PROGRAMMING TECHIE

Spring Boot Angular Project – Build a Youtube Clone – Part 1

1 Comments

BY SAI UPADHYAYULA | July 10, 2021

Share Tweet Pin

In this tutorial series, **Spring Boot Angular Project**, you will learn how to build a complete application from the scratch using **Spring Boot** and **Angular**, you will be building a Video Streaming Application like Youtube. Of course, we are not going to build the complete Clone of Youtube, but only a minimal set of functionality, which is common for Video Streaming applications like Youtube.

Source Code

You can find the source code of this project at Github – https://github.com/SaiUpadhyayula/youtube-clone

Video Tutorial

If you are a visual learner like me, you can check out the video tutorial I created

Functional Requirements

We will be implementing the following features in this tutorial series, where we will cover each feature in a different part:

- User can Upload new Videos
- User can Upload Thumbnails for the Videos
- User can View Videos
- User can Like/Dislike a Video
- User can Subscribe to another User, to receive updates about future videos
- User can Login/Logout using Single Sign On
- User can comment on Videos
- User can view the History of Videos he/she watched
- User can view the List of Videos he/she Liked

Technologies Used

So to develop this project, we are going to use the following technologies:

- Spring Boot
- MongoDB
- Angular
- AWS S3 – to store Videos and Thumbnails

Similar Tutorials

If you are interested in tutorials like these, you can find a couple of other Full Stack Project Oriented Tutorials I created on my Youtube Channel –

https://www.youtube.com/playlist?list=PLSVW22jAG8pAGrwFjsUERCu9WSo2-uEMg

If you like to read you can also find the whole tutorial in a text format like this – https://programmingtechie.com/2020/05/14/building-a-reddit-clone-with-spring-boot-and-angular/

Discussing Application Architecture

As we have the **Functional Requirements** ready, now it's time to think about the high-level architecture of the application we are going to build.

We are going to follow a **3 Tier Architecture**, where we have a Client Application ie. the Frontend, we will develop this using Angular Framework, the Backend Application which will be running on a server, will be developed using Spring Boot, and finally the Database, we are going to use MongoDB.

Why MongoDB ?

If you are looking for a super thoughtful reason, then you are going to be disappointed 🙂

There is no special reason for that, if you check my previous tutorials, I already implemented a couple of applications from the scratch using a Relational Database, so I am planning to cover also the NoSQL Databases and show you, how we will work with them in a big project like this.

Storing Media in AWS S3

As this is a video streaming application, we are going to host video and image files (Thumbnails) which will be uploaded by the user, to handle this effectively, we are going to use **AWS S3** to store and retrieve our Media Files (Video and Image).

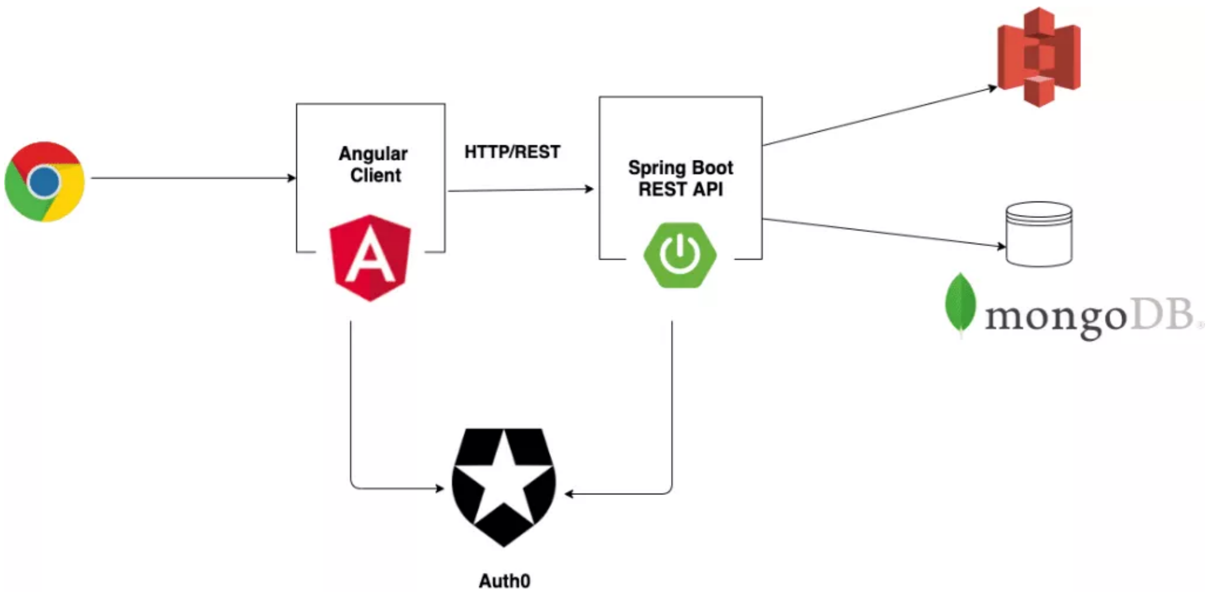
By outsourcing the storage and streaming of media content to AWS, we can simply focus on implementing the main Functional Requirements and Business Logic of the application.

