Angular2 and Spring Boot: Getting Started

Posted on December 13, 2016 by Jeroen Bruinink Tweet

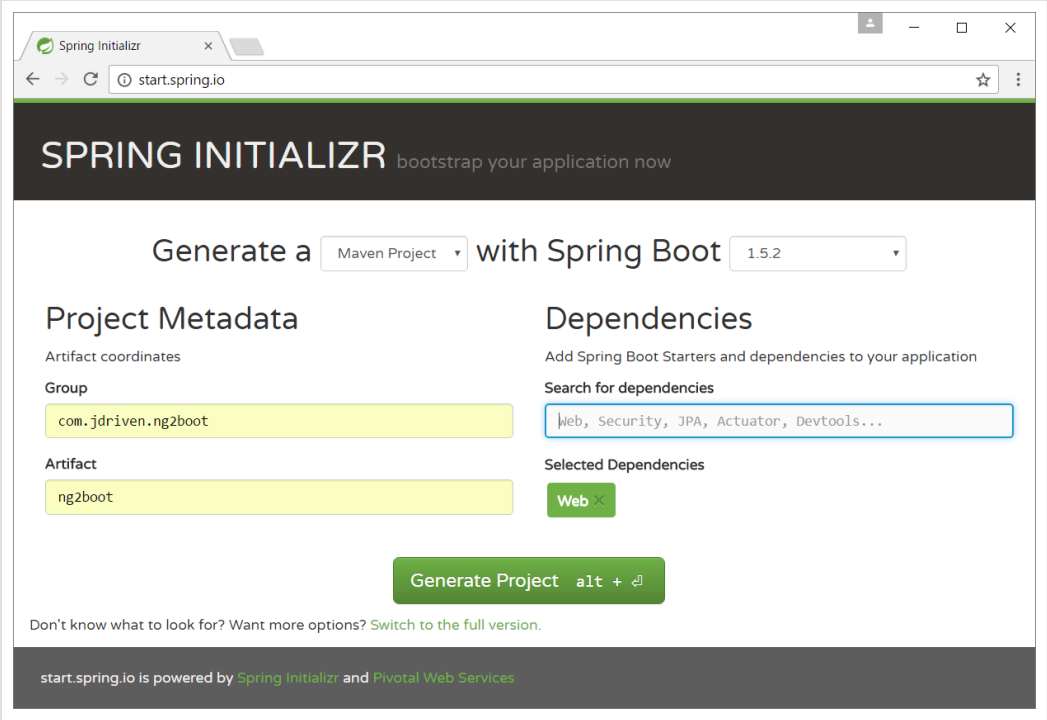
Combining Angular2 and Spring Boot is a great way of getting up and running quickly with a new web application. However, it can be challenging to fit all the different pieces together in the beginning. In this blog post we will create a new project that is easy to build and run across different environments. The goal is to create a minimal, but runnable, application with as little dependencies and setup as possible. You can then start expanding the application however you see fit.

We will create the Angular2 application using angular-cli. This allows us to easily generate a new project with a clear structure. This will also make it easy to add new elements to our Angular2 application. When adding new elements with angular-cli, we will maintain the same structure and wire the new elements together automatically. Angular-cli is an npm module, so it requires Node and npm to install and run. Make sure to install a recent version of both of these applications before continuing with the steps below. We will use Maven as the overall build manager for our application.

The steps should be easy to follow and will not require a lot of work. However, you can get the result directly in this GitHub repository if you want to. There is a seperate commit for every step.

Step 1: Generate a new Spring Boot project

The first step is to generate a new Spring Boot project at https://start.spring.io. I've used the values in the screenshot below. Of course you can use your own group and artifact ids. We need at least the Web plugin to let Spring Boot serve the Angular2 application. If you already know you need more dependencies for your project, feel free to add them.



Click on *Generate Project* to download the zip file with your new Spring Boot project. Extract the zip file somewhere on your computer. Then open a command prompt and go to the newly created project directory.

Step 2: Split the project into seperate modules

We will want to use seperate Maven modules for the frontend and backend. Let's start by creating the correct directory structure and copying the pom file.

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\Jeroen>cd c:\ng2boot

C:\ng2boot>mkdir backend frontend\src\main

C:\ng2boot>move src backend
        1 dir(s) moved.

C:\ng2boot>copy pom.xml backend
        1 file(s) copied.

C:\ng2boot>
```

Edit pom.xml in the top level project directory and make it look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apa
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.jdriven.ng2boot</groupId>
    <artifactId>parent</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>pom</packaging>

    <name>parent</name>
```

```
    <description>The ng2boot parent project</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncodi
        <java.version>1.8</java.version>
    </properties>

    <modules>
        <module>frontend</module>
        <module>backend</module>
    </modules>
</project>
```

Next, edit pom.xml in the backend directory and make it look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apa
    <modelVersion>4.0.0</modelVersion>

    <artifactId>backend</artifactId>

    <name>backend</name>
    <description>The ng2boot backend project</description>

    <parent>
        <groupId>com.jdriven.ng2boot</groupId>
        <artifactId>parent</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </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>
            </plugin>
        </plugins>
    </build>
</project>
```

Finally, copy the pom.xml from the backend directory to the frontend directory and edit it to look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
```

```
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apa
    <modelVersion>4.0.0</modelVersion>

    <artifactId>frontend</artifactId>

    <name>frontend</name>
    <description>The ng2boot frontend project</description>

    <parent>
        <groupId>com.jdriven.ng2boot</groupId>
        <artifactId>parent</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>

    <build>
        <plugins>
        </plugins>
    </build>
</project>
```

Step 3: Add the Angular2 application to the project

If angular-cli is not installed on your system already, use npm to install it now.

```
C:\ng2boot>npm install -g @angular/cli
```

To keep the blog post readable, I'm not showing the output of the command. You can ignore any warnings about optional dependencies. The -g flag tells npm to install it *globally.* This means you can run the command *ng* from the command line anywhere on your computer. Let's use angular-cli to generate our Angular2 application in the frontend\src\main\frontend directory.

```
C:\ng2boot\frontend\src\main>ng new --skip-git --directory frontend ng2boot
installing ng
 create .editorconfig
 create README.md
 create src\app\app.component.css
 create src\app\app.component.html
 create src\app\app.component.spec.ts
 create src\app\app.component.ts
 create src\app\app.module.ts
 create src\assets\.gitkeep
 create src\environments\environment.prod.ts
 create src\environments\environment.ts
 create src\favicon.ico
 create src\index.html
 create src\main.ts
 create src\polyfills.ts
 create src\styles.css
 create src\test.ts
 create src\tsconfig.app.json
 create src\tsconfig.spec.json
 create src\typings.d.ts
 create .angular-cli.json
 create e2e\app.e2e-spec.ts
 create e2e\app.po.ts
 create e2e\tsconfig.e2e.json
 create karma.conf.js
 create package.json
 create protractor.conf.js
 create tsconfig.json
 create tslint.json
Installing packages for tooling via npm.
Installed packages for tooling via npm.
Project 'ng2boot' successfully created.
```

```
34
35 C:\ng2boot\frontend\src\main>
```

We just told angular-cli to generate a *new* project. We let it skip creating a git repository, because we are not in the top level directory of our project. We specified that the output directory should be *frontend* and called the new application *ng2boot.* We chose src\main\frontend to prevent maven from putting the source files, including the node_modules directory, in our jar.

Step 4: Configure Maven to build the Angular2 application

We will use the frontend-maven-plugin to build the Angular 2 application with Maven. First, let's add the plugin to the fronted pom in the build/plugins section.

```
1         <plugins>
2             <plugin>
3                 <groupId>com.github.eirslett</groupId>
4                 <artifactId>frontend-maven-plugin</artifactId>
5                 <version>1.3</version>
6
7                 <configuration>
8                     <nodeVersion>v6.10.1</nodeVersion>o
9                     <npmVersion>4.4.1</npmVersion>
10                    <workingDirectory>src/main/frontend</workingDirectory>
11                </configuration>
12
13                <executions>
14                    <execution>
15                        <id>install node and npm</id>
16                        <goals>
17                            <goal>install-node-and-npm</goal>
18                        </goals>
19                    </execution>
20
21                    <execution>
22                        <id>npm install</id>
23                        <goals>
24                            <goal>npm</goal>
25                        </goals>
26                    </execution>
27
28                    <execution>
29                        <id>npm run build</id>
30                        <goals>
31                            <goal>npm</goal>
32                        </goals>
33
34                        <configuration>
35                            <arguments>run build</arguments>
36                        </configuration>
37                    </execution>
38                </executions>
39            </plugin>
40        </plugins>
```

We specify the node and npm versions and the working directory in the configuration section. We also add three executions. The first execution downloads and installs node and npm to the directories *node* and *node_modules*. Npm will download a lot of packages, so add both directories to the ignore list of your version control system. The second execution runs *npm install* in the working directory to download all npm

dependencies of our project. The third execution runs *npm run build* in the working directory to build the Angular2 application.

