Angular video applicaton

Developed by Josip Kenjerić

1. Getting started

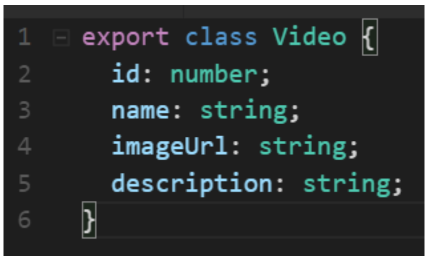
Technologies used for this solutions are Angular JS 2, TypeScript, Node JS and Express JS. Angular 2 in combination with TypeScript is used to develop the application itself and Node with Express to develop API that will serve the application needs. Application was developed in Visual Studio Code.

2. Solution

Project is setup with angular-seed project that is found "*https://github.com/angular/quickstart.git*". This is a starting point for developing our application.

2.1. Classes

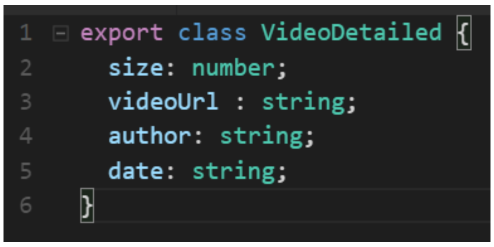
Classes are written in TypeScript. There are two classes used for this solution: video class and VideoDetailed class. This approach is used because it is easier to have all videos saved at one JSON (JSON in this example) and get additional information for selected video trough new JSON. Like this it is possible to distinguish information that is needed for different views (although in this example both information will be used in presented views). Video class is going to present basic information about the videos such as id of the video, description and name of the video and thumbnail or image that is going to be set next to the information (Figure 1).



```
export class Video {
  id: number;
  name: string;
  imageUrl: string;
  description: string;
}
```

Figure 1 - Video class properties

VideoDetailed class on the other hand saves different information about the video such as size, author, date and link to video (Figure 2). In this example for each entry in videos JSON, application needs JSON that shares the name of the id property of video class to be able to fetch additional information about the video. If the file does not exist server will return 404.

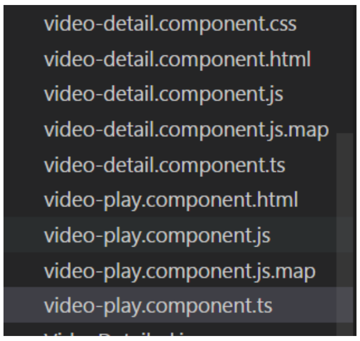


```
export class VideoDetailed {
  size: number;
  videoUrl : string;
  author: string;
  date: string;
}
```

Figure 2 - VideoDetailed class properties

2.2. Angular Components

As mentioned earlier Angular 2 is used to develop this project. There are two main components apart from app.component (Figure 3). Components are great way to make code reusable.



```
video-detail.component.css
video-detail.component.html
video-detail.component.js
video-detail.component.js.map
video-detail.component.ts
video-play.component.html
video-play.component.js
video-play.component.js.map
video-play.component.ts
```

Figure 3 - List of projects components

First component is video-detail component. Video-detail component is used to show all the videos saved in JSON containing basic video information. This component upon initialisation fetches all the videos and places them on the site. When site is being initialised *ngOnInit()* method is called what triggers *getVideos()* method to be called and load information from JSON to scope. Videos are fetched trough service that will be explained later. There are three more methods used: *onSelect(video)* which highlights selected video, *goToDetail()* that will navigate to other view where video can be played and *generatePopUp(video)* which will fetch additional information from the selected video and present it in a pop up box above the additional information tab this method also grabs information trough service. *ngFor directive is used to fill list with videos and *ngIf directive is used to check if the object is ready. Rest of HTML and CSS for this example is basic and it will not be explained.