Outsourcing Authentication to Auth0

We are also going to outsource the Authentication in our application, by using an External Authorization Server like Auth0, you can certainly use also other Authorization Server like **AWS Cognito**, **Microsoft Azure AD**, and **Okta** or you can also maintain your own Authorization Server by using a software like **Keycloak**.

If you want to learn how to implement Authentication and Authorization using Spring Boot and Keycloak – have a look at this tutorial series – https://www.youtube.com/playlist?list=PLSVW22jAG8pAXU0th247M7xPCekzeNdrH

You can view the Architecture Diagram of the Project in the below image:



Frontend Architecture

We are going to implement a Component Oriented Architecture in our Angular application, which means, we will try to divide each meaningful element, you will see on the screen, into a different component, in that way, we can re-use the components across different places.

Backend Architecture

If you zoom in to the Spring Boot REST API, we are going to follow a standard 3-Layer architecture also in our backend side.

We are going to expose the REST API and maintain it separately in a **Presentation/Controller Layer**, this layer will only deal with receiving REST calls from the clients, validating whether the requests are valid or not and then delegating the request to the **Service Layer**.

The **Service Layer** will perform the actual business logic of our application, it is the core of our application. This layer will not deal with storing (or) persisting of the data, this will be the responsibility of the **Persistency Layer**

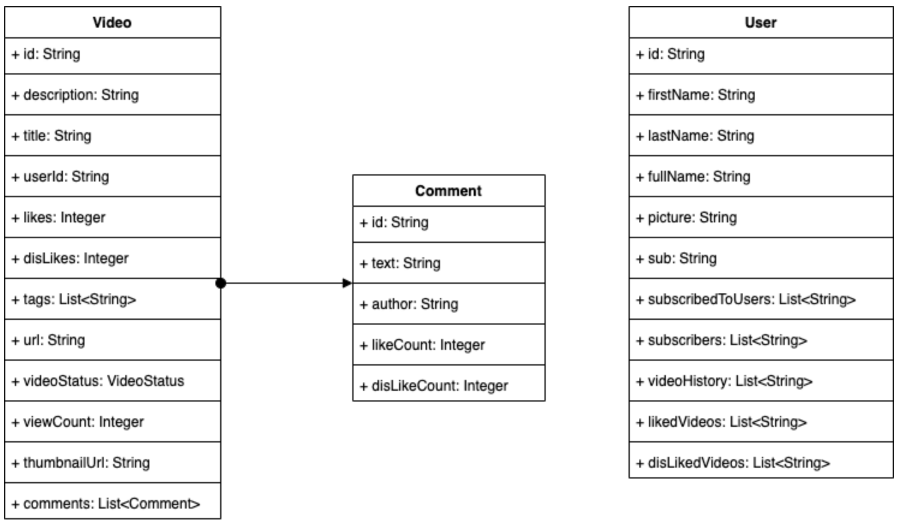
The **Persistence Layer** is responsible to make sure that the objects are stored properly in the database, it will not deal with any kind of business logic.