By default, angular-cli will write the final Angular2 application in the src\main\frontend\dist directory. We can adhere to the Maven standard directory layout better by configuring angular-cli to write it to the target directory. This will also delete the built application, along with all other build artifacts, when we run *mvn clean*. Edit .angular-cli.json and change the *outDir* in the *apps* section.

```
"apps": [
  {
    "root": "src",
    "outDir": "../../../target/frontend",
    "assets": [
      "assets",
      "favicon.ico"
    ],
    "index": "index.html",
    "main": "main.ts",
    "polyfills": "polyfills.ts",
    "test": "test.ts",
    "tsconfig": "tsconfig.app.json",
    "testTsconfig": "tsconfig.spec.json",
    "prefix": "app",
    "styles": [
      "styles.css"
    ],
    "scripts": [],
    "environmentSource": "environments/environment.ts",
    "environments": {
      "dev": "environments/environment.ts",
      "prod": "environments/environment.prod.ts"
    }
  }
]
```

Step 5: Let Spring Boot serve the Angular2 application

Maven will now build (and clean) the Angular2 application, but it will not end up in our final jar. Spring Boot will serve static content from a number of directories on the Java classpath. So we have to make sure that the Angular2 application ends up in one of those directories. Files on the classpath that are not Java classes, are known as resources. Maven copies all resources (normally found in src\main\resources) into the final jar. Add the packaged Angular2 application to the resources by adding the snippet below to the *build* section in pom.xml. The given targetPath will put it in /static on the classpath and Spring Boot will serve it from there.

```
        <resources>
            <resource>
                <directory>target/frontend</directory>
                <targetPath>static</targetPath>
            </resource>
        </resources>
```

Because the Spring Boot backend is in another module, we will need to add a dependency to the Angular2 application. Edit the backend pom file and add the following to the list of dependencies.

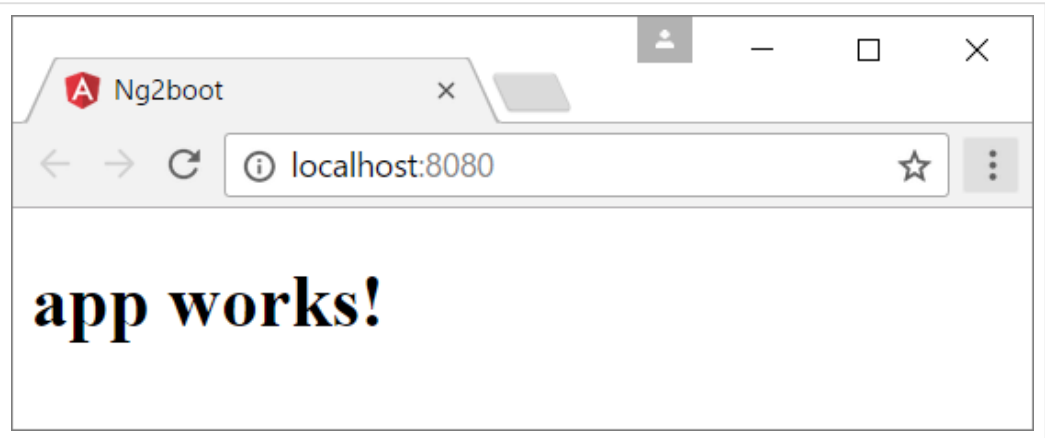
```
        <dependency>
            <groupId>com.jdriven.ng2boot</groupId>
            <artifactId>frontend</artifactId>
            <version>${project.version}</version>
            <scope>runtime</scope>
        </dependency>
```

Step 6: Fire it up!

We are now ready to build and run our application. From the top level directory of our project, run:

```
mvn clean install
cd backend
mvn spring-boot:run
```

Wait for the application to start and then point your browser to http://localhost:8080



If you have configured everything properly, your application should look just like the above image. If it doesn't, have a look at the GitHub repository which has the finished application in it. And of course, don't hesitate to leave a reply if you're having trouble getting the application to build or start.

Step 7: Using the angular-cli development server with your Spring Boot backend

One of many nice features of angular-cli is the development server. It will serve the Angular2 application, just like Spring Boot. However, every time we save a source file, it will automatically rebuild the application and refresh the browser.

There is one problem though. Maven (running our backend) and the development server (running our frontend) are seperate processes listening on seperate ports. This prevents Angular2 from making backend requests because it violates the

Same Origin Policy of our web browser. Web browsers only allow backend requests to the same origin that the web application making the requests was downloaded from.

Thankfully, we can let angular-cli act as a proxy for our Spring Boot backend. The Angular2 application will send backend requests to thedevelopment server, which will forward them to Spring Boot. Now, the Angular2 application can make backend requests to the same origin it came from.

Edit package.json and change the *start* script to add the proxy configuration.

```
"scripts": {
  "ng": "ng",
  "start": "ng serve --proxy-config proxy.conf.json",
  "build": "ng build",
  "test": "ng test",
  "lint": "ng lint",
  "e2e": "ng e2e"
}
```

The *start* script now references proxy.conf.json. Create that file, with the following content:

Content of proxy.conf.json — JavaScript

```
  "/api": {
    "target": "http://localhost:8080",
    "secure": false
  }
}
```

This configuration assumes all backend requests will be made to (sub paths of) /api. You can of course choose any url you like. Now, when you run *npm start* in the frontend\src\main\frontend directory, the development server will run your Angular2 application. It can be reached at http://localhost:4200. Don't forget you have to start your backend seperately. You can do this by running *mvn spring-boot:run* in the backend directory.

I hope you enjoyed this blog post. If you found it useful, or if you have any questions, please let me know by leaving a reply. Happy coding!

This entry was posted in Angular2, Featured, Spring Boot and tagged Angular2, Spring Boot by Jeroen Bruinink. Bookmark the permalink [https://blog.jdriven.com/2016/12/angular2-spring-boot-getting-started/] .

---

155 THOUGHTS ON "ANGULAR2 AND SPRING BOOT: GETTING STARTED"

Evgenij
on **December 29, 2016 at 08:46** said:

Hi Jeroen! First of all, thank you for the great post! It helped me a lot just on the eve of the new year :-) But I had one problem: I have springboot back-end on myip:8080 and front-end on myip:4200, My back-end started on tomcat container with context name /ng2boot and in my angular app I have base href /ng2boot to make it work together, also in proxy.conf.json “target”: “http://myip:8080/ng2boot” and in package.json “start”: “ng serve –proxy-config proxy.conf.json”, I call backend from service by private testDataUrl: string = ‘/backend/test’ but I got 404 because in console there is zone.js:1645 GET myIp:4200/backend/test 404 (Not Found). I dont understand what is wrong with proxying requests, have you any suggestions what I do wrong? Thanks in advance!

> Jeroen Bruinink
> on **December 29, 2016 at 09:29** said:
>
> Hi,
>
> Nice to know my post has been helpful :-)
>
> It looks like there is a wrong path somewhere in your application. Could you please post the entire contents of your proxy.conf.json?
>
> > Evgenij
> > on **December 29, 2016 at 09:46** said:
> >
> > proxy.conf.json
> >
> > {
> > “/api”: {
> > “target”: “http://localhost:8080”,
> > “secure”: false
> > }
> > }
> >
> > Service:
> >
> > @Injectable()
> > export class ApiService {
> > constructor(private http: Http) {
> > }
> > private testDataUrl: string = ‘/ng2boot/backend/test’;
> >
> > getTestData(): Observable {
> > return this.http.get(this.testDataUrl)
> > .map(this.extractData)

.catch(this.handleError);
}
……
}

After the query I get response:

Request URL:http://localhost:4200/ng2boot/backend/test
Request Method:GET
Status Code:404 Not Found
Remote Address:127.0.0.1:4200

For some reason the request goes to :http://localhost:4200/ not :http://localhost:8080/

> Jeroen Bruinink
> on **December 29, 2016 at 09:55** said:
>
> Your proxy configuration currently proxies all requests to http://localhost:4200/api to http://localhost:8080/api. Try changing /api to /ng2boot/backend in proxy.conf.json. This assumes that your endpoint is at /ng2boot/backend, if not you might want to rewrite the requests like so:
>
> proxy: {
> '/ng2boot/backend': {
> target: 'https://localhost:8080',
> pathRewrite: {'^/ng2boot/backend' : '/ng2boot/something-else'}
> }
> }
>
> You can find more information about configuring the proxy of the webpack dev server here: https://webpack.github.io/docs/webpack-dev-server.html#proxy

> Kim
> on **January 6, 2017 at 23:15** said:

Great tutorial, this has helped us so much in our time of need :)

Thank you!

By the way I've noticed that there has been some sort of breaking change in a library on the front end, and to fix it I had to upgrade angular-cli and its dependencies. I think your version on Github will have this issue. You'll see an error about listlazyroutes not being defined when building.

