@EventListener và @Async là các Annotatiob hỗ trỡ việc tạo các sự kiệ từ  **nguồn – source,**  thông qua các  **publisher**  và được nhận bởi các **Listener**.

**- Ở Nguồn – source: là các đối tượng**  cần extend: **ApplicationEvent**

public class GoFishingEvt extends ApplicationEvent {  
 private int numberOfPeople;  
 private String leader;  
  
 public GoFishingEvt(Object source, int numberOfPeople, String leader) {  
 super(source);  
 this.numberOfPeople = numberOfPeople;  
 this.leader = leader;  
 }

**- publisher là ApplicationEventPublisher – component** được tạo sẵn còn spring context.

@Component  
public class Publisher {  
  
 public Publisher(ApplicationEventPublisher applicationEventPublisher) {  
 this.applicationEventPublisher = applicationEventPublisher;  
 System.*out*.println("Start publishing events...");  
 }  
  
 private ApplicationEventPublisher applicationEventPublisher; // PUBLISHER  
  
 public void goFishingPublisher() {  
 System.*out*.println("\n\*\*\*\*\*\*" + "publisher: Asking anyone go fishing...");  
 applicationEventPublisher.publishEvent(new GoFishingEvt(this, 5, "Quanaqt"));  
 }  
  
 public void playSoccerPublisher(Soccer soccer, LocalDateTime localDateTime) {  
 System.*out*.println("\n\*\*\*\*\*\*" + "publisher: Asking anyone go to play soccer at " + localDateTime);  
 applicationEventPublisher.publishEvent(soccer);  
 }

- **listener :** là các component sẽ lắng nghe các event khi publisher tạo event.

Trong 1 @EventListênr chