Bean Initialization in Spring

1. Parsing configurations across annotations:
   1. AnnotationConfigApplicationContext(Config.class or “package name”). There are two fields AnnotatedBeanDefinitionReader and ClassPathBeanDefinitionScanner in it;
2. Creating all BeanDefinition(s) – special interface, with it we can get access to meta-information about future bean;
   1. ClassPathBeanDefinitionScanner scans target package to find all classes with @Component/@Repository/@Service/@Controller and creates BeanDefinitions;
   2. AnnotatedBeanDefinitionReader works in two stages:
      1. Registering all @Configuration for next parsing. If it is with @Conditional then is will be register only if it returns true;
      2. Registering special BeanFactoryPostProcessor, naturally, BeanDefinitionRegisryPostProcessor, which with ConfigurationClassParser parses JavaConfig and creates BeanDefinition.
3. Configure BeanDefinition:
   1. All BeanDefinitions stores in Map<id, BeanMetaInfo> in ConfigurableListableBeanFactory;
   2. We can configure BeanMetaInfo with interface BeanFactoryPostProcessor, which includes the only one method – void postProcessBeanFactory(ConfigurableListableBeanFactory beanFactory). For example, database properties will be injected with postProseccBeanFactory in Java DB Connection’s BeanMetataInfo. @PropertySource shows to BeanFactoryPostProcessor what we need to inject to BeanMetaInfo before bean creating;
4. FactoryBean – is for creating custom Class which will be managing bean creating;
5. Bean creating – BeanFactory (if we have not custom FactoryBean ):
   1. First of all class BeanPostProcessor let us jump into bean creating process and do two methods:
      1. postProcessBeforeInitialization(Object bean, String beanName) is a method which do smthng before bean’s init() method;
      2. postProcessAfterInitialization(Object bean, String beanName) is a method which do smthng after bean’s init() method;
6. Bean is created.