Thanks again,
Kim

> Jeroen Bruinink
> on **January 7, 2017 at 11:46** said:
>
> Hi Kim,
>
> Thank you very much for pointing out the error. I actually ran into the same problem at my client project but never realised it would also affect my example here. I updated the code on Github so it builds again. I rebased master so the commits still correspond to the steps in the blog.
>
> Regards,
> Jeroen

RICARDO ESPERGUE
on **January 26, 2017 at 21:42** said:

This is the solution I was looking for!! Many post I have seen use node js as Backend but here we can use Spring Boot.

> Evgenij
> on **January 30, 2017 at 14:34** said:
>
> Hi Jeroen! In my project uses war-packaging and context path, something like localhost:8080/application/. Please advise the way of integration Angular2 front-end into this back-end! It turns out the confusion in resources due the entry point of context name of tomcat application. In the mode of development I have to fix some points that need to return back again when I need to prepare all tohether for packaginig in war :-(Is there a way to avoid such problems? Thanks…
> By the way, the same project works perfectly when packaged in .jar and has no context name and base href..

Jeroen Bruinink
on **January 30, 2017 at 14:47** said:

Hi,

You should be able to solve this by changing the file proxy.conf.json. Just add your context path to the existing proxy path. Using your example of ‘application’, it should look like this:

{
“/application/api”: {
“target”: “http://localhost:8080”,
“secure”: false
}
}

This configuration assumes your endpoint can be reached at http://localhost:8080/application/api

> Evgenij
> on **January 30, 2017 at 15:26** said:
>
> Many thanks, Man! Everything worked as it should! Good job, Keep up the good work! :-)
>
> > Jeroen Bruinink
> > on **January 30, 2017 at 15:51** said:
> >
> > Glad I could help. Good luck!

Noah
on **January 31, 2017 at 12:38** said:

Hi Jeroen:
Thx for the help. I used to combine this two but failed.
Step by step with your article, now work fine.
But how to update frontend code? I changed app.component.ts’s title variable , but it still show default “app works!”
I mvn clean install and spring boot again, but still can’t get any update.
Help!

Jeroen Bruinink
on **January 31, 2017 at 12:54** said:

Hi Noah,

That is really weird. If you are using the dev server by connecting to localhost:4200, you don't even need to recompile or restart anything. It just updates as soon as you save the file.

What are you using to edit the code? Are you sure you are editing the file you are running? Are you sure the changes you made were saved to disk?

> Noah
> on **January 31, 2017 at 13:11** said:
>
> Hi Jeroen,
> Thx for replying so fast. localhost:4200 now works fine.
>
> But I am wondering should localhost:8080 keep up-to-date ? Or just test my code with localhost:4200 and make mvn install when publish to localhost:8080. Which port is for the user?
>
> Sorry, just a beginner hope the questions not too dump :O
>
> > Jeroen Bruinink
> > on **January 31, 2017 at 14:10** said:
> >
> > I normally use the development server at port 4200 because it automatically reloads the changes I make to the frontend code.

Ian Rae
on **January 31, 2017 at 14:18** said:

Very useful. Building angular with maven is fantastic. Thanks very much for this tutorial.

Shuaib
on **February 6, 2017 at 16:11** said:

Hey man

I really want to be able to package the backend and frontend into a single jar, and also run them with the same command.

Is this possible? Currently it's only possible to start the backend and the frontend seperately.

> Jeroen Bruinink
> on **February 6, 2017 at 17:08** said:
>
> The frontend is actually packaged into the backend jar. You can simply start the backend and go to http://localhost:8080. The development server at port 4200 is optional. It just makes it easier to develop the frontend, because all changes are reloaded automatically.
>
> > Shuaib
> > on **February 8, 2017 at 10:25** said:
> >
> > Thanks for the reply
> >
> > I followed your instructions, and packaged the app like you said, but when I run localhost:8080, I only get the Json from the backend.
> >
> > What am I doing wrong?
> >
> > > Jeroen Bruinink
> > > on **February 8, 2017 at 12:07** said:
> > >
> > > What commands are you executing and what output do you see?
> > >
> > > Are you using the code from my Github repository or your own?
> > >
> > > > Shuaib
> > > > on **February 8, 2017 at 15:32** said:
> > > >
> > > > Hi
> > > >
> > > > I was using my own code. Turns out it was application specific problems that stopped it from working. It's working now, thanks!
> > > >
> > > > One other question. I see that the frontend folder is quite dense. Is there a way to make it simpler or is the

structure necesarry?

> Jeroen Bruinink
> on **February 8, 2017 at 15:58** said:
>
> I did not change the structure of the frontend directory after generating it with angular-cli. This way, when there's a new version of angular-cli, I just generate a new directory, copy it over my current directory and use git to revert some of the changes and I'm done.
>
> I'm sure it's possible to change the structure, you may need to adjust a few paths in some configuration files.

> Nitin
> on **February 8, 2017 at 22:04** said:

Fantastic… I was struggling to do this.

> Gaurav
> on **February 8, 2017 at 23:27** said:

can you please provide some info on what commands to run when the development is done and one is ready to get all the changes in development mode to prod mode, how will the development changes be copied to maven generated files.?

when i again did mvn clean install, it still shows the initial page only, and when i am running via npm start on the dev server all my dev changes are there, so wanted to know what to do to get those changes on mvn install.

> Jeroen Bruinink
> on **February 9, 2017 at 11:48** said:

My first guess would be that you ran mvn clean install in the backend directory instead of in the top level directory of the project.

> Gaurav
> on **February 10, 2017 at 06:12** said:
>
> that was it…!! excellent post by the way.. helps save a ton of time while figuring out where to get started with angular and java… this saved a great deal of time.. i was up and running in 20 mins with this tutorial..!! apologies for the late complements! thank you!
>
> > Jeroen Bruinink
> > on **February 10, 2017 at 09:21** said:
> >
> > Glad I could help. It's very rewarding to read that my blog post is helping people getting stuff done!

> Shuaib
> on **February 9, 2017 at 11:22** said:
>
> Hi again.
>
> I am trying to package this into a war file instead of a jar file. However, when I run the war on a Tomcat server. it can't find any js or css files.
>
> Do you have any idea what I would need to change in order to make it see these files?
>
> > Jeroen Bruinink
> > on **February 9, 2017 at 11:55** said:
> >
> > I have not tried building this project as a war file. So I don't know what you would need to change.
> >
> > If I can find the time, I will try to add it to the blog later.
> >
> > > Gaurav
> > > on **February 10, 2017 at 06:19** said:

would love to know how you'd do that.. I am sure a lot of people would be keen on knowing how to get this done with a .war file. Is there a way to get notified when you update the blog?

> Jeroen Bruinink
> on **February 10, 2017 at 09:43** said:
>
> There is an RSS feed here: http://blog.jdriven.com/feed/ or you can follow us on Twitter: https://twitter.com/jdriven_nl

> Daniel Lesko
> on **April 27, 2017 at 16:39** said:
>
> I figured out the solution to deploy to war for this particular project setup. Please see:
>
> http://stackoverflow.com/questions/43602712/angular-2-and-spring-boot-deploy-to-war

> cgeo
> on **February 11, 2017 at 00:02** said:
>
> Why would you convert the angular-cli project into a maven project. Just use a normal non-maven frontend project and use maven-frontend-plugin in the backend project to build the frontend directory and copy files to your static folder.
> Why mix up a .js project with a maven project?

> Jeroen Bruinink
> on **February 13, 2017 at 07:41** said:
>
> This way, I don't have to rebuild the frontend every time I make a change to the backend.

vovkor
on **February 11, 2017 at 18:22** said:

Hello Jeroen.
Very good job!!!
Can you write an article “How create CRUD using Spring boot and Angular2” ? It will be very helpful post. Please)))

Anisha
on **February 13, 2017 at 21:51** said:

Thank you for this, it is incredibly helpful for someone who is just getting started with AngularJS (like me).

I did have a question though. How do I add Maven dependencies to the frontend? I want to be able to reference some third-party Java classes from the app.component.ts. I added the dependency and corresponding repository location to the frontend’s pom. Doing a mvn clean install I can see that the dependency is being downloaded. But where is it stored? In my local m2? How do I reference its classes from, say, app.component.ts?

Thanks again. I hunted around for 1.5 days and this post was the first helpful resource I found.

> Jeroen Bruinink
> on **February 17, 2017 at 08:29** said:
>
> Please note that app.component.ts is a TypeScript file, so it won’t be possible to use Java classes from there.
>
> What are you trying to accomplish? Maybe I can point you in the right direction.

