all the components and services

**Description-** First we will have to create all the components and services that is to be used in the project.

Steps to create component and services

**Step 1-**Type ng generate component component_name in terminal

**Step 2-**Type ng generate service service_name in terminal

Note - repeat step 1 for generating all the components character,gadget,gallery,search,add,message.

***Task 2-*** To give routing to each component

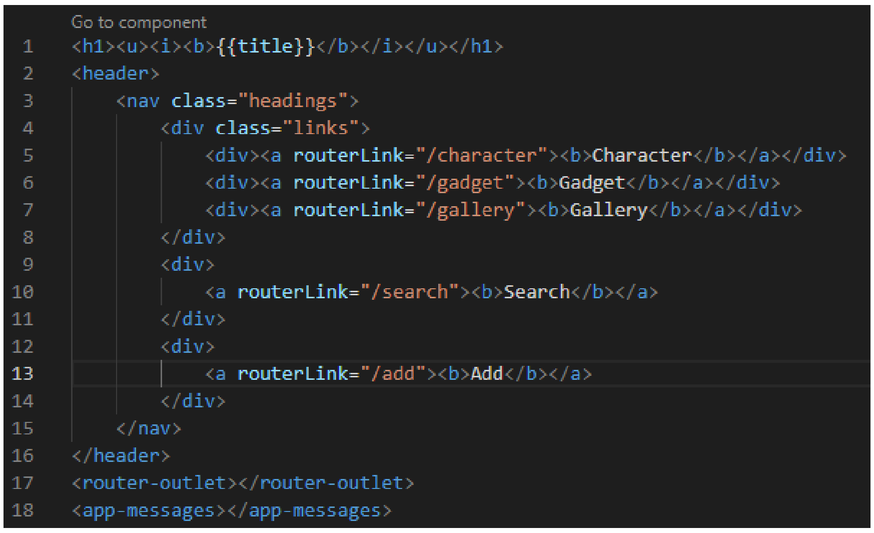
**Description-**We will be creating navbar in app.html file and give routerLink to each element of navbar.

Steps to give routerLink

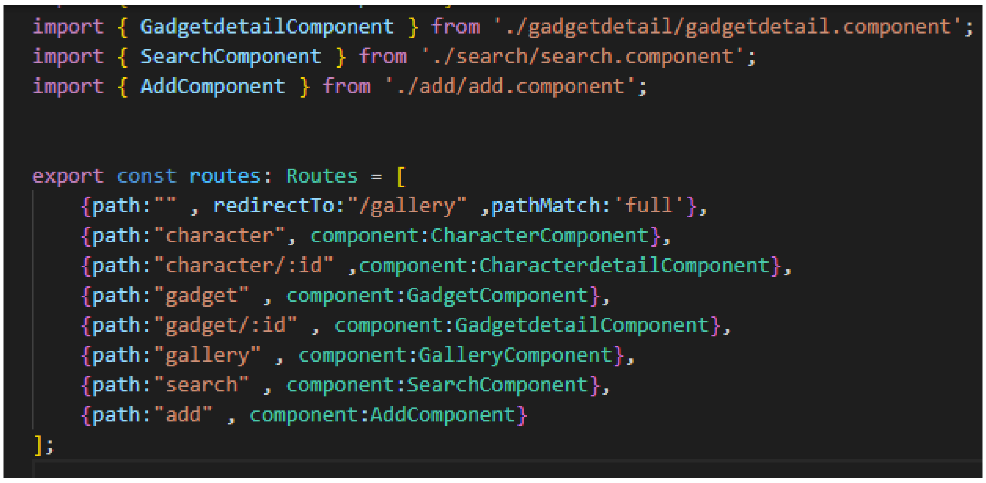
**Step 1-** Give path and component for each routing in routes.ts file.

**Step 2-** Create navbar in app.component.html file and give routerLink to each element and add router outlet in it.

**Step 3-** Import RouterModule in app.component.ts file for using routerLink.



```html
Go to component
<h1><u><i><b>{{title}}</b></i></u></h1>
<header>
    <nav class="headings">
        <div class="links">
            <div><a routerLink="/character"><b>Character</b></a></div>
            <div><a routerLink="/gadget"><b>Gadget</b></a></div>
            <div><a routerLink="/gallery"><b>Gallery</b></a></div>
        </div>
        <div>
            <a routerLink="/search"><b>Search</b></a>
        </div>
        <div>
            <a routerLink="/add"><b>Add</b></a>
        </div>
    </nav>
</header>
<router-outlet></router-outlet>
<app-messages></app-messages>
```



```
import { GadgetdetailComponent } from './gadgetdetail/gadgetdetail.component';
import { SearchComponent } from './search/search.component';
import { AddComponent } from './add/add.component';


export const routes: Routes = [
    {path:"" , redirectTo:"/gallery" ,pathMatch:'full'},
    {path:"character", component:CharacterComponent},
    {path:"character/:id" ,component:CharacterdetailComponent},
    {path:"gadget" , component:GadgetComponent},
    {path:"gadget/:id" , component:GadgetdetailComponent},
    {path:"gallery" , component:GalleryComponent},
    {path:"search" , component:SearchComponent},
    {path:"add" , component:AddComponent}
];
```

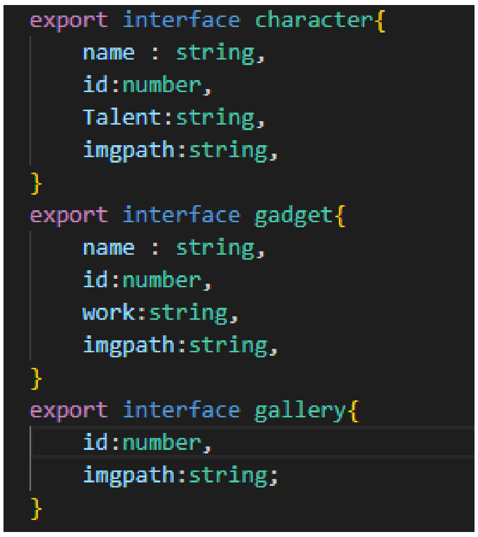
***Task 3-*** To create interfaces

**Description** - We will be creating interfaces of character, gadget, gallery

Steps to create interface

**Step 1-** Create .ts file of any name

**Step 2-** Create interface inside that for character, gadget and galley



```
export interface character{
    name : string,
    id:number,
    Talent:string,
    imgpath:string,
}
export interface gadget{
    name : string,
    id:number,
    work:string,
    imgpath:string,
}
export interface gallery{
    id:number,
    imgpath:string;
}
```

***Task 4-***To create data server

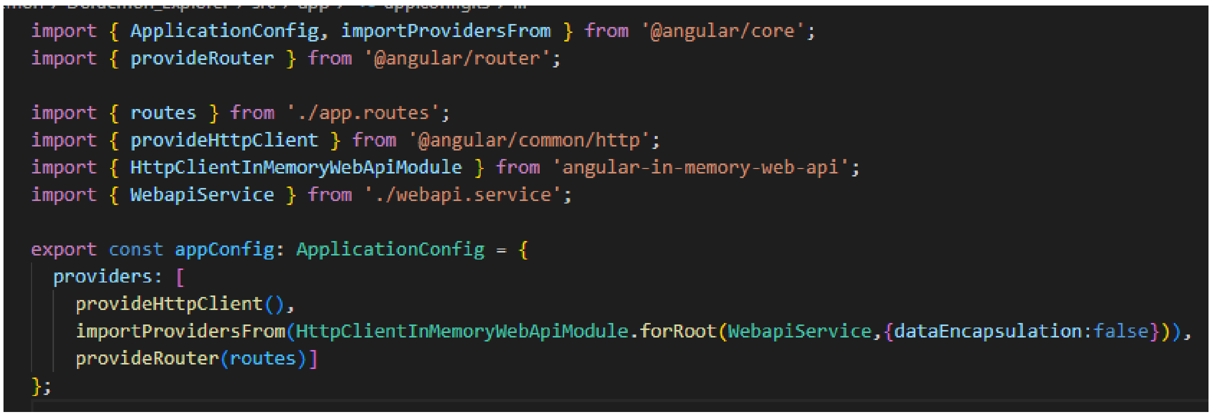
**Description-** Now we will be creating data server using in memory db service

