

This is what you will need to download from the git repository :

nginx-1.11.13 folder

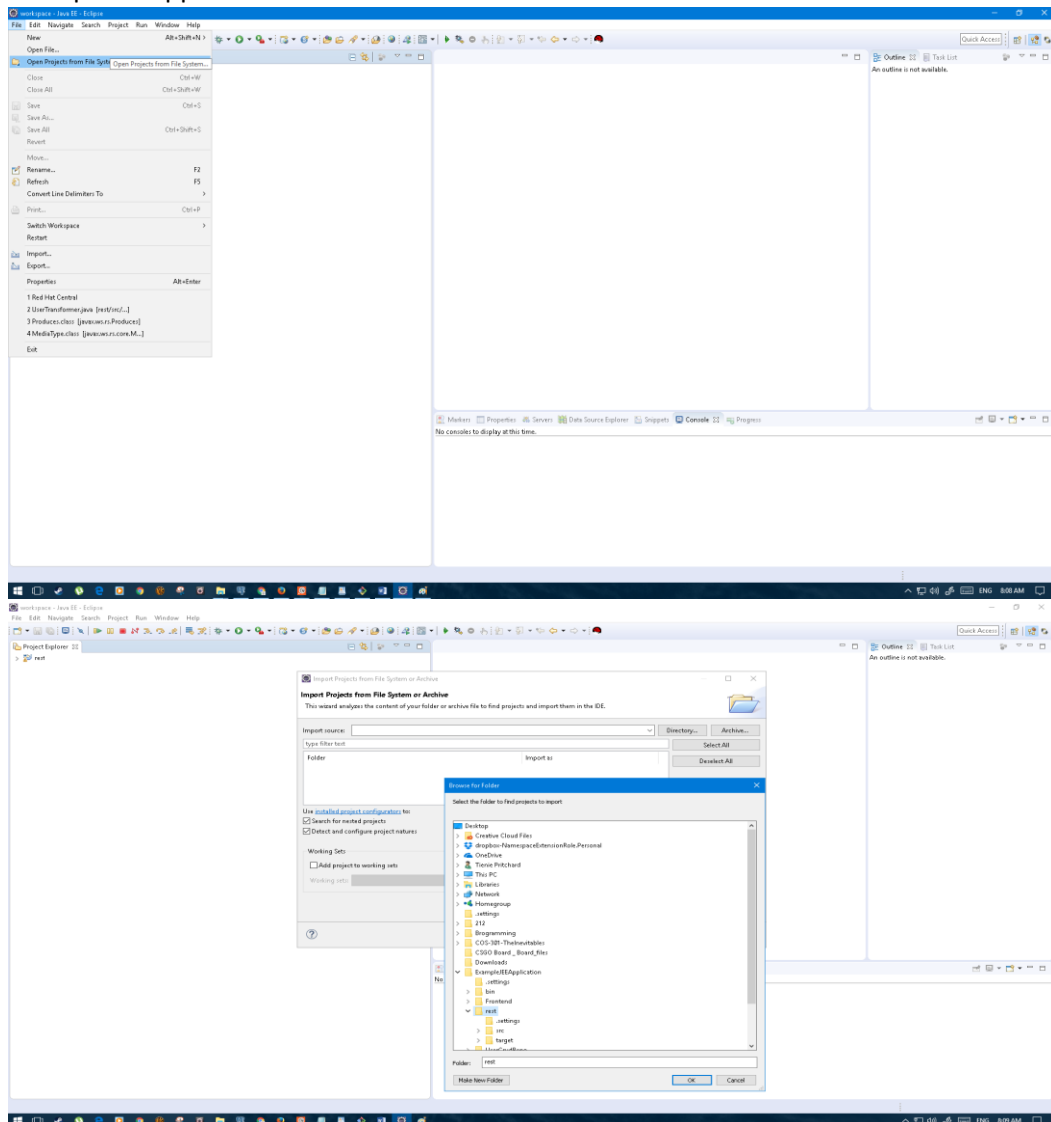
download maven on your machine

So that is all, don't mess with it since the configuration has already been done.

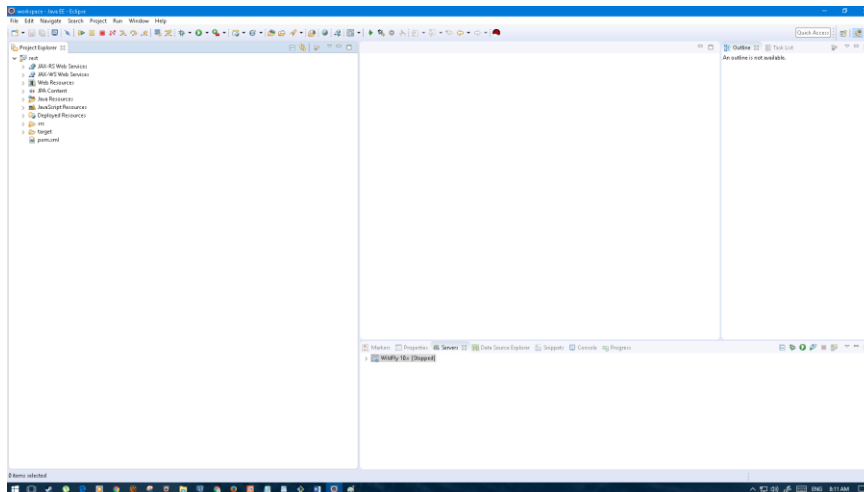
A problem I had was the I kept getting 405 errors which were CORS issues. I learnt that is a massive problem in the dev world. So what I am doing now is I am using a proxy server called nginx to bypass the cors issues. Then I am using webpack(frontend/webpack) to run the code on localhost:9000 and using that proxy to access the data. Also we are barely using eclipse anymore, only to run wildfly since eclipse has its own servers which messes up everything else. I would suggest downloading atom to work with the front end code

Here is the step by step instructions to get everything working with the proxy server and webpack.

