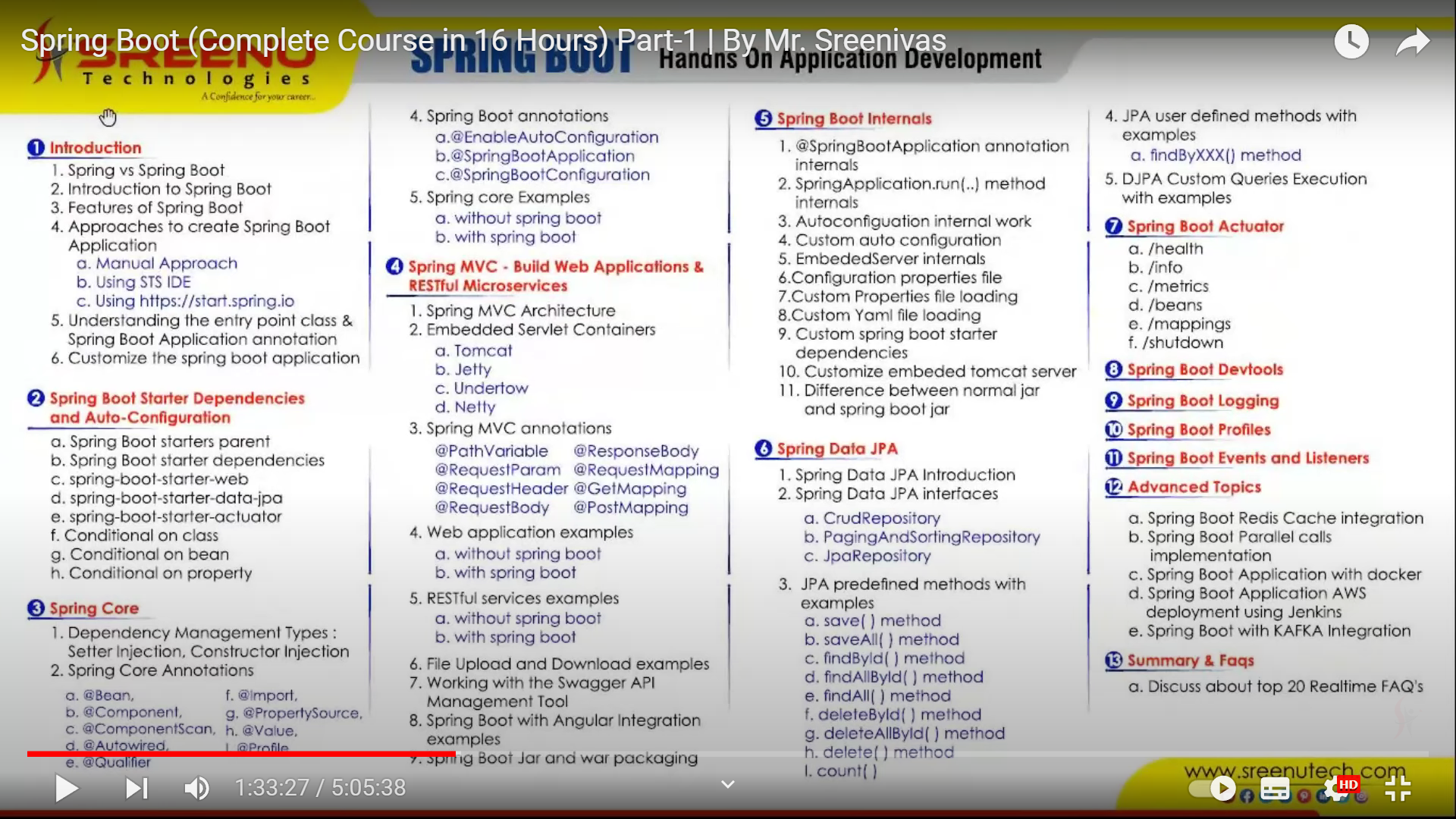
Spring Boot



Introduction:

#1. The main objective of spring framework is to make J2EE application development was easier