Todd Fielder
on **February 16, 2017 at 23:53** said:

Hi, great post…hoping to get this to work, but I am getting the following error when trying to build the frontend…any ideas?
Thanks in advance…

[INFO] Reactor Summary:
[INFO]
[INFO] parent ……. SUCCESS [0.231 s]
[INFO] frontend ……. FAILURE [0.821 s]
[INFO] backend …….. SKIPPED
[INFO] ————————————————————————
[INFO] BUILD FAILURE
[INFO] ————————————————————————
[INFO] Total time: 1.537 s
[INFO] Finished at: 2017-02-16T15:51:15-07:00
[INFO] Final Memory: 16M/219M
[INFO] ————————————————————————
[ERROR] Failed to execute goal com.github.eirslett:frontend-maven-plugin:1.3:install-node-and-npm (install node and npm) on project frontend: Could not download Node.js from: https://nodejs.org/dist/v6.9.1/win-x64/node.exe: Could not download https://nodejs.org/dist/v6.9.1/win-x64/node.exe: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target -> [Help 1]
[ERROR]

> Todd Fielder
> on **February 16, 2017 at 23:58** said:
>
> by the way, I'm using eclipse as my build tool
>
> > Jeroen Bruinink
> > on **February 17, 2017 at 08:24** said:
> >
> > It looks like the keystoree of the JDK running Maven is missing a certificate. It should "just work" with a recent version of the JDK with the default keystore. You can add certificates to your keystore using the keytool: http://docs.oracle.com/javase/7/docs/technotes/tools/solaris/keytool.html

> ruman2693
> on **February 17, 2017 at 04:26** said:
>
> Hey, I am having trouble!!!
> I tried your solution with simple spring boot application it works fine.
> In fact this is the solution I wanted.
> However, I started adding REST controller with some rest-endpoints, the fronted stopped working. I am getting error page on localhost:8080. and when I remove the

rest controller part, it again works fine.
I think dispatcher is unable to resolve the path in case we add rest controller.
A quick help is appreciated! Thanks in advance…

> Jeroen Bruinink
> on **February 17, 2017 at 08:27** said:
>
> Could you post the errors or exceptions you're seeing?
>
> > ruman2693
> > on **February 17, 2017 at 12:24** said:
> >
> > I am getting this, when I ping on 'localhost:8080/' after adding rest controller, which was previously working perfectly on the same address.
> >
> > Whitelabel Error Page
> >
> > This application has no explicit mapping for /error, so you are seeing this as a fallback.
> >
> > Fri Feb 17 16:25:57 IST 2017
> > There was an unexpected error (type=Not Found, status=404).
> > No message available
> >
> > > Jeroen Bruinink
> > > on **February 20, 2017 at 12:38** said:
> > >
> > > Could be anything really, is your code online somewhere (like Github?)

> Todd Fielder
> on **February 17, 2017 at 15:25** said:

Hi Jeroen,
This is really helpful…you have saved many ppl untold number of hours…

I am struggling with the last bit about setting up the proxy. When I go to http://localhost:4200/, the app live updates as it should, but http://localhost:8080/ does not…It should, shouldn't it? That's the point of the proxy?

I didn't modify the path at all, so maybe the /api is what is throwing me off…I tried changing the contextPath=/api, but then angular didn't load at all…and tried setting the proxy.conf.json path to / and also no luck

Any insight into what behavior I should see and what might be going on?

thanks again
-Todd

> Jeroen Bruinink
> on **February 17, 2017 at 15:56** said:
>
> Hi Todd,
>
> The behaviour you describe seems correct to me. The point of the proxy is to serve the backend(running at port 8080) from the development server(running at port 4200). This way, all content is served from the development server(port 4200) and your browser will not complain about cross origin requests. The backend server does not live update, so changes to the backend require a restart.
>
> > Todd Fielder
> > on **February 17, 2017 at 21:40** said:
> >
> > I see, I think I had expected port 8080 to be my entry into the app, but for development, you're saying it's 4200, and then for prod it is 8080? makes sense (I think)
> > thanks again

Evgeny
on **February 18, 2017 at 15:31** said:

Many thanks!!!

Aurélien
on **February 21, 2017 at 00:00** said:

Hello,

Thanks for your tutorial, however I have a problem, I get to do routing on localhost: 4200 however I have the impression that the back take over the localhost 8080 and I arrive on a 404 error.

> Jeroen Bruinink
> on **February 21, 2017 at 07:27** said:
>
> Hi,
>
> Are you using html5 routing? Your app will then request urls that Spring Boot doesn't know about, so it will result in a 404. My colleague wrote about how to support html5 routing in his blog post about integrating an angular2 app in a Gradle build over here . Have a look at the section "Support Angular HTML5 mode".
>
> > Aurélien
> > on **March 7, 2017 at 10:24** said:
> >
> > Thank you very much

> Evgeny
> on **February 28, 2017 at 00:17** said:

Hi, Jeroen!
Thanks a lot for your post.
Everything is great but i've got one issue about http service.
When I click the button first time, response is undefined, but then everything works fine.
My code:
Raised button

```
import { Component } from '@angular/core';
import { TestService } from './test.service'
import { Resp } from './resp'
@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css'],
providers: [ TestService ]
})

export class AppComponent {
resp: Resp;
```

```
constructor(private testService: TestService){}

onClick(): void {
this.testService.getTest().then(r => this.resp = r);
console.log(this.resp);
}

}

import { Injectable } from '@angular/core';
import { Http, Response } from '@angular/http';
import { Observable } from 'rxjs/Observable';
import { Resp } from './resp'
import 'rxjs/add/operator/toPromise';

@Injectable()
export class TestService {
private url = '/api/test';

constructor(private http:Http) { }

getTest() : Promise {
return this.http.get(this.url)
.toPromise()
.then(response => response.json() as Resp)
.catch(this.handleError);
}

private handleError(error: any): Promise {
console.error('An error occurred', error); // for demo purposes only
return Promise.reject(error.message || error);
}
}
```

Thanks in advance.

> Evgeny
> on **February 28, 2017 at 22:05** said:
>
> I found the problem.

kishore
on **March 3, 2017 at 19:00** said:

Hi, Jeroen!
Great tutorial, this has helped us so much …!

We combined both forntend and banckend as per your instructions .
if we run the spring boot application. We are able access spring rest api using postman .I am not able to access http://localhost:8080/.
I am expecting to see my agular app.
nothing is show up.

angular-cli.json


```
{
“project”: {
“version”: “1.0.0-beta.18”,
“name”: “frontend”
},
“apps”: [
{
“root”: “src”,
“outDir”: “../../../target/frontend”,
“assets”: [
“assets”,
“favicon.ico”
],
“index”: “index.html”,
“main”: “main.ts”,
“test”: “test.ts”,
“tsconfig”: “tsconfig.json”,
“prefix”: “app”,
“mobile”: false,
“styles”: [
“styles.css”
],
“scripts”: [],
“environments”: {
“source”: “environments/environment.ts”,
“dev”: “environments/environment.ts”,
“prod”: “environments/environment.prod.ts”
}
}
],
“addons”: [],
“packages”: [],
“e2e”: {
“protractor”: {
“config”: “./protractor.conf.js”
}
},
“test”: {
“karma”: {
“config”: “./karma.conf.js”
}
```

},
“defaults”: {
“styleExt”: “css”,
“prefixInterfaces”: false,
“inline”: {
“style”: false,
“template”: false
},
“spec”: {
“class”: false,
“component”: true,
“directive”: true,
“module”: false,
“pipe”: true,
“service”: true
}
}
}
package.json

“scripts”: {
“ng”: “ng”,
“start”: “ng serve –proxy-config proxy.conf.json”,
“lint”: “tslint \”src/**/*.ts\””,
“test”: “ng test”,
“pree2e”: “webdriver-manager update”,
“e2e”: “protractor”,
“build”: “ng build”
}

proxy.conf.json

{
“/api”: {
“target”: “http://localhost:8080”,
“secure”: false
}
}

Please advise what might be the reason.

Many thanks,
Kishore.

Fatality
on **March 5, 2017 at 00:54** said:

hey many thanks for the article.

Could you please help me also to add security. I want to add security like oath2 and also add more pages to have more options to see for a starter person? :) I know that takes time and I have to dig more for those matters but if you would I think your article would really rock :)

> Jeroen Bruinink
> on **March 8, 2017 at 15:30** said:
>
> Adding security would be a nice follow up to this blog post. If I ever find the time, I think that will be my next subject.

Phil
on **March 6, 2017 at 17:59** said:

Hello and many thanks for the article. Unfortunately I cannot get the app to run (at the Fire it up! step). Spring serves a static “Loading…” screen from index.html, but no JS-code is loaded later. The only things I did differently when following the tutorial is changing the names of the project and versions of used modules. Also my Angular-CLI version is probably different (latest). Where might be the error?

> Jeroen Bruinink
> on **March 8, 2017 at 15:28** said:
>
> Are there any errors in the javascript console of your browser?

