Жизненный цикл бина.

- В Spring есть возможность выполнять код после внедрения всех зависимостей в бин и перед уничтожением бина.
- Для специализированного метода инициализации гарантируется, что он будет вызван после того, как все зависимости бина готовы.

- Создать такой метод можно тремя способами:
- Реализовать в классе интерфейс InitializingBean, переписать метод afterPropertiesSet();
- Указать в декларации бина:

<bean id="bean1" class="Bean1" init-method="init(или любое другое название)"/>

Или общий метод по умолчанию для всех бинов в контейнере

<beans default-init-method="init">

- С использованием аннотации **@PostConstruct** к методу

Аналогично для выполнения кода перед уничтожением бина существуют способы:

- Реализовать в классе интерфейс DisposableBean, переписать метод destroy();
- Указать в декларации бина:

<bean id="bean1" class="Bean1" destroy-method="название_метода"/>

Или общий метод по умолчанию для всех бинов в контейнере

<beans default-destroy-method=“destroy">

- С использованием аннотации **@PreDestroy**

При комбинировании разных методов они отработают в след.порядке:

- Аннотации
- метода из интерфейсов
- Установленный в XML пользовательский метод.

- @PostConstruct и @PreDestroy были полностью удалены в Java 11 . Чтобы продолжать их использовать, вам нужно добавить JAR-файл javax.annotation-api в ваши зависимости.

- В примере наш бин exampleBean реализует интерфейсы InitializingBean и DisposableBean

При запуске main видим, что

- сеттеры установили свойства,
- затем отработал метод afterPropertiesSet(),
- код из main'a
- в последнюю очередь метод destroy().

- Если изменить скоуп бина на прототип, то вывод изменится: метод destroy() не отработает. Он не вызывается для бинов со скоупом прототип.

- В следующем примере (27) показан второй способ установки метода инициализации и destroy-метода. Через указание этих методов в декларации бина в XML.