#2. The main Objecvive of spring boot is to make Spring application development was easier

**Spring Application Development - Drawbacks:**

1. lot of manual configuration - xml config, java configuration, autowiring & component scan

autowired and componentscan - should work for userdefined classes (we have sourcecode)

framework related classes - DataSource, JdbcTemplate, RedisTemplate...( @Bean)

"Regular spring ligter interms of code but heavy weight with configuration"

-Time consuming

-complex

-understand memorise tags

2.Dependency Management is too hard(If we want to convert spring mvc 3 to hibernate,the hibernate supports which version 2 or 3 not sure)

3.Difficult for new comers to tryout spring features

To overcome all these problems spring was introduced a module called “spring boot”.

**Spring Boot:**

1.Spring Boot doesn't replacement of spring framework. i.e. it is one of the module like spring core, mvc, dac

Spring Boot = all spring modules

2.Using spring we can develop End 2 End Applications, but it will take lot of time to deliver the product

1. standlone applications

2. web applications 3. distributed applications ..etc

3.Whatever regular spring framework is doing, same will do by spring boot but to deliver the product quickly into market with the the help of spring boot features

4.Spring Boot is new way of creating spring based applications

5.Spring boot aims to simplify the process of developing production ready spring based applications

6.SpringBoot Latest version – 5.x ,SpringBoot version – 3.x,Industry using version -2.5.x(2.5.9)

**Spring Boot Features:**

#1. Dependency Management is easier #2. Auto Configuration #3. Embeded Server

#4. Actuator - Address all non functional requirements like health, monitor, metrics ..production ready/deployable features

#5. Dev tools

#6. Spring Application without xml configuration

#7. Opinionated but highly customizable

#8. Spring Boot CLI

**1.Dependency Management:**

If we want to develop web applications we need many jars

Like spring -core 4.x,spring context 4.x, hibernate 3.x,Jackson,validator

In order to overcome that, spring boot introduced starter dependency(starter dependency will take care of all dependencies for ex starter web will take care of versions to add for developing web applications)

**1. spring boot parent starter pom**

**2. spring boot starter <feature>**

**1. spring boot parent starter pom:**

>spring would integrate with so many third party technologies like mongodb, redis cache, kafka,..etc

>It will configure all the spring modules and third party libraries version information

**spring boot 1.x --> spring boot starter pom --> spring core-3.x, spring mvc 3.x, jackson 1.x..ete**

**} i.e., so many libraries version information**

**-->Spring boot starter dependency is used to bring all the required dependencies of that features**

web application -- **add spring-boot-starter-web** -- it will bring all the required dependencies

to develop web application

jdbc -- **add spring-boot-starter-jdbc** -- it will bring all the requited jars to develop jdbc application

spring-boot-starter-parent pom - is used to provide the information of all the libraries

spring-boot-starter-feature dependency - is used to get the required libraries to develop that

feature i.e. developer no need to add all the libraries, just add only

one starter dependency

2.Autoconfiguration:

configuration - configure the beans using xml or java config or autowired with component

\*there are some drabacks on XML config, we moved to java config or autowired with component

\*autowrire is used to to reduce/remove the configuration either in xml config or java config

\*eventhough we are using autowire but still we are depending on manual configuration

ie. we cant apply @Component for all the classes

1. to remove @Bean or <bean> configruation we are using @Component but we can’t apply @Component for all the classes

2. We can apply @Component only which we have source code but can't apply for framework

clases like DataSource, JdbcTemplate, RestTemplate, MongoTemplate

3.If we want use framework class then we should configure them as either @Bean or <bean> element

@configuration

public class JavaConfig{

@Bean

public DataSource datasource(){

return new DriverManagerDataSource();

}

@Bean

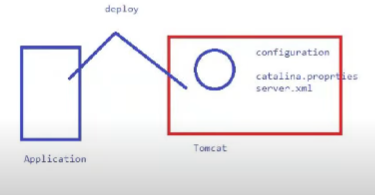
public JdbcTemplate jdbcTemplate(){

return new JdbcTemplate();

}

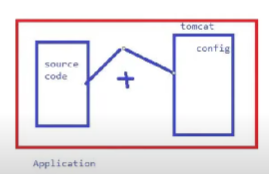
**Instead of writing all these configuration spring boot will enable auto configuration with the help of @SpringBootApplication(**Springboot will remove all the configuration code**)**

**3.Embedded Server:**



In traditional method we will deploy the application from outside into server like tomcat.(Inside tomcat some properties like Catalina.properties,server.xml will be present)

But in SpringBoot Application..inside SpringBootApplication will have Application source code and tomcat server(configuration) is present. During Application Start up, our application start and whole will be deployed.This is called **Embedded Tomcat Server.**



server inside the application is called embedded server

->Spring boot application itself having server is called embedded server

->This more useful in cloud

**4. Actuator** : It is used to address all non functional requirements like health check, monitor, metrics, beans etc. This is production ready features.

**5. Dev tools :**

-->It is used to improve the developer productivity

-->If any changes done by the developer, dev tools will take care compile, regenerate jar, deploy

appn into server automatically

**6. spring boot without xml configuration**

7. opinionate but highly customizable :

spring boot has provided default version, default required libraries, but still we can

use our own dependencies

8. spring boot CLI : command line interface, used to quick application development from command line interface

**Summary on Spring Boot features :**

Developer should not waste their time by spending on manual work like

a. manual configuration b. manual dependencies c. setup monitor application

d.deploy application into external server e. redeployment for every changes.

->Spring boot will take care all the above things and ask to developer focus on only coding

**How to create Spring boot application** :

1. Using Eclipse IDE 2. Using http: /start.spring.io 3. Using STS IDE

**1.Using Eclipse IDE :**

🡪Developer need add the dependencies manually in pom.xml

🡪spring configuration need to add manually, i.e. ide will not help predefined confiuration

**2. Using https:/start.spring.io :**

🡪developer will choose all the required dependencies, plugins. .etc

🡪It will generate the zip file and download it

🡪we can unzip and import into eclipse ide

**3.Using STS IDE:**

https://start.spring.io/starter.zip?nam&groupId=com.sreenutechartifactId=SpringBootDemoApp&version=1.@SNAPSHOT&description=Demo+project+for+Spring+Boot&packageName=com.sreenutech&type=mavenproject&packaging=javaversion&bootVersionSpringBcotDemoApp15.9. RELEASE



Spring core:

🡪SpringBoot used to develop end 2 end applications(We can develop Controller,can develop DAO, can develop BusinessLayer )

-->using spring boot we can develop standlone application , web application, distributed application

a. standlone :ApplicationContext context = new AnnotationConfigApplciatoncontext (JavaConfig);

b. webapplication : new AnnotationConfigServletWebServerApplicationContext (JavaConfig);

c. reactive application(both spring and web) : AnnotationConfigreactiveWebServerApplicationContext (JavaConfig);

**Dependency Injection:**

1. Setter Injection 2. Constructor Injection 3. Field level Injection (possible only with autowired)

if we use spring core then, developer responsible to

a. write configuration(use @Component & Component scan)

b. create the ioc container

**how to load the properties file - @PropertySource("classpath: filename")**

**how many ways read the properties file**

**1. Environment Object 2. @Value annotation**

**Q) What will happen when SpringApplication.run() will execute?**

1. Spring Application constructor will take care to identify which container(standalone,web or reactive ) is required

deduceFromClasspath() method will check in classpath, based on classes find in

class path then corresponding spring container will be created

2. create a empty envionrment object?

3. detects the configuration for our application and loads into environment Object

(application.properties/application.yaml)

**SpringBootApplication:**

It is entry class for the Application.If this annotation not mentioned at starting, throws error called No such bean definition exception, no qualifying bean of type A available.