Steps to create data server

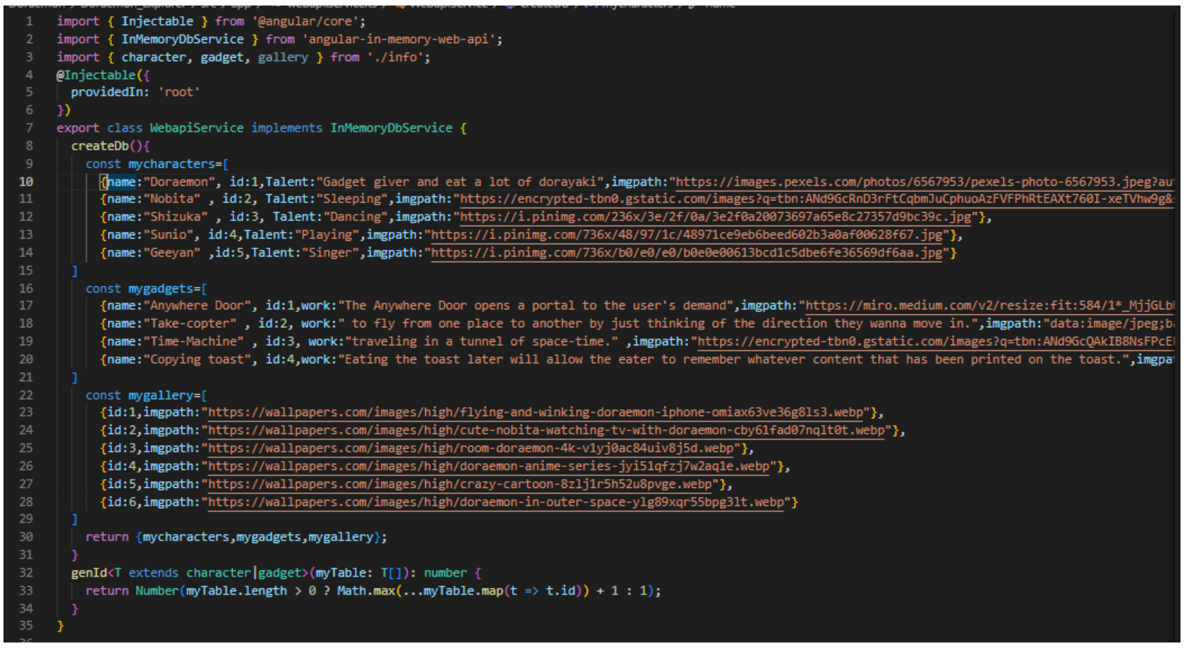
**Step 1-** Write npm install angular-in-memory-web-api - -save in terminal and ng generate service webapi

**Step 2-** Implement InMemoryDbService in webapi service and creteDb in that and make genId function in that.

**Step 3-** add some imp imports in app.config file.



```
import { ApplicationConfig, importProvidersFrom } from '@angular/core';
import { provideRouter } from '@angular/router';

import { routes } from './app.routes';
import { provideHttpClient } from '@angular/common/http';
import { HttpClientInMemoryWebApiModule } from 'angular-in-memory-web-api';
import { WebapiService } from './webapi.service';

export const appConfig: ApplicationConfig = {
  providers: [
    provideHttpClient(),
    importProvidersFrom(HttpClientInMemoryWebApiModule.forRoot(WebapiService,{dataEncapsulation:false})),
    provideRouter(routes)]
};
```



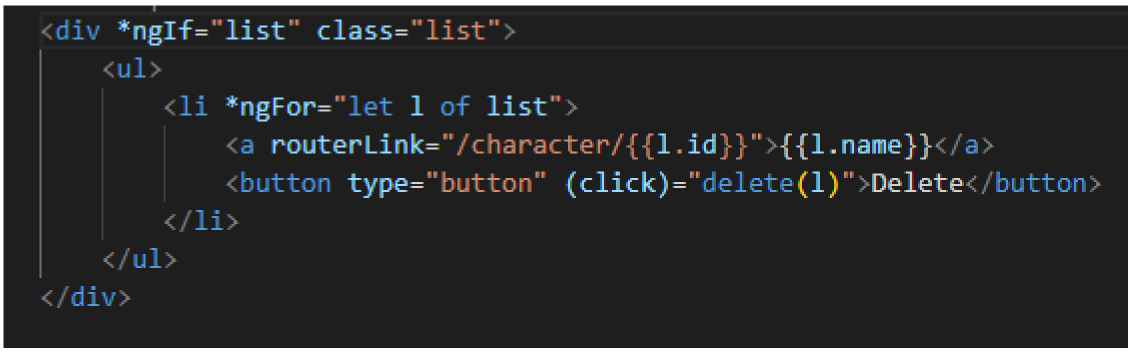
```
import { Injectable } from '@angular/core';
import { InMemoryDbService } from 'angular-in-memory-web-api';
import { character, gadget, gallery } from './info';
@Injectable({
  providedIn: 'root'
})
export class WebapiService implements InMemoryDbService {
  createDb(){
    const mycharacters=[
      {name:"Doraemon", id:1,Talent:"Gadget giver and eat a lot of dorayaki",imgpath:"https://images.pexels.com/photos/6567953/pexels-photo-6567953.jpeg?au
      {name:"Nobita" , id:2, Talent:"Sleeping",imgpath:"https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRnD3rFtCqbmJuCphuoAzFVFPhRtEAXt760I-xeTVhw9g&
      {name:"Shizuka" , id:3, Talent:"Dancing",imgpath:"https://i.pinimg.com/236x/3e/2f/0a/3e2f0a20073697a65e8c27357d9bc39c.jpg"},
      {name:"Sunio", id:4,Talent:"Playing",imgpath:"https://i.pinimg.com/736x/48/97/1c/48971ce9eb6beed602b3a0af00628f67.jpg"},
      {name:"Geeyan" ,id:5,Talent:"Singer",imgpath:"https://i.pinimg.com/736x/b0/e0/e0/b0e0e00613bcd1c5dbe6fe36569df6aa.jpg"}
    ]
    const mygadgets=[
      {name:"Anywhere Door", id:1,work:"The Anywhere Door opens a portal to the user's demand",imgpath:"https://miro.medium.com/v2/resize:fit:584/1*_MjjGLb
      {name:"Take-copter" , id:2, work:" to fly from one place to another by just thinking of the direction they wanna move in.",imgpath:"data:image/jpeg;b
      {name:"Time-Machine" , id:3, work:"traveling in a tunnel of space-time." ,imgpath:"https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQAkIB8NsFPcE
      {name:"Copying toast", id:4,work:"Eating the toast later will allow the eater to remember whatever content that has been printed on the toast.",imgpa
    ]
    const mygallery=[
      {id:1,imgpath:"https://wallpapers.com/images/high/flying-and-winking-doraemon-iphone-omiax63ve36g8ls3.webp"},
      {id:2,imgpath:"https://wallpapers.com/images/high/cute-nobita-watching-tv-with-doraemon-cby61fad07nqlt0t.webp"},
      {id:3,imgpath:"https://wallpapers.com/images/high/room-doraemon-4k-v1yj0ac84uiv8j5d.webp"},
      {id:4,imgpath:"https://wallpapers.com/images/high/doraemon-anime-series-jyi51qfzj7w2aq1e.webp"},
      {id:5,imgpath:"https://wallpapers.com/images/high/crazy-cartoon-8zlj1r5h52u8pvge.webp"},
      {id:6,imgpath:"https://wallpapers.com/images/high/doraemon-in-outer-space-ylg89xqr55bpg31t.webp"}
    ]
    return {mycharacters,mygadgets,mygallery};
  }
  genId<T extends character|gadget>(myTable: T[]): number {
    return Number(myTable.length > 0 ? Math.max(...myTable.map(t => t.id)) + 1 : 1);
  }
}
```

***Task 5-*** Creating character component

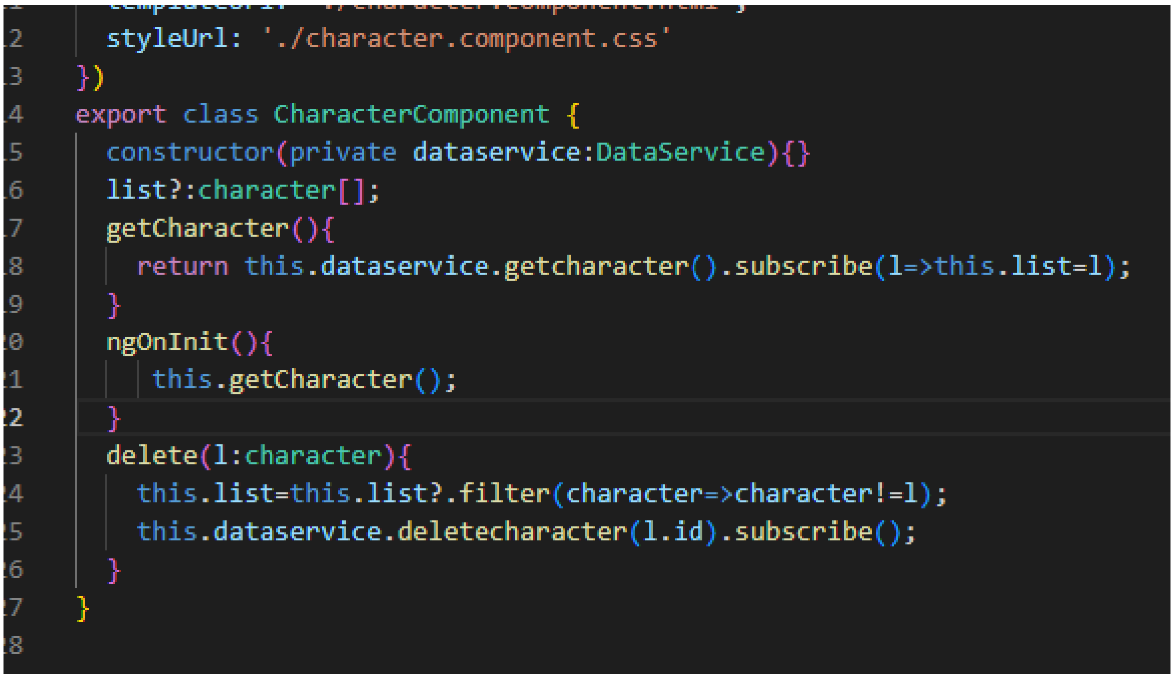
**Description-** We will be printing the list of characters in character component.