1. Open eclipse go to open projects from file system and click directory. go to ExampleJEEApplication and click on the rest folder and click ok



This is how your screen is supposed to look like when you start up eclipse.

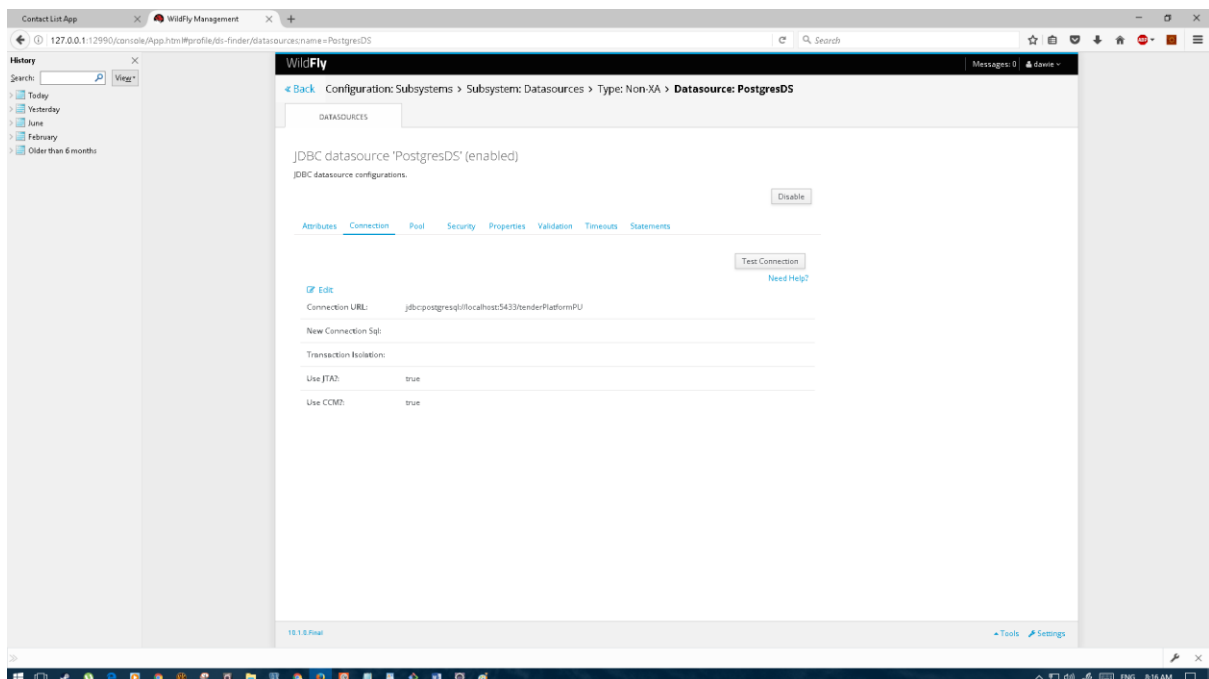


And this is what your wildfly datasource should look like in the management when you start wildfly:

IMPORTANT NOTES SINCE WE ARE USING A PROXY THERE IS AN OFFSET OF 3000 ON EACH PORT:

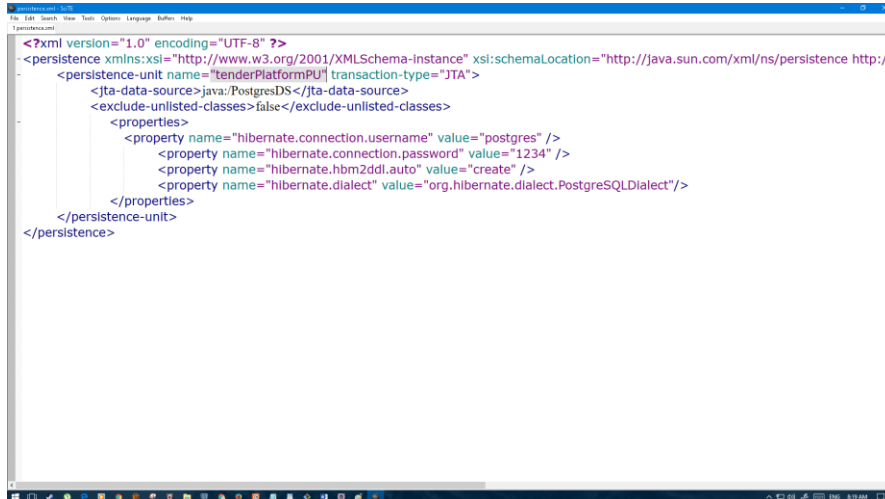
<http://127.0.0.1:12990> for the management of wildfly.