**Q) What will happen when Spring Application. run() will execute?**

1. creates an empty environment object

2. detects external configuration of our application and loads into environment object

3. print boot banner

4. identifies type of WebApplicationType

a. if spring mvc jars found in the classpath then treat the WebApplicationType=wEB

and it creates AnnotationConfigServiethebServerApplcationContext

b. if spring webflux jars are found then treat the WebApplicationType=Reactive then

and creates AnnotationConfigReactiveliebServerApplcationContext

c. if none of the above then treats the WebApplicationType=NONE then it creates

AnnotationApplicatonContext Object will be created

5. Instantiate the spring factories and register with ioc container

6. executes ApplicationContextIntializer //it will detect all the configuration

7. prepareContext

8. refreshContext // all the beans will be stored in container

9. during the above stages, it will publish various different type of events and invokes listeners to perform operation

**How the autoconfiguration will work?**

Sol : @SpringBoctApplication annotation is responsible for enabling spring boot auto confiugration

@SpringBootApplication = @ComponentScan + @EnableAutoConfiguration +

@SpringBootConfiguration

1. @SpringBootConfiguration : ( used for configuration)@Configuration

>This is like spring configuration class

>just for naming convention spring boot uses @SpringBootConfiguration

2. @Componentscan : //it is used to scan the our project source code @Component classes like A,B

based on the specified classes, packages, if not specified default is base package

>It will scan the component class at package level, class level

>default package name is current package

>It is used to scan userdefined packages

ex : @ComponentScan(basepackage=” com. sreenutech. service")

@ComponentScan(basepackage = “com. sreenutech. controller")

@ComponentScan(basepackage=” com.sreenutech") //it will scan com.sreenutech and child packages also

[3.@EnableAutoConfiguration](mailto:3.@EnableAutoConfiguration) :

//It is used to create the framework(In sts proj expand META-INF u will see some claases like jdbc,Jackson etc)/predefined classes with the help of spring. factories file

🡪It is used to enabled autconfiguration for the framework related classes

Spring -boot- starter.jar

|

/META-INF/spring.factories

--------------------------------------------------

**How to load the properties file ?**

1. In spring boot properties filename should be either application.propertes or application.yml file

2. spring boot will load these properties during application startup and no need to use @PropertySource

3. if we want load custom/our own properties file then we should load using @PropertySource annoation

4. If we want load custom/our own yml file then either load @PropertySource() annotation or

1. convert yml file into properties file

2. load the porperties files using @PropertySource annotation

Configuration Properties

application.properties

book.isbn=89778

book.author=sreenu

book.title=spring boot in action

book.price=400

public class BookController{

@Value(“${book.isbn}”)

:

:

}

No of properties =No of @Value annotations

@Component

@ConfigurationProperties(Prefix=”book”) is used to read properties

@PropertySource(“classpath:products-details.yml”)

Spring Actuator:

It is a feature of Spring Boot with the help of which you can see what is happening inside a running application. So, whenever you want **to debug your application**, and need to **analyze the logs,access production ready rest points** .

-------------------------------------------------------------------------------

**How to read type based properties file or**

**How to read the properties without @Value Annotation**

no.of properties = no.of @Value annotations

Inorder to inject the depedent values into BookController then we have to write @Value annotation on each property of the class, if the no.of attributes more then it takes time to write the annotations an inject values

To overcome this problem spring boot has introduced ConfigurationPropertiesBeanPostProcessor, this will take care to create the bean definition, so that it would be automatically register with ioc container and willbe invoked for each bean definition

**we should use 2 annotations**

1. @EnableConfigurationProperties- this is used to create the ConfigurationPropertiesBeanPostProcessor book")

2. @ConfigurationProperties(prefix=”book”)

**spring profiles**

=>In realtiem we have different enviornment like dev,

=>each envionment will have their own confiuration

**db-dev. properties:**

db.driverclassname=com.mysql . jdbc .Driver

db.url=jdbc:mysql://10.0.0.120:3306/db

db.username: root

db. password: root

**db-test. properties:**

db.driverclassname=com.mysql . jdbc .Driver

db.url=jdbc:mysql://10.0.0.130:3306/db

db.username: testuser

db. password: pwd123

**What are the profiles?What is the purpose of it?(@Profile)**

We use profiles to switch from one environment(dev to test,test to dev..)to another environment.

