ApplicationContextAware, Ивенты.

И ещё раз: чтобы получить доступ к контексту, для публикации ивента, класс должен поддерживать интерфейс ApplicationContextAware

```
public class Manager implements ApplicationContextAware {

    private ApplicationContext applicationContext;

    @Override
    public void setApplicationContext(ApplicationContext applicationContext) throws
BeansException {
        this.applicationContext = applicationContext;
    }

    public ApplicationContext getApplicationContext() {
        return applicationContext;
    }
}
```

- Такой класс сможет публиковать ивенты через контекст через метод publishEvent(), в который нужно передать параметр - ивент

```
manager.getApplicationContext().publishEvent();
```

- Для получения ивентов класс должен реализовать интерфейс ApplicationListener<>

```
public class RefreshListenerBean implements ApplicationListener<ContextRefreshedEvent> {
```

```
public class CustomListenerBean implements ApplicationListener<CustomEvent> {
```

```
public class CloseListenerBean implements ApplicationListener<ContextClosedEvent>
```

- В методе onApplicationEvent() прописывается необходимая логика.

- Кастомные(свои) ивенты: класс кастомного ивента должен наследоваться от ApplicationEvent класса.

```
public class CustomEvent extends ApplicationEvent {
    private static final long serialVersionUID = 21L;

    public CustomEvent(Object source) {
        super(source);
    }
}
```

В следующем примере мы разберём, как можно использовать кастомный ивент.

Когда происходит ивент, все бины в контейнере, которые реализуют ApplicationListener интерфейс оповещаются.

Две главные реализации ApplicationEvent:

- ContextRefreshedEvent - при создании или обновлении ApplicationContext
- ContextClosedEvent - после использования close() на контексте.