Anil Jagtap
on **March 8, 2017 at 10:31** said:

Hi,
My back end app implemented in spring boot with spring security. And front end app in angular 2(angular-cli). I’m deploying parent app to tomcat server. I’m unable to render on localhost:4200/dashboard after login(or any other page).

My proxy.conf.json code:
{
“/abc/api”: {

“target”: “http://localhost:8080”,
“secure”: false
}
}

Where I’m doing wrong? please suggest me.

> Jeroen Bruinink
> on **March 8, 2017 at 15:25** said:
>
> Hi Anil,
>
> There was a comment earlier from someone who was trying to run it on Tomcat. Here is what I replied then:
>
> *Just add your context path to the existing proxy path. Using your example of ‘application’, it should look like this:*
>
> ```
> {
> “/application/api”: {
> “target”: “http://localhost:8080”,
> “secure”: false
> }
> }
> ```

Anil Jagtap
on **March 8, 2017 at 15:10** said:

Front end build is not running…

> Jeroen Bruinink
> on **March 8, 2017 at 15:19** said:
>
> Hi Anil,
>
> The build still seems to be working properly at Travis CI. Please compare your source code with the Github repository.

Babe
on **March 9, 2017 at 06:36** said:

Hi, Jeroen!
Thanks a lot for your post.

Step 6: Fire it up! error T_T

[INFO] Scanning for projects…
[ERROR] [ERROR] Some problems were encountered while processing the POMs:
[FATAL] Non-resolvable parent POM for com.jdriven.ng2boot:frontend:0.0.1-SNAPSHO
T: Failure to find com.jdriven.ng2boot:parent:pom:0.0.1-SNAPSHOT in http://repo1
.maven.org/maven2 was cached in the local repository, resolution will not be rea
ttempted until the update interval of central has elapsed or updates are forced
and 'parent.relativePath' points at wrong local POM @ line 11, column 10
[FATAL] Non-resolvable parent POM for com.jdriven.ng2boot:backend:0.0.1-SNAPSHOT
: Failure to find com.jdriven.ng2boot:parent:pom:0.0.1-SNAPSHOT in http://repo1.
maven.org/maven2 was cached in the local repository, resolution will not be reat
tempted until the update interval of central has elapsed or updates are forced a
nd 'parent.relativePath' points at wrong local POM @ line 11, column 10
@
[ERROR] The build could not read 2 projects -> [Help 1]
[ERROR]
[ERROR] The project com.jdriven.ng2boot:frontend:0.0.1-SNAPSHOT
(D:\SourceCode
\SpringProject\ng2boot\frontend\pom.xml) has 1 error
[ERROR] Non-resolvable parent POM for com.jdriven.ng2boot:frontend:0.0.1-SNA
PSHOT: Failure to find com.jdriven.ng2boot:parent:pom:0.0.1-SNAPSHOT in http://r
epo1.maven.org/maven2 was cached in the local repository, resolution will not be
reattempted until the update interval of central has elapsed or updates are for
ced and 'parent.relativePath' points at wrong local POM @ line 11, column 10 ->
[Help 2]
[ERROR]
[ERROR] The project com.jdriven.ng2boot:backend:0.0.1-SNAPSHOT
(D:\SourceCode\
SpringProject\ng2boot\backend\pom.xml) has 1 error
[ERROR] Non-resolvable parent POM for com.jdriven.ng2boot:backend:0.0.1-SNAP
SHOT: Failure to find com.jdriven.ng2boot:parent:pom:0.0.1-SNAPSHOT in http://re
po1.maven.org/maven2 was cached in the local repository, resolution will not be
reattempted until the update interval of central has elapsed or updates are forc
ed and 'parent.relativePath' points at wrong local POM @ line 11, column 10 -> [
Help 2]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e swit
ch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]

[ERROR] For more information about the errors and possible solutions, please rea
d the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/ProjectBuildin
gException
[ERROR] [Help 2]
http://cwiki.apache.org/confluence/display/MAVEN/UnresolvableMo
delException

Babe
on **March 9, 2017 at 06:57** said:

Ok is Pass !!!

helene
on **May 8, 2017 at 09:46** said:

Sorry i am having same issue you pasted above. How did you solve the parent pom errors?

helene
on **May 8, 2017 at 09:59** said:

I fixed it myself looking at the POMs on github repo. Perhaps a correction precising need to be indicated in the “frontend” Pom not the main one, it was unclear reading

Jeroen Bruinink
on **May 8, 2017 at 10:02** said:

Make sure that you are running “maven install” in the top level directory

Helene
on **May 8, 2017 at 18:10** said:

i did. Issue was that code was in the wrong pom, should be frontend pom not main pom (block)

Aditya
on **March 10, 2017 at 07:41** said:

Hi Jeroen,
Thanks for the awesome article !!

I have 2 doubts now –
1. In My system, Node and NPM are already installed. I don't want them to install again but to use the installed versions of Node and NPM. How will I specify this in my frontend-maven-plugin ?
2. I am also using Thymeleaf to server the HTML contents. In this case how will I specify the path for my templateUrl in the Angular Component ?

> Jeroen Bruinink
> on **March 10, 2017 at 09:49** said:
>
> Hi Aditya,
>
> 1: That is a feature that has been requested for the frontend maven plugin, but has not been implemented. I would recommend just letting the frontend maven plugin download it. This way, there will be no problems caused by different versions of node or npm if you share your project with other developers.
>
> 2: I did a quick search for using dynamically generated html files in an Angular2 app. I found this example, but it looks really complicated and I wouldn't recommend it.

C Magro
on **March 11, 2017 at 01:19** said:

Hi, fantastic post, never used angular and it worked immediately.

The only issue I have is that I am used to running my Spring Boot application during development from the IDE. After this blog however; this works well if I 'mvn clean package' and run however if I just 'mvn clean' and try to run the main spring boot class, it will throw a ClassNotFound exception.

I understand that with clean I have removed everything but I also expect that the backend will at least rebuild (as is normal operation)… most likely, being my first ever angular experience, I do not understand something.

Well done on the blog.

> Jeroen Bruinink
> on **March 11, 2017 at 09:42** said:
>
> Hi,
>
> Which IDE are you using and which version? I can run this project from IntelliJ just fine.
>
> > 
> > C Magro
> > on **March 12, 2017 at 14:30** said:
> >
> > Intellij 2016.3.5 I noticed since the post that even when I run a single unit test from Intellij it throws a class not found, empty test suite error… It could be intellij related
> >
> > > 
> > > C Magro
> > > on **March 12, 2017 at 17:43** said:
> > >
> > > Resolved… IntelliJ was being funny. I had a word with him (assuming its a he) and he agreed to invalidate the cache and restart! We are now friends!
> > >
> > > > Jeroen Bruinink
> > > > on **March 12, 2017 at 17:46** said:
> > > >
> > > > Ah yes, the dreaded IntelliJ cache. Glad tot know it's working now.

Pingback: 03.13.2017 – 03.17.2017 – gvr4wd

Pingback: O strukturze projektu słów kilka – Devs&Deadlines

> GUN
> on **March 21, 2017 at 20:08** said:
>
> [INFO] Using 'UTF-8' encoding to copy filtered resources.
> [INFO] Copying 1 resource

[INFO] Copying 0 resource
[INFO]
[INFO] — maven-compiler-plugin:3.1:compile (default-compile) @ backend —
[INFO] Changes detected – recompiling the module!
[INFO] Compiling 1 source file to D:\Formation Angular 2\ng2boot\backend\target\c
lasses
[INFO] ————————————————————————
[INFO] Reactor Summary:
[INFO]
[INFO] parent …….. SUCCESS [0.655s]
[INFO] frontend …… SUCCESS [40.812s]
[INFO] backend …….. FAILURE [0.914s]
[INFO] ————————————————————————
[INFO] BUILD FAILURE
[INFO] ————————————————————————
[INFO] Total time: 43.022s
[INFO] Finished at: Tue Mar 21 20:08:18 WET 2017
[INFO] Final Memory: 17M/491M
[INFO] ————————————————————————
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-
plugin:3.1
:compile (default-compile) on project backend: Fatal error compiling: invalid tar
get release: 1.8 -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switc
h.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read
the following articles:
[ERROR] [Help 1]
http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionEx
ception
[ERROR]
[ERROR] After correcting the problems, you can resume the build with the command
???

> StJimmy
> on **May 1, 2017 at 13:37** said:
>
> Hi Jeroen,
>
> I got the same error, have you an idea where it comes from ?
>
> > StJimmy
> > on **May 1, 2017 at 15:10** said:
> >
> > It was just my JAVA_HOME variable which was not updated… mvn clean install does work perfectly now :)

Leonardo Oliveira
on **March 21, 2017 at 20:45** said:

When I push the app in git server and I pull it in another machine, if I execute the command ‘mvn clean install’ I receive an error as following:

Cannot find module ‘acorn’
[ERROR] Error: Cannot find module ‘acorn’

I’ve tried a lot of things like delete node modules and executing ‘npm install –save again’, save depedencies might be instaled globally ?

Jim
on **March 23, 2017 at 00:33** said:

When I type ‘ng new –skip-git –directory frontend ng2boot’ I get something different to what you show.

I get what looks like a cmd-line editor open up (like vi but with different cmds.) At the bottom of the screen I get :

–**-Mg: ng2boot (fundamental)—-Bot

I have no idea why this is. I installed Node.js with npm and followed the instructions to the tee… until I get the above.

Any help would be much appreciated.

> Jim
> on **March 23, 2017 at 15:30** said:
>
> Just an update…
> It appears the command: ‘npm install -g angular-cli’, did not succeed.
>
> As part of the process, it appears that a link is supposed to be generated to ‘ng’ under the node-modules/angular-cli/bin directory. See snippet:
>
> C:\ng2boot>npm install -g angular-cli
> C:\Users\Jeroen\AppData\Roaming\npm\ng ->
> C:\Users\Jeroen\AppData\Roaming\npm\node_modules\angular-cli\bin\ng

C:\Users\Jeroen\AppData\Roaming\npm
`– angular-cli@1.0.0-beta.22-1

So I think what has happened is that when I run ng … it is running some editor I have installed on my machine by default.

could someone confirm, please: how many commands am I supposed to execute in the above snippet?

> Jeroen Bruinink
> on **March 23, 2017 at 21:50** said:
>
> Hi Jim,
>
> *could someone confirm, please: how many commands am I supposed to execute in the above snippet?*
>
> It is just one command: `npm install -g angular-cli`
>
> Are you doing this on Windows or Linux? On windows you can check which executable will run using the “where” command. So if you run `where ng` it shows the full path of the executable ng. On Linux you can do the same with the “which” command, so `which ng`
>
> > Jim
> > on **March 24, 2017 at 00:19** said:
> >
> > Linux.
> >
> > I ran ‘which’ and it shows a different directory path to the one in the instructions.
> >
> > So it seems that first cmd is not completing as expected.
> >
> > > Jeroen Bruinink
> > > on **March 24, 2017 at 06:34** said:
> > >
> > > Hi Jim,
> > >
> > > I installed Node.js on my Linux (Ubuntu) server by following these instructions. I ran these commands:
> > > ```
> > > curl -sL https://deb.nodesource.com/setup_6.x
> > > | sudo -E bash -
> > > sudo apt-get install -y nodejs
> > > ```
> > >
> > > When I tried to install angular-cli, I noticed there has been a change since I wrote my blog, I’ll update it when I have more time. I ended up installing it with this

command:
`sudo npm install -g @angular/cli`

The command `which node npm ng` gives this output:
`/usr/bin/node`
`/usr/bin/npm`
`/usr/bin/ng`

`/usr/bin/ng` is a symbolic link.
Running `ls -l /usr/bin/ng` gives this output:
`lrwxrwxrwx 1 root root 39 Mar 24 06:24 /usr/bin/ng -> ../lib/node_modules/@angular/cli/bin/ng`

If you already have an executable named ng on your `$PATH` you could also run ng by going around the symbolic link in /usr/bin and run `/usr/lib/node_modules/@angular/cli/bin/ng` instead.

Jim
on **March 24, 2017 at 18:51** said:

Right…

Hurdle cleared.

Thanks for your attention and assistance, Jeroen. I followed the instructions at then end of that link and I have an ng executable which doesn't open an editor.

I have another query but its not related so will start a new thread.

mauricio
on **March 23, 2017 at 12:50** said:

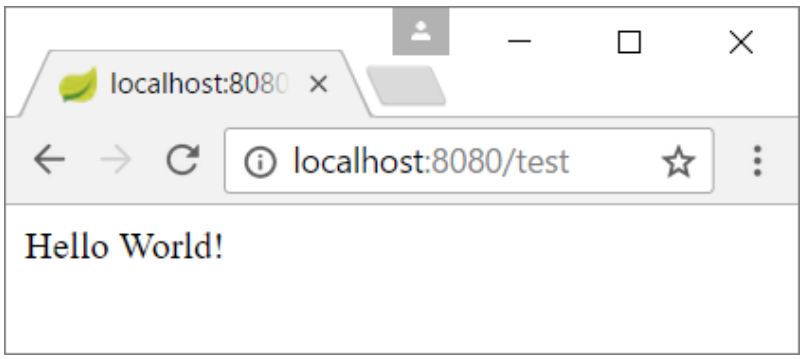
Hello,
The application works fine, but I create one restController on backend and maped it to:
@RequestMapping(value = "/test")

public String HelloWorld() {
return “Hello world!”;
}
When I try to access url localhost:8080/test I have the following message error:

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

How can I acess my rest endpoint on backend?

> Jeroen Bruinink
> on **March 23, 2017 at 22:04** said:
>
> Hi Mauricio,
>
> I’m not sure what went wrong there. Are there any more specific error messages or stack traces in your console?
>
> I wrote my own Hello World controller. WordPress won’t format code properly, so I saved it as a gist here
>
> When I visit the url with my browser, it seems to work:
>
> 
>

Chandu
on **March 23, 2017 at 23:03** said:

Hey Jeroen

Great blog post. Thank you so much for the knowledge share. I am new to Angular2 and Spring boot. I was able to set up and run everything with in 30 mins.

However I a question for you. So far, we have been displaying static content on the Angular2 front end. If I were to fetch data from the back end and display on the front

end, is adding a new RestController on the back end and make a rest call from front end is the only way to go about it??
Please advise. if you can, point me in the right direction. Much appreciated.

Thanks again.

> Jeroen Bruinink
> on **March 24, 2017 at 10:21** said:
>
> Hi Chandu,
>
> *...is adding a new RestController on the back end and make a rest call from front end is the only way to go about it?*
>
> Pretty much. It doesn't have to be REST though, you could also consider GraphQL for instance. Ofcourse you can communicate with your backend in any way you like, but by choosing a popular method you can be pretty sure it will be well supported by Angular2 and supporting libraries. I would **really** avoid rendering any html or javascript dynamically on the server side, because that would require that you keep user state in your backend.

Jim
on **March 24, 2017 at 21:51** said:

The mvn install is working but not the mvn run:

Failed to execute goal org.springframework.boot:spring-boot-maven-plugin:1.4.2.RELEASE:run (default-cli) on project parent: Unable to find a suitable main class, please add a 'mainClass' property -> [Help 1

In response to which I added:

com.mycompany.ng2boot.Ng2bootApplication
… to the parent pom.xml

But it still I get the same message. Moving the property into backup's pom.xml didn't seem to make a difference either.

Any ideas?

> Jim
> on **March 25, 2017 at 16:05** said:
>
> My bad…. it works now.

Thanks, Jaroen.

> Jeroen Bruinink
> on **March 25, 2017 at 17:15** said:
>
> You're welcome. Good to know it worked out.

> mustapha
> on **April 3, 2017 at 17:07** said:
>
> I have the same issue, what was the sollution in your case.

> Jeroen Bruinink
> on **April 6, 2017 at 11:27** said:
>
> Are you running the `mvn spring-boot:run` command from the backend directory?

Mykola Ilkiv
on **March 25, 2017 at 23:13** said:

Hello, Jeroen.
Thank you for a good tutorial. It works from the first time.

But I have one question according to this part:
"However, every time we save a source file, it will automatically rebuild the application and refresh the browser."

Am I understood right? Every time I am changing something in frontend module it should also automatically update it on the server (I mean should I see changes after refreshing the page)? Because unfortunately, it does not work in that way for me.

If I understood it correctly, then, can you please point it out where to look for the solution or in which part problem could be.

I will appreciate any help. Thank you in advance.

> Jeroen Bruinink

on **March 26, 2017 at 00:41** said:

Hi Mykola,

Please make sure you are using the Live Development Server by running `npm start` in the directory `ng2boot\frontend\src\main\frontend` and that you are connecting to http://localhost:4200

If you change (and save to disk) a source file of the frontend, the page in your browser should reload automatically and show the changes. There are some exceptions, for instance the proxy.conf.js file, where you need to restart the Live Development Server.

> Mykola Ilkiv
> on **March 26, 2017 at 04:34** said:
>
> Thank you for your prompt reply, Jeroen.
> You were right, I didn't run my Development Server but it works properly now.
>
> Thanks a lot.

mauricio
on **March 26, 2017 at 18:07** said:

Hellow, I have a question. I mapped my routes on angular 2 and it works fine on localhost:4200 'cause angular take care of it. But how can I acess this routes using localhost:8080? For example:

localhost:4200/products -> angular loads products page
localhost:8080/products -> error, spring havent this mappad

> Jeroen Bruinink
> on **March 27, 2017 at 07:36** said:
>
> Hi Mauricio,
>
> My collegue wrote a blog post about integrating Angular2 with Spring Boot using Gradle. It also covers html5 routing: Integrate Angular in Spring Boot with Gradle

John Masset
on **March 27, 2017 at 17:39** said:

Hi Jeroen,

Thanks again for the very helpful article. Is it possible with this setup to just have a parent and frontend pom? Or does the parent and backend have to be split up?