Instead of @Configuration,@PropertySource,@Profile for each env in Spring..We can directly use @Profiles in SpringBoot.

**For dev env**:@Configuration

@PropertySoruce("classpath:db-dev. properties")

@Profile("dev")

public class DevConfig{ }

@Configuration

@PropertySoruce("classpath:db-test. properties")

@Profile("test”)

public class DevConfig{

}

@Configuration

@PropertySoruce("classpath:db-dev. properties")

@Profile("uat”)

public. class DevConfig{}

‘**Spring Boot - no need write any configuration to load profiles**

In realtime spring.profiles.active={envrionmentname} should send dockerfile, cicd pipeline

**Runners :**

If we want perform onetime startup activites activity after ioc container has been created

1. read values from cache

2. some application will display the countries list, state list

1. Commandline Runner (In order to perform startup activities like starting of the application we need to run batch jobs)

2. ApplicationRunner(If we want to see detail implementation)

-----------------------------------

**Using Spring Boot - we can avoid @Bean, @PropertySource, @Profile, @Import annoations**

**We will use @Value, @Component, @Autowired, @Qualifer**

**Spring Core:**

1.how spring boot will proivde the IOC container (AnnotationApplicationContext)

2.DI - CI, SI without configuration **--@SpringBootApplication**(Combination of 3 -: **EnableAutoConfiguration SpringBootConfiguration Component Scan**)

3.How @SpringBootApplication will work

4.what happend when SpringApplication.run() method will be executed?

5.how to load the properties file

6.configuration properties

7.profiles - spring.profiles.active

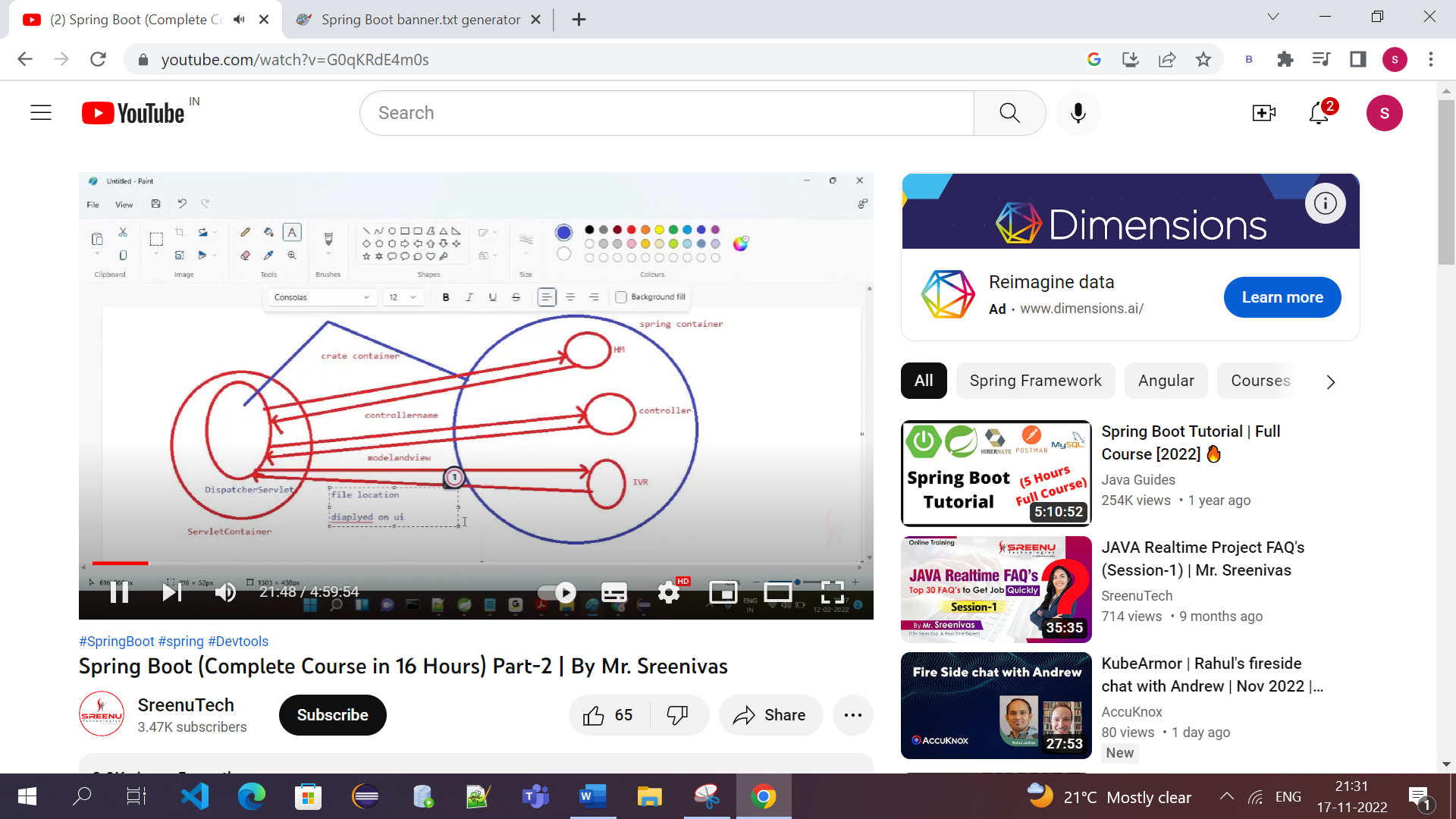
8.runners - Commandline,ApplicationRunner 1

