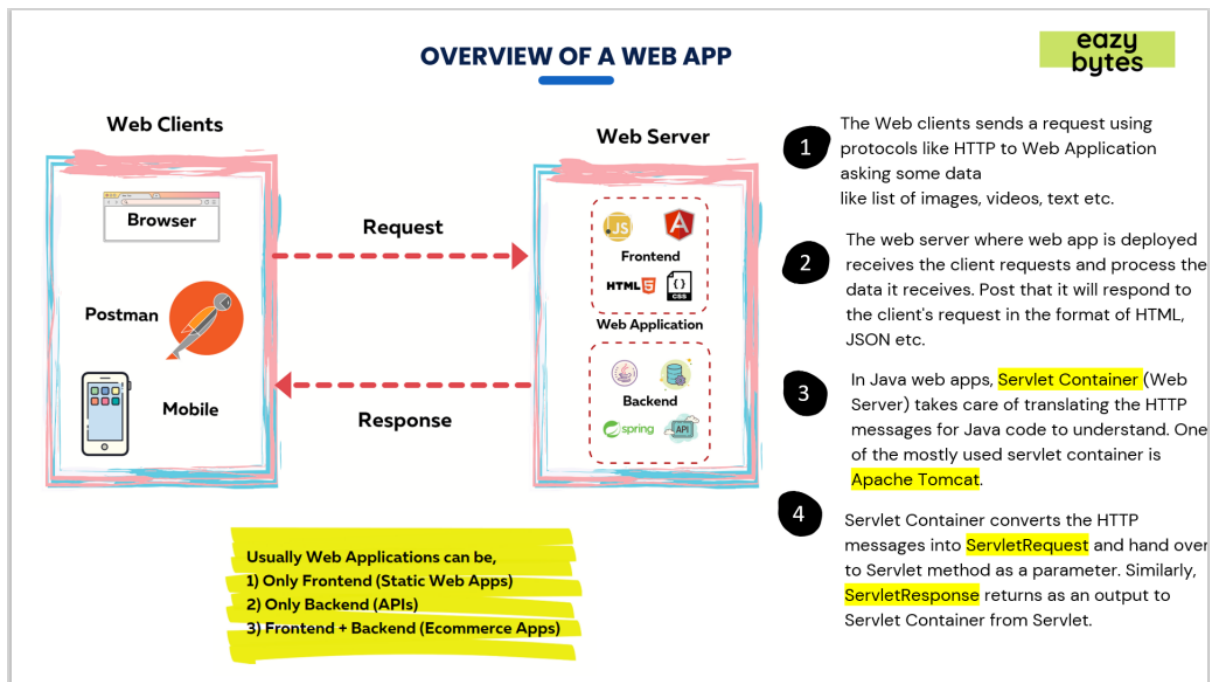
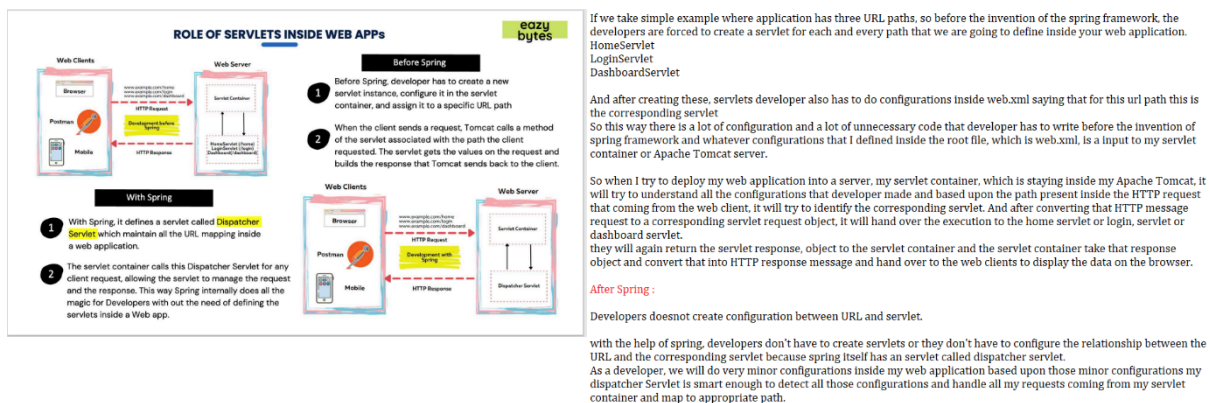


