```
@Autowired
@Component ====> It  is used for UserDefined class to create an object using
Stereotype annotation by the container automatically.
@ComponentScan
@Bean     ======> It is used for PreDefined class to create an Object and code is
written inside Configuration class.
@Controller
@Service
@Repository
@value
@PropertyResource
@Configuration
@Scope
@Import
@Qualifier
@PostConstruct
@PreDestroy
@Lazy
@Primary

Building application using SpringBoot
=====================================
   Step1:: Keep the following software ready
                        => Eclipse IDE with STS plugin(SpringToolSuite)

            To install sts plugin :: Help menu-> Eclipse market place -> search for
sts(3.9.14)
                                              select all -> click on install->
accept terms and conditions -> restart IDE.


=>Plugin is a patch software that provides additional features/functionalities to
existing software.
=>STS plugin makes eclipse to develop spring,spring boot apps very easily ... more
over it brings STS IDE features to eclipse IDE.


@SpringBootApplication
      =>@EnableAutoConfiguration(It enables AutoConfiguration)
      =>@ComponentScan(Scan for the stereo type annotations in the given package
and subpackage)
      =>@Configuration(Marking the class as Configuration class)

SpringApplication.run() internally uses AnnotationConfigApplicationContext class to
create an IOC container by taking java class
as @Configuration class(in fact it takes current class nothing but ClientApp cum
ConfigurationClass)

Note: By default all the components are of Singleton, we can explicitly make it as
other scopes using the anotation called
        @Scope(value="")


Difference b/w Spring vs SpringBoot
===================================
1. Spring
         It is a framework for JEE technologies/Application framework
         The main feature is DependancyInjection and DependancyLookUp.
         It supports XML driven configuration as a inputs to the IOC-Container.
```

```
          Programmer creates IOC container explicitly.
          Allows to devleop spring apps using
                   a. XML
                   b. XML + Annotation
                   c. Pure Java(No XML)
          Doesn't give embeded server to use in webapplications.
          Doesn't give embeded database/inMemory Database
        It is light weight because no autoconfiguration.
          No support for "Microservices architecture" based application development.

2. SpringBoot
          It provides abstraction for Spring framework and simplifies SpringApp
development.
          The main feature is AutoConfiguration(giving common things automatically)
          Doesn't support XML driven configuration as a inputs to the IOC-Container.
          Programmer doesn't create IOC container explicitly it gets created
automatically using
                   SpringApplication.run().
          Supports only one style of configuration that is AutoConfiguration where
inputs are supplied
          through application.properties/.yml file.
          It gives embeded server(tomcat server,jetty server) to use in web
applications.
          It gives embeded database/InMemory database called "H2".
          It is heavy weight because of AutoConfiguration.
          Support of Microservices architecture is extensively avilable.


How to change the dependant beans names dynamically in springboot application at
runtime through softcoding process to coninue
loose coupling for programmer?

@SpringBootApplication
@ImportResource(locations = "in/ineuron/cfg/applicationContext.xml")
public class BootProj02DependancyInjectionApplication {

}

applicationContext.xml
======================
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd">

      <!-- No need of keeping context namespace as the data is kept in
application.properties file -->
      <alias name="${course.choose}" alias="courseId"/>
</beans>


application.properties
======================
course.choose=dotnet


ICourseMaterial
       |=> JavaCourseMaterial("java")
       |=> DotNetCourseMaterial("dotnet")

Student
      |
      |   @Autowired
      |   @Qualifier("${courseId}");
      |=> IMaterial material
```

Usage of @Import(In 100%purejava code configuration)
==

```
@Import(value = {Persistence.class,Service.class,Controller.class})
@Configuration
public class AppConfig{

}
```