Your connection url is when you press view on the datasource. Make sure it points to tenderPlatformPU, localhost 5433 as previously discussed is just on my machine you can use localhost 5432, also make sure the datasource name is PostgresDS



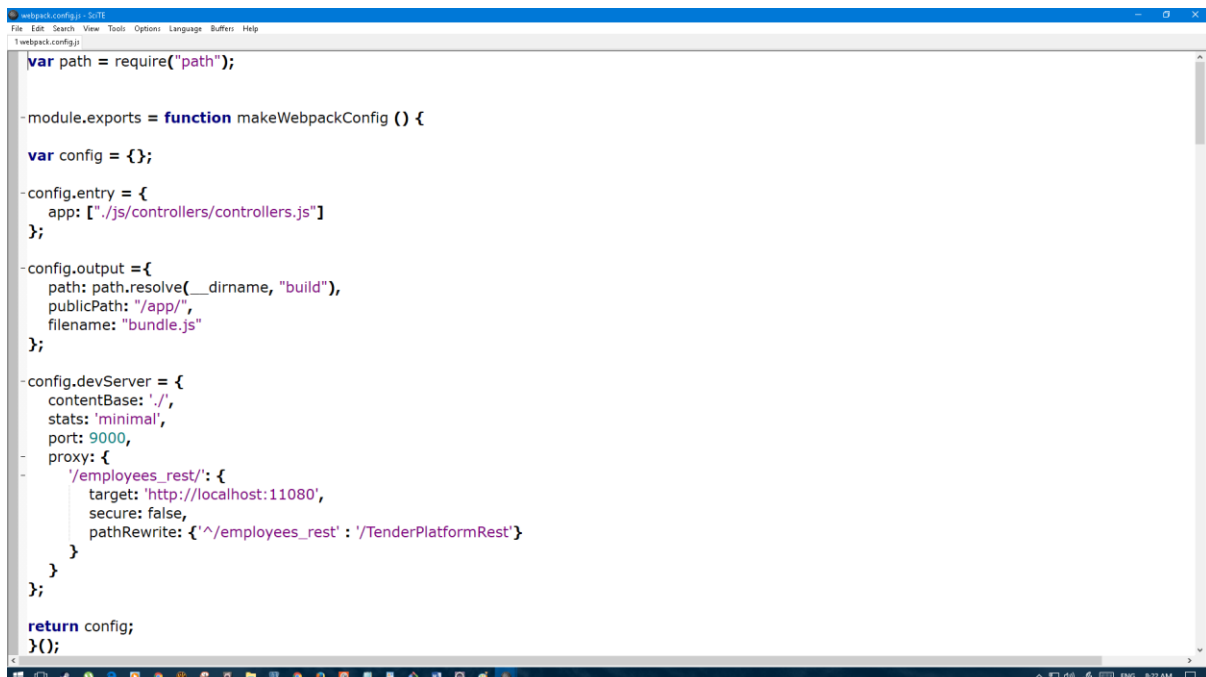
Now lets quickly go throw what nginx and webpack does and also persistence.xml which is located in the rest folder, \ExampleJEEApplication\rest\src\main\java\META-INF

Persistence.xml: this is the hibernate settings to connect to postgres, that is why your datasource should have tenderPlatformPU, username password is same for all and dialect is same for all, but its massively important otherwise you wont be able to persist.



```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://
<persistence-unit name="tenderPlatformPU" transaction-type="JTA">
  <jta-data-source>java:PostgresDS</jta-data-source>
  <exclude-unlisted-classes>false</exclude-unlisted-classes>
  <properties>
    <property name="hibernate.connection.username" value="postgres" />
    <property name="hibernate.connection.password" value="1234" />
    <property name="hibernate.hbm2ddl.auto" value="create" />
    <property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect"/>
  </properties>
</persistence-unit>
</persistence-->
```

Now we look at webpack.config.js ExampleJEEApplication\Frontend



```
var path = require("path");

module.exports = function makeWebpackConfig () {

  var config = {};

  config.entry = {
    app: [ './js/controllers/controllers.js' ]
  };

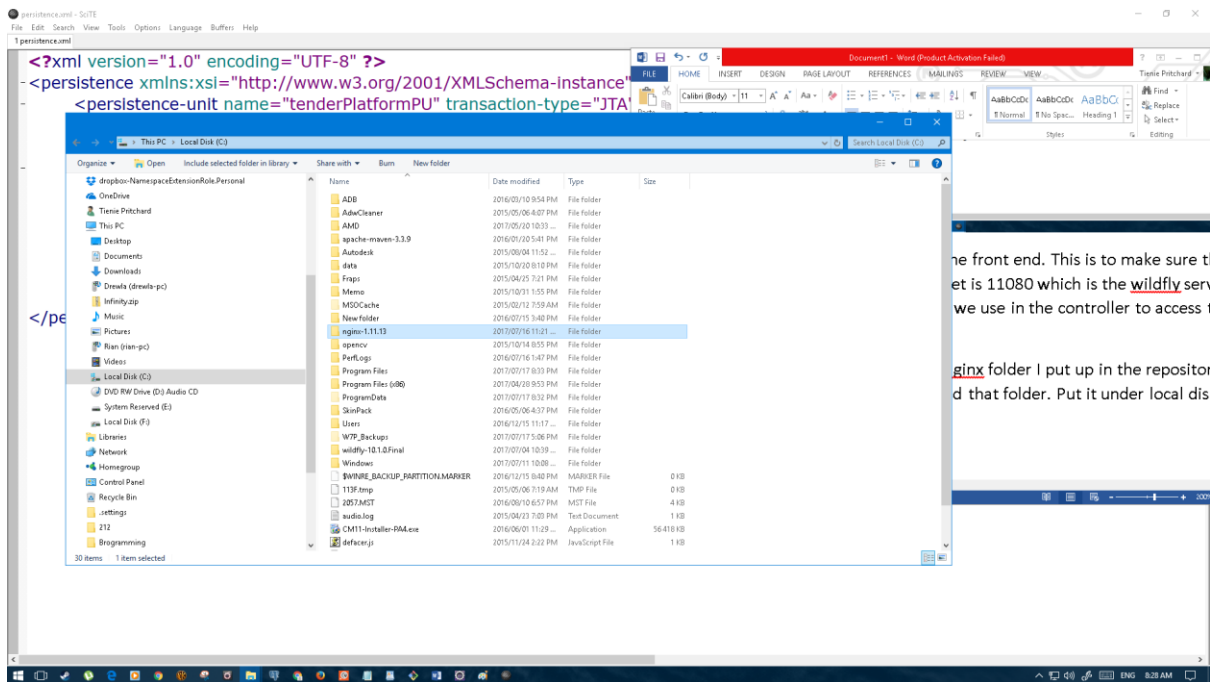
  config.output = {
    path: path.resolve(__dirname, "build"),
    publicPath: "/app/",
    filename: "bundle.js"
  };

  config.devServer = {
    contentBase: './',
    stats: 'minimal',
    port: 9000,
    proxy: {
      '/employees_rest': {
        target: 'http://localhost:11080',
        secure: false,
        pathRewrite: { '^/employees_rest' : '/TenderPlatformRest' }
      }
    }
  };

  return config;
}();
```

App is located at the controller.js file in the front end. This is to make sure the controller can make calls to that path, as you can see the target is 11080 which is the wildfly server and our proxy directory is employees_rest, that is what we use in the controller to access the backend guys.

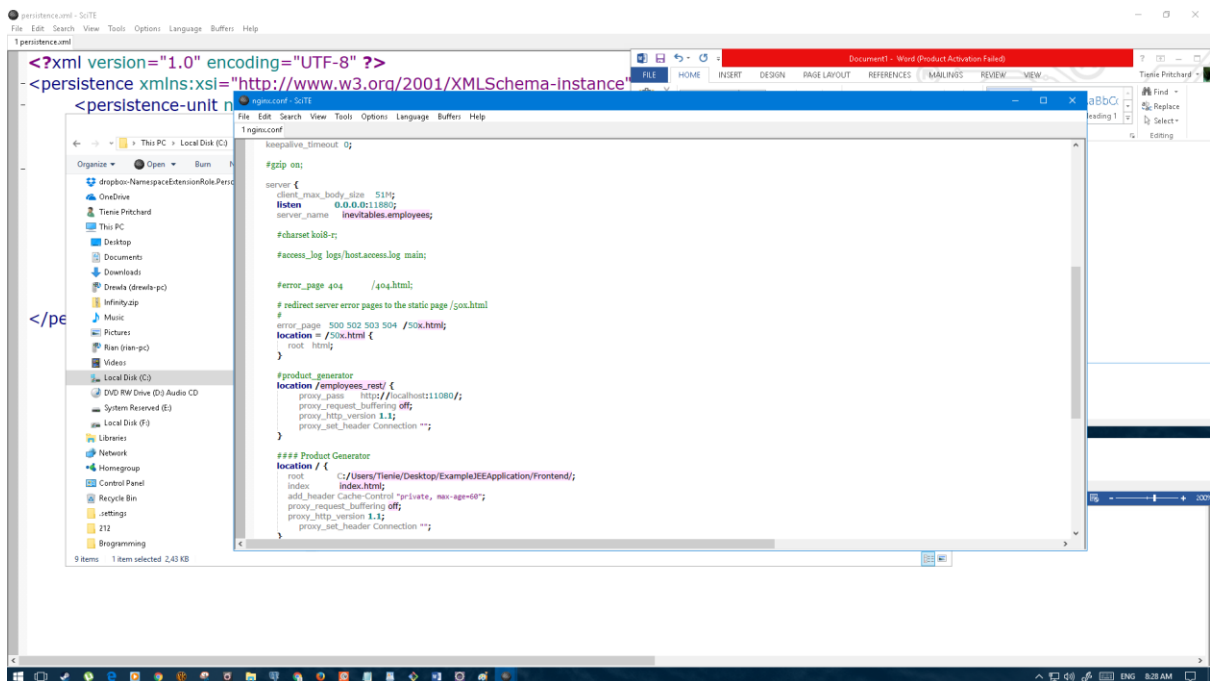
Lets take a look at NGINX. Retrieve the nginx folder I put up in the repository it should be the folder nginx-1.11.13. After you have downloaded that folder. Put it under local disk C



the front end. This is to make sure that the port is 11080 which is the **wildfly** server we use in the controller to access it

the **nginx** folder I put up in the repository and that folder. Put it under local disk