9.customize SpringBootApplications

spring.main.banner-mode=off (If we write in application.properties..while executing SBA Spring boot symbol will not appear)

banner.txt - >google sb banner->give any name->copy banner n paste in banner.txt->run sba





Spring MVC

->Using MVC we can develope web applications and distributed application (Rest API's)

webapplication :

MVC where M-model V-View C-Controller

-->In Spring MVC developer has the responsible to write configurations manually and deploy application

a. configuration - xml or java config

xml - web.xml and dispatcher-servlet.xml

javaconfig - ApplicatoinIntializer extends

AbstractAnnotationConfigDispatcherServletinitializer MvcConfig extends

WebMvcConfigurerAdapter

b. deploy application manually

**Spring Boot**

->No configuration. i.e. no xml and no java configurations

=>no need to deploy application into server

->In spring boot just write controller and add mvc view properties in application.properties file

Q)HoW Springboot will create the DispatcherServiet, HanadlerMapping, ViewResolver..etc

ol : All these object will be created with the help of @EnableAutoConfiguration annotation

)How Autoconfiguration work in spring boot?

ol : Autoconfiguration is implemented with @Configuration class

#1. @ConditionalXxx annotations are used to constraints when autoconfiguration should apply

#2.Usually autoconfiguration classes uses @ConditionalXXX annotations

#3.Location of all autoconfiguration classes

Spring-boot-autoconfigure.jar ->Meta INF ->spring factories

#4. There are 150+ autoconfigure classes in spring.factories file, but in my project i will use

only 4 or 5 classes, then why we should load all the autoconfigure classes

#5. What are the classes are present in my class path and present in my contianer their classes

only create the beans

ie. this can be possible with the help of @ConditonalXX annottaions

#6. DispatcherServletAutoConfiguration class is taking care of creating DispatcherServiet Object

#7. EmbeddedwebServerFactoryCustomizerAutoConfiguration is responsible to create the Embeded Server

based on EmbededServer present in the classpath

#8. In spring boot main() method is the entry point if we use embeded server then no need pakc

war, it should be jar only

#9. SpringAppleation.run() will start the application and intialize the tomcat then deploy the

application into tomcat server

#10.all tomcat server configuration should be keep in application.properties file|

**->In springboot there are 4 embeded servers**

**Tomcat 2. Jetty 3. undertow 4. Netty(to develop Reactive applications)**

**->Default Embeded Server is Tomcat**

->We can override default behaviour

->we dont need to add any additional dependency, it will come along with spring-boot-starter Web

->By default tomcat exposes 8080 port no, we can customize using application.properties file

server. port

Q) How to remove default emebded server?