Steps to display list

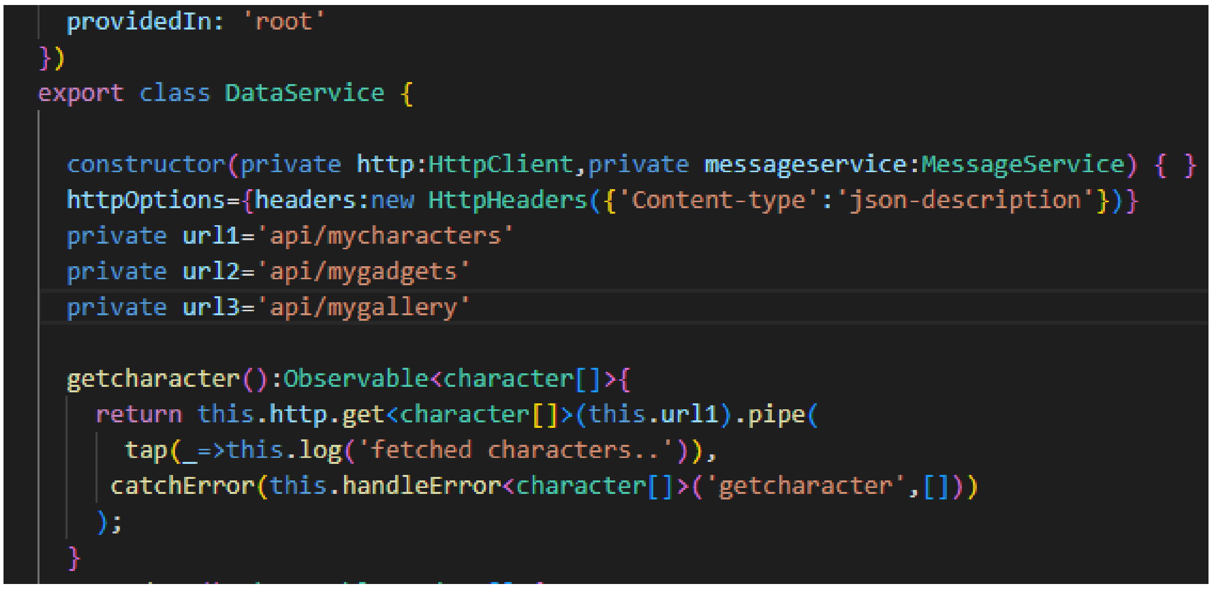
**Step 1-** To create get function and delete function in ts file of character component for getting list of character from service and giving delete functionality.
**Step 2-** Now loop over that list of character in its html file to display character name and giving delete button to each character name.
**Step 3-** Create function in service for getting character from data server using http client using url.
***Note -***Give router link to each character so that when we click on that character name then we can redirect to its detail component.



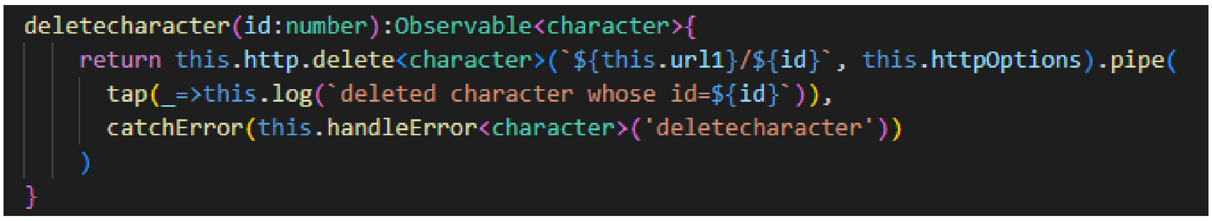
```html
<div *ngIf="list" class="list">
    <ul>
        <li *ngFor="let l of list">
            <a routerLink="/character/{{l.id}}">{{l.name}}</a>
            <button type="button" (click)="delete(l)">Delete</button>
        </li>
    </ul>
</div>
```



```typescript
  styleUrl: './character.component.css'
})
export class CharacterComponent {
  constructor(private dataservice:DataService){}
  list?:character[];
  getCharacter(){
    return this.dataservice.getcharacter().subscribe(l=>this.list=l);
  }
  ngOnInit(){
    this.getCharacter();
  }
  delete(l:character){
    this.list=this.list?.filter(character=>character!=l);
    this.dataservice.deletecharacter(l.id).subscribe();
  }
}
```

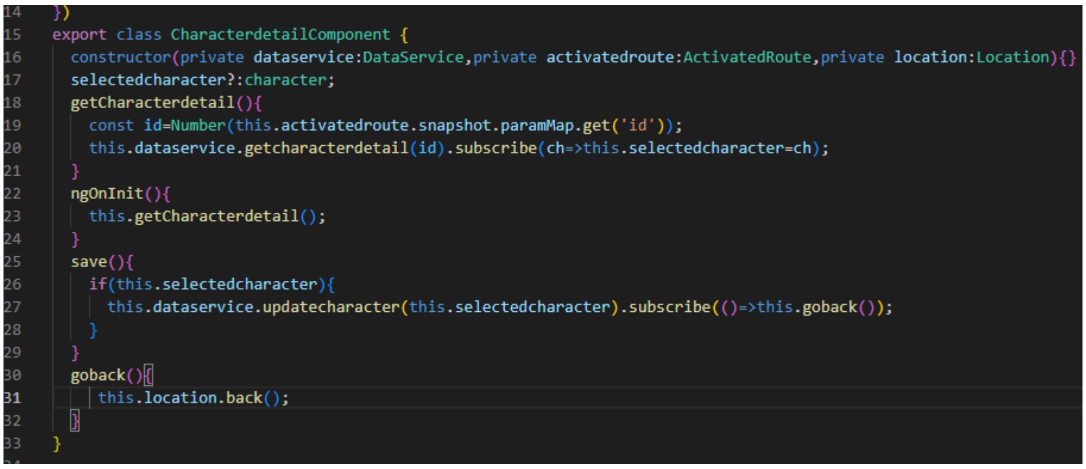


```
  providedIn: 'root'
})
export class DataService {

  constructor(private http:HttpClient,private messageservice:MessageService) { }
  httpOptions={headers:new HttpHeaders({'Content-type':'json-description'})}
  private url1='api/mycharacters'
  private url2='api/mygadgets'
  private url3='api/mygallery'

  getcharacter():Observable<character[]>{
    return this.http.get<character[]>(this.url1).pipe(
      tap(_=>this.log('fetched characters..')),
     catchError(this.handleError<character[]>('getcharacter',[]))
    );
  }
```

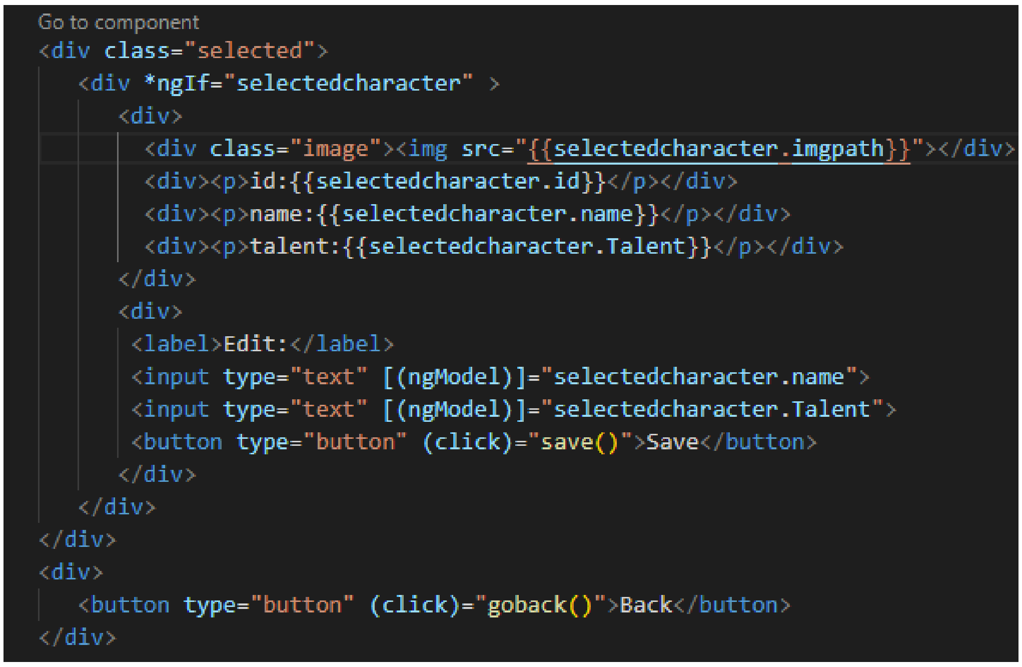


```
deletecharacter(id:number):Observable<character>{
    return this.http.delete<character>(`${this.url1}/${id}`, this.httpOptions).pipe(
      tap(_=>this.log(`deleted character whose id=${id}`)),
      catchError(this.handleError<character>('deletecharacter'))
    )
}
```

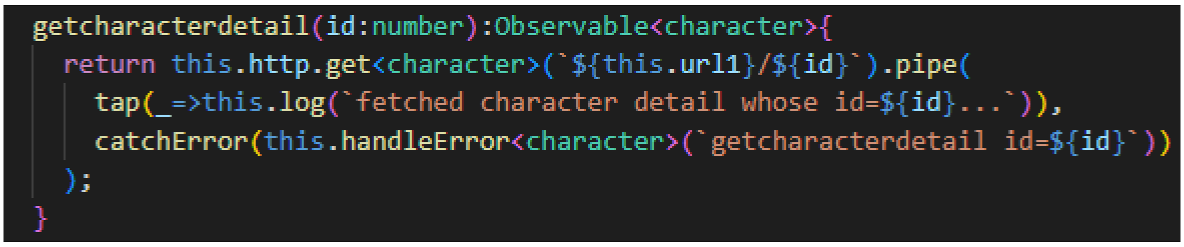
***Task 6-*** Creating character detail component
**Description-** Now we will be creating character detail component for displaying details of character on clicking on any character name
Steps to create character detail
**Step 1-** Create get, save and goback function in its .ts file for getting, updating and going back respectively , we will be using service inside these functions so for that first inject the service in its ts file .
**Step 2-** Now use that details in its html inorder to display that on the screen
**Step 3-** Create get by id  and update function in the service so as to get details of character that we clicked and update the details.
***Note-*** For getting id of particular character we will be using activated route and for going back we will be using location.