Lets look at the config file of nginx



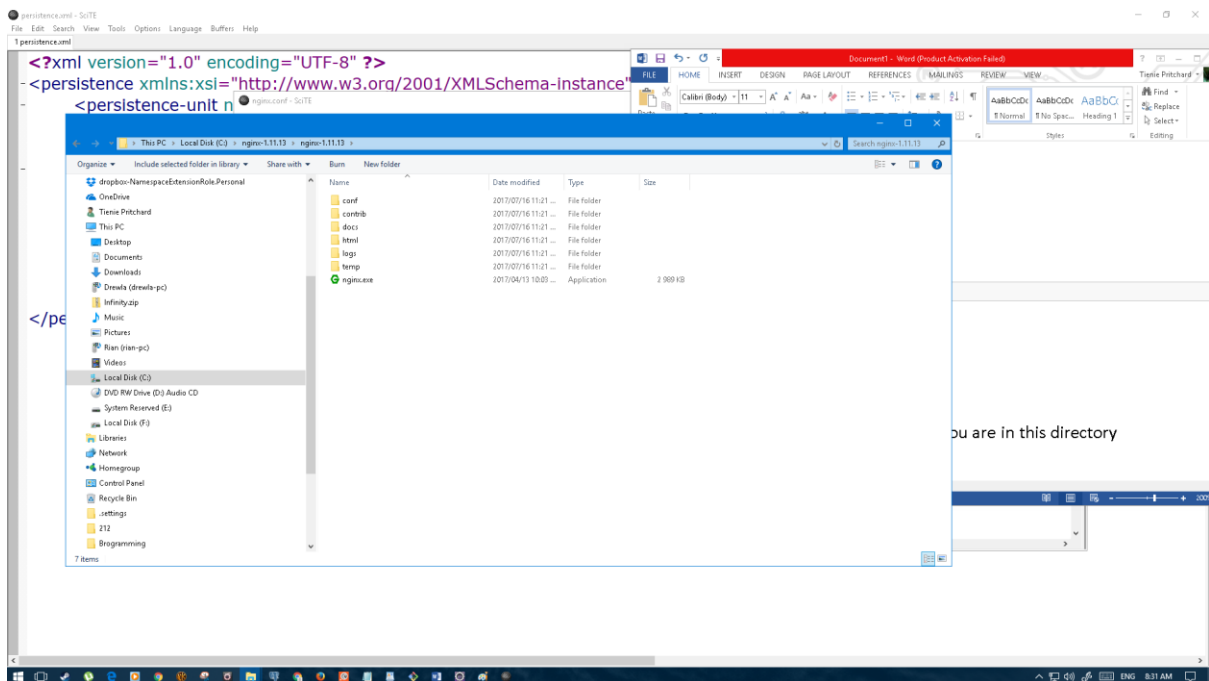
As you can see it sets up the proxy to employees_rest so you can use that in your controller. And with webpack you specify that proxy

Now we can finally go and run the JavaEE project guys!

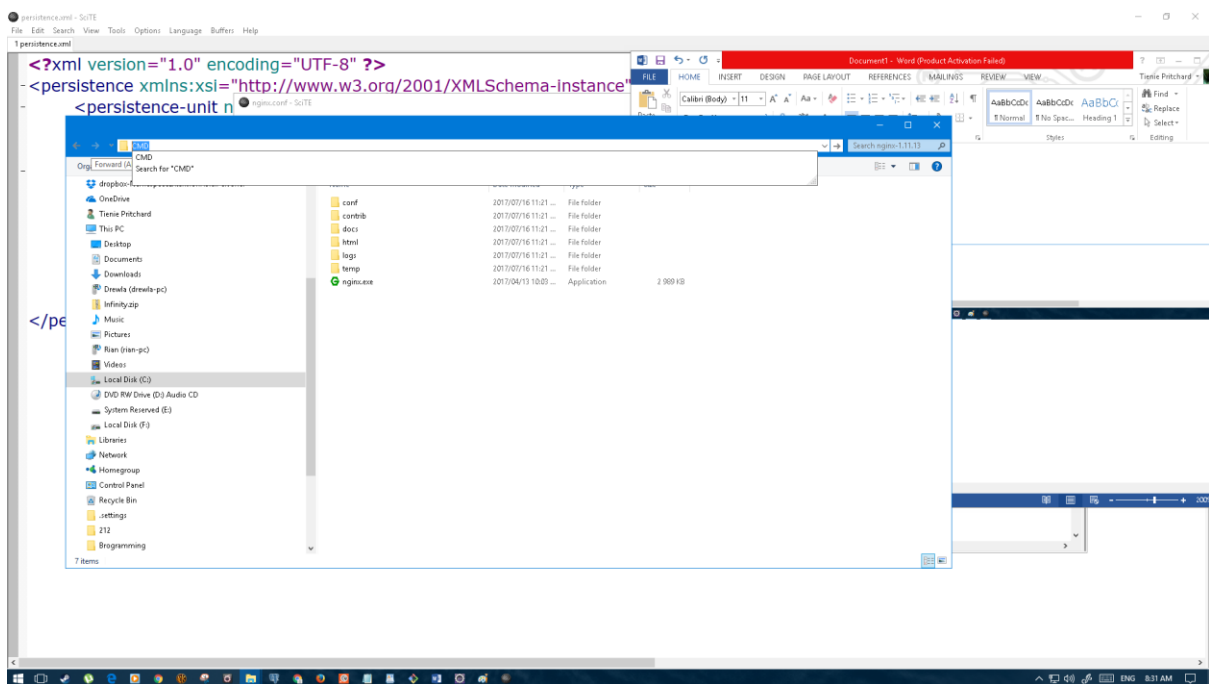
We will be using cmd A TON SO PLEASE GET USED TO IT I WILL GIVE EXAMPLES BECAUSE YOU NEED IT FOR MAVEN

