1-Using @Bean annotation

@Bean is a method-level annotation and a direct analog of the XML <bean/> element.

We use it to declare bean. During startup time the bean definition is registered to the ApplicationContext.

We can use the @Bean annotation in a @Configuration class.

Example

Bean class

```
public class Calculator {
  public int sum(int x, int y) {
      return x + y;
  }
}
```

Using @Bean in the @Configuration class

```
@Configuration
public class AppConfig {

  @Bean
  Calculator calculator() {
      return new Calculator();
  }

  public static void main(String[] args) {
```

```
    AnnotationConfigApplicationContext context =
            new AnnotationConfigApplicationContext(AppConfig.class);
    Calculator calculator = context.getBean(Calculator.class);
    int sum = calculator.sum(5, 7);
    System.out.println(sum);
  }
}
```

Output: 12

2-Elements of @Bean Annotation

The @Bean annotation is used on Java based configuration method. It has following elements:

name : The optional bean name.

```
@Configuration
public class AppConfig {
   @Bean(name = "myBean")
   public MyBean createBean() {
      ......
   }
}
```

By default Spring container associate the bean with method name .

A bean can be associated with multiple names, the extra ones are considered aliases.

Specifying a unique meaningful name to a bean is necessary if the configuration provides more than one implementations for a bean. In that case, we have to use @Qualifier at the injection point which has an option to specify the bean name.

autowire : The autowiring mode.

```
@Configuration
public class AppConfig {

    @Bean(autowire = Autowire.BY_TYPE)
    public MyBean createBean(){
      ....
    }
}
```

Autowire.NO : This is the default. In this case, we have to explicitly use @Autowired at injection point.

Autowire.BY_TYPE : we don't need @Autowired at the injection point, given that there is only one bean available for the injection. In this mode of autowiring, the field injection doesn't work. There must be a setter.

Autowire.BY_NAME : If this mode of autowiring is specified and injection provider bean has specified name element with the some value in it's @Bean annotation, we have to use @Qualifier along with @Autowired at injection point.

initMethod/destroyMethod : Optional initialization and destruction callback method names.

```
@Configuration
public class AppConfig {

    @Bean(initMethod = "init", destroyMethod = "destroy")
    public MyBean createBean() {
     ......
    }
}
```

The destroy method will only be called for singletons but not for other scoped as Spring does not manage complete life cycle of the other beans.

In case of singleton this method is called upon closing the application context.

Since Spring also supports Java SE Common Annotations (JSR-250), we can annotate bean's methods with @PostConstruct and @PreDestroy instead of these elements.