Spring Core Module Revision
==========================

Single Line statements about various topics of the Core module

a) Spring Bean :: The Java class , whose obj is created and managed by the Spring Container/IOC container
b)IOC container :: It is Spring /Spring boot supplied Container who manages the spring Bean life cycle
and also takes care of Dependency Management among the Spring beans
c) what is IOC :: IOC (Inversion of Controler) is a specification providing set of rules and guidelines to manage
the dependency among the spring beans .. Depedency Lookup and Dependency Injection
are the implementation models of the IOC

d) Dependency Lookup :: The target spring bean searches in different resources to get the dependent spring bean

e) Dependency Injection :: The underlying Container(IOC container) assigns the dependent spring bean class object
to target spring bean class obj dynamically at runtime

f) setter Injection :: IOC container using setter method of tarrget class to assign dependent class obj to target class obj
g) Constructor Injection :: IOC container using constructor of the tarrget class to create the target class obj
and to assign dependent class obj to target class obj
h) Filed Injection :: IOC container assings Dependent spring bean class obj to target spring Bean obj's HAS-A property

i) Arbitary MEthod Injection:: IOC container uses arbitary (random method) method of target class to assign dependent
spring bean class obj to target spring bean class object

j) Stereo type annotations :: Multiple Annotations performing similar or same operations with minor changes
a)@Component b) @Service c) @Repository d)@Controller e)@Configuration and etc..
(note: all these annotations can make java classes as the spring beans)

k) @Autowired :: useful for Dependency Injection configurations (One spring bean injection to another spring bean)
l) @Value :: USeful for injecting simple values collected from the
a) properties file
b) System properties
c) Env variables
d) Hard Coded values
and etc..
m) @ComponentScan :: makes the IOC container searching in the given packages and their sub packages
to make stereo type annotation based java classes as the spring beans

n) @PropertySource :: To Configure the Properties file with Spring Application

o) @Primary :: Useful to solve the ambiguity Problem if the multiple same type dependent
spring beans are trying to get injected into single HAS-A property of
target spring bean class object

note:: Constructor injection is fast
but Filed injection is best becoz
it does not need separate methods
for injection process.

p)@Qualifer :: To solve the ambiguity Problem in another way

q) @ImportResource :: To map spring bean configuration (xml file) with @Configuration class

r) @Configuration :: To make java class as the spring bean cum Configuration class
required for the IOC container to provide inputs to the IOC container

s) @Bean :: makes the IOC container to take the java method of @Configuration class returned
object as the spring bean.. very useful to make pre-defined class obj as the spring bean

t) @Scope :: Makes the spring bean class object going to certain scope

the scopes are : a)singleton (default)
b) prototype
c) request
d) session
e) application
f)websocket

v) @Lazy :: the value "true" of this annotation , makes IOC container
to enable Lazy Instantiation of spring bean though the scope of the spring bean is
singleton

w) Spring Bean Life cycle :: Talks about the life cycle events that are raised in the
spring bean life cycle .. they are instanitation event , destruction event

we can configure these life cycle event related life cycle methods
using @PostConstruct and @PreDestroy methods

Spring Core Module Revision

Single Line statements about various topics of the Core module

a) Spring Bean :: The java class, whose obj is created and managed by the Spring Container/IOC container
b)IOC container :: It is Spring/Spring boot supplied Container who manages the spring Bean life cycle and also takes care of Dependency Management among the Spring beans

c) what is IOC :: IOC (Inversion of Controler) is a specification providing set of rules guidelines to manage the dependency among the spring beans.. Depedency Lookup and Dependency Injection are the implementation models of the IOC

target

different

d) Dependency Lookup :: The spring bean searches in resources to get the dependent spring bean

bean

e) Dependency Injection: The underlying Container(IOC container) assigns the dependent spring class object to target spring bean class obj dynamically at runtime

f) setter Injection :: IOC container using setter method of tarrget class to assign dependent class obj to target class obj g) Constructor Injection :: IOC container using constructor of the tarrget class to create the target class obj

and to assign dependent class obj to target class obj

h) Filed Injection :: IOC container assings Dependent spring bean class obj to target spring Bean obj's HAS-A property (random method)

i) Arbitary Method Injection:: IOC container uses arbitary method of target class to assign dependent spring bean class obj to target spring bean class object

j) Stereo type annotations ::

Multiple Annotation3 performing similar or same operations with minor changes a)@Component b) @Service c) @Repository d)@Controller e)@Configuration and etc.. (note: all these annotations can make java classes as the spring beans)

k) @Autowired :: useful for Dependency Injection configurations (One spring bean injection to another spring bean) I) @Value :: Useful for Injecting simple values collected from the

a) properties file

m) @ComponentScan ::

b) System properties

c) Env variables

d) Hard Coded values and etc..

makes the IOC container searching in the given packages and their sub packages to make stereo type annotation based java classes as the spring beans

n) @PropertySource :: To Configure the Properties file with Spring Application

o) @Primary ::

Useful to solve the ambiguity Problem if the multiple same type dependent spring beans are trying to get Injected into single HAS-A property of target spring bean class object

p)@Qualifer :: To solve the ambiguity Problem in another way

q) @ImportResource:: To map spring bean configuration (xml file) with @Configuration class

r) @Configuration :: To make java class as the spring bean cum Configuration class

s) @Bean ::

required for the IOC container to provide inputs to the IOC container

note:: Constructor injection is fast but Filed Injection is best becoz it does not need separate methods for injection process.

makes the IOC container to take the java method of @Configuration class returned object as the spring bean.. very useful to make pre-defined class obj as the spring bean

t) @Scope :: Makes the spring bean class object going to certain scope

the scopes are: a)singleton (default)

b) prototype

c) request

d) session

e) application

f) websocket

v) @Lazy :: the value "true" of this annotation, makes IOC container

to enable Lazy Instantiation of spring bean though the scope of the spring bean is singleton

*) Spring Bean Life cycle :: Talks about the life cycle events that are raised in the

spring bean life cycle .. they are instanitation event, destruction event

we can configure these life cycle event related life cycle methods using @PostConstruct and @PreDestroy methods

y) Additional features of ApplicationContext container over the BeanFactory Container

i) pre-instantiation of singleton scope spring beans

ii) Annotation driven cfgs/Java config driven configurations

iii) ability to work with properties file by recognizing the place holders with out additional cfgs iv) support of I18n

v) Event handling

vi) Ability to stop or close the IOC container

values

z) Environment obj :: The IOC container managed internal spring bean object holding various the that

are collected dynamically from different places like properties files, system properties, env variables and etc.. This Environment gives those values to spring bean properties based on the place holders placed in the @Value annotation

a1) @Import annotation:: To link one @Configuration class with another @Configuration class

a2) @PostConstruct

:: To configure user-defined method as custom init life cycle method for the instantiation event of spring bean life cycle

a3) @PreDestroy :: To configure user-defined method as custom destroy life cycle method for the destruction event of spring bean life cycle

a4) Solving Ambiguity Problem with 100% Loose Coupling

=>Here we need to change one dependent spring bean with another Dependent spring bean with out touching the source code of target spring bean.. This can be done in two ways a) using properties file + @Qualifier annotation + bean aliasing + spring bean cfg file (or)

b) Using spring profiles (best)

a5) Implementation of Dependency Lookup ::

Calling ctx.getBean(-) method in the client app to get certain spring bean class obj ref fall under Dependency Lookup

a6) Implementation of Dependency Injection :: @Value for injecting simple values to spring

Bean proeprites.. @Autowired for

injecting one spring bean with another spring bean