Step 1: Start nginx

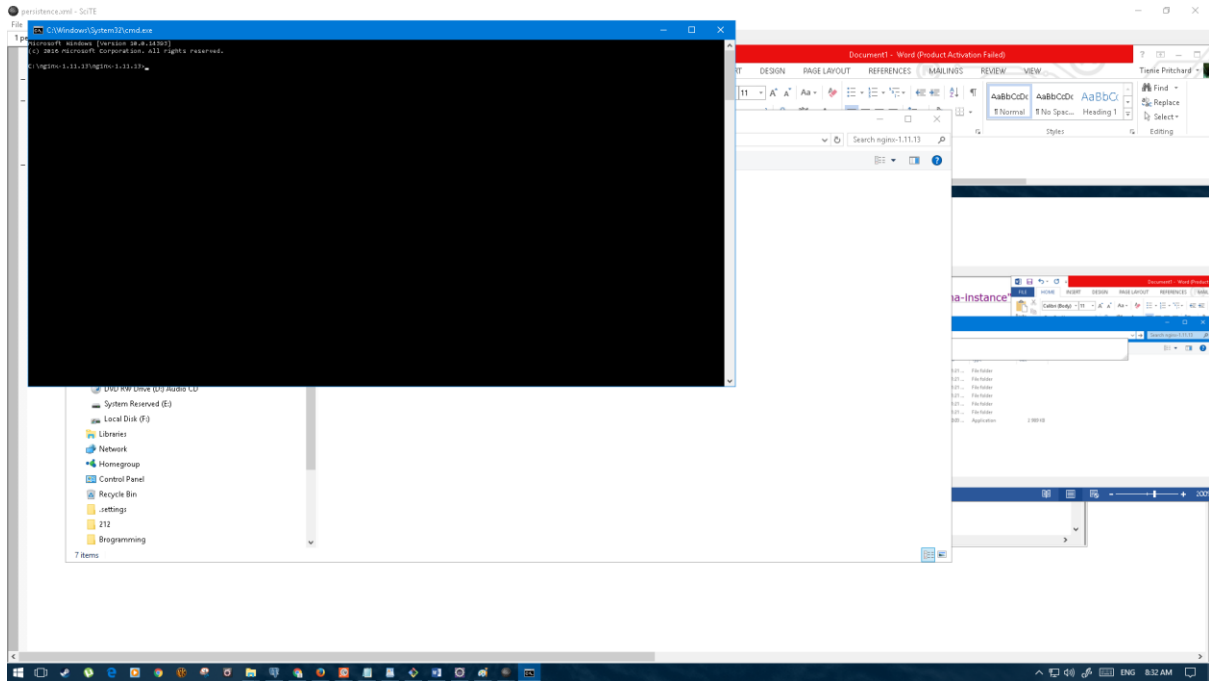
Go to your nginx folder on C drive until you are in this directory



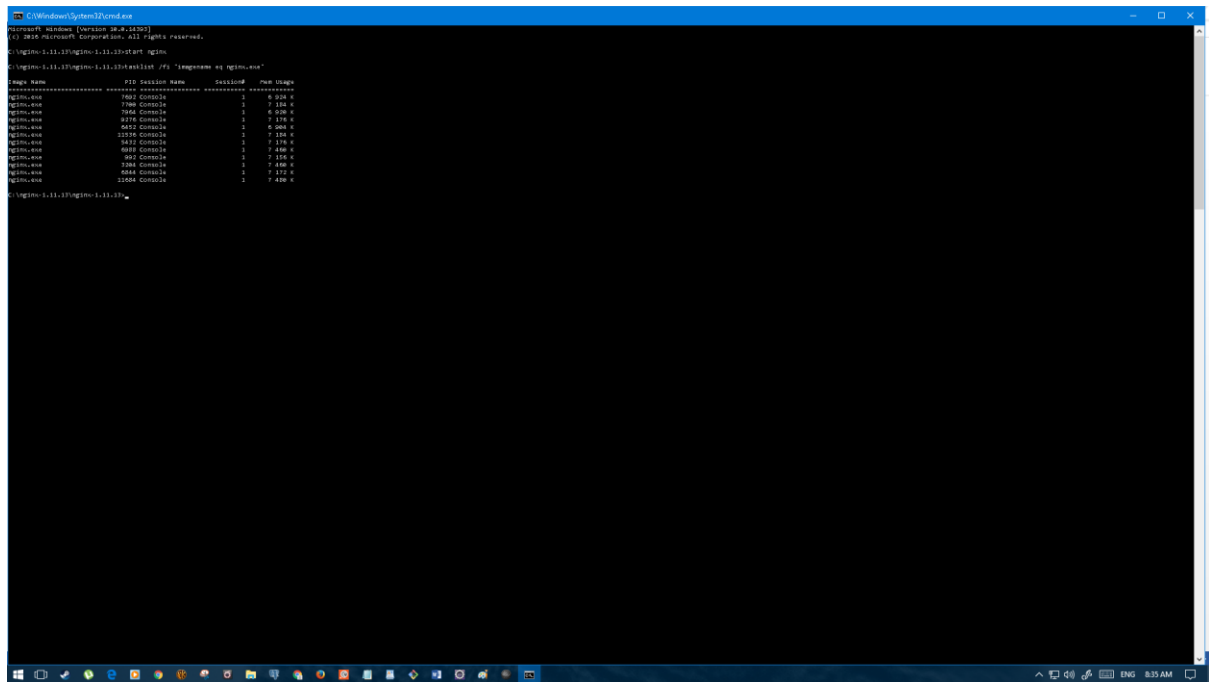
Then cmd the top url as follows:



If you did that correctly it should look as follows your screen:



Then to start do this:



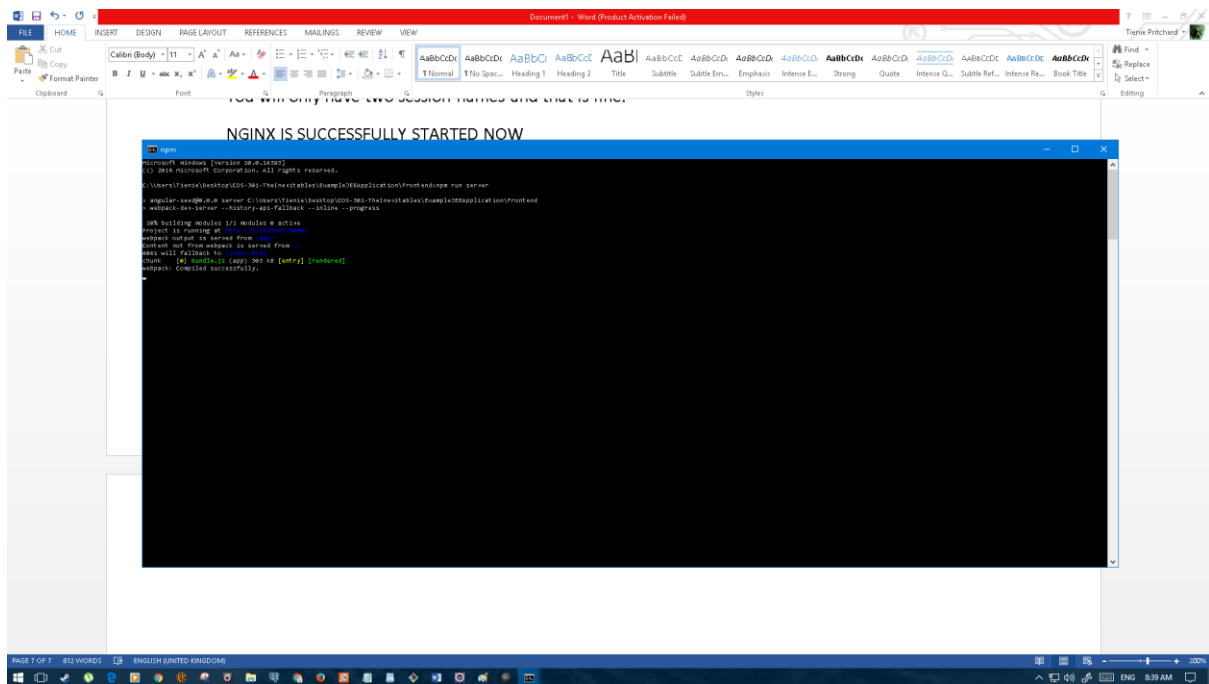
You will only have two session names and that is fine.

NGINX IS SUCCESSFULLY STARTED NOW

Now we need to start webpack so it can proxy and we can use localhost:9000 to actually see the front end.

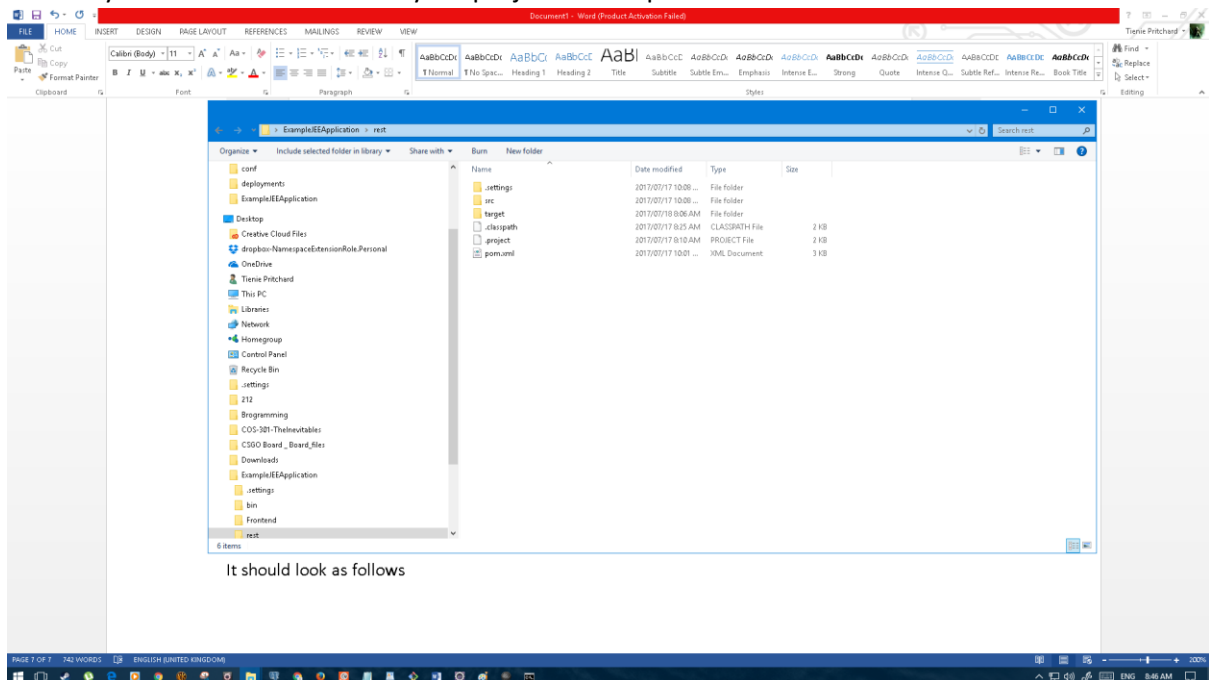
Go to the directory of webpack.config.js : ExampleJEEApplication\Frontend