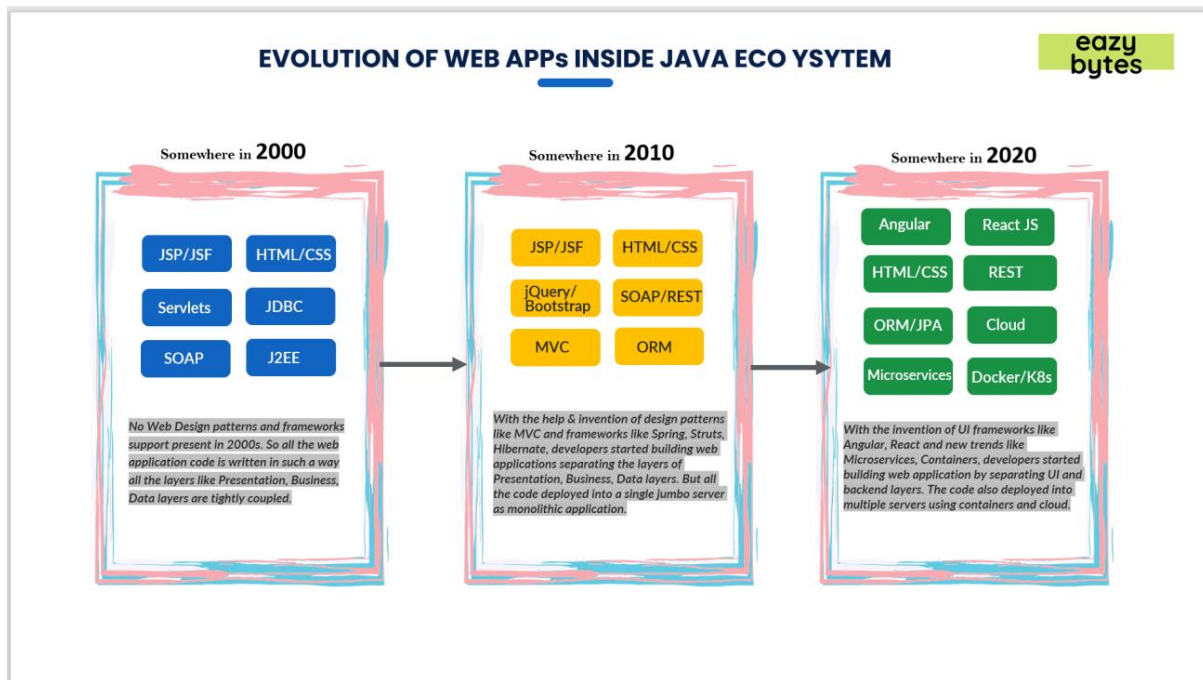
61: Quick Introduction about web Applications



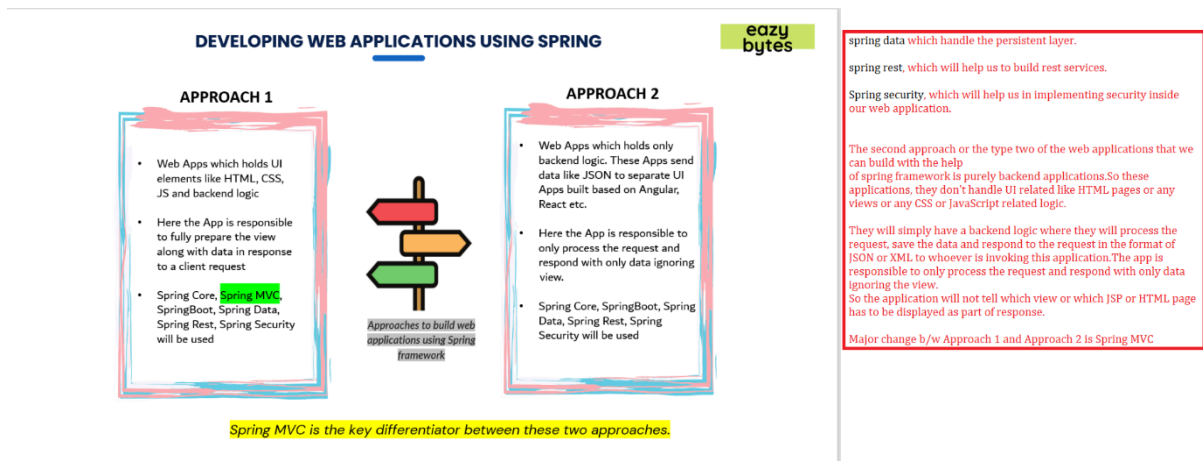
62: Role of Servlets inside the web Application:



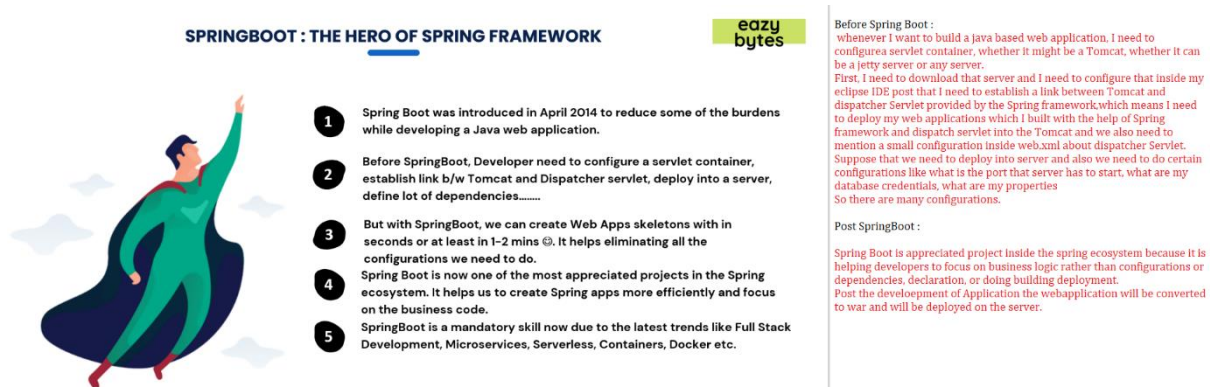
63: Evolution of Web Applications inside Java Eco System.

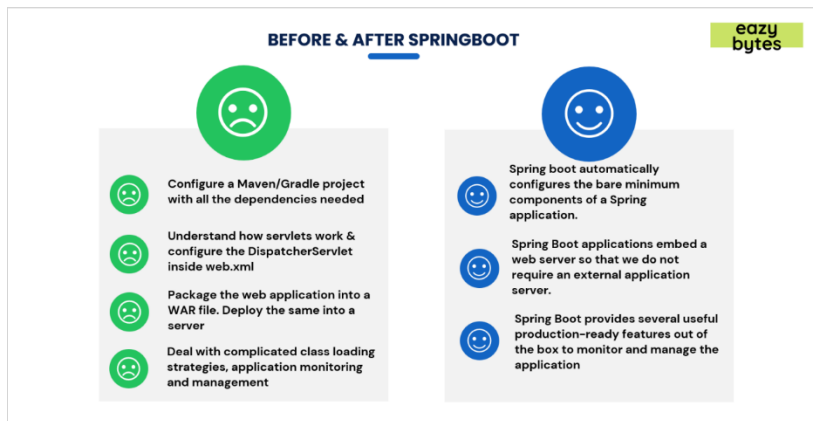


64: Types of Web Applications we can build using Spring



65: Introduction to Spring Boot - The Hero of Spring Framework

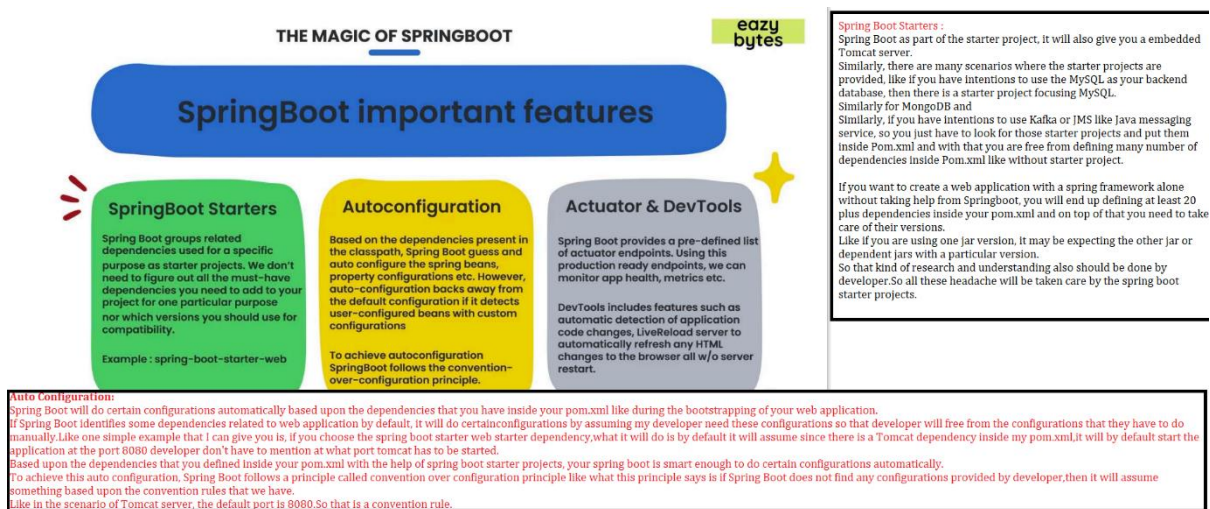




Once the Spring Boot came into picture we will be specifying which application we are developing then Spring Boot will identify the libraries that you need and automatically configure them. Spring Boot also provides an embedded web server like Tomcat, which means you don't have to download or configure a separate server for your web application.