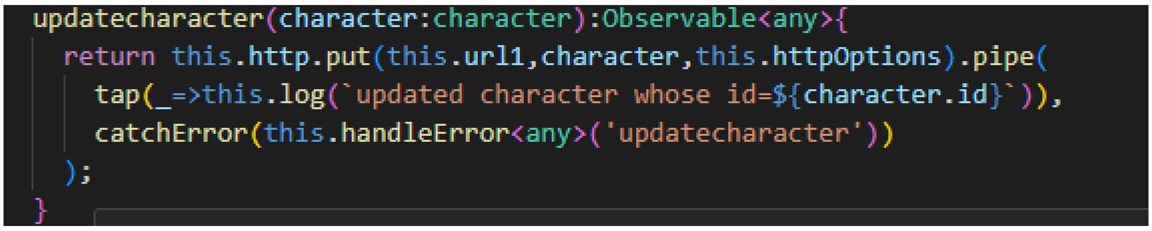
```
})
export class CharacterdetailComponent {
  constructor(private dataservice:DataService,private activatedroute:ActivatedRoute,private location:Location){}
  selectedcharacter?:character;
  getCharacterdetail(){
    const id=Number(this.activatedroute.snapshot.paramMap.get('id'));
    this.dataservice.getcharacterdetail(id).subscribe(ch=>this.selectedcharacter=ch);
  }
  ngOnInit(){
    this.getCharacterdetail();
  }
  save(){
    if(this.selectedcharacter){
      this.dataservice.updatecharacter(this.selectedcharacter).subscribe(()=>this.goback());
    }
  }
  goback(){
    this.location.back();
  }
}
```



```
Go to component
<div class="selected">
   <div *ngIf="selectedcharacter" >
      <div>
        <div class="image"><img src="{{selectedcharacter.imgpath}}"></div>
        <div><p>id:{{selectedcharacter.id}}</p></div>
        <div><p>name:{{selectedcharacter.name}}</p></div>
        <div><p>talent:{{selectedcharacter.Talent}}</p></div>
      </div>
      <div>
       <label>Edit:</label>
       <input type="text" [(ngModel)]="selectedcharacter.name">
       <input type="text" [(ngModel)]="selectedcharacter.Talent">
       <button type="button" (click)="save()">Save</button>
      </div>
   </div>
</div>
<div>
   <button type="button" (click)="goback()">Back</button>
</div>
```



```
getcharacterdetail(id:number):Observable<character>{
  return this.http.get<character>(`${this.url1}/${id}`).pipe(
    tap(_=>this.log(`fetched character detail whose id=${id}...`)),
    catchError(this.handleError<character>(`getcharacterdetail id=${id}`))
  );
}
```

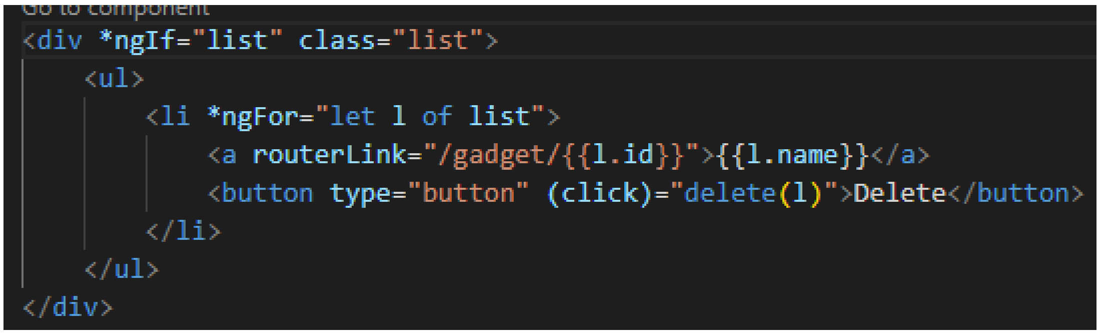


```
updatecharacter(character:character):Observable<any>{
  return this.http.put(this.url1,character,this.httpOptions).pipe(
    tap(_=>this.log(`updated character whose id=${character.id}`)),
    catchError(this.handleError<any>('updatecharacter'))
  );
}
```

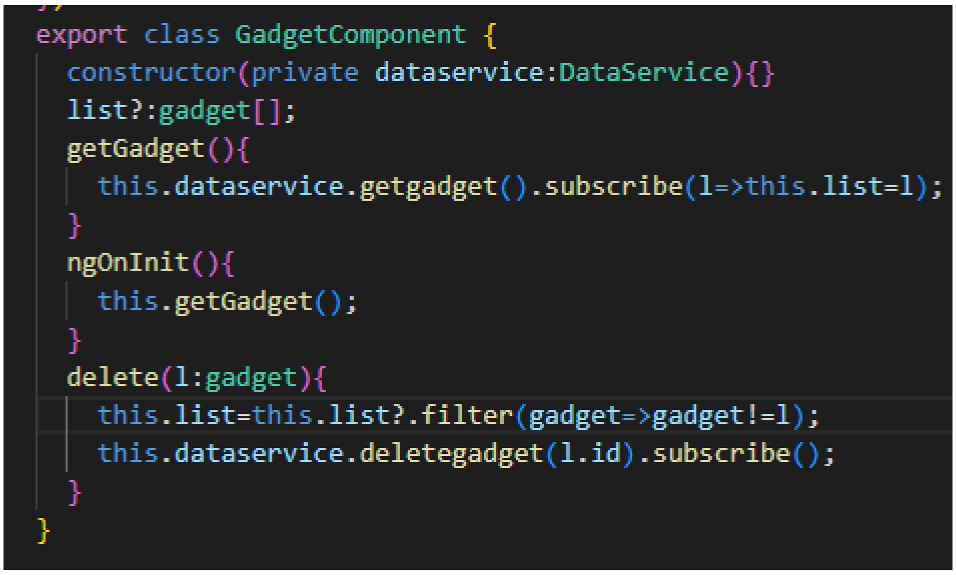
***Task 7-*** Creating Gadget component and gadget detail component

**<u>Description-</u>**Now we will be doing the same things for gadget and its details as we did for the character and its details component.

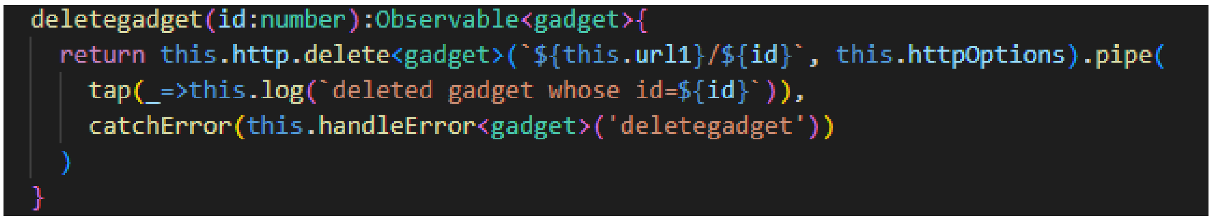
## Gadget-



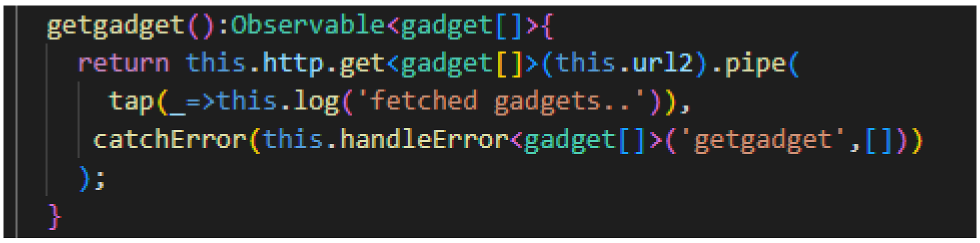
```
Go to component
<div *ngIf="list" class="list">
    <ul>
        <li *ngFor="let l of list">
            <a routerLink="/gadget/{{l.id}}">{{l.name}}</a>
            <button type="button" (click)="delete(l)">Delete</button>
        </li>
    </ul>
</div>
```



```
export class GadgetComponent {
  constructor(private dataservice:DataService){}
  list?:gadget[];
  getGadget(){
    this.dataservice.getgadget().subscribe(l=>this.list=l);
  }
  ngOnInit(){
    this.getGadget();
  }
  delete(l:gadget){
    this.list=this.list?.filter(gadget=>gadget!=l);
    this.dataservice.deletegadget(l.id).subscribe();
  }
}
```