Cmd the directory as we did with nginx and type in :npm run server, that should start webpack and you will have a terminal looking as follows:



If you have this then you will have nginx running and you will have webpack running. WELL DONE!

Now we will focus on maven. This is what you will be using to compile and deploy to wildfly. For a ton of reasons but especially to make sure you have a newly compiled project we will do this manually. Go to the rest folder of your project where pom.xml is as follows :



It should look as follows

Then cmd the directory

Once the terminal has loaded enter the command: mvn clean install, this will compile the project and tell you if there are any errors if not it should say build success.

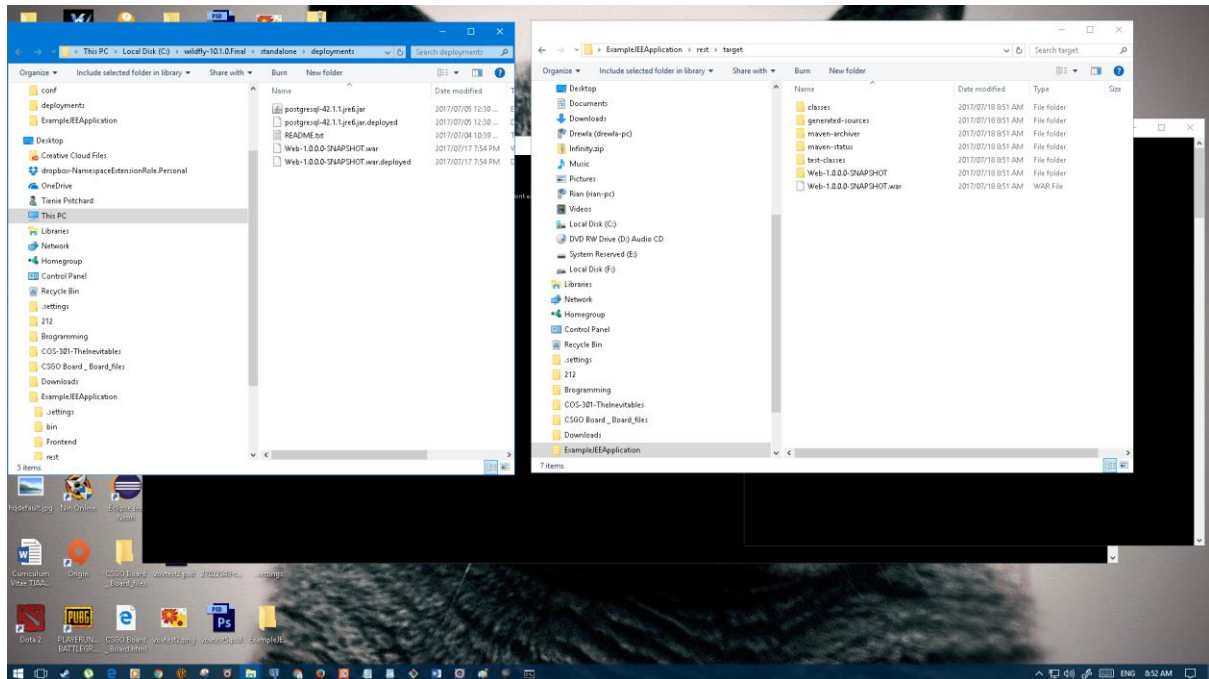
NOW TO DO THE DEPLOYMENT MANUALLY:

2 DIRECTORIES WILL NEED TO BE OPEN. YOUR WILDFLY DIRECTORY:

1.wildfly-10.1.0.Final\standalone\deployments

2.ExampleJEEApplication\rest\target

It should look as follows



Copy the.war file from target to wildfly and just overwrite it. This will deploy the project to wildfly, you should see that on your wildfly server console in eclipse.

THERE YOU GO NOW YOU CAN SUCCESSFULLY GO TO LOCALHOST:9000 AND ACCESS THE USER API.

IF YOU HAVE ANY ISSUES PLEASE CONTACT ME ASAP.

THANKS GUYS