Spring Boot will do all the magic internally and it will deploy the web application into an embedded Tomcat server. Spring Boot provides all the endpoint URLs needed for me where I can simply look at them to understand the health and status of my application at any point of time.

66:Spring Boot Important Features:



67:Creating Simple Web Application using Spring Boot:

we can select the SpringBoot starter project by using the add dependencies option in the start.spring.io. Based on the dependencies provided we can generate the web application.

By default for all SpringBoot Applications spring-boot-starter-test dependency is going to be added by the framework and this dependency is related to the unit testing.

Under test folder :we can write the unit test case or Junit related logic.

We will have @SpringBootApplication which indicates the main method of the program.

Resources:

static :

images,css,java script files.

template :

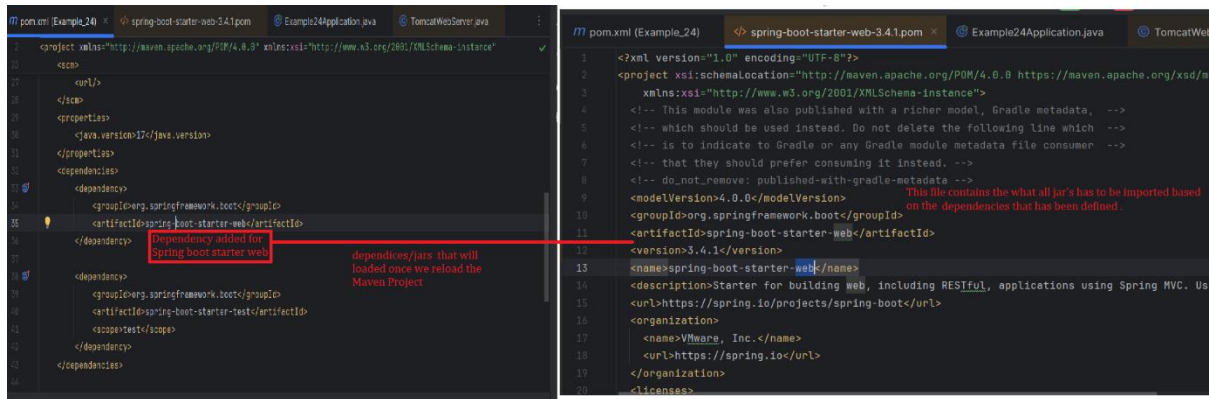
we can maintain any template files related to the UI logic.

application.properties :

we can define any properties that control the behaviour of your web application.

68: Running Spring Web Application using Spring Boot:

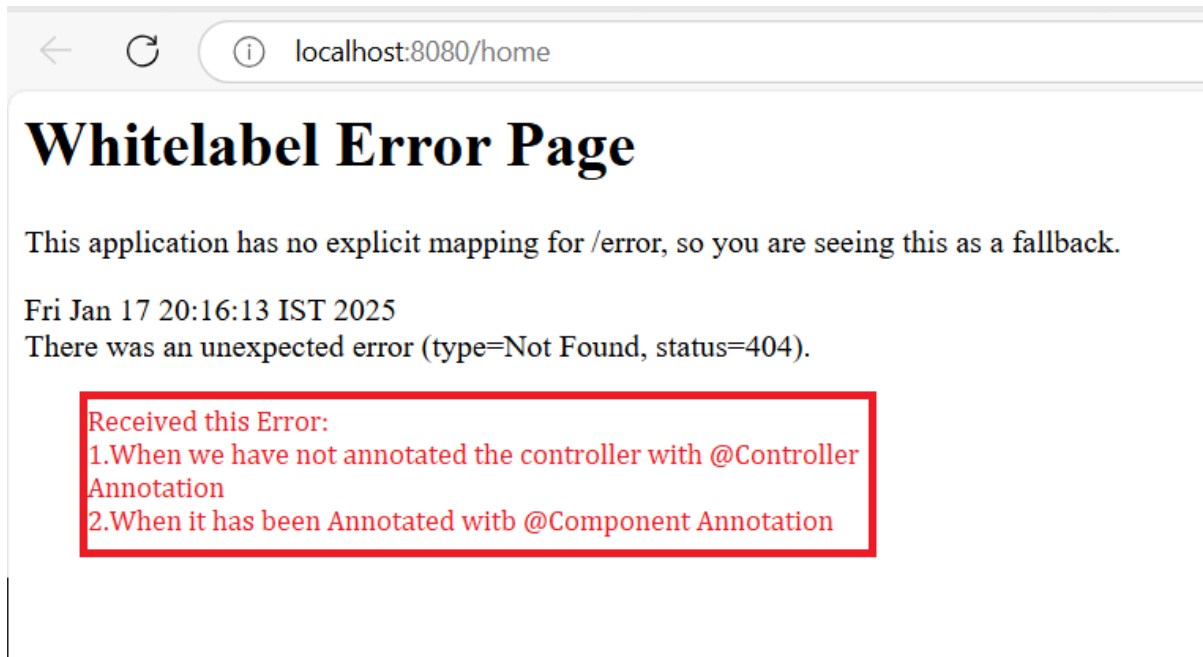
Once we define the dependencies all the corresponding jar's will be downloaded.