By following this layered architecture approach, we can keep the code clean and maintainable.

MongoDB Schema

Now let's have a look at the MongoDB schema, we are going to maintain 2 Collections inside our Database:

- Video – We will store the meta data related to videos
- User – We will store the meta data related to users



Backend – Setting up Spring Boot Project

Now let's go ahead and start the development, we will start off by going to https://start.spring.io/.

In there add the following project details:

Project – Maven Project

Group: com.programming.techie

Artifact: youtube-clone

Package-name: com.programming.techie

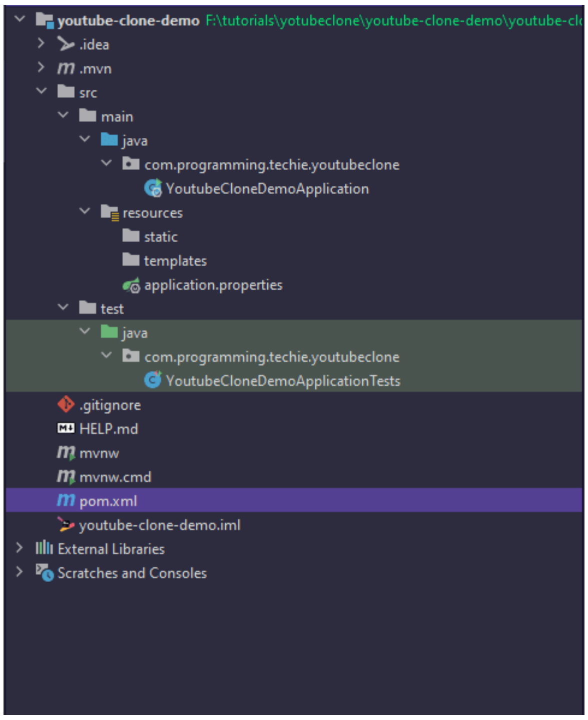
Dependencies:

- Spring Web
- Spring Data MongoDB
- Lombok

Once we start the development, we will add other dependencies to enable AWS Cloud Support and Spring Security Resource Server Support for OAuth2.

Once you added all those dependencies, click on **Generate** button, the project will be downloaded to your machine.

Here is how the package structure of the project we just downloaded, looks like:



```
youtube-clone-demo F:\tutorials\yotubeclone\youtube-clone-demo\youtube-cl
  > .idea
  > .mvn
  v src
    v main
      v java
        v com.programming.techie.youtubeclone
            YoutubeCloneDemoApplication
      v resources
          static
          templates
          application.properties
    v test
      v java
        v com.programming.techie.youtubeclone
            YoutubeCloneDemoApplicationTests
    .gitignore
    HELP.md
    mvnw
    mvnw.cmd
    pom.xml
    youtube-clone-demo.iml
> External Libraries
> Scratches and Consoles
```