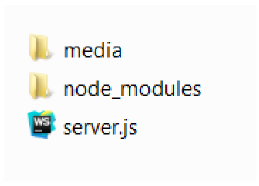
Second component is video-play. This component is called when user navigates from videos to desired video (*goToDetail()* navigates trough selected id). Here upon initialisation *ngOnInit()* method is called which grabs information about the video and once again this information is grabbed trough service. In HTML additional information about the video is presented. Video is played trough HTML5 video tag. Framework video.js is added to project and it is used for a player but somehow it is not giving desired result. As well three buttons are added to help navigate between videos: one button that returns user to videos page, one button for next and one for previous video. *ngIf directive is used once again to check if object is ready.

Service is used to easily manage data calls - like this we don't have to rewrite the code over and over. Video service is used to get and parse JSON from our server. There are two main methods *getVideos()* that fetch JSON with all videos and *getVideo(id)* that fetch JSON with additional information about the video. Third method *getVideoOriginal(id)* should return basic information about the video in play video view.

In app.component selector router-outlet is used. Trough this selector in app.module and ngModule we can assign our selectors that we have created in our components (my-video-detail and play-video) to different paths. Basically path will define component or selector that will be served.

2.3. Backend

To setup backend we have to install Node.js first on our machine. Framework express.js for node is used. Our sever is simple (Figure 4). Media folder have all JSON and static files images and videos.

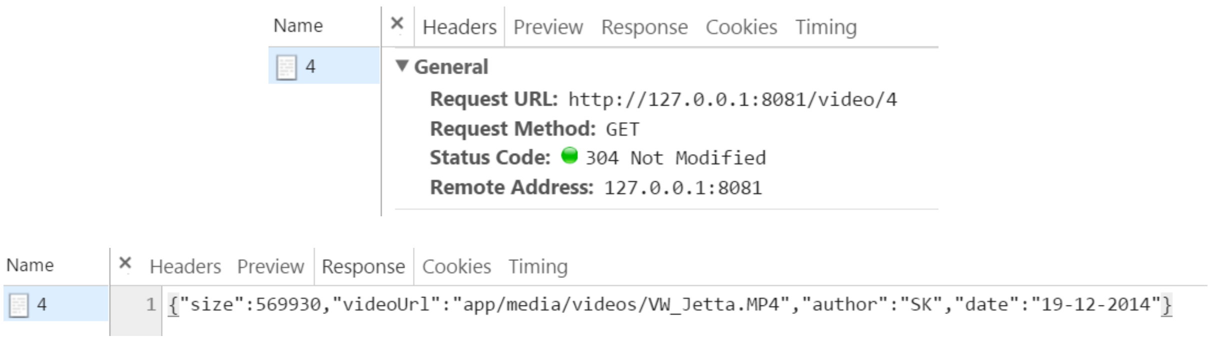


```
media
node_modules
server.js
```

Figure 4 - Backend folder structure

Our server file is setup as follows: express is included, file reading is included, cross origin is enabled and media is set as static folder. To be able to read JSON files with data for our application we need read their content to be able to serve them (JSON files are not served as static). To be able to get resources from different domains Cross-Origin HTTP request has to be enabled. Since backend works on port 8081 and application on port 3000. All videos and images are served statically.

To get our JSON two different gets are in server file. When user sends request to "*get_videos*" JSON with all videos will be in response. But when user sends request to "*video/:uid*" depending on id that was sent in :uid JSON with additional details about video will be in response (Figure 5).



```
▼ General
   Request URL: http://127.0.0.1:8081/video/4
   Request Method: GET
   Status Code: ● 304 Not Modified
   Remote Address: 127.0.0.1:8081
```

```
{"size":569930,"videoUrl":"app/media/videos/VW_Jetta.MP4","author":"SK","date":"19-12-2014"}
```