Sol : By default Tomcat is emebded server which will bring with spring-boot-starter-web

If we want any other embeded server then exlude tomcat and add new embded server

**If we want to use jetty instead of tomcat for running application use this code in pom.xml**

**Exclude Tomcat:**

<exclusions>

<exclusion>

<groupid>org. springframework .boot</grouptd>

<artifactId>spring-boot-starter-tomcat</artifact>

</exclusion>

</exclusions>

**Include Jetty:**

<dependency>

<groupid>org. springframework.boot</groupid>

<artifactId>spring-boot-starter-jetty</artifactid>

</dependency>



If we want register a bean when(Here B is dependent on A, if in A class B is present then only create bean for A)

1. A specific class present in classpath

2. A specific type of bean doesn't already registered in Applicationcontext

3. A specific file exists on location

4. A specific property value is configured in a configuration file

5. A specific System property is present/absent

1, class level conditions – 1 . @Conditionalonclass (Ex : if B is present in classpath)

2. @ConditionalonMissingClass (Ex : if A is present in classpath)

2. bean conditions -- 3. @Conditionalsean (Ex : if B is present in container)

4, @ConditionalonMissingBean(Ex : if A is not present in Container)

3. property conditions --- 5. @CondtionalonProperty

4. Resource Conditions —-- 6. @ConditionalonResource

5. WebApplication Conditions --7. @ConditionalWebApplication

8. @ConditionalNotwWebApplication

6. SPEL expression condition -- 9. @ConditionalOnExpression

-->Springboot default support packaging model is jar file but we can enable packaging as war also

To enable as war, springboot main class ( which is representing with @SpringBootApplication)

should be extends with SpringBootServletInitializer

🡪SpringBootServletInitializer will have cofigure() method which returns SpringApplicationBuilder object

SpringApplicationBuilder class will take care to identify the required container and start the instantiating all the required beans

**Q)What difference between normal jar and spring boot jar?**

normal jar : no main manifest attribute, in SpringBoot-Jar-app-1.0-SNAPSHOT. jar

normal jar : it contains only project related .class files will be there, we can't execute the

the spring boot application with only .class files

To execute the spring boot application we need project .class files + all required jar files +class loaders

**FAT or Uber or Shadded jar**

-->These jars are executable jar which contains classes along with required dependencies

Q)How to make your spring boot application as executable jar?

Sol : Just add spring boot maven plugins in pom.xml, so that that jar would become as fat(executable) jar

<build>

<plugins>

<plugin>

<groupId>org. springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

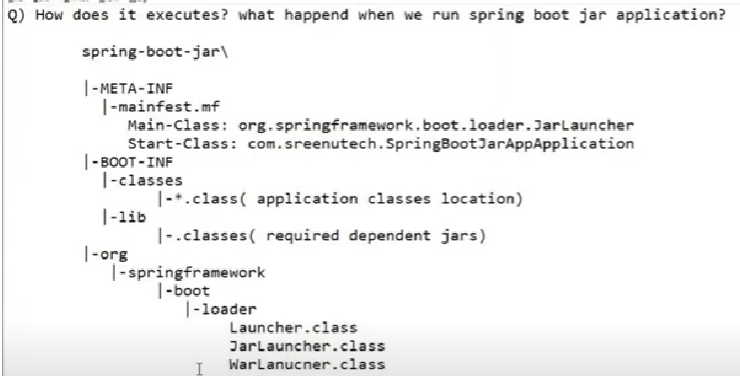
</plugin>

</plugins>

</build>

**how to extract the jar files? how to run the spring boot application?**

**>jar -xvf filename.jar >java -jar filename.jar**



How to implement rest api using spring boot