Converting to a Multi-Module Maven Project

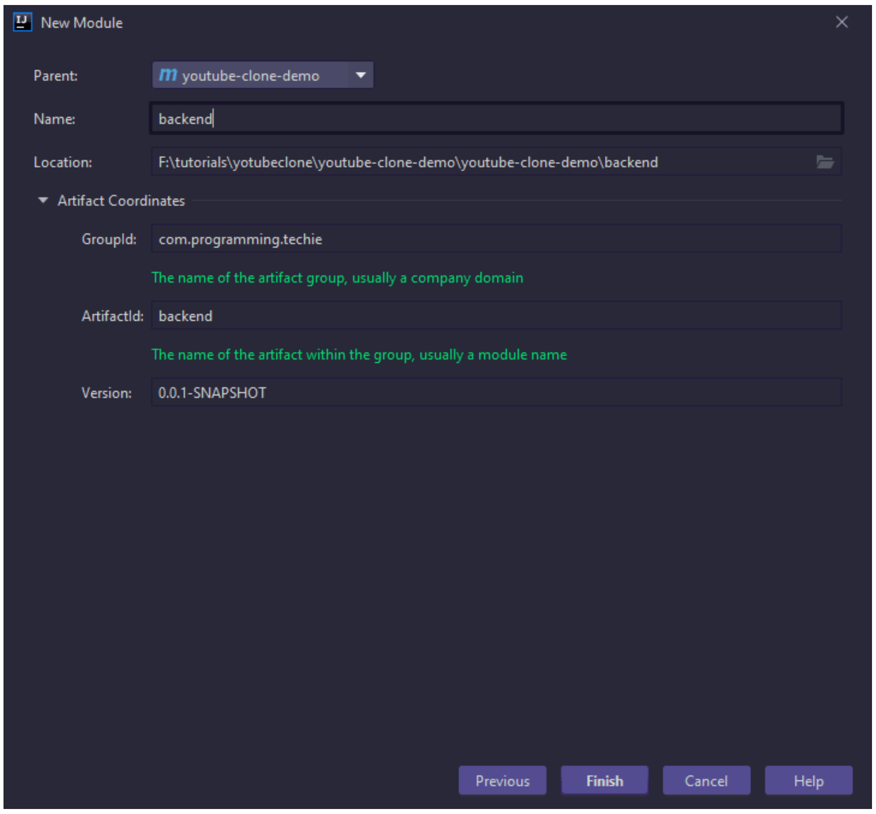
As you can see above, the project generated by Spring Initializr has a single module, and we are going to develop the project using a Multi Module Approach, i.e. we are going to maintain two modules, one for backend and another for frontend code.

Maven provides us the option, to modularize our code into different pieces, which will help us to maintain each area of our code base as independent projects. If you want to know how Maven works and how also the Multi Module System works, check out my video tutorial from my Youtube Channel – https://www.youtube.com/watch?v=JhSBS2OpGdU

So let's go ahead and create these 2 modules in our existing project, for that **Right Click** on the root project (these instructions are for **Intellij IDEA**, if you are following this tutorial using VS Code or Eclipse, you may have to follow different steps to create the Maven Module) and go to **New** -> **Module**

A new pop-up will open with name **"New Module"**, in there make sure you selected the option **Maven** to the left side of the window, and also make sure to point the Module SDK to the JDK location, then click on **Next**.

This will create a default Maven Module, the first module we are going to create is going to be called as **backend**



Parent: youtube-clone-demo
Name: backend
Location: F:\tutorials\yotubeclone\youtube-clone-demo\youtube-clone-demo\backend
GroupId: com.programming.techie
The name of the artifact group, usually a company domain
ArtifactId: backend
The name of the artifact within the group, usually a module name
Version: 0.0.1-SNAPSHOT

Make sure you have similar values filled in, and click on **Finish** to create the module.

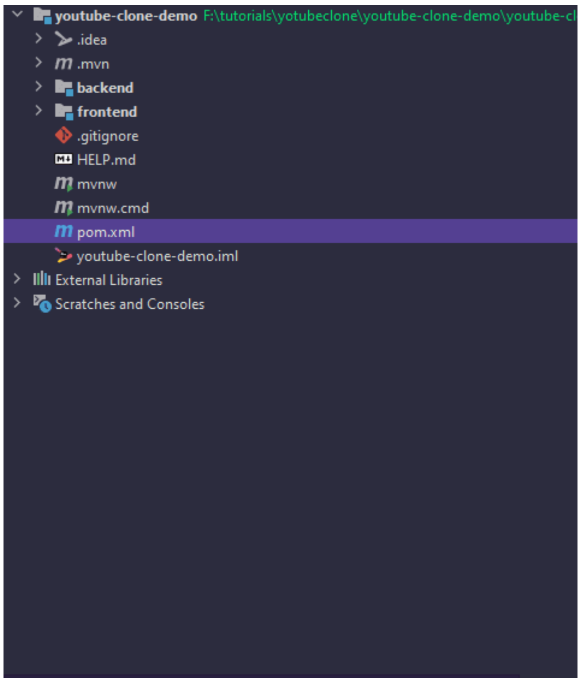
Follow the similar process, to create another module called as **frontend**.