Figure 5 - Response for video with id 4

3. Application

3.1. Navigating and views

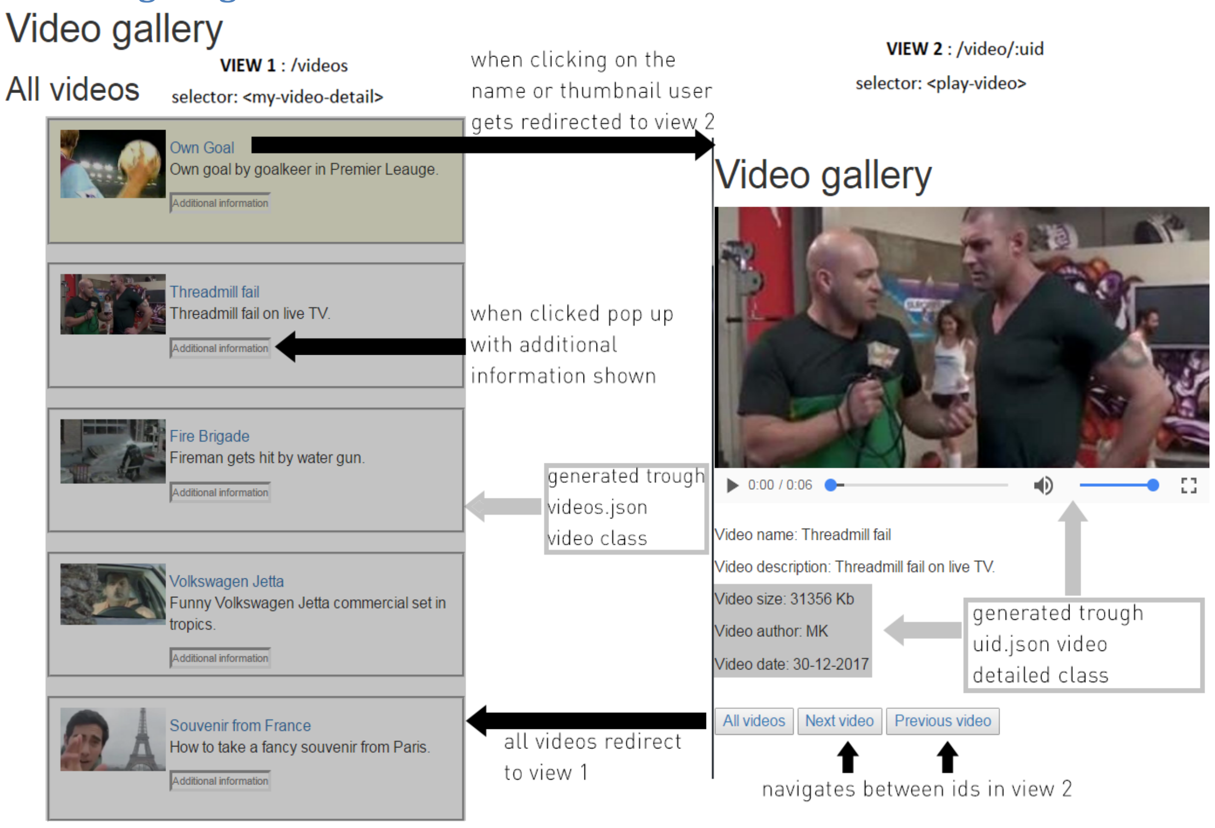

Video gallery
All videos
VIEW 1 : /videos
selector: <my-video-detail>
Own Goal
Own goal by goalkeer in Premier Leauge.
Additional information
Threadmill fail
Threadmill fail on live TV.
Additional information
Fire Brigade
Fireman gets hit by water gun.
Additional information
Volkswagen Jetta
Funny Volkswagen Jetta commercial set in tropics.
Additional information
Souvenir from France
How to take a fancy souvenir from Paris.
Additional information
when clicking on the name or thumbnail user gets redirected to view 2
when clicked pop up with additional information shown
generated trough videos.json video class
all videos redirect to view 1
VIEW 2 : /video/:uid
selector: <play-video>
Video gallery
0:00 / 0:06
Video name: Threadmill fail
Video description: Threadmill fail on live TV.
Video size: 31356 Kb
Video author: MK
Video date: 30-12-2017
generated trough uid.json video detailed class
All videos
Next video
Previous video
navigates between ids in view 2

Figure 6 - Views in application and their relations

3.2. Starting application

Application runs on two ports (8081 backend and 3000 frontend). To get application running clone/copy git repo "*https://github.com/joekenj/VideoStreamApp*" to your local machine. It is important that you have node.js installed!

First we start backend:

```
cd ../backend                 //navigate to backend folder
npm install express --save    //install express framework
node server.js                //start server on localhost port 8081
```

Now we can try to send request to our server. We can try grab information about video with id 4, we do so with running "*http://127.0.0.1:8081/video/4*" in web browser. If everything went smoothly we should see information about the video.

Now to start frontend:

```
npm install                   //to install dependencies in our project
npm start                     //start our project on localhost port 3000
```

Application should open url "*http://localhost:3000/videos*" with loaded list of videos. Now it is possible to work with application.

3.3. Tutorial for application

Application homepage is "*/videos*" from there user is able to navigate to video user wants. Each video in the list has 4 main points (Figure 7):

- Thumbnail or image (imageUrl property of video class) that is as well link to clicked video.

- Name of the video (name property of video class) also link to clicked video.

- Description (description property of video class) that gives brief overview of the video.

- Additional information label that when clicked will call pop up with additional information (Figure 8).

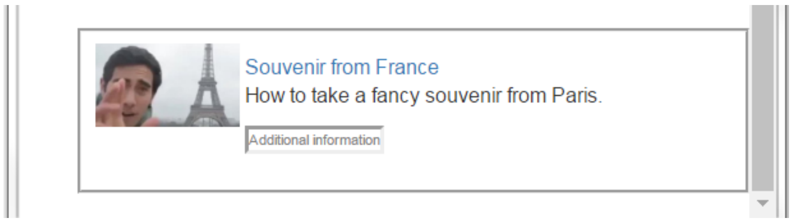


Figure 7 - Video entry

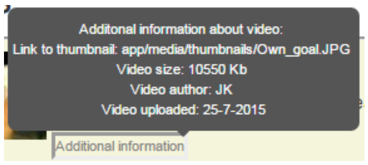

Additonal information about video:
Link to thumbnail: app/media/thumbnails/Own_goal.JPG
Video size: 10550 Kb
Video author: JK
Video uploaded: 25-7-2015

Figure 8 - Additional information for video entry

When user navigates to desired video it will redirect to "*video/:uid*". There user is able to play video with video player and navigate to other videos (Figure 9). When clicking on "*All videos*" user will be redirected to homepage ("*/videos*") and previous and next video will navigate us trough videos. User is provided with data from VideoDetailed object (size, author and date), videoUrl property is within video tag. On this page user can see name, description and thumbnail as well.

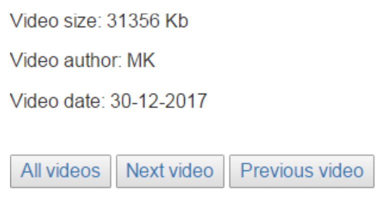

Video size: 31356 Kb
Video author: MK
Video date: 30-12-2017

Figure 8 - Additional information and navigation

4. Literature

This application was developed trough reading several Angular 2 tutorials, those tutorials can be found on following links:

https://angular.io/docs/ts/latest/tutorial/

https://scotch.io/tutorials/routing-angular-2-single-page-apps-with-the-component-router

https://scotch.io/tutorials/angular-2-http-requests-with-observables

https://thinkster.io/tutorials/angular-2-http

For Node.js and Express all information was found on following link:

https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm

And of course heaps of StackOverflow :) .