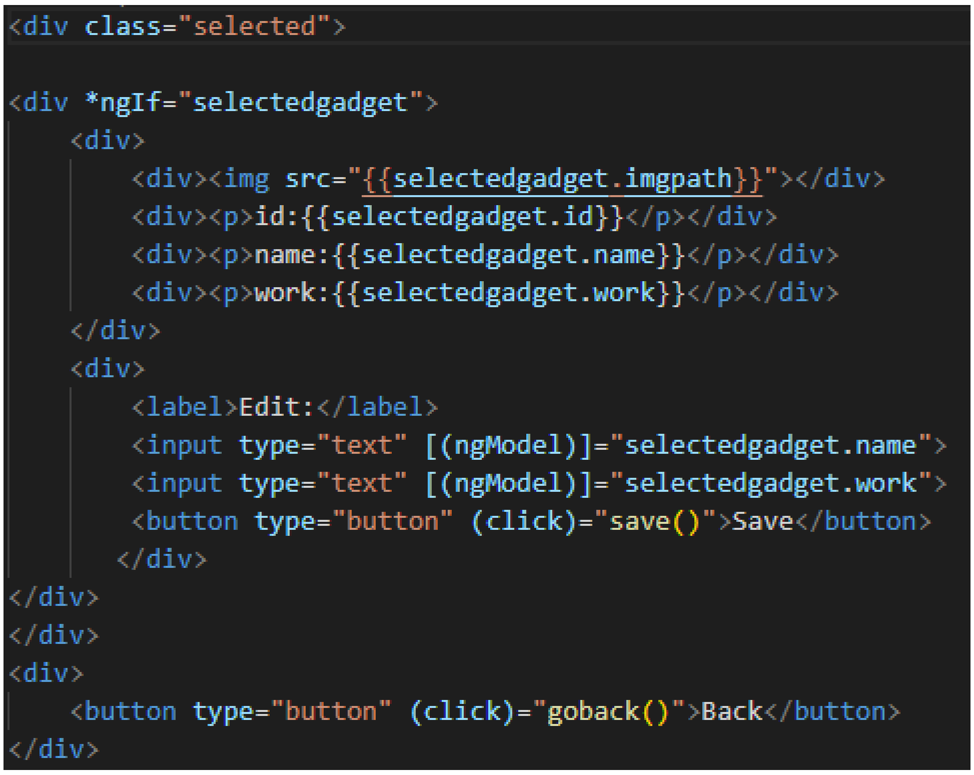


```
deletegadget(id:number):Observable<gadget>{
  return this.http.delete<gadget>(`${this.url1}/${id}`, this.httpOptions).pipe(
    tap(_=>this.log(`deleted gadget whose id=${id}`)),
    catchError(this.handleError<gadget>('deletegadget'))
  )
}
```

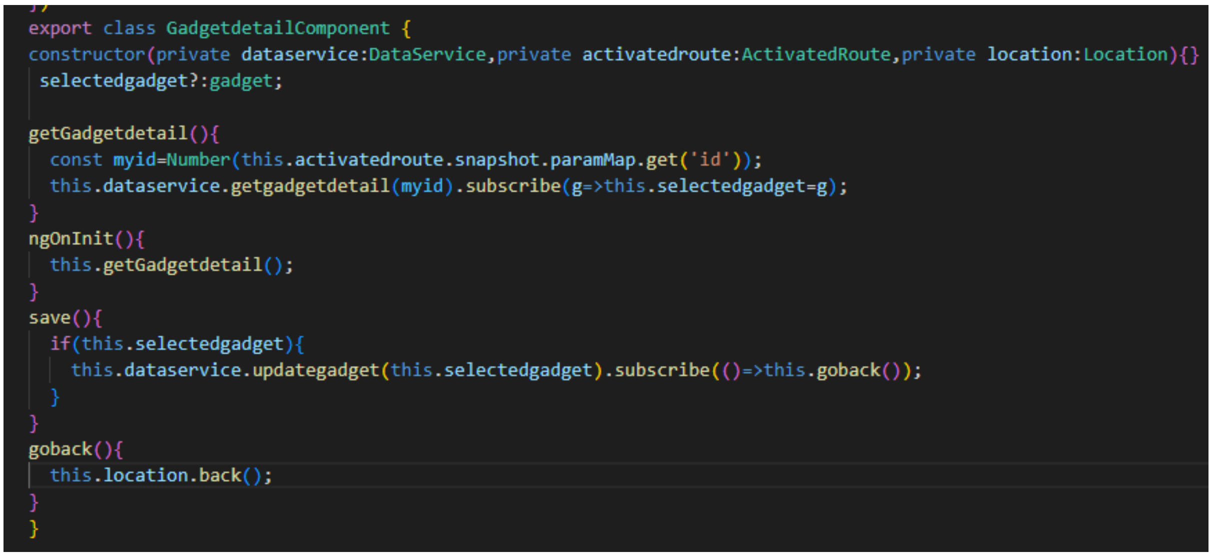


```
getgadget():Observable<gadget[]>{
  return this.http.get<gadget[]>(this.url2).pipe(
    tap(_=>this.log('fetched gadgets..')),
   catchError(this.handleError<gadget[]>('getgadget',[]))
  );
}
```

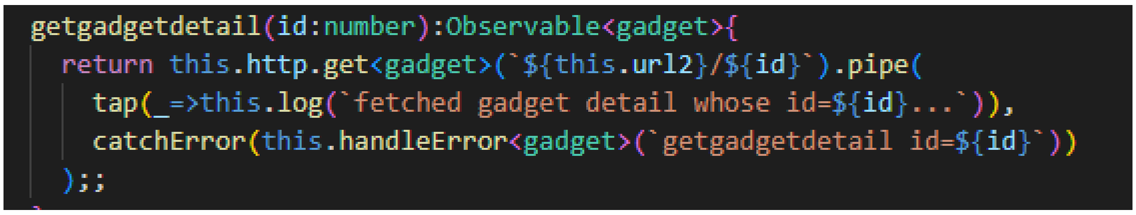
## Gadget Detail



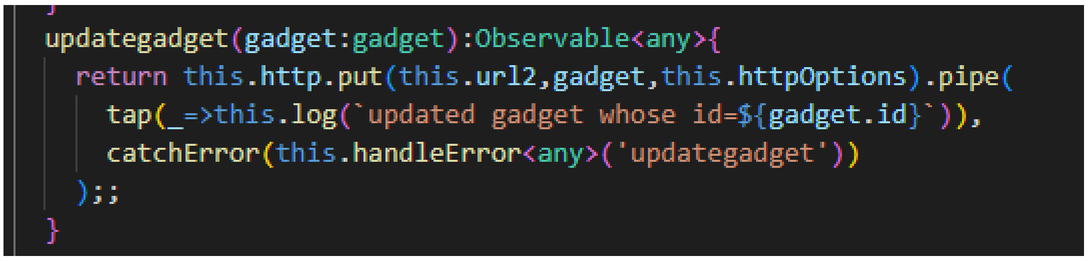
```html
<div class="selected">

<div *ngIf="selectedgadget">
    <div>
        <div><img src="{{selectedgadget.imgpath}}"></div>
        <div><p>id:{{selectedgadget.id}}</p></div>
        <div><p>name:{{selectedgadget.name}}</p></div>
        <div><p>work:{{selectedgadget.work}}</p></div>
    </div>
    <div>
        <label>Edit:</label>
        <input type="text" [(ngModel)]="selectedgadget.name">
        <input type="text" [(ngModel)]="selectedgadget.work">
        <button type="button" (click)="save()">Save</button>
       </div>
</div>
</div>
<div>
    <button type="button" (click)="goback()">Back</button>
</div>
```



```typescript
export class GadgetdetailComponent {
constructor(private dataservice:DataService,private activatedroute:ActivatedRoute,private location:Location){}
 selectedgadget?:gadget;

getGadgetdetail(){
  const myid=Number(this.activatedroute.snapshot.paramMap.get('id'));
  this.dataservice.getgadgetdetail(myid).subscribe(g=>this.selectedgadget=g);
}
ngOnInit(){
  this.getGadgetdetail();
}
save(){
  if(this.selectedgadget){
    this.dataservice.updategadget(this.selectedgadget).subscribe(()=>this.goback());
  }
}
goback(){
  this.location.back();
}
}
```



```typescript
getgadgetdetail(id:number):Observable<gadget>{
  return this.http.get<gadget>(`${this.url2}/${id}`).pipe(
    tap(_=>this.log(`fetched gadget detail whose id=${id}...`)),
    catchError(this.handleError<gadget>(`getgadgetdetail id=${id}`))
  );;
```



```typescript
 updategadget(gadget:gadget):Observable<any>{
   return this.http.put(this.url2,gadget,this.httpOptions).pipe(
     tap(_=>this.log(`updated gadget whose id=${gadget.id}`)),
     catchError(this.handleError<any>('updategadget'))
   );;
 }
```

***Task 8-*** Creating Gallery Component

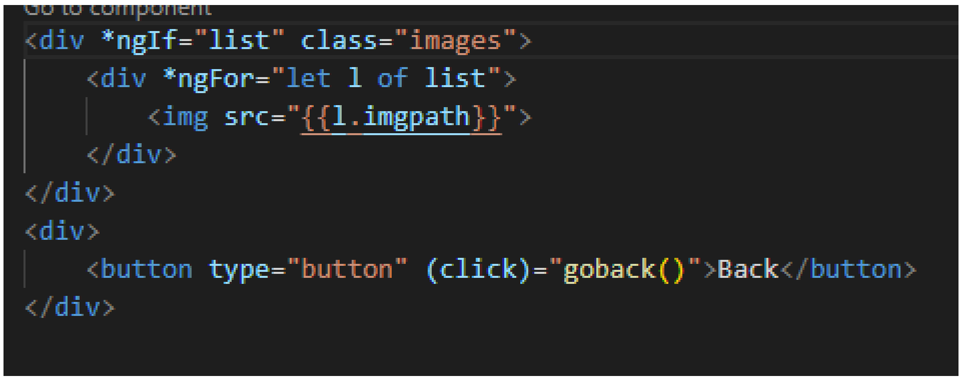
**Description-** We will be displaying the images in gallery component.