Package Structure

After you created both these modules, this is how the package structure looks like:

```
├─.mvn
│   └── wrapper
│       ├── maven-wrapper.jar
│       └── maven-wrapper.properties
├── backend
├── frontend
├── mvnw
├── mvnw.cmd
├── pom.xml
├── src
│   ├── main
│   └── test
```

We still have the **src/main** and **src/test** packages, generated from the project, these packages should now be part of the **backend** module, so delete the **src** directory inside the **backend** module, which is generated and move the **src** folder into the **backend** module, now the package structure, should look like this:



Update the contents of the **pom.xml** by adding the **backend** and **frontend** projects to the module section, as shown below:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-ins
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.2</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.programming.techie</groupId>
    <artifactId>youtube-clone-demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>youtube-clone-demo</name>

    <packaging>pom</packaging>
    <modules>
        <module>backend</module>
        <module>frontend</module>
    </modules>

    <description>Demo project for Spring Boot</description>

</project>
```

Now update the contents of the **pom.xml** inside the **backend** module as shown below:

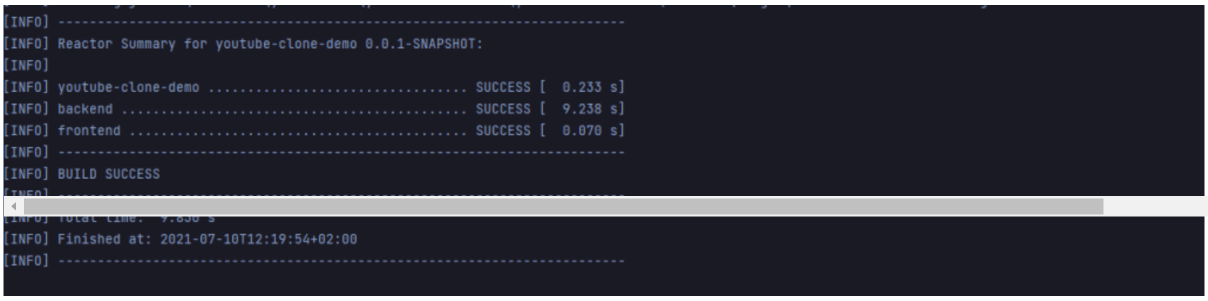
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4
    <parent>
        <artifactId>youtube-clone-demo</artifactId>
        <groupId>com.programming.techie</groupId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>backend</artifactId>

    <properties>
        <maven.compiler.source>16</maven.compiler.source>
        <maven.compiler.target>16</maven.compiler.target>
        <java.version>16</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-mongodb</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

We basically took all the dependencies defined inside the Root Pom.xml file and move them to the Backend module.

Now, verify if everything is working fine or not, by running the below command

```
mvn clean verify
```

This will build the whole project, including the **backend** and **frontend** modules. You should see the below output:



```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for youtube-clone-demo 0.0.1-SNAPSHOT:
[INFO]
[INFO] youtube-clone-demo ................................. SUCCESS [  0.233 s]
[INFO] backend ............................................ SUCCESS [  9.238 s]
[INFO] frontend ........................................... SUCCESS [  0.070 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  [illegible] s
[INFO] Finished at: 2021-07-10T12:19:54+02:00
[INFO] ------------------------------------------------------------------------
```

Frontend – Setting up Angular Project

So in the last section, we created the **frontend** module, but didn't create the Angular project inside the module, so let's do that now.

Before we create the project, make sure you have the latest Angular CLI installed on your machine.

You can install Angular CLI, using the following command:

```
$ npm install -g @angular/cli
```

Now, open the Terminal and make sure you are under the **frontend** directory, and then run the below command to create new Angular Application.

