

1. Przygotowanie projektu. Ściągnij projekt przykładowy z portalu edux. Zainstaluj moduły dla node'a
Npm install
Oraz uruchom aplikację
Ng serve

(wszystkie komponenty opisane w tym dokumencie będą tworzone komendą `ng generate component <nazwa-komponentu>`)

2. Naszym zadaniem będzie napisanie interfejsu użytkownika do wyszukiwania filmików z serwisu youtube. W tym celu trzeba będzie napisać klasy które są w stanie odpytać api serwisu o informacje o filmach – klasa `SearchResult` do przechowywania danych z odpowiedzi zapytania oraz klasa `YoutubeSearchService` która będzie posiadać logikę do wykonania zapytań http do api restowego youtube'a.
Zacznijmy od klasy `SearchResult`

W katalogu `app` do podkatalog `youtube` a w nim plik `search-result.ts`

```
export class SearchResult {
  id: string;
  title: string;
  description: string;
  thumbnailUrl: string;
  videoUrl: string;

  constructor(obj?: any) {
    this.id = obj && obj.id || null;
    this.title = obj && obj.title || null;
    this.description = obj && obj.description || null;
    this.thumbnailUrl = obj && obj.thumbnailUrl || null;
    this.videoUrl = obj && obj.videoUrl ||
      `https://www.youtube.com/watch?v=${this.id}`;
  }
}
```

Dodajmy teraz klasę `YoutubeService` w pliku `youtube.service.ts`

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

@Injectable()
export class YoutubeService {

  constructor() { }
}
```

Oraz zadeklarujmy ją w module aplikacji – będzie także potrzeba zadeklarowania modułu `HttpModule`

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule
  ],
  providers: [YoutubeService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Do klasy dodajmy pola odnośnie adresu URL do api restowego oraz klucza do usług

```
@Injectable()
export class YoutubeService {

  private apiKey = 'AIzaSyDxugcveCG_gLvxiAvS01jGE8QV6rdw4U4';
  private apiUrl = 'https://www.googleapis.com/youtube/v3/search';

  constructor() { }

}
```

Należy pamiętać, że powyższy klucz jest tylko kluczem tymczasowym – może on przestać działać- dobrze jest sobie samemu wygenerować własny klucz według instrukcji:

https://developers.google.com/youtube/registering_an_application#Create_API_Keys

(interesuje nas sekcja Create Api Keys)

Do konstruktora naszej klasy dodajmy referencje do angularowego serwisu http – będzie nam potrzebny do komunikacji z youtube'em.

```
@Injectable()
export class YoutubeService {

  private apiKey = 'AIzaSyDxugcveCG_gLvxiAvS01jGE8QV6rdw4U4';
  private apiUrl = 'https://www.googleapis.com/youtube/v3/search';

  constructor(private http : Http) { }

}
```

Teraz zaimplementujemy funkcję search, która przyjmie jako argument query z adresu URL a zwróci obiekt Observable, który będzie emitował strumień SearchResult[] (czyli kolekcje naszych wyników).

```

search(query: string):Observable<SearchResult[]>{
  const params : string =[
    `q=${query}`,
    `key=${this.apiKey}`,
    `part=snippet`,
    `type=video`,
    `maxResults=10`
  ].join('&');

  const queryUrl = `${this.apiUrl}?${params}`;
  return this.http.get(queryUrl)
    .map((response:Response)=>{
      return (<any>response.json()).items.map(item=>{
        return new SearchResult({
          id: item.id.videId,
          title: item.snippet.title,
          description: item.snippet.description,
          thumbnailUrl: item.snippet.thumbnails.high.url
        });
      });
    });
}

```

Jeszcze warto zwrócić uwagę na importy w klasie

```

import { Injectable } from '@angular/core';
import { Http, Response } from '@angular/http';
import { Observable } from "rxjs/Observable";
import "rxjs/add/operator/map";
import { SearchResult } from "../search-result";

```

3. Komponent do wyszukiwania wyników. W tej sekcji dodamy nowy component SearchBoxComponent, którego odpowiedzialność będzie się skupiała na obserwowaniu zdarzeń wpisywania fraz do wyszukiwania filmów, emitowania zdarzeń do ładowania wyników oraz emitowanie zdarzenia odbioru wyników.

Zacznijmy od zdefiniowania podstawowych pól tej klasy, konstruktora oraz szablonu html

```

import { Component, OnInit, EventEmitter, Output, ElementRef } from
'@angular/core';
import { SearchResult } from "app/youtube/search-result";
import { YoutubeService } from "app/youtube/youtube.service";

@Component({

```

```

    selector: 'app-search-box',
    templateUrl: './search-box.component.html',
    styleUrls: ['./search-box.component.css']
  })
  export class SearchBoxComponent implements OnInit {

    @Output() loading: EventEmitter<boolean>
      = new EventEmitter<boolean>();
    @Output() results: EventEmitter<SearchResult[]>
      = new EventEmitter<SearchResult[]>();

    constructor(private youtube: YoutubeService,
                 private el: ElementRef) { }

    ngOnInit() {
    }
  }
}

```

Gdzie w konstruktorze el : ElementRef jest wrapperem na natywny element szablonu html
Szablon html:

```
<input type="text" class="form-control" placeholder="Search" autofocus>
```

Teraz wykorzystajmy właściwości Observable tak aby przypisać zdarzeniom wpisywania tekstu to search box'a właściwości:

- Akcja szukanie nie wykona się dopóki wpisano za mało znaków
- Event się wywoła z opóźnieniem 250 ms
- Gdy zacznie się wywołanie eventu keyup ma zostać wywołany event Loading
- Gdy zakończy się ściąganie danych o filmach ma zostać wywołany event results

W tym celu należy zmodyfikować metodę ngOnInit w następujący sposób:

```

ngOnInit() {
  Observable.fromEvent(this.el.nativeElement, 'keyup')
    .map((e: any) => e.target.value)
    .filter((text: string) => text.length > 1)
    .debounceTime(250)
    .do(() => this.loading.emit(true))
    .map((query: string) => this.youtube.search(query))
    .switch()
    .subscribe((results: SearchResult[]) => {
      this.loading.emit(false);
      this.results.emit(results)
    }, (err) => {
      console.log(err);
      this.loading.emit(false);
    },
    () => this.loading.emit(false));
}

```

4. Komponent do reprezentacji wyników. Utworzymy tutaj nowy komponent `SearchResultComponent`, które będzie nam w ładny sposób reprezentował wynik który otrzymamy z serwisu youtube'a

```
import { Component, OnInit, Input } from '@angular/core';
import { SearchResult } from "app/youtube/search-result";

@Component({
  selector: 'app-search-result',
  templateUrl: './search-result.component.html',
  styleUrls: ['./search-result.component.css']
})
export class SearchResultComponent implements OnInit {

  @Input() result: SearchResult;

  constructor() { }

  ngOnInit() {
  }

}
```

Jak widać jest to prosty komponent do którego będziemy ładować wynik naszego zapytania. Jego szablon będzie wyglądał następująco:

```
<div class="col-sm-6 col-md-3">
  <div class="thumbnail">
    
    <div class="caption">
      <h3>{{result.title}}</h3>
      <p>{{result.description}}</p>
      <p>
        <a href="{{result.videoUrl}}" class="btn btn-default">Watch</a>
      </p>
    </div>
  </div>
</div>
```

5. Komponent wiążący wszystko w całość. Ostatnim krokiem będzie dodanie komponentu `YoutubeSearchComponent` który będzie reprezentował całą naszą pracę. Będzie on odpowiadał za pokazanie ikonki ładowania wyników, reagowanie na zdarzenia wywoływane przez poprzednio napisany komponent oraz reprezentowanie wyników. Zaczniemy od napisania prostej klasy komponentu

```
import { Component, OnInit } from '@angular/core';
import { SearchResult } from "app/youtube/search-result";

@Component({
```

```

    selector: 'app-youtube-search',
    templateUrl: './youtube-search.component.html',
    styleUrls: ['./youtube-search.component.css']
  })
  export class YoutubeSearchComponent implements OnInit {

    results: SearchResult[];
    loading: boolean;
    constructor() { }

    ngOnInit() {
    }

    updateResults(results: SearchResult[]): void {
      this.results = results;
    }
  }
}

```

Na koniec szablon:

```

<div class="conatainer">
  <div class="page-header">
    <h1>Youtube Search
      <img style="float:right;"
        *ngIf="loading"
        src='assets/images/loading.gif' />
    </h1>
  </div>

  <div class="row">
    <div class="input-group input-group-lg col-md-12">
      <app-search-box
        (loading)="loading=$event"
        (results)="updateResults($event)">
      </app-search-box>
    </div>
  </div>

  <div class="row">
    <app-search-result
      *ngFor="let result of results"
      [result]=result>
    </app-search-result>
  </div>
</div>

```

Nie zapomnij dodać tego komponentu do app.component.html.

Uruchom aplikację komendą ng serve

HttpDemo

localhost:4200


Adam

⌵

🔍 ⭐ 📶 🔊 🔌

Youtube Search


cats



CATS you will remember and LAUGH all day! - World's funniest cat videos

Here's a challenge for you: watch this and try not to smile or move your lips! It's simply impossible! These cats are too funny! Just look how all these cats & kittens ...


Watch



Cats are so funny you will die laughing - Funny cat compilation

Cats are simply the funniest and most hilarious pets, they make us laugh all the time! Just look how all these cats & kittens play, fail, get along with dogs and ...


Watch



CATS will make you LAUGH YOUR HEAD OFF - Funny CAT compilation

Cats are amazing creatures because they make us laugh all the time! Watching funny cats is the hardest try not to laugh challenge! Just look how all these cats ...

Watch



Funny Cats Compilation [Most See] Funny Cat Videos Ever Part 1

Funny Cats Compilation - Funny Cat Videos - Funny Videos
Check out the NEW Part 2 Compilation!
<https://youtu.be/XHr2cvkA8rQ> Funny Cats Compilation Try ...

Watch