Steps to display images

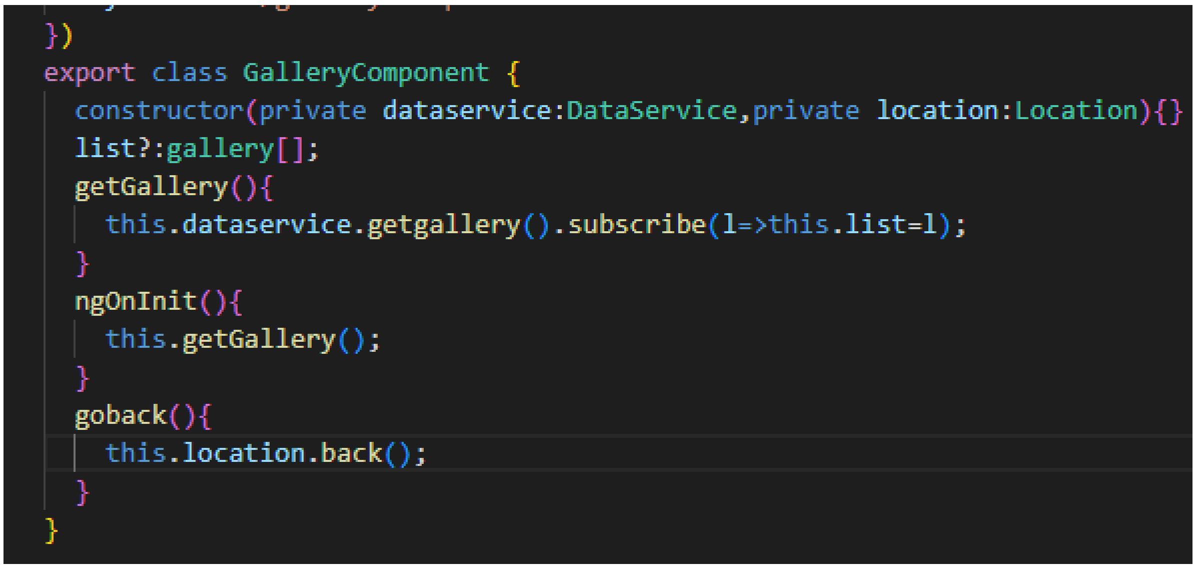
**Step 1-** To create get function in ts file of gallery component for getting those images from service.

**Step 2-** Now loop over that images in its html file to display the images.

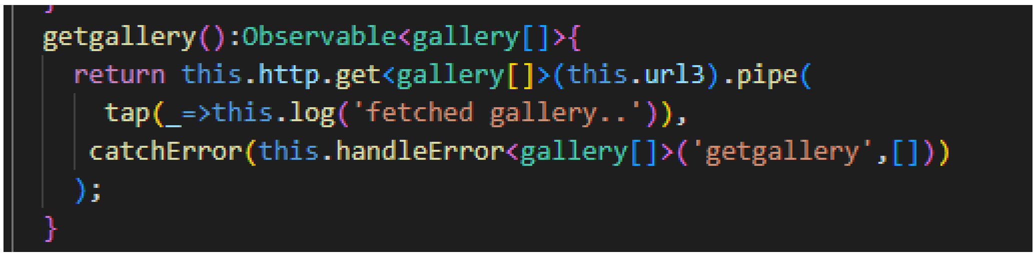
**Step 3-** Create function in service for getting images from data server using http client using url.



```
<div *ngIf="list" class="images">
    <div *ngFor="let l of list">
        <img src="{{l.imgpath}}">
    </div>
</div>
<div>
    <button type="button" (click)="goback()">Back</button>
</div>
```



```
})
export class GalleryComponent {
  constructor(private dataservice:DataService,private location:Location){}
  list?:gallery[];
  getGallery(){
    this.dataservice.getgallery().subscribe(l=>this.list=l);
  }
  ngOnInit(){
    this.getGallery();
  }
  goback(){
    this.location.back();
  }
}
```



```
getgallery():Observable<gallery[]>{
  return this.http.get<gallery[]>(this.url3).pipe(
    tap(_=>this.log('fetched gallery..')),
   catchError(this.handleError<gallery[]>('getgallery',[]))
  );
}
```

***Task 9-*** Creating Search Component

**Description -** Now we will be creating search component for searching any character or gadget.

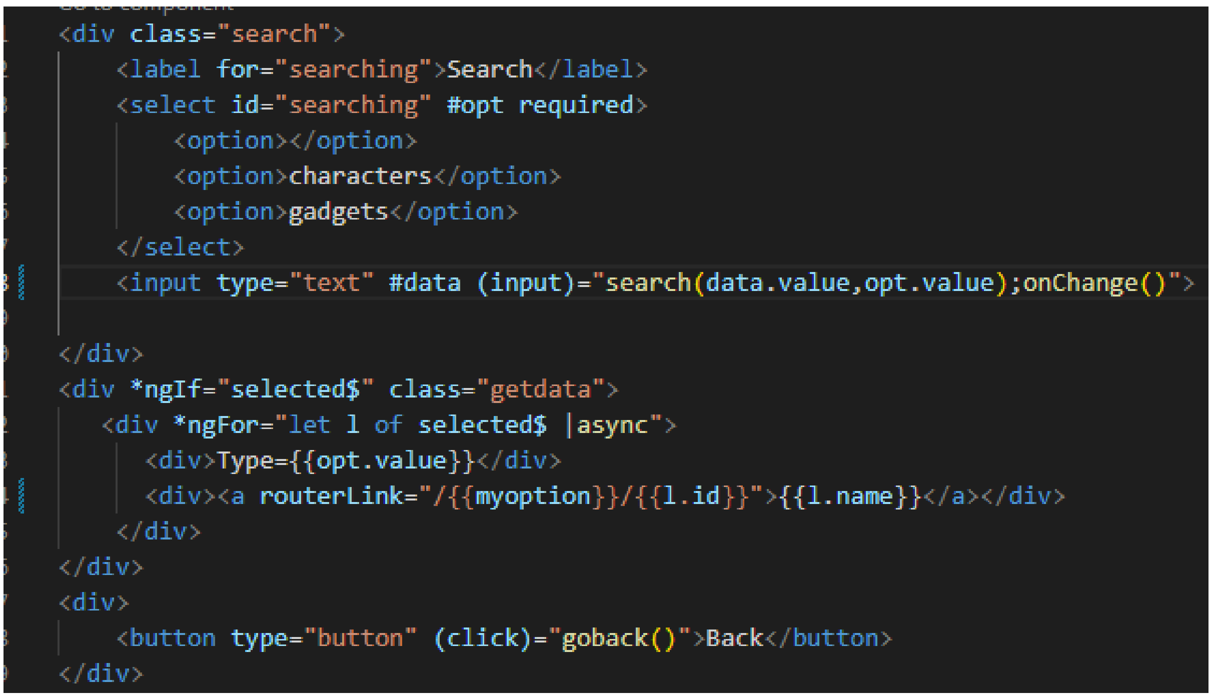
Steps to create search component

**Step 1-** We will be asking from user whether to search for character or gadget through options.so we will add options in its html file. And then we will put search bar for searching

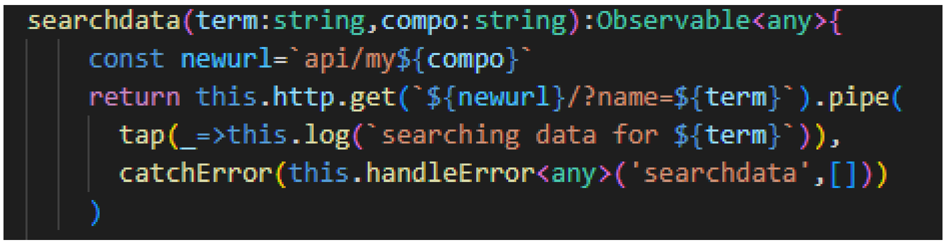
**Step 2-** We will display the display the thing according to the user search