```
$ ng new youtube-clone-ui --directory .
```

Make sure you select the following options:

- Angular Routing – Y
- Stylesheet Format – CSS

After you selected the required options, this will generate the project in the **frontend** directory and install all the dependencies. Once the install is completed, you can run the application using the below command:

```
$ ng serve
```

The above command will start the application locally at http://localhost:4200/, you can also use the **npm** package manager to start the application by using the **npm start** command.

Frontend – Installing Angular Material

We are going to use Angular Material as our styling framework to build the frontend of our application, as Angular Materrial is created and maintained by Google, you will get similar look and feel of Youtube because, the components which are present in Angular Material are the same components which are used also in Youtube.

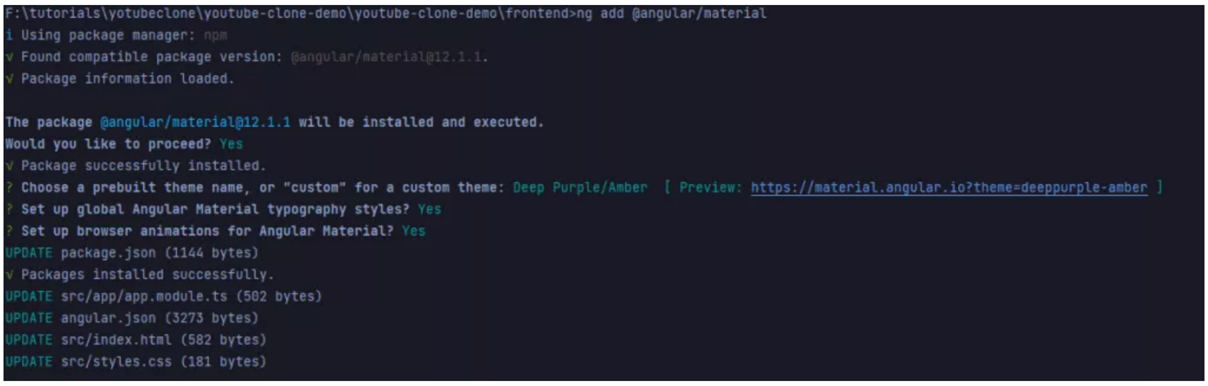
So to install Angular Material, all you need to do is run the following command in your project directory.

```
$ ng add @angular/material
```

Once you type the command, you will be asked for some input, first of all it will ask you whether you want to install the latest angular material package or not, select **Yes**.

Then you will be asked to choose a theme, pick something you like, as part of this tutorial, I picked the Deep Purple and Amber theme.

As the follow-up questions, **Enable** the Angular Material Typography Styles as well as the Browser Animations for Angular Material.
Once you selected all these options, the package will be installed to your machine. The output should look like below:



```
F:\tutorials\yotubeclone\youtube-clone-demo\youtube-clone-demo\frontend>ng add @angular/material
i Using package manager: npm
√ Found compatible package version: @angular/material@12.1.1.
√ Package information loaded.

