Difference between BeanPostProcessor and BeanFactoryPostProcessor

BeanPostProcessor is an interface that allows you to create extensions to Spring Framework that will modify Spring Beans objects during initialization. This interface contains two methods:

- `postProcessBeforeInitialization`
- `postProcessAfterInitialization`

Implementing those methods allows you to modify created and assembled bean objects or even switch object that will represent the bean.

Main difference compared to BeanFactoryPostProcessor is that BeanFactoryPostProcessor works with Bean Definitions while BeanPostProcessor works with Bean Objects.

BeanFactoryPostProcessor and BeanPostProcessor in Spring Container Lifecycle

1. Beans Definitions are created based on Spring Bean Configuration.
2. `BeanFactoryPostProcessors` are invoked.
3. Instance of Bean is Created.
4. Properties and Dependencies are set.
5. `BeanPostProcessor::postProcessBeforeInitialization` gets called.
6. @PostConstruct method gets called.
7. InitializingBean::afterPropertiesSet method gets called.
8. @Bean(initMethod) method gets called
9. `BeanPostProcessor::postProcessAfterInitialization` gets called.

Recommended way to define BeanPostProcessor is through static @Bean method in Application Configuration. This is because BeanPostProcessor should be created early, before other Beans Objects are ready.

```
@Bean
public static CustomBeanPostProcessor customBeanPostProcessor() {
    return new CustomBeanPostProcessor();
}
```

It is also possible to create BeanPostProcessor through regular registration in Application Configuration or through Component Scanning and @Component annotation, however because in that case bean can be created late in processes, recommended way is options provided above.

Initialization method is a method that you can write for Spring Bean if you need to perform some initialization code that depends on properties and / or dependencies injected into Spring Bean.

You can declare Initialization method in three ways:

- Create method in Spring Bean annotated with @PostConstruct
- Implement InitializingBean::afterPropertiesSet
- Create Bean in Configuration class with @Bean method and use @Bean(initMethod)

Destroy method is a method in Spring Bean that you can use to implement any cleanup logic for resources used by the Bean. Method will be called when Spring Bean will be taken out of use, this is usually happening when Spring Context is closed.

You can declare destroy method in following ways:

- Create method annotated with @PreDestroy annotation
- Implement DisposableBean::destroy
- Create Bean in Configuration class with @Bean method and use @Bean(destroyMethod)

When using AnnotationConfigApplicationContext support for @PostConstruct and @PreDestroy is added automatically.

Those annotations are handled by CommonAnnotationBeanPostProcessor with is automatically registered by AnnotationConfigApplicationContext.

Context is Created:

1. Beans Definitions are created based on Spring Bean Configuration.
2. BeanFactoryPostProcessors are invoked.

Bean is Created:

1. Instance of Bean is Created.
2. Properties and Dependencies are set.
3. BeanPostProcessor::postProcessBeforeInitialization gets called.
4. `@PostConstruct method gets called.`
5. `InitializingBean::afterPropertiesSet method gets called.`
6. `@Bean(initMethod) method gets called`
7. BeanPostProcessor::postProcessAfterInitialization gets called.

Bean is Ready to use.

Bean is Destroyed (usually when context is closed):

1. `@PreDestroy method gets called.`
2. `DisposableBean::destroy method gets called.`
3. `@Bean(destroyMethod) method gets called.`