Spring Boot

1

Spring Boot

1 Spring Boot was introduced in April 2014 to reduce some of the burdens while developing a Java web application.

2 Before SpringBoot, Developer need to configure a servlet container, establish link b/w Tomcat and Dispatcher servlet, deploy into a server, define lot of dependencies........

3 But with SpringBoot, we can create Web Apps skeletons with in seconds or at least in 1-2 mins ☺. It helps eliminating all the configurations we need to do.

4 Spring Boot is now one of the most appreciated projects in the Spring ecosystem. It helps us to create Spring apps more efficiently and focus on the business code.

5 SpringBoot is a mandatory skill now due to the latest trends like Full Stack Development, Microservices, Serverless, Containers, Docker etc.

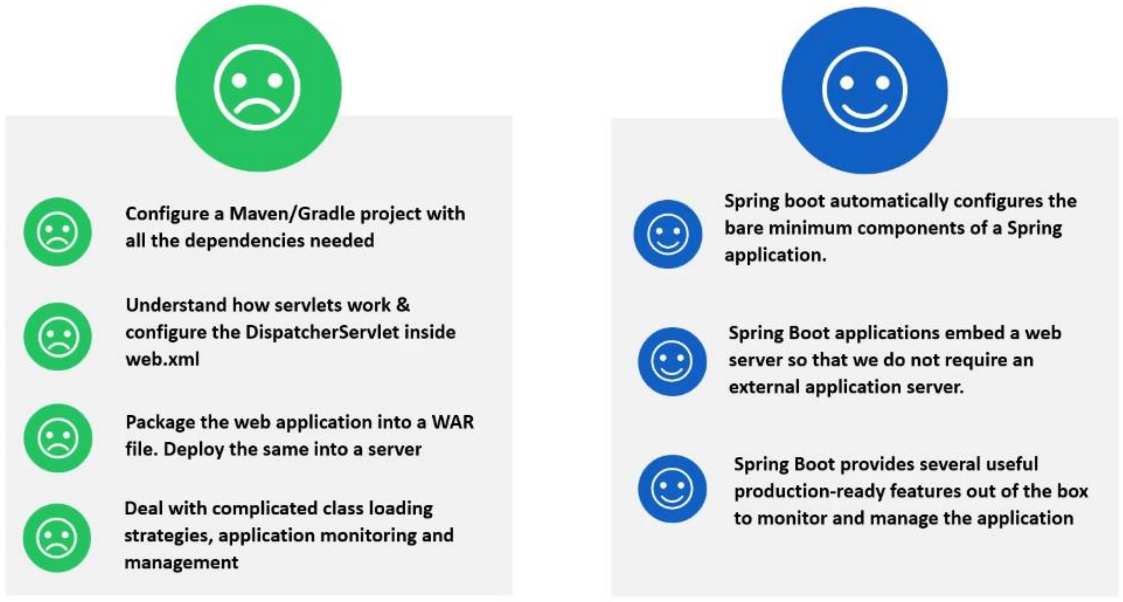
Spring Boot Makes Getting Started with Spring Quick and Easy
EASY
Quick
Easy

“Spring Boot makes it easy to create stand-alone, production-grade, Spring-based applications that you can ‘just run’.”

Spring Boot Project

2

Before & After Spring Boot



Configure a Maven/Gradle project with all the dependencies needed

Understand how servlets work & configure the DispatcherServlet inside web.xml

Package the web application into a WAR file. Deploy the same into a server

Deal with complicated class loading strategies, application monitoring and management

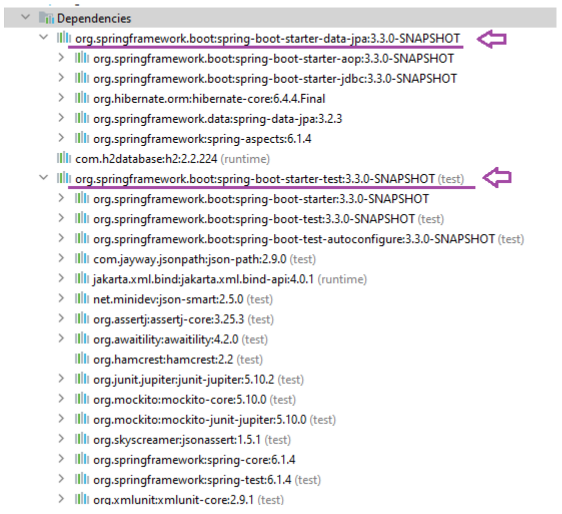
Spring boot automatically configures the bare minimum components of a Spring application.

Spring Boot applications embed a web server so that we do not require an external application server.

Spring Boot provides several useful production-ready features out of the box to monitor and manage the application

Spring Boot important features

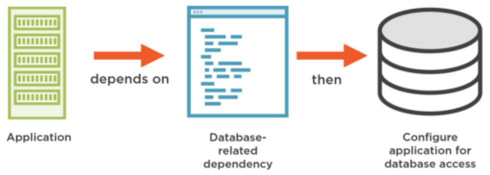
- **Spring Boot starters**: it groups related dependencies used for a specific purpose as starter projects.
  - We don't need to figure out all the must-have dependencies you need to add to your project for one particular purpose nor which versions you should use for compatibility.
  - E.g.
    - spring-boot-starter-web
    - spring-boot-starter-data-jpa
    - spring-boot-starter-test



```
Dependencies
  org.springframework.boot:spring-boot-starter-data-jpa:3.3.0-SNAPSHOT
    org.springframework.boot:spring-boot-starter-aop:3.3.0-SNAPSHOT
    org.springframework.boot:spring-boot-starter-jdbc:3.3.0-SNAPSHOT
    org.hibernate.orm:hibernate-core:6.4.4.Final
    org.springframework.data:spring-data-jpa:3.2.3
    org.springframework:spring-aspects:6.1.4
  com.h2database:h2:2.2.224 (runtime)
  org.springframework.boot:spring-boot-starter-test:3.3.0-SNAPSHOT (test)
    org.springframework.boot:spring-boot-starter:3.3.0-SNAPSHOT
    org.springframework.boot:spring-boot-test:3.3.0-SNAPSHOT (test)
    org.springframework.boot:spring-boot-test-autoconfigure:3.3.0-SNAPSHOT (test)
    com.jayway.jsonpath:json-path:2.9.0 (test)
    jakarta.xml.bind:jakarta.xml.bind-api:4.0.1 (runtime)
    net.minidev:json-smart:2.5.0 (test)
    org.assertj:assertj-core:3.25.3 (test)
    org.awaitility:awaitility:4.2.0 (test)
    org.hamcrest:hamcrest:2.2 (test)
    org.junit.jupiter:junit-jupiter:5.10.2 (test)
    org.mockito:mockito-core:5.10.0 (test)
    org.mockito:mockito-junit-jupiter:5.10.0 (test)
    org.skyscreamer:jsonassert:1.5.1 (test)
    org.springframework:spring-core:6.1.4
    org.springframework:spring-test:6.1.4 (test)
    org.xmlunit:xmlunit-core:2.9.1 (test)
```

Spring Boot important features…

- **Auto-configuration**: Based on the dependencies present in the classpath, Spring Boot guesses and auto-configures the spring beans, property configurations, etc.
  - E.g.
    - if it is a web app, then the **Tomcat port** will be auto-configured to **8080**.
    - If it identifies a **mysql** dependency, it will look for the connection details in the ***application.properties*** file and during the start of the app, it will create a connection object based on the details that we provided!
  - To achieve auto-configuration Spring Boot follows the **convention-over-configuration** principle.
    - **convention-over-configuration:** If Spring Boot doesn't find a configuration, it will assume something based on the convention rules that we have (e.g. Tomcat port 8080).
    - It aims to improve developer productivity, reduce boilerplate code, and promote best practices by providing sensible defaults and conventions. However, developers still have the flexibility to override these defaults and configure their applications according to their specific requirements when necessary.