The package @angular/material@12.1.1 will be installed and executed.
Would you like to proceed? Yes
√ Package successfully installed.
? Choose a prebuilt theme name, or "custom" for a custom theme: Deep Purple/Amber  [ Preview: https://material.angular.io?theme=deeppurple-amber ]
? Set up global Angular Material typography styles? Yes
? Set up browser animations for Angular Material? Yes
UPDATE package.json (1144 bytes)
√ Packages installed successfully.
UPDATE src/app/app.module.ts (502 bytes)
UPDATE angular.json (3273 bytes)
UPDATE src/index.html (582 bytes)
UPDATE src/styles.css (181 bytes)
```

Frontend – Build Angular Project using Maven

We are all ready and setup with the Angular project, but as of now to build the Angular Project, we have to use the Angular CLI. But when deploying the project, it would be easier for us to build the whole project with one command, as we are using Maven as the Build Tool, now let's configure our **frontend** module to build also the Angular project using Maven. In this way, we can bundle the backend and frontend modules into a single JAR.

Fortunately, we have a maven plugin – frontend maven plugin which will do this, this plugin provides a lot of built-in goals that we can use to do different tasks like Downloading and Installing Node and NPM, Build and Test using NPM, run different commands using other build tools like Yarn, Grunt and Gulp.

To enable this plugin, replace the contents of **frontend/pom.xml** with the below contents:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4
    <parent>
        <artifactId>youtube-clone-demo</artifactId>
        <groupId>com.programming.techie</groupId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>frontend</artifactId>

    <properties>
        <maven.compiler.source>16</maven.compiler.source>
        <maven.compiler.target>16</maven.compiler.target>
    </properties>

  <build>
    <resources>
      <resource>
        <directory>dist/META-INF/resources</directory>
      </resource>
    </resources>
    <plugins>
      <plugin>
        <groupId>com.github.eirslett</groupId>
        <artifactId>frontend-maven-plugin</artifactId>
        <version>1.12.0</version>
        <configuration>
          <nodeVersion>v14.17.3</nodeVersion>
          <nodeDownloadRoot>https://nodejs.org/dist/</nodeDownloadRoot>
        </configuration>
        <executions>
          <execution>
            <id>install node and npm</id>
            <goals>
              <goal>install-node-and-npm</goal>
            </goals>
            <phase>generate-resources</phase>
          </execution>
          <execution>
            <id>npm install</id>
            <goals>
              <goal>npm</goal>
            </goals>
            <configuration>
              <arguments>install</arguments>
            </configuration>
          </execution>
          <execution>
            <id>npm build</id>
            <goals>
              <goal>npm</goal>
            </goals>
            <phase>generate-resources</phase>
            <configuration>
              <arguments>run-script build</arguments>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

The above plugin configuration, will first download node and npm to your machine and then will invoke the goals like **npm install** and **npm build** during the Maven Build Lifecycle.

You can run the **mvn clean verify** command to install the project dependencies and build the project.

Once you build the Angular project, the build output will be placed inside the dist directory of the **frontend** module.

After making this change, also make sure to update the **build** section inside the **package.json** file to below:

```
"build": "ng build --prod",
```

Finally, make sure to add the **frontend** module as a dependency to the **backend** module, so that the **frontend** will be built first, and then in the resulting JAR file, which is generated by running the **backend** you can also find the contents of the Angular Project in a single JAR file.

This is how the **backend/pom.xml** file should look like:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4
    <parent>
        <artifactId>youtube-clone-demo</artifactId>
        <groupId>com.programming.techie</groupId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>backend</artifactId>

    <properties>
        <maven.compiler.source>16</maven.compiler.source>
        <maven.compiler.target>16</maven.compiler.target>
        <java.version>16</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-mongodb</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>com.programming.techie</groupId>
            <artifactId>frontend</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Now run the below command

```
mvn clean verify
```

Both the **backend** as well as **frontend** modules should be built successfully without any errors, now you can find the JAR file under the **backend/target** folder.

Run the jar file with the command:

```
$backend/target > java -jar backend-0.0.1-SNAPSHOT.jar
```

You should now see the Login Page from Spring Security, but don't worry, we will fix this later towards the end of the tutorial series, when we completed the development part.

Backend – Configuring MongoDB in our Project

Now let's get started with the backend development, by first configuring the details of MongoDB inside our **backend** module, before that make sure you have **MongoDB** installed on your machine, you can either use a Docker Image or download Community edition of MongoDB from the official website.

You can find the docker image below:

https://hub.docker.com/_/mongo

You can find the download link of the Community Edition of MongoDB below:

https://www.mongodb.com/try/download/community

Once you downloaded and installed MongoDB, we have to update the **application.properties** file inside our **backend** module like below:

```
##################### MongoDB ######################
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=youtube-clone
```

Backend – Create Data Model in Spring Boot Project

Now, it's time to configure the Model Entiteis in our Spring Boot Project, as already shown in the MongoDB Schema section, we are going to create mainly 3 classes – **Video**, **Comment** and **User**

The **Comment** class will be an embedded inside the **Video** class, so in MongoDB terms, we are embedding the Comment document inside the Video Document.

This is how each of the classes looks like, starting with **Video.java**

Video.java

```
package com.programming.techie.youtubeclone.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.data.mongodb.core.mapping.Document;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;

@Data
```

```java
@AllArgsConstructor
@NoArgsConstructor
@Document(value = "Videos")
@Builder
public class Video {
    private String id;
    private String title;
    private String description;
    private String userId;
    private AtomicInteger likes = new AtomicInteger(0);
    private AtomicInteger disLikes = new AtomicInteger(0);
    private List<String> tags;
    private String url;
    private VideoStatus videoStatus;
    private AtomicInteger viewCount = new AtomicInteger(0);
    private String thumbnailUrl;
    private List<Comment> comments = new ArrayList<>();

    public int likeCount() {
        return likes.get();
    }

    public int disLikeCount() {
        return disLikes.get();
    }

    public void increaseViewCount() {
        viewCount.incrementAndGet();
    }

    public void increaseLikeCount() {
        likes.incrementAndGet();
    }

    public void decreaseLikeCount() {
        likes.decrementAndGet();
    }

    public void increaseDisLikeCount() {
        disLikes.incrementAndGet();
    }

    public void decreaseDisLikeCount() {
        disLikes.decrementAndGet();
    }

    public void addComment(Comment comment) {
        comments.add(comment);
    }
}
```

VideoStatus.java

```java
package com.programming.techie.youtubeclone.model;

public enum VideoStatus {
    PUBLIC, PRIVATE, UNLISTED
}
```

Comment.java

```java
package com.programming.techie.youtubeclone.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.data.annotation.Id;

import java.util.concurrent.atomic.AtomicInteger;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Comment {
    @Id
    private String id;
    private String text;
    private String author;
    private AtomicInteger likeCount = new AtomicInteger(0);
    private AtomicInteger disLikeCount = new AtomicInteger(0);

    public int likeCount() {
        return likeCount.get();
    }

    public int disLikeCount() {
        return disLikeCount.get();
    }
}
```

User.java

```
package com.programming.techie.youtubeclone.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.data.mongodb.core.mapping.Document;

import java.util.HashSet;
import java.util.LinkedHashSet;
import java.util.Set;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Document(value = "Users")
public class User {
    private String id;
    private String firstName;
    private String lastName;
    private String fullName;
    private String picture;
    private String emailAddress;
    private String sub;
    private Set<String> subscribedToUsers = new HashSet<>();
    private Set<String> subscribers = new HashSet<>();
    private Set<String> videoHistory = new LinkedHashSet<>();
    private Set<String> likedVideos = new HashSet<>();
    private Set<String> disLikedVideos = new HashSet<>();

    public void addToLikedVideos(String videoId) {
        likedVideos.add(videoId);
    }

    public void removeFromLikedVideos(String videoId) {
        likedVideos.remove(videoId);
    }

    public void addToDisLikedVideo(String videoId) {
        disLikedVideos.add(videoId);
    }

    public void removeFromDisLikedVideo(String videoId) {
        disLikedVideos.remove(videoId);
    }
```

After you added all these dependencies, you can now start the backend application by running the class –

`YoutubeCloneDemoApplication.java`

Conclusion

In **Part 1** of the Spring Boot Angular Project Tutorial Series, we setup the project and discussed about Functional Requirements and App Architecture, from next part onwards, we will start developing the application.

I will see you in the next tutorial, until then Happy Coding 🙂

About the author

Sai Upadhyayula

Mateusz

July 25, 2021 at 10:52 pm

Nice tutorial. I can't wait for second part. When it will be available?

Comments are closed.