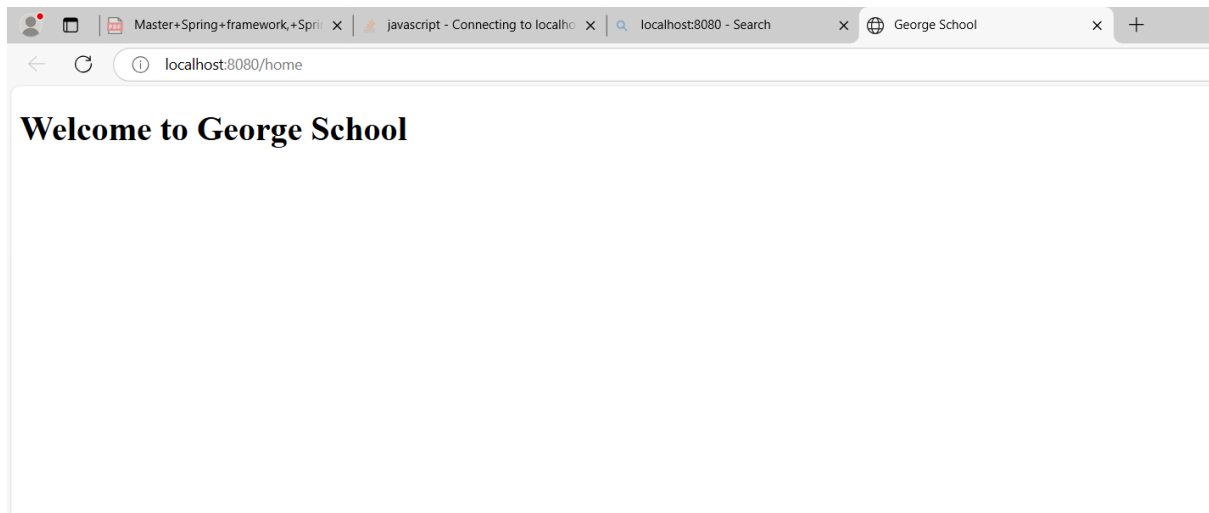
Spring boot will have its own version and spring core will have different versions.

@SpringBootApplication:

Class which has above is the class from which the application starts/initializes.



If the data is not updated even after the updating the files clear the cache and try.



@SpringBootApplication: It is combination of below three Annotations

@SpringBootConfiguration

@EnableAutoConfiguration--To Enable all the default Configurations.

@ComponentScan-Scans all the packages and manages the beans

How Spring Boot will be aware that there is HTML file under static folder?

By default spring boot look for the static files under the static folder so you don't have to do any extra configuration mentioning to your spring boot web application about the static folder location.

Until unless we want to use a different folder location, we should be good with this default configurations without providing any extra configurations to your web application.

home.html --This is a view to your web application, but how do you invoke that page when you invoke that path in the browser. Spring Application has to understand that it needs to re-direct that request to home.html

@Controller :

Indicates that this particular class serves the role of controller.

In the example here if some one is trying to access /home path they need to be re-directed to home.html

@RequestMapping with the path value /home(@RequestMapping("/home"))

This indicates to my spring web application if anyone is trying to access /home path inside my web application, they need to be redirected to this controller with the name HomeController and inside this method, which is displayHomePage() and this method, as of now, it is not doing anything except to return the html name that it needs to be displayed on the browser.