Application
depends on
Database-related dependency
then
Configure application for database access

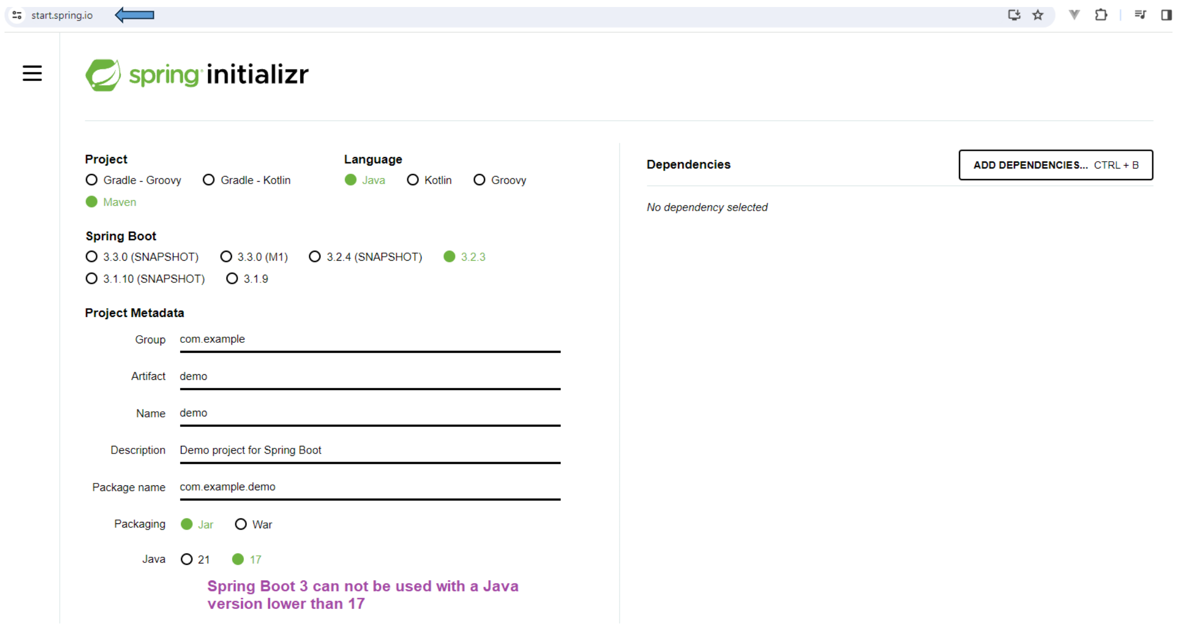
5

Spring Boot important features…

- **Actuator**: It is a starter project inside Spring Boot. It provides a pre-defined list of actuator endpoints.
  - Using these production-ready endpoints, we can monitor app health, metrics, etc.
- **DevTools:** It includes features such as automatic detection of application code changes, a LiveReload server to automatically refresh any HTML changes to the browser.
  - No need to restart the server after changes… It is done automatically!
  - We use this feature only in development phase… not in production.

6

Create a simple web application with Spring Boot



start.spring.io

spring initializr

Project
Gradle - Groovy Gradle - Kotlin [✓ Maven]

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.3.0 (SNAPSHOT) 3.3.0 (M1) 3.2.4 (SNAPSHOT) [✓ 3.2.3]
3.1.10 (SNAPSHOT) 3.1.9

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	[✓ Jar] War
Java	21 [✓ 17]

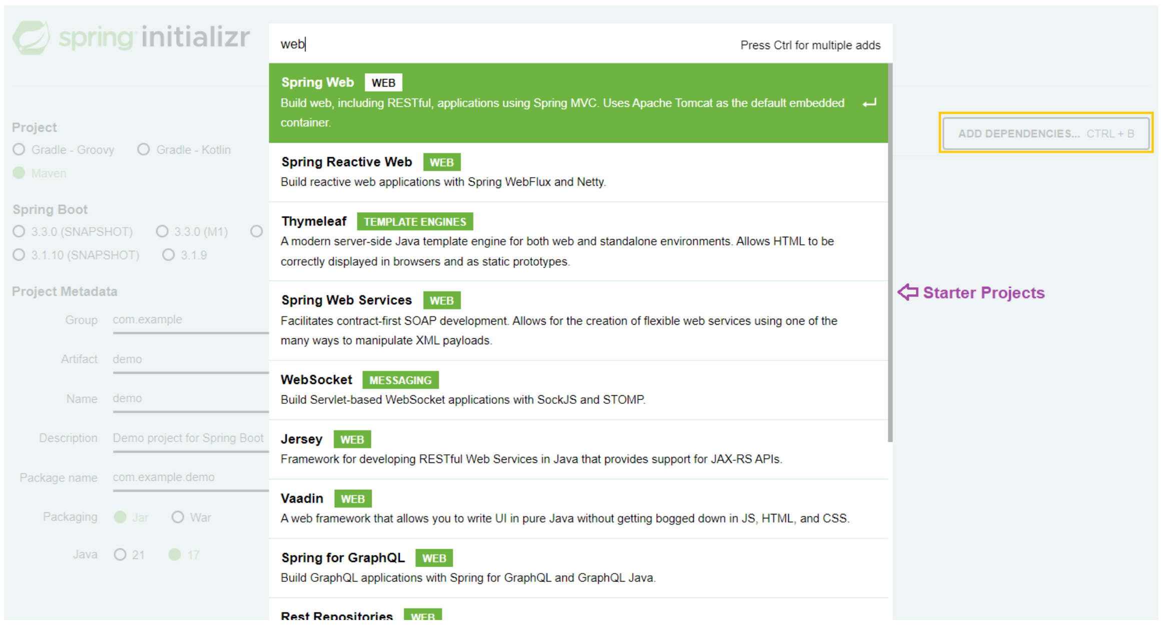
Spring Boot 3 can not be used with a Java version lower than 17

