Diagram

Description automatically generated

Diagram

Description automatically generated with medium confidence

Getting started with spring boot.

Go ahead to the spring initializer.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Start.spring.io website you can initialize the spring project.

Then we can select dependencies.

1. Spring web
2. Spring data JPA
3. PostgreSQL driver

Once you click on generate button you can download the zip file and then you can open it with IntelliJ IDE.

There’s a pom.xml file where we can see all the dependencies listed as xml scrip.

It is good make this folder a git repo. And push it to the remote github.com.

Command – git init

Command – git branch main

Command – git checkout main

Command – git remote add origin <url to the git repo>

Command – git add -A

Command – git commit -m ”initial commit to the remote”

Command – git push -u origin main

I want to learn

Unit testing

Integration testing and mocking and tdd.

Graphical user interface, text, application, chat or text message

Description automatically generated

All the testing happens inside the test folder.

Graphical user interface, text, application

Description automatically generated

This is where we put our code

Inside the com.example.demo

Annotations in java solves two problems in common programming

1. Commenting without
2. Its easy to debug.

There are pre build annotations made in java. But we can also create annotations.

Pre built annotations in Java

1. @Deprecated
2. @Override
3. @SuppressWarnings(“unchekced”)

We can use annotations on classes, we can use annotations on methods and also on variables.we can create annotations using interfaces and use them on our codes for our specific purposes.

In springboot itself there are annotations specified for the framework.

In resources folder we have two folders and one file.

Static folder – where we save HTML, CSS, JavaScript

Templates

Application.properties – this is where we write all the properties as well as environment specific properties of our applications like .env in node.js

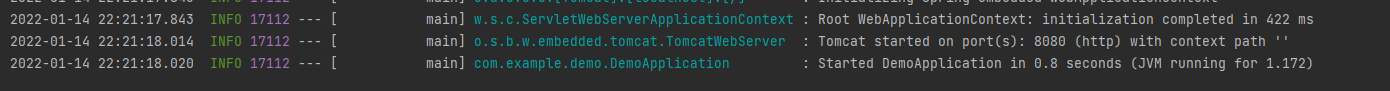
When I run the application now which is DemoApplication.java then I get an error called failed to determine a suitable driver class.

Only for now I can comment that dependency in pom.xml. which is the

Spring-boot-starter-data-jpa

Then I can right click on pom.xml and under maven theres option called reload project.

Once you run the DemoApplication.java the tomcat server software stat on port 8080.

that’s the msg we want to receive.

So this means we have web server up and running on port 8080 and if we try to hit our web server on this port we will get nothing back because we haven’t implemented any end points.

Lets look at how easy to implement a Restful API then we can use it here.

**Simple API with Spring Boot**

Once you start the server with tomcat you can go to the browser and type port then you get the following page.

Graphical user interface, text, application, email

Description automatically generated

Status code is 404

Lets create an end point.

Annotations starts with capital letters since all annotations are interfaces. Interfaces are also classes which are different from normal classes and abstract classes. So when you create an annotation we also want to use that capitalize convention.

Text

Description automatically generated

So we convert the main application into a Rest API by annotation using “RestController”.

We also need to put @GetMapping annotation since we need to output this return function to the frontend of the web application.

Once you done with changes you have to kill the server and start again the web server. Then only you can see the changes.

Like @GetMapping we have

1. @GetMapping
2. @PostMapping
3. @PutMapping

and so on and so for. Lets learn all of these mappings in this tutorial.

We cant put two @GetMapping inside the same class I tried and does not work.

So only one

Text

Description automatically generated@GetMapping will work

Diagram

Description automatically generatedWe just build the Get but we need to build all of the following things.

**Creating the student class**