Date : 12/12/2020
Spring Boot 9AM
Mr. RAGHU

Srikanth (admin): +91-630 29 68 665 (whatsapp also)
https://www.facebook.com/groups/thejavatemple
email : javabyraghu@gmail.com

==

Spring Core Session#1
https://www.youtube.com/watch?v=xusQhpQuODk

Spring Core Session#2
https://www.youtube.com/watch?v=-FlszP92JVM

Spring Boot Sessions Links:
Spring Boot Day -1: https://youtu.be/L7zUhVLgoBA
Spring Boot Day -2: https://youtu.be/oCG4w6Rkcag
Spring Boot Day -3: https://youtu.be/diWOnew5VLk
Spring Boot Day -4: https://youtu.be/tmxJZlBB7Jw

Maven Sessions:
https://www.youtube.com/c/NareshIT/search?query=maven%20raghu

==

Spring Boot : Java Configuration
(@Configuration - @Bean)

*) @Bean indicates to container create object. Generally this is used for
Pre-defined classes.

Syntax: 1 object ---- 1 method

```
@Bean
public  <ClassName>  <objectName>() {
   //object creation code
   return obj;
}
```

----Examples---------------------

Ex#1 class : PdfExport (pre-defined class)

```
@Bean
public  PdfExport  pob() {
  PdfExport p = new PdfExport();
  return p;
}
```

*) Above code indicates creating object in spring container.
Above object can be used in any where of application.
It is re-usable object.

--Core Java Snippets----

```
class A{
  void m1(){
     int a=10;
  }
  void m2() {
     sysout(a); //error, a is a local variable, not access
  }
}

class A{
  int a=10;
  void m1(){

  }
  void m2() {
     sysout(a); //this is vaid
  }
}
class B {
  void m3(){
     // sysout(a); //this is invaid (a is not accessable directly)
     A oa = new A();
     sysout(oa.a);// this valid.
  }
}
```

*) in similer way, if we create any object inside
a method or a class that is restricted to
only that class/method, not a global object.

```
class A {
   Sample s = new Sample();
   void m1(){
      Test t = new Test();
   }
}
class B{
  void m2(){
    A oa = new A();
    sysout(oa.s);
    //Sample s = new Sample(); //duplciate
  }
}
```

*) But if we follow above Spring Syntax then
object created in Spring container that can be
accessed where even we want.

*) We can not write a direct method without class in java.
So, we need to keep all our object(@Bean) methods inside
any one class and apply @Configuration (that indicates
input to Spring container)

==

Pre-defined class : MyDataSource (driver, url)

```
@Configuration
public class AppConfig {
   @Bean
   public MyDataSource dsObj() {
      MyDataSource d = new MyDataSource();
      //provide data to variables using set method
      d.setDriver("Oracle");
      d.setUrl("jdbc:oracle");
      return d;
   }
}
==================================================================
```