Dependencies ADD DEPENDENCIES... CTRL + B

No dependency selected

7

Create a simple web application with Spring Boot



web Press Ctrl for multiple adds

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Reactive Web WEB
Build reactive web applications with Spring WebFlux and Netty.

Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

Spring Web Services WEB
Facilitates contract-first SOAP development. Allows for the creation of flexible web services using one of the many ways to manipulate XML payloads.

WebSocket MESSAGING
Build Servlet-based WebSocket applications with SockJS and STOMP.

Jersey WEB
Framework for developing RESTful Web Services in Java that provides support for JAX-RS APIs.

Vaadin WEB
A web framework that allows you to write UI in pure Java without getting bogged down in JS, HTML, and CSS.

Spring for GraphQL WEB
Build GraphQL applications with Spring for GraphQL and GraphQL Java.

Rest Repositories WEB

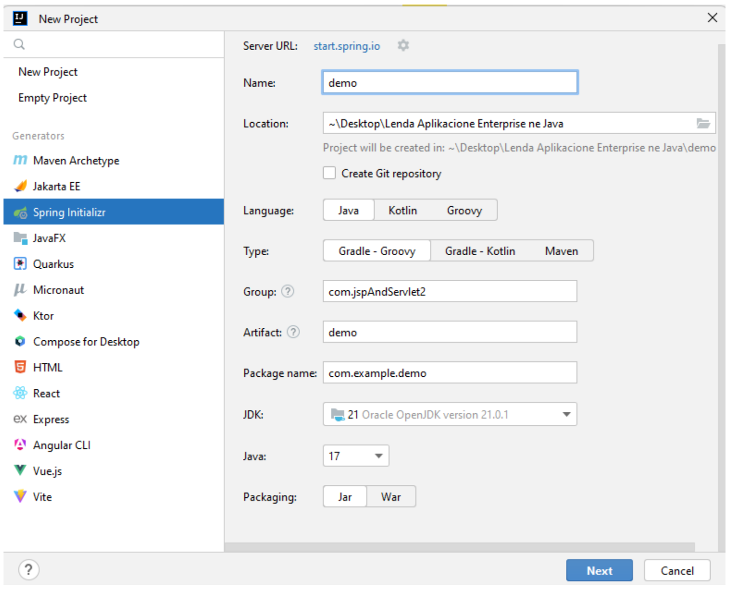
ADD DEPENDENCIES... CTRL + B

Starter Projects

8

Use the IDE

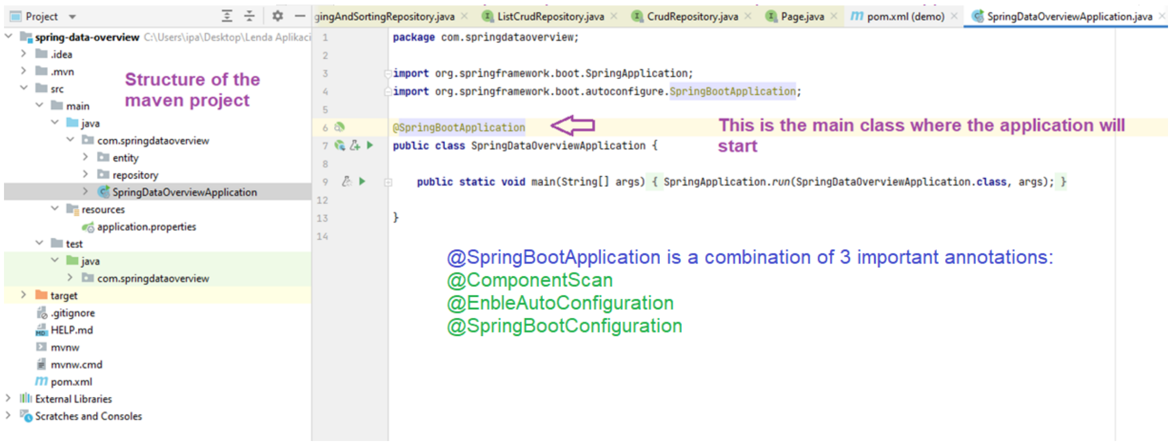
- We can also use the IDE to generate a Spring Boot project.



Server URL: start.spring.io
Name: demo
Location: ~\Desktop\Lenda Aplikacione Enterprise ne Java
Project will be created in: ~\Desktop\Lenda Aplikacione Enterprise ne Java\demo
Create Git repository
Language: Java | Kotlin | Groovy
Type: Gradle - Groovy | Gradle - Kotlin | Maven
Group: com.jspAndServlet2
Artifact: demo
Package name: com.example.demo
JDK: 21 Oracle OpenJDK version 21.0.1
Java: 17
Packaging: Jar | War

9

Spring Boot application - example



```
spring-data-overview C:\Users\ipa\Desktop\Lenda Aplikaci
  .idea
  .mvn
  src
    main
      java
        com.springdataoverview
          entity
          repository
          SpringDataOverviewApplication
      resources
        application.properties
    test
      java
        com.springdataoverview
  target
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
Scratches and Consoles
```

Structure of the maven project

```java
package com.springdataoverview;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringDataOverviewApplication {

    public static void main(String[] args) { SpringApplication.run(SpringDataOverviewApplication.class, args); }

}
```

This is the main class where the application will start

@SpringBootApplication is a combination of 3 important annotations:
@ComponentScan
@EnbleAutoConfiguration
@SpringBootConfiguration

