**Spring boot :**

It is spring framework that has

1. Auto configuration based on Jar files added to classpath

@EnableAutoConfiguration

1. Auto Injection dependency resolution

We d not need to use @Autowire we can use construction

1. Embedded Tomcat
2. Actuator for managing End points <https://start.spring.io/>
3. CLI command line interface to run application independent from maven  
   CLI command test it

**What is @SpringBootApplication**

It is combination of

@Configuration   
 @EnableAutoConfiguration  
 @ComponentScan

**Disable a service a or class?**@SpringBootApplication (exclude=””)

**Disable auto configure class ?**

@enableAutoConfiguration(exclude=”)

Also from application.yml file

Spring.autoconfigure.exclude=

**What is Spring Actuator?**

Managing end point

**How to enable or disable actuator ?   
remove dependcy from maven**

**Spring boot component** :

1. Boot Starter

It is just JAR files that boot uses to do auto-configuration

1. Boot Actuator

Spring boot uses Actuator to manage endpoints.

It is a features to help us monitor and manage application in production

We can choose to monitor your application by using HTTP endpoints or Auditing, health, and metrics gathering can also be automatically applied to your application.

Disable ;

management.endpoint.shutdown.enabled=true

1. Boot CLI (command line interface)
2. Boot initializer

Spring initializer:

The Spring Initializer is a web application that generates a Spring Boot project with everything you need to start it quickly.

<https://start.spring.io/>

1. What is java Swagger ?

It is open source software allowing developers to intarct with APIS

Generate client template for our APIs

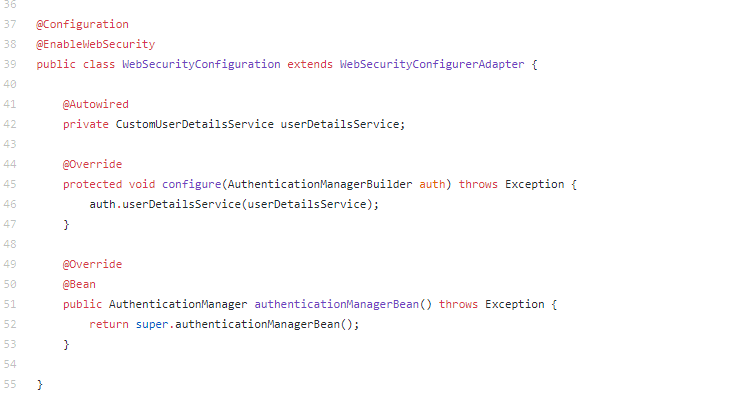
1. Replace or override the Tomcat
   1. Spring-boot-starter-jetty
   2. Spring-boot-starter-undertow
2. Disable Tomcat :
   1. Spring.main.web-application-type = none.

**Spring boot security**

* 1. **Create Config ( implement WebSecurityConifigurerAdapter)**

**It will give us 3 beans**

* + 1. **Filter**
    2. **PasswordEncoder**
    3. **DAoAuthenticatiponPorvider**
    4. **Configurere-> limit access to HTTP methods**

****

Unit Test Spring boot :

1. DAO layer
   1. memory DB such as H2 database
   2. @DataJpaTest that provide TestEntityManager
2. Service layer
   1. We mock persistent layer we do not need it
   2. We use @mockBean on Repository dependency injection

@RunWith(SpringRunner.class)  
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM\_PORT)

https://spring.io/guides/gs/testing-web/

@Before

public void setUp() {

   Employee alex = new Employee("alex");

   Mockito.when(employeeRepository.findByName(alex.getName()))

     .thenReturn(alex);

}

@Test

public void whenValidName\_thenEmployeeShouldBeFound() {

    String name = "alex";

    Employee found = employeeService.getEmployeeByName(name);

     assertThat(found.getName())

      .isEqualTo(name);

 }

Controller Layer test

We mock Service layer using @MockBean

@RunWith(SpringRunner.class)

@WebMvcTest(EmployeeRestController.class)

public class EmployeeRestControllerTest {

    @Autowired

    private MockMvc mvc;

    @MockBean

    private EmployeeService service;

    // write test cases here

}