> John Masset
> on **March 28, 2017 at 14:26** said:
>
> Ha, I answered my own question, thanks again!
>
> > Jeroen Bruinink
> > on **March 29, 2017 at 08:01** said:
> >
> > Hi John,
> >
> > Great!
> >
> > In case other people are wondering: It is possible to do this by combining the backend and frontend directories and pom files into a single Maven module.

> Steve Nader
> on **March 29, 2017 at 20:32** said:
>
> Jeroen,
>
> This has been tremendously useful! One question, if I am bringing angular 2 into an application (aka localhost:8080 is already setup as a separate app) and I want to link to the angular2 app from localhost:8080 is there a way to do that without hard coding localhost:4200?
>
> Thanks!
>
> > Jeroen Bruinink
> > on **April 3, 2017 at 17:22** said:
> >
> > If you want to use relative links between two web applications, they need to be served by the same process (for instance, the same Tomcat instance), or they both need to be behind the same proxy server (Apache, Nginx, HAproxy)

Gaurav Nasit
on **March 28, 2017 at 20:00** said:

A few people include me had earlier mentioned and had interest in generating a war for the above example.

Below is what I have done to package it as a war as well, incase you want to update the blog to include it or if someone else wants to refer to it.

After the above, to generate a war, do the below steps.
1. In your backend/pom.xml,
add the below tag as a sibling to :
war
add the below to your dependencies list:

org.springframework.boot
spring-boot-starter-tomcat
provided

2. In your frontend package.json:
replace “build”: “ng build”,
with
“build”: “ng build -prod –base-href .”,
3. In your Application.java
add “extends SpringBootServletInitializer” as class extension
and override the configure method as below:
@Override
protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
return application.sources(AngularSeedApplication.class);
}

> Jeroen Bruinink
> on **March 29, 2017 at 07:54** said:
>
> Hi Gaurav,
>
> Thanks for the input! I’ll add it to the article when I have time. That way, I can format it a little better. I’m a bit busy at the moment, so it might take a few days.

Oli K
on **April 15, 2017 at 14:22** said:

“add the below tag as a sibling to :
war”

what does that mean?

> Sgt. Lee
> on **April 27, 2017 at 19:20** said:
>
> To get it working, I had to set the “packaging” element in the backend pom to “war”.

John McDonald
on **March 29, 2017 at 06:25** said:

How can I get this to work in Eclipse? I created a spring starter project and selected “web” and it runs. However, once I copy in the files from the Ng2Boot project, the Ng2boot pom wrecks my project. Eclipse no longer sees it as a spring boot project and I get errors on my bootApplication.java complaining that it can’t find the @SpringBootApplication annotation or ANY of the spring framework classes.

How do I configure eclipse to work with the parent/child pom files?

Thanks for your help

> Jeroen Bruinink
> on **March 29, 2017 at 08:02** said:
>
> You should be able to tell Eclipse to reload the Maven pom files somehow. I don’t have Eclipse at the moment, so I can’t test it myself right now.
>
> > Brian Smith
> > on **April 29, 2017 at 15:29** said:
> >
> > I use STS myself, I was able to import the project from Github as a maven project and everything just worked. I haven’t tried creating one from scratch myself.
> >
> > By the way, I also have Angular2 Eclipse Plugin installed in my STS, and some of its fancy functions apparently require the .angular-cli.json

to be at the root of the folder structure. I was able to move everything from the src/main/frontend directory down two levels, change the outdir in .angular-cli.json files to just '../target/frontend', and remove the workingDirectory from the frontend-maven-plugin configuration, and it still builds the jar correctly, without including the source files.

I'm only about a week into messing with Angular, but so far this is the best and simplest configuration I've found for an Angular2 / Boot app, and having the build process completely in Maven will let us avoid disruption to our existing Jenkins CI environment. So thank you so much for this!

Narsimlu
on **March 29, 2017 at 14:27** said:

Thanks for your post.I did the setup quickly in 30 mins after following the steps.I have one question.
Now if i made any changes to TS file,i am running mvn clean install and then only my changes are reflecting on spring boot server localhost:8080.

Instead this,Can we have all the JS in files backend after the transpiling.so if we refresh the server localhost:8080,it should pick the latest changes without builing code.

Can we achieve this?

Jeroen Bruinink
on **April 3, 2017 at 17:10** said:

If you want to develop your Javascript files more quickly, why don't you use the live development server?

Mo
on **March 29, 2017 at 19:00** said:

Hi Jeroen,

Thank you so much for the great post! I have a question hope that you have some insight.

We are trying to setup a CI using Jenkins for our Angular2-boot project, which uses the same setup described in your post. When “mvn clean install” is run, we are getting the error output listed below. Note that, we do not install Angular CLI globally on the CI server before running “mvn clean install”; and I think that might be why we are getting the error. However, we believe that CI server should be able to just pull down about any project and make a build without having to pre-install anything.

Looking forward to hear your thought on this. Thank you very much in advance.

15:37:35 [ERROR] npm ERR! Linux 3.10.0-514.6.1.el7.x86_64
15:37:35 [ERROR] npm ERR! argv “/var/jenkins_home/workspace/myproject_maven/frontend/src/main/frontend/node/node” “/var/jenkins_home/workspace/myproject_maven/frontend/src/main/frontend/node/node_modules/npm/bin/npm-cli.js” “run” “build”
15:37:35 [ERROR] npm ERR! node v6.9.5
15:37:35 [ERROR] npm ERR! npm v3.10.10
15:37:35 [ERROR] npm ERR! code ELIFECYCLE
15:37:35 [ERROR] npm ERR! ng2boot@0.0.0 build: `ng build -bh /myproject/`
15:37:35 [ERROR] npm ERR! Exit status 1
15:37:35 [ERROR] npm ERR!
15:37:35 [ERROR] npm ERR! Failed at the ng2boot@0.0.0 build script ‘ng build -bh /myproject/’.
15:37:35 [ERROR] npm ERR! Make sure you have the latest version of node.js and npm installed.
15:37:35 [ERROR] npm ERR! If you do, this is most likely a problem with the ng2boot package,
15:37:35 [ERROR] npm ERR! not with npm itself.
15:37:35 [ERROR] npm ERR! Tell the author that this fails on your system:
15:37:35 [ERROR] npm ERR! ng build -bh /myproject/
15:37:35 [ERROR] npm ERR! You can get information on how to open an issue for this project with:
15:37:35 [ERROR] npm ERR! npm bugs ng2boot
15:37:35 [ERROR] npm ERR! Or if that isn’t available, you can get their info via:
15:37:35 [ERROR] npm ERR! npm owner ls ng2boot
15:37:35 [ERROR] npm ERR! There is likely additional logging output above.
15:37:35 [ERROR]
15:37:35 [ERROR] npm ERR! Please include the following file with any support request:
15:37:35 [ERROR] npm ERR! /var/jenkins_home/workspace/myproject_maven/frontend/src/main/frontend/npm-debug.log
15:37:36 [INFO] ————————————————————————
15:37:36 [INFO] Reactor Summary:
15:37:36 [INFO]
15:37:36 [INFO] myproject …….. SUCCESS [0.081 s]
15:37:36 [INFO] backend …….. SUCCESS [14.312 s]
15:37:36 [INFO] frontend ……. FAILURE [01:26 min]
15:37:36 [INFO] ————————————————————————
15:37:36 [INFO] BUILD FAILURE

15:37:36 [INFO] ————————————————————————————————
15:37:36 [INFO] Total time: 01:43 min
15:37:36 [INFO] Finished at: 2017-03-16T15:37:36+00:00
15:37:36 [INFO] Final Memory: 41M/111M
15:37:36 [INFO] ————————————————————————————————
15:37:36 [ERROR] Failed to execute goal com.github.eirslett:frontend-maven-plugin:1.3:npm (npm run build) on project frontend: Failed to run task: 'npm run build' failed. (error code 1) -> [Help 1]
15:37:36 [ERROR]
15:37:36 [ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
15:37:36 [ERROR] Re-run Maven using the -X switch to enable full debug logging.
15:37:36 [ERROR]
15:37:36 [ERROR] For more information about the errors and possible solutions, please read the following articles:
15:37:36 [ERROR] [Help 1]
http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
15:37:36 [ERROR]
15:37:36 [ERROR] After correcting the problems, you can resume the build with the command
15:37:36 [ERROR] mvn -rf :frontend
15:37:36 Waiting for Jenkins to finish collecting data

> Jeroen Bruinink
> on **April 3, 2017 at 17:17** said:
>
> *However, we believe that CI server should be able to just pull down about any project and make a build without having to pre-install anything.*
>
> First of all: excellent idea, I couldn't agree more.
>
> Did you check the contents of `/var/jenkins_home/workspace/myproject_maven/frontend/src/main/frontend/npm-debug.log`? The standard npm output doesn't give a lot of information beyond "something went wrong".

