```
                                      Date : 10-Jul-21
                                      Spring Boot 9AM
                                         Mr. RAGHU
                ---------------------------------------------
                              Spring Boot internals

1. Starter class work flow
> Presetup (@SpringBootApplication)
  a. Readbase package as starter class package name
  b. Load all Input file details with locations
  c. Activate Conditional Annotations

> run() method work flow
 a. Start StopWatch(C) object

 b. Prepare Environment
    > Create Empty Memory
    > Load Option args/VM Args
    > proeprties/YAML Inputs
    > Systems/OS args
    > Convert into Environment memory

 c. Print banner at console

 d. Create Container Reference with Environment setup

 e. Create Objects Required (DataSource, Scheduling,
ViewResolver...etc)
    (Refresh Context)

 f. Execute/Call Runners

 g. Finalize Container Creation or throw exception

 h. Return ApplicationContext
------------------------------------------------------------
2. Conditional Annotations

-> Create object inside Spring container based on some conditions.
 Which is called as AutoConfiguration done using
     @EnableAutoConfiguration on Refresh Context


a. @ConditionalOnProperty
=> if a given key=val is present in Environment memory then only
Create Bean/
   execute Configutation

@Configuration
@ConditionalOnProperty("spring.datasource.url")
//if this key is added then only do this task
class  DatabaseConfiguration {
  ...
}

b. @ConditionalOnExpression : it will execute one expression, if it
```

```
returns
                true then execute task, else no

@Configuration
@ConditionalOnExpression("${spring.jpa.show-sql}") //if true
class  PrintSqlsAtConsole {

}

c. @ConditionalOnBean : If given class object is exist then only allow
                executing logic/creating object

@Configuration
@ConditionalOnBean(DataSource.class)
//is DataSource (impl) class object is created if yes, continue
public JpaConfiguration {

}

d. @ConditionalOnMissingBean :If given class object is not exist then
only allow
                executing logic/creating object


@Configuration
@ConditionalOnMissingBean(DataSource.class)
public CreateDataSourceWork {

}

e. @ConditionalOnResource : If given file/details exist in project
then only
                                implement this , else no

@Configuration
@ConditionalOnResource(resources="bootstrap.properties")
class SetupPropertiesBeforeInputs {
  //...
}

f. @ConditionalOnClass : if given .class exist in classpath(jar is
added to project)
                then only

Ex: spring-boot-starter-jdbc you added then DataSource.class is added
by that jar

@Configuration
@ConditionalOnClass(DataSource.class)
class ConfigureJdbcSetup {
  //...
}



f. @ConditionalOnMissingClass : if given .class is not exist in
classpath
```

```
                (jar is not added to project) then only

@Configuration
@ConditionalOnMissingClass(ConfigNativeClient.class)
class ConfigureExternalClient {
  //...
}


g. @ConditionalOnWebApplication : if application is running under
server
                        then only

@Configuration
@ConditionalOnWebApplication
class OnWebAppConfig {
        ....
}

 @ConditionalOnNotWebApplication : For standalone only


Q) How can we work with other Servers?
A)
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                        <exclusions>
                                <exclusion>

<groupId>org.springframework.boot</groupId>
                                        <artifactId>spring-boot-
starter-tomcat</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
jetty</artifactId>
                </dependency>
```