**Step 3-** We will create function in its ts file for searching

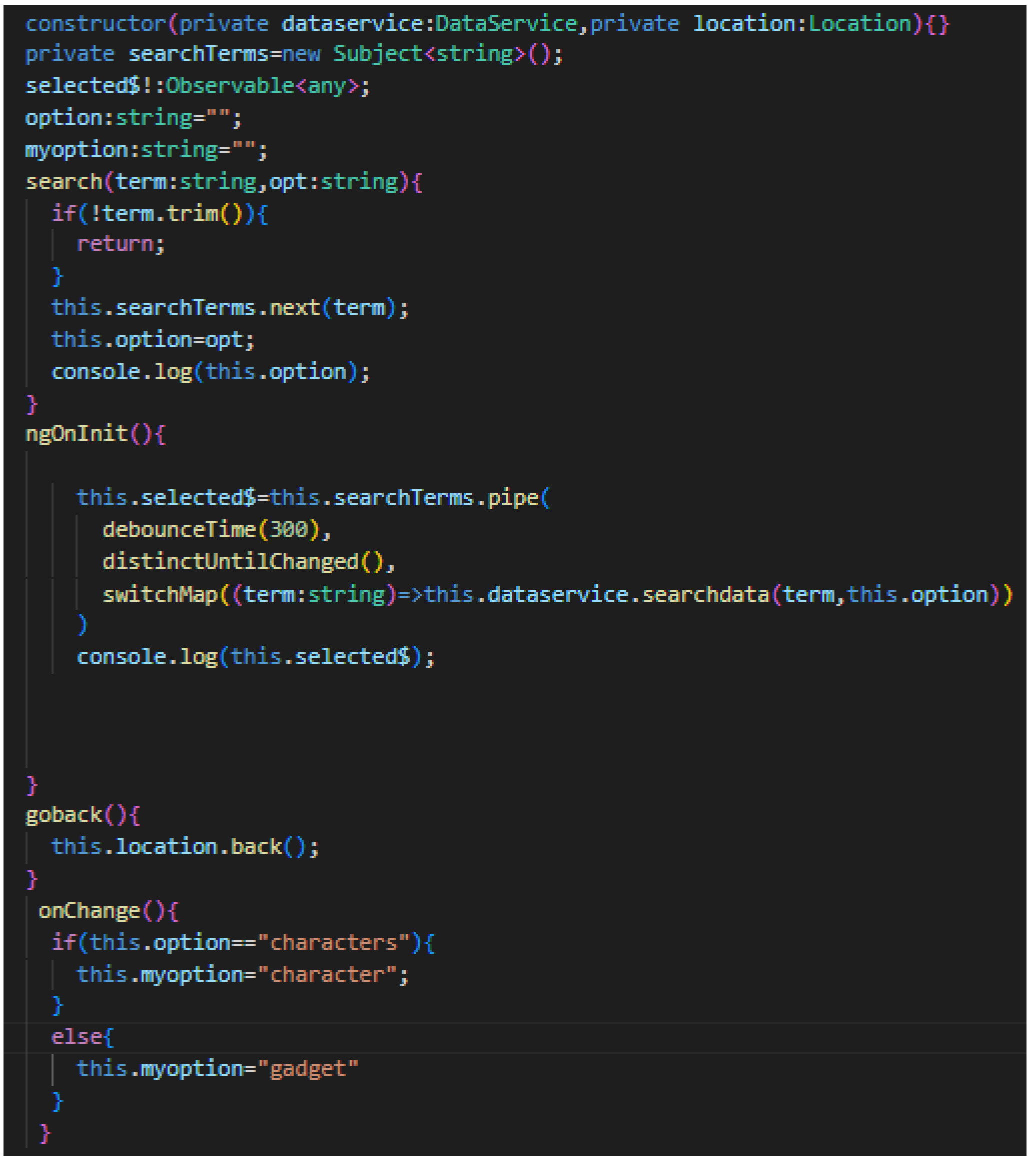
**Step 4-** Creating function for search in service using http



```
<div class="search">
    <label for="searching">Search</label>
    <select id="searching" #opt required>
        <option></option>
        <option>characters</option>
        <option>gadgets</option>
    </select>
    <input type="text" #data (input)="search(data.value,opt.value);onChange()">

</div>
<div *ngIf="selected$" class="getdata">
   <div *ngFor="let l of selected$ |async">
      <div>Type={{opt.value}}</div>
      <div><a routerLink="/{{myoption}}/{{l.id}}">{{l.name}}</a></div>
    </div>
</div>
<div>
    <button type="button" (click)="goback()">Back</button>
</div>
```



```
searchdata(term:string,compo:string):Observable<any>{
    const newurl=`api/my${compo}`
    return this.http.get(`${newurl}/?name=${term}`).pipe(
      tap(_=>this.log(`searching data for ${term}`)),
      catchError(this.handleError<any>('searchdata',[]))
    )
```



```
constructor(private dataservice:DataService,private location:Location){}
private searchTerms=new Subject<string>();
selected$!:Observable<any>;
option:string="";
myoption:string="";
search(term:string,opt:string){
  if(!term.trim()){
    return;
  }
  this.searchTerms.next(term);
  this.option=opt;
  console.log(this.option);
}
ngOnInit(){

    this.selected$=this.searchTerms.pipe(
      debounceTime(300),
      distinctUntilChanged(),
      switchMap((term:string)=>this.dataservice.searchdata(term,this.option))
    )
    console.log(this.selected$);



}
goback(){
  this.location.back();
}
 onChange(){
  if(this.option=="characters"){
    this.myoption="character";
  }
  else{
    this.myoption="gadget"
  }
 }
```

***<u>Task 10-</u>*** Creating Add Component

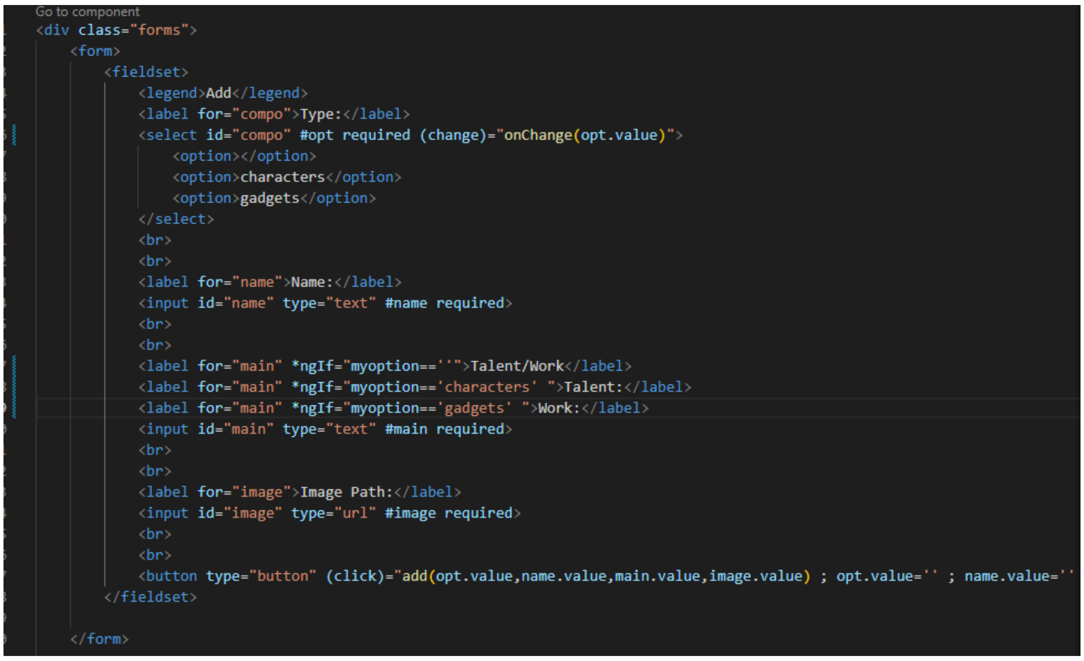
**<u>Description-</u>** Now we will be creating add component for adding new character or new gadget.

<u>Steps for creating add component</u>

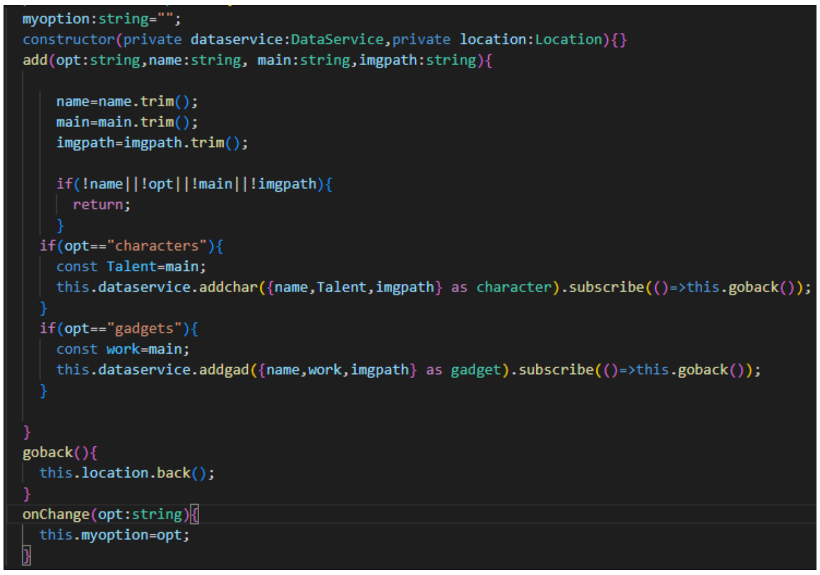
**Step 1-** First we will be asking from user all the information about what to be added.in its html file.

**Step 2-** We will be creating add function in its ts file for making the data added and for that we will be using service .

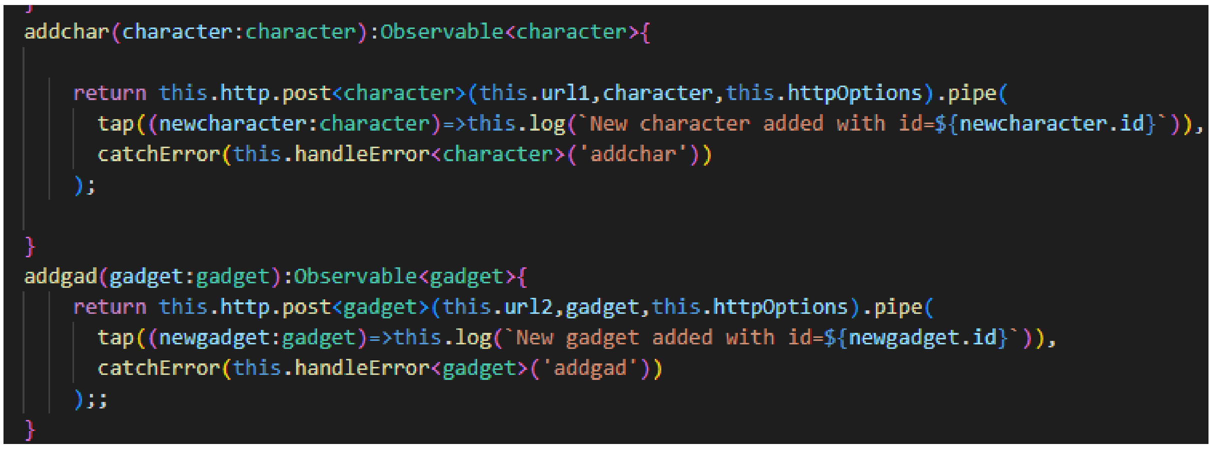
**Step 3-** In service create function for adding data.



```html
Go to component
<div class="forms">
    <form>
        <fieldset>
            <legend>Add</legend>
            <label for="compo">Type:</label>
            <select id="compo" #opt required (change)="onChange(opt.value)">
                <option></option>
                <option>characters</option>
                <option>gadgets</option>
            </select>
            <br>
            <br>
            <label for="name">Name:</label>
            <input id="name" type="text" #name required>
            <br>
            <br>
            <label for="main" *ngIf="myoption==''">Talent/Work</label>
            <label for="main" *ngIf="myoption=='characters' ">Talent:</label>
            <label for="main" *ngIf="myoption=='gadgets' ">Work:</label>
            <input id="main" type="text" #main required>
            <br>
            <br>
            <label for="image">Image Path:</label>
            <input id="image" type="url" #image required>
            <br>
            <br>
            <button type="button" (click)="add(opt.value,name.value,main.value,image.value) ; opt.value='' ; name.value=''
        </fieldset>

    </form>
```



```typescript
myoption:string="";
constructor(private dataservice:DataService,private location:Location){}
add(opt:string,name:string, main:string,imgpath:string){

    name=name.trim();
    main=main.trim();
    imgpath=imgpath.trim();

    if(!name||!opt||!main||!imgpath){
      return;
    }
  if(opt=="characters"){
    const Talent=main;
    this.dataservice.addchar({name,Talent,imgpath} as character).subscribe(()=>this.goback());
  }
  if(opt=="gadgets"){
    const work=main;
    this.dataservice.addgad({name,work,imgpath} as gadget).subscribe(()=>this.goback());
  }

}
goback(){
  this.location.back();
}
onChange(opt:string){
  this.myoption=opt;
}
```



```typescript
addchar(character:character):Observable<character>{

    return this.http.post<character>(this.url1,character,this.httpOptions).pipe(
      tap((newcharacter:character)=>this.log(`New character added with id=${newcharacter.id}`)),
      catchError(this.handleError<character>('addchar'))
    );

}
addgad(gadget:gadget):Observable<gadget>{
    return this.http.post<gadget>(this.url2,gadget,this.httpOptions).pipe(
      tap((newgadget:gadget)=>this.log(`New gadget added with id=${newgadget.id}`)),
      catchError(this.handleError<gadget>('addgad'))
    );;
}
```

***Task 11-*** Creating Message component

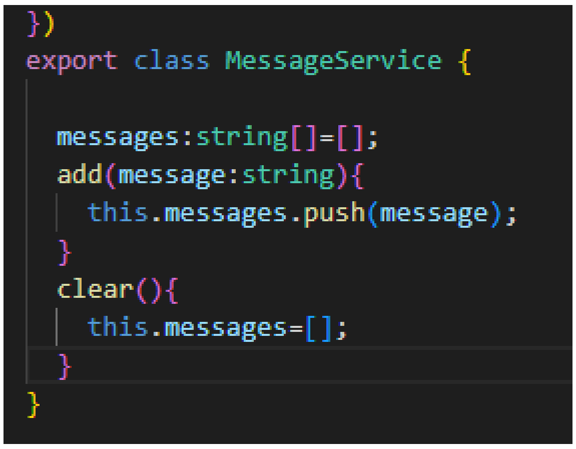
**Description-** We will be creating message component for displaying the messages .

Steps to create message component

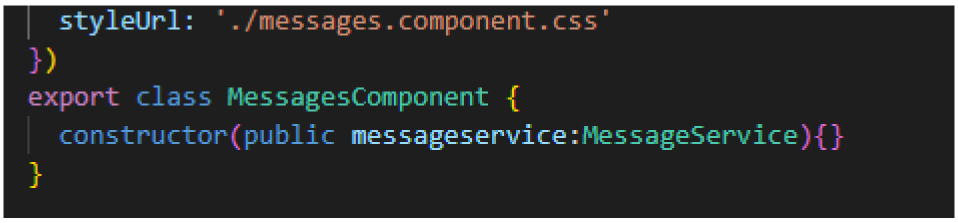
**Step 1-** Create message service and in that service create array of message for storing all the messages and create add and clear message functions

**Step 2-** Inject the message service in its ts file

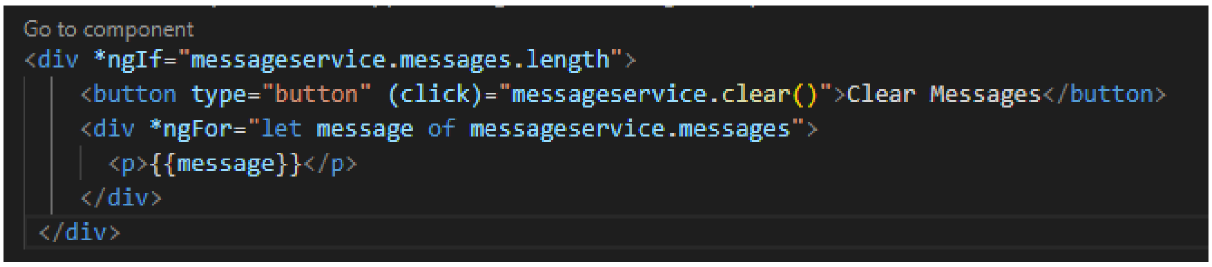
**Step 3-** Loop over the messages to display them on the screen.



```
})
export class MessageService {

  messages:string[]=[];
  add(message:string){
    this.messages.push(message);
  }
  clear(){
    this.messages=[];
  }
}
```



```
  styleUrl: './messages.component.css'
})
export class MessagesComponent {
  constructor(public messageservice:MessageService){}
}
```



```
Go to component
<div *ngIf="messageservice.messages.length">
    <button type="button" (click)="messageservice.clear()">Clear Messages</button>
    <div *ngFor="let message of messageservice.messages">
      <p>{{message}}</p>
    </div>
</div>
```

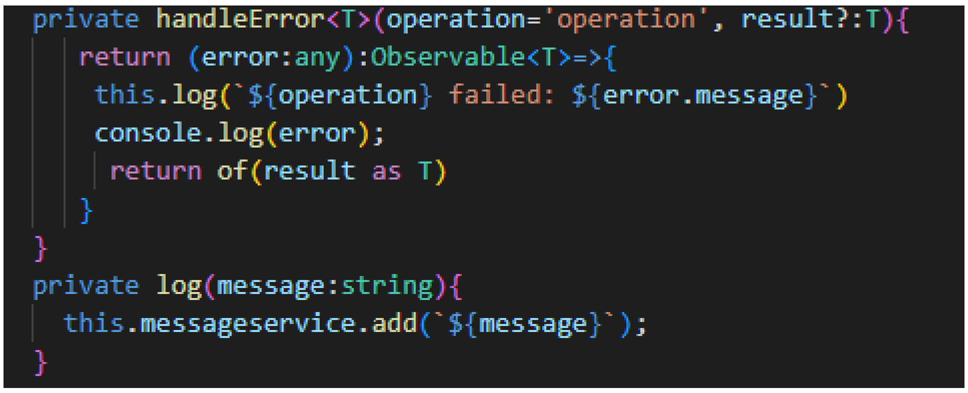
***Task 12-*** Implementing Error Handling

**Description -** Now we will be implementing error handling in service.

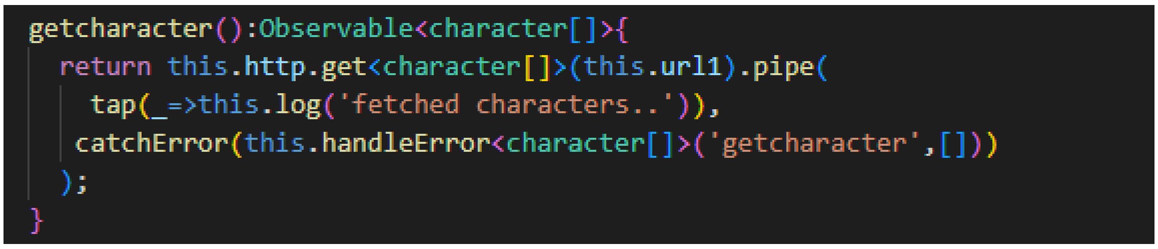
Steps for implementing error handling

**Step 1-** First we have to import tap, catchError, map , pipe

**Step 2-** Use catch error and tap after all the http requests.
**Step 3-** Implement handle error function for handling the error if any occur.



```
private handleError<T>(operation='operation', result?:T){
    return (error:any):Observable<T>=>{
     this.log(`${operation} failed: ${error.message}`)
     console.log(error);
      return of(result as T)
    }
}
private log(message:string){
  this.messageservice.add(`${message}`);
}
```



```
getcharacter():Observable<character[]>{
  return this.http.get<character[]>(this.url1).pipe(
    tap(_=>this.log('fetched characters..')),
   catchError(this.handleError<character[]>('getcharacter',[]))
  );
}
```