10

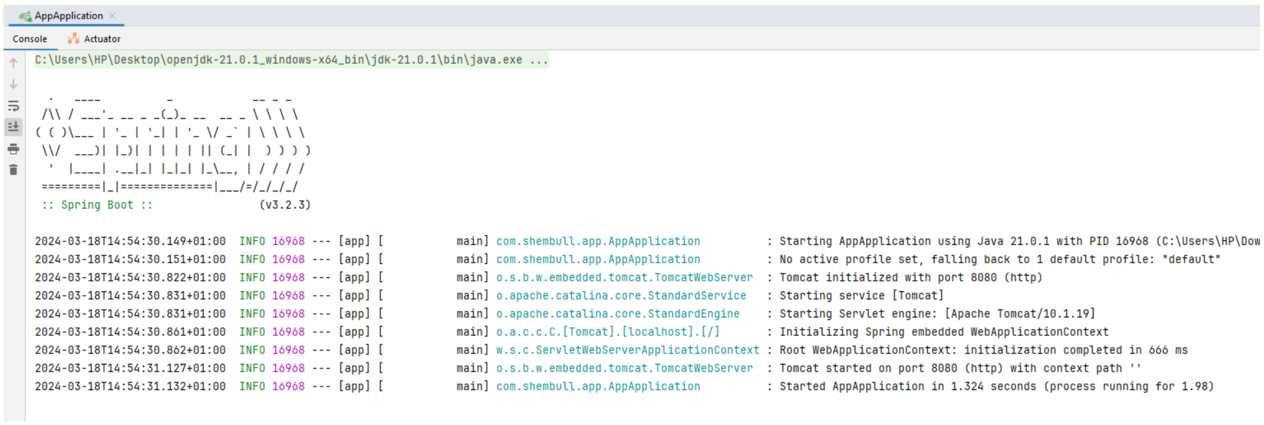
@SpringBootApplication

- Annotations included in @SpringBootApplication:
  - @SpringBootConfiguration:
    - This annotation is used to mark a class as a configuration class in a Spring Boot application.
    - It's equivalent to annotating a class with **@Configuration**.
  - @EnableAutoConfiguration:
    - This annotation enables Spring Boot's auto-configuration mechanism.
    - Spring Boot auto-configuration automatically configures beans and infrastructure based on the dependencies present in the classpath.
    - It's responsible for configuring various features such as DataSource, JPA, Security, and others based on the dependencies detected in the project.
  - @ComponentScan:
    - This annotation enables component scanning in the specified package and its sub-packages.
    - Component scanning automatically detects and registers Spring components such as **@Component, @Service, @Repository,** and **@Controller** beans.
    - It allows Spring to find and instantiate beans without explicitly declaring them in configuration classes.

11

Run the application

- There is no need to create the dispatcher servlet => it is created automatically when the application is started.
- There is no need to deploy the application.
- There is no need to start the server.

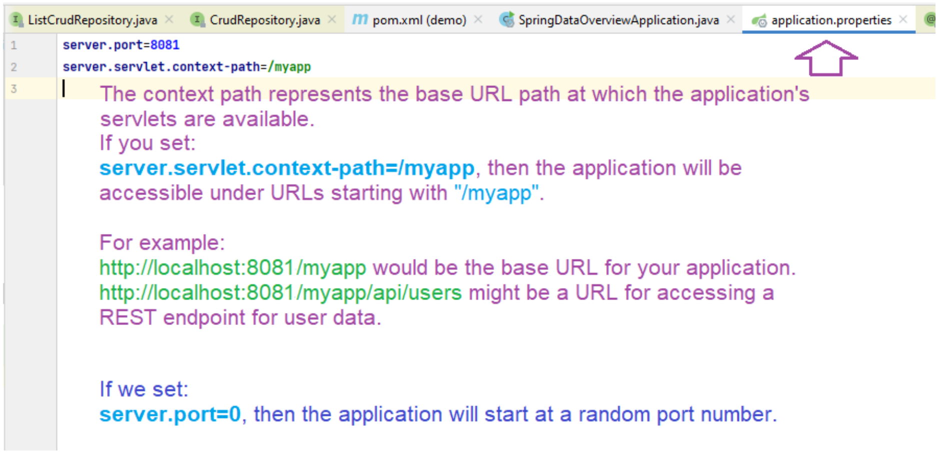


```
C:\Users\HP\Desktop\openjdk-21.0.1_windows-x64_bin\jdk-21.0.1\bin\java.exe ...

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.2.3)

2024-03-18T14:54:30.149+01:00  INFO 16968 --- [app] [           main] com.shembull.app.AppApplication          : Starting AppApplication using Java 21.0.1 with PID 16968 (C:\Users\HP\Dow
2024-03-18T14:54:30.151+01:00  INFO 16968 --- [app] [           main] com.shembull.app.AppApplication          : No active profile set, falling back to 1 default profile: "default"
2024-03-18T14:54:30.822+01:00  INFO 16968 --- [app] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2024-03-18T14:54:30.831+01:00  INFO 16968 --- [app] [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-03-18T14:54:30.831+01:00  INFO 16968 --- [app] [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.19]
2024-03-18T14:54:30.861+01:00  INFO 16968 --- [app] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-03-18T14:54:30.862+01:00  INFO 16968 --- [app] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 666 ms
2024-03-18T14:54:31.127+01:00  INFO 16968 --- [app] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path ''
2024-03-18T14:54:31.132+01:00  INFO 16968 --- [app] [           main] com.shembull.app.AppApplication          : Started AppApplication in 1.324 seconds (process running for 1.98)
```

12

Changes in the default configurations

- The default configurations can be overwritten… the configurations should be added in the **application.properties** file.
  - Example:



```
server.port=8081
server.servlet.context-path=/myapp
```

The context path represents the base URL path at which the application's servlets are available.
If you set:
server.servlet.context-path=/myapp, then the application will be accessible under URLs starting with "/myapp".

For example:
http://localhost:8081/myapp would be the base URL for your application.
http://localhost:8081/myapp/api/users might be a URL for accessing a REST endpoint for user data.

If we set:
server.port=0, then the application will start at a random port number.

13