Internally Spring Boot is going to create a Dispatcher Servlet. So whenever someone is trying to invoke /home path, your Dispatch Servlet knows there is a controller inside my web application with the name HomeController and it has a method which is handling the path /home and it will re-direct that requests to that path.

```
2025-01-17T20:36:54.658+05:30 INFO 22956 --- [Example_24] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.34]
2025-01-17T20:36:54.761+05:30 INFO 22956 --- [Example_24] [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-01-17T20:36:54.763+05:30 INFO 22956 --- [Example_24] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
in 1723 ms
2025-01-17T20:36:55.349+05:30 INFO 22956 --- [Example_24] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path
 '/'
2025-01-17T20:36:55.360+05:30 INFO 22956 --- [Example_24] [main] c.c.g.GeorgeSchoolApplication : Started GeorgeSchoolApplication in 3.167 seconds
```

Example_24:

69: Changing the default server port and context path of Spring Application:

Default port:8080:

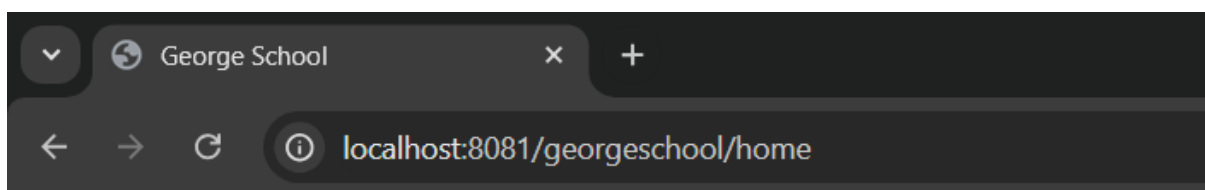
Context path:"

We can use application.properties to override the existing configuration and to define our own custom configurations.

Default configurations of SpringBoot can be overridden by mentioning our own custom configurations in application.properties.

```
2 server.port=8081
3 server.servlet.context-path=/georgeschool
```

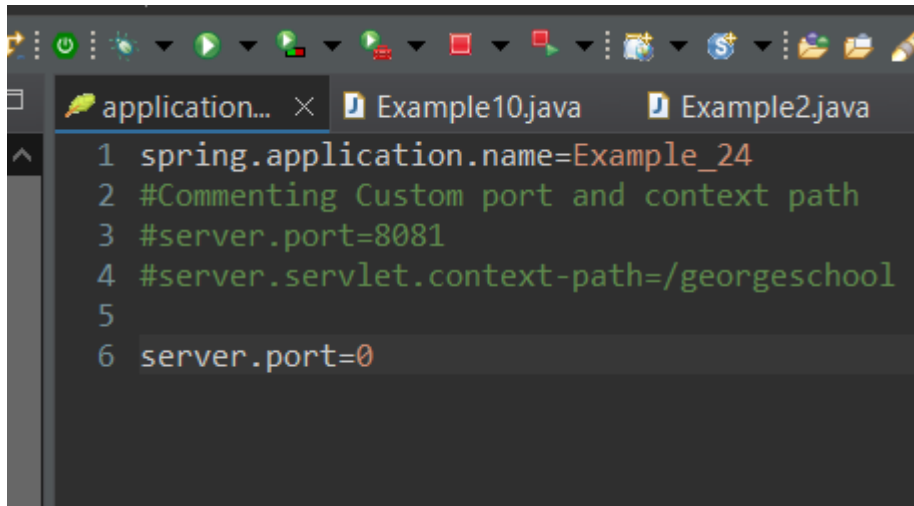
```
main] c.c.g.GeorgeSchoolApplication : No active profile set, falling back to 1 default profile: "default"
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8081 (http)
main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.34]
main] o.a.c.c.C.[.localhost].[/georgeschool] : Initializing Spring embedded WebApplicationContext
main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1326 ms
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8081 (http) with context path '/georgeschool'
main] c.c.g.GeorgeSchoolApplication : Started GeorgeSchoolApplication in 2.71 seconds (process running for 3.675)
8081-exec-1] o.a.c.c.C.[.localhost].[/georgeschool] : Initializing Spring DispatcherServlet 'dispatcherServlet'
8081-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
8081-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 4 ms
```

A screenshot of a web browser window. The address bar shows the URL 'localhost:8081/georgeschool/home'. The page title is 'George School'. The browser interface includes back, forward, and refresh buttons, as well as a search icon.

Welcome to George School

We could see in the above image that Spring Boot Application is running on port 8081 with context path as 'georgeschool'.

70:Random server port number inside Spring Boot



```
1 spring.application.name=Example_24
2 #Commenting Custom port and context path
3 #server.port=8081
4 #server.servlet.context-path=/georgeschool
5
6 server.port=0
```

Indicates that Spring Boot Application should start at random port everytime.

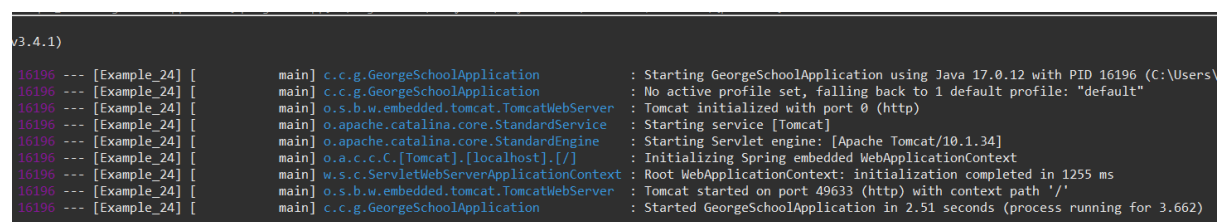
There might be a scenario where you may want to run the same application multiple instances like your automation tester or your automation team, they want to execute certain scripts to test various scenarios inside your web application and in those scenarios may be they want to run multiple instances of your web application.

But if you try to run the application with the same port number, obviously you will get an error that port is already being used by another web application.

So to avoid those kind of scenarios, we can configure the port value as zero, which indicates to the spring boot framework that the port number has to be assigned randomly.

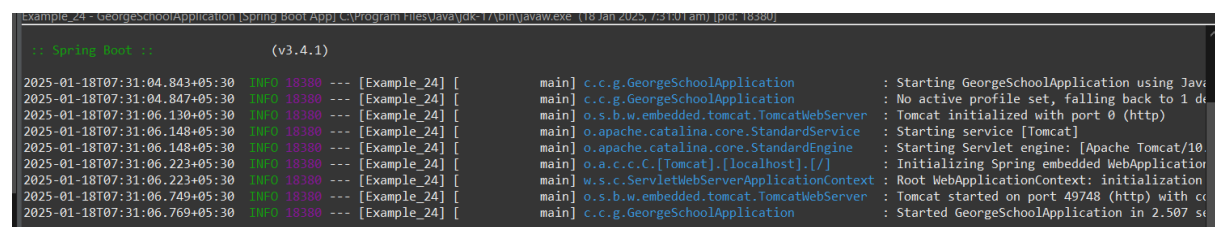
Setting the serve.port=0 indicates the springboot application that it needs to start at a random port at run-time.

First instance



```
v3.4.1)
16196 --- [Example_24] [main] c.c.g.GeorgeSchoolApplication : Starting GeorgeSchoolApplication using Java 17.0.12 with PID 16196 (C:\Users\
16196 --- [Example_24] [main] c.c.g.GeorgeSchoolApplication : No active profile set, falling back to 1 default profile: "default"
16196 --- [Example_24] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 0 (http)
16196 --- [Example_24] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
16196 --- [Example_24] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.34]
16196 --- [Example_24] [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
16196 --- [Example_24] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1255 ms
16196 --- [Example_24] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 49633 (http) with context path '/'
16196 --- [Example_24] [main] c.c.g.GeorgeSchoolApplication : Started GeorgeSchoolApplication in 2.51 seconds (process running for 3.662)
```

Second instance



```
Example_24 - GeorgeSchoolApplication [Spring Boot App] C:\Program Files\Java\jdk-17\bin\javaw.exe (18 Jan 2025, 7:31:01 am) [pid: 18380]
:: Spring Boot :: (v3.4.1)
2025-01-18T07:31:04.843+05:30 INFO 18380 --- [Example_24] [main] c.c.g.GeorgeSchoolApplication : Starting GeorgeSchoolApplication using Java
2025-01-18T07:31:04.847+05:30 INFO 18380 --- [Example_24] [main] c.c.g.GeorgeSchoolApplication : No active profile set, falling back to 1 default profile: "default"
2025-01-18T07:31:06.130+05:30 INFO 18380 --- [Example_24] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 0 (http)
2025-01-18T07:31:06.148+05:30 INFO 18380 --- [Example_24] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-01-18T07:31:06.148+05:30 INFO 18380 --- [Example_24] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.34]
2025-01-18T07:31:06.223+05:30 INFO 18380 --- [Example_24] [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-01-18T07:31:06.223+05:30 INFO 18380 --- [Example_24] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1255 ms
2025-01-18T07:31:06.749+05:30 INFO 18380 --- [Example_24] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 49748 (http) with context path '/'
2025-01-18T07:31:06.769+05:30 INFO 18380 --- [Example_24] [main] c.c.g.GeorgeSchoolApplication : Started GeorgeSchoolApplication in 2.507 seconds (process running for 3.662)
```

71: Demo Spring Boot Auto Configuration

Spring Boot, while starting your web application, it will look at what dependencies you have by looking at the pom.xml. So based upon the dependencies that you define, it will try to create many beans and configurations that you may need, while you are developing web applications.

To check the auto-configuration details we need to set debug=true post that spring boot framework is please go ahead and start the web

application in a debug mode. we can see debug statements on your console, which you will also have the AutoConfiguration details.

Once we run the application we will be able to see the Debug Summary Report

CONDITIONS EVALUATION REPORT

Positive Matches in the report:

Indicates that if it found the dependencies corresponding to that in pom.xml then auto-configurations will be made for that.

Negative Matches indicates that Spring Boot did not create these beans or configurations because it didn't identify any dependencies related to them.

Exclusion list: we can consider in the scenarios where I don't want the spring boot to create the bean. Maybe I want to create that manually by considering special or custom scenarios that I have. so In those scenarios you can mention that exclusion list.

Unconditional Classes:

unconditional classes indicates that these classes and the beans around them are created regardless of what dependencies that you have defined, because these unconditional classes are the base classes that are needed for your spring boot application to run.

To Mention the Exclusion list:

We can mention it with @SpringBootApplication Annotation:

```
@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class}) -- Here  
DataSourceAutoConfiguration will be considered as Exclusion list and it will not be managed by Spring  
boot framework.
```



```
7 @SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})
8 public class GeorgeSchoolApplication {
9
10     public static void main(String[] args) {
11         SpringApplication.run(GeorgeSchoolApplication.class, args);
12     }
13 }
```

Problems Servers Terminal Data Source Explorer Properties Console × Git Staging

Example_24 - GeorgeSchoolApplication [Spring Boot App] C:\Program Files\Java\jdk-17\bin\javaw.exe (18 Jan 2025, 12:20:20 pm) [pid: 18844]

Exclusions:

org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration

We could see in the above image as the Class has been added into Exclusion list .So it will not be managed by the Spring boot framework .We could see the same in the Exclusion list

@SpringBootApplication:

It is combination of three important Annotations

@EnableAutoConfiguration --- Telling to the spring boot framework, go ahead and do the auto configuration for all the dependencies that we have defined inside pom.xml and spring boot configuration

@ComponentScan --ComponentScan will help us to tell spring framework to go ahead and scan all the packages that we need for the beans that should be created with the help of stereotype annotations like @Component, @Controller or @Service.

If other classes are in the same package or subpackage of where the main method/@SpringBootApplication Annotation is Present then we doesn't have to explicitly specify the component scan.Spring boot will create the beans for them but they should have been annotated with Corresponding Annotations(@Service,@Repository,@Controller,@Component,,etc).

@SpringBootConfiguration

if your package is staying somewhere outside of this main package where this spring boot main application class is staying, then definitely you need to define that package details with the help of @ComponentScan Annotation.

72.Quick Recap:

GETTING STARTED WITH SPRINGBOOT

Quick Recap

1

<https://start.spring.io/> is the website which can be used to generate web projects skeleton based on the dependencies required for an application.

2

We can identify the Spring Boot main class by looking for an annotation `@SpringBootApplication`.

3

A single `@SpringBootApplication` annotation can be used to enable those three features, that is:

- `@EnableAutoConfiguration`: enable Spring Boot's auto-configuration mechanism
- `@ComponentScan`: enable `@Component` scan on the package where the application is located
- `@SpringBootConfiguration`: enable registration of extra beans in the context or the import of additional configuration classes. An alternative to Spring's standard `@Configuration` annotation

GETTING STARTED WITH SPRINGBOOT

Quick Recap

- 4 The `@RequestMapping` annotation provides “routing” information. It tells Spring that any HTTP request with the given path should be mapped to the corresponding method. It is a Spring MVC annotation and not specific to Spring Boot.
- 5 `server.port` and `server.servlet.context-path` properties can be mentioned inside the `application.properties` to change the default port number and context path of a web application.
- 6 Mentioning `server.port=0` will start the web application at a random port number every time.
- 7 Mentioning `debug=true` will print the Autoconfiguration report on the console. We can mention the exclusion list as well for SpringBoot auto-configuration by using the below config,

```
@SpringBootApplication(exclude = { DataSourceAutoConfiguration.class })
```

Example_24 : Contains all the Code for the this module.