> Valerii Sosliuk
> on **April 3, 2017 at 22:38** said:

Thank you! That's a kind of tutorial I was missing!

Sumanth
on **April 4, 2017 at 12:27** said:

Thank you for your post Jeroen. I am able to start both frontend and backend server. But http://localhost:4200/ works and shows ‘app works’ but with 8080 it gives 404. Where i am getting wrong.

Thanks In Advance.

Phanindra Kumar
on **April 4, 2017 at 12:27** said:

Hi

thanks for you post.

I followed the all step in you post. But after step 6 I am getting below ms in the browser

“Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Apr 04 15:55:34 IST 2017
There was an unexpected error (type=Not Found, status=404).
No message available”

I could able to figure what went wrong.

Ilse Haanstra
on **April 4, 2017 at 16:56** said:

Thanks for the elaborate and clear example.
I deploy the frontend jar to our maven dependency repository (Nexus) and the backend is build by our buildserver. Is this the setup you would envision?
When build by buildserver, the backend jar is deployed on the development environment, and then test environment and eventually production. I’m looking for possibilities to have an external configuration file for angular (like application.yml for

spring boot). The external configuration file could have environment specific values. Do you maybe have any suggestions?
(Initially i thought the environment.ts mechanisme could be of help however because the frontend package is build only once, before the backend is build, i don't think i can use it.)

> Jeroen Bruinink
> on **April 5, 2017 at 09:47** said:
>
> Hi Ilse,
>
> *I deploy the frontend jar to our maven dependency repository (Nexus) and the backend is build by our buildserver. Is this the setup you would envision?*
>
> Pretty much. I would probably also build the frontend jar on the build server and deploy the backend jar to the Nexus repository, but it's fine either way.
>
> As for the environment specific configuration: I would put it in an application.yml and build a backend service that can serve the necessary configuration values to the frontend. Make sure that service is very picky about which values it will expose to the frontend.
>
> > Ilse Haanstra
> > on **April 7, 2017 at 08:48** said:
> >
> > Thank you so much Jeroen :-)

> ManuelBk
> on **April 5, 2017 at 13:13** said:
>
> Hi,
>
> thanks for the great tutorial, however even your checked out git project seems to fail, when i try.
>
> Everything works fine, until I trie to point to localhost:8080:
> Only HTTP Status 404 by tomcat.
>
> I downloaded your project and startet everything like described in the tutorial.
>
> Bye,
> Manuel

Jeroen Bruinink
on **April 6, 2017 at 11:17** said:

Hi Manuel,

Could you please copy/paste the commands that you executed to start the application? I just did a download from the Gihub repository on another machine, started the application and was able to get the App Works! page.

What OS are you running on? Can you try executing mvn install and see if it succeeds?

> ManuelBk
> on **April 28, 2017 at 15:51** said:
>
> Hi Jeroen,
>
> sorry, that it took so long for me to answer.
> Meanwhile however it seems to work, when i swap the spring boot project, it seems that the configuration don't map resource/static folder to the web.
> However now its working and I could by maven config and build process ready for development, thanks a lot, your tutorial enabled me to kickoff my project.
>
> Bye,
> Manuel

Sumanth
on **April 6, 2017 at 05:53** said:

Hi Jeroen,

As i posted earlier, i followed setup instructions and brought up both backend and frint end servers. But app is not accessible with 8080, but with developments setup 4200 it works fine.

Can you please help me out to come out of this?

> Jeroen Bruinink
> on **April 6, 2017 at 11:20** said:
>
> Hi Sumanth,

I would love to help you, but I'm not sure how. Everything seems to work fine for me. ManuelBk reported the same problem, could you please answer the questions I asked him?

> Sumanth
> on **April 14, 2017 at 06:56** said:
>
> Hi Jeroen,
>
> I have used below commands to start the application.
> mvn clean install
> cd server(backend)
> mvn spring-boot:run.
>
> My backend server came up message Tomcat started on port(s): 8080 (http). But when i try to access with localhost:8080 it shows whitelabel error page 404. For some reason, request are not reaching server.
>
> Below is the package.conf.json
> {
> "/api": {
> "target": "http://localhost:8080",
> "secure": false
>
> }
> }
> Let me know if you need any other information.

> Eric
> on **April 6, 2017 at 10:00** said:
>
> Hi all !
> I'm a beginner with spring boot and I have some questions, can please help me?
> I followed the whole process and it worked fine, but now I want to add my .java files (backend) and my css,html and js files (frontend).
> Can you please show me how to do that properly ?
>
> Thank you.

Raj
on **April 6, 2017 at 19:49** said:

Hi

I already have node.exe install in my machine, would i still need to add this in pom.xml.

install node and npm

install-node-and-npm

when i comment this part i get below error

Failed to execute goal com.github.eirslett:frontend-maven-plugin:1.3:npm (npm install) on project frontend: Failed to run task: 'npm install' failed. java.io.IOException: Cannot run program "C:\Users\ad99473\Downloads\ng2boot-master\ng2boot-master\frontend\src\main\frontend\node\node.exe" (in directory "C:\Users\ad99473\Downloads\ng2boot-master\ng2boot-master\frontend\src\main\frontend"): CreateProcess error=2, The system cannot find the file specified -> [Help 1]

> Raj
> on **April 6, 2017 at 20:19** said:
>
> I downloaded the project from github and imported in STS. only thing i did is commented the npm plugins things from parent pom.xml, I did run npm install and build & npm start from the folder itself using existing npm install in my machine. now when i run the spring boot app, http://localhost:8080 , getting 404 errror, everthing is same from github.
> i created one rest controller, and able to get result from http://localhost:8080/home,
>
> But not sure how to run the frontend code on spring,
>
> > Jeroen Bruinink
> > on **April 7, 2017 at 12:04** said:
> >
> > Hi Raj,
> >
> > The frontend-maven-plugin does not support using a previously installed node.exe. If you really want to, you could use the maven-exec-plugin. However, to ensure that your builds are repeatable (over time and on different environments) you shouldn't rely on what is installed on the machine running the build. Downloading a Node and NPM version specified in your build files makes the build more stable. It also allows anyone(with an Internet connection) to just check out your code from a repository and run the build. You can (and probably should) still have a locally installed Node on your development machine to run your build manually.

Raj
on **April 7, 2017 at 21:28** said:

If I have to deploy in production, I can only deploy the war file, will i need node.exe installed there as well? how does it work. for working in angular one , we just need the js file reference. Not sure , we can deploy same way for angular 2 as well?

Raj
on **April 6, 2017 at 20:21** said:

I commented this part,

com.github.eirslett
frontend-maven-plugin
1.3

v6.10.1o
4.4.1
src/main/frontend

<!–
install node and npm

install-node-and-npm

–>

<!–
npm install

npm

npm run build

npm

run build

–>

Raj

on **April 6, 2017 at 20:53** said:

when i unzip the jar files and checked the angular2 files are not copied in frontend.jar files, do you know why?

Ouss
on **April 7, 2017 at 10:47** said:

Hello Jeroen,
Thank you a lot for this tutorial, it is so helpfull.

I have a beginner question, I created an HTML page under frontend/src/main/frontend/src/research/mypage.html and I created a restController under backend/src/main/java/projectname/researchController/ReasearchController.java as follows

```
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ResearchController {

@RequestMapping(value =”/research”)
public String helloWorld() {
return ” “; //
}
}
```

What value do I have to return ? and is there any updates I need to make on my project files ?

Thand for your response.

Daniel
on **April 9, 2017 at 20:31** said:

Are there any solutions how to use angular 2 + spring mvc (without spring boot) ?

Nino
on **April 12, 2017 at 15:06** said:

Hi Jeroen. Thank you for this tutorial.

proxy.conf.json:
{
“/api”: {
“target”: “http://localhost:8080”,
“secure”: false
}
}

But if i try to call the rest api with:

this.http.post(‘/authenticate’, JSON.stringify({ username: username, password: Md5.hashStr(password) }))

I get this error:

Response with status: 404 Not Found for URL: http://localhost:4200/authenticate

So it’s still trying to call in localhost:4200 instead of localhost:8080
Have you any hint or solution for this?

Kind Regards

> Nino
> on **April 17, 2017 at 22:31** said:
>
> Hello again, after trying several things, I got it to work.
> If anyone has the same problems, I changed proxy.conf.json to:
>
> {
> “/”: {
> “target”: “http://localhost:8080”,
> “changeOrigin”: true,
> “secure”: false,
> “pathRewrite”: {
> “^/”: “”
> },
> “logLevel”: “debug”
> }
> }
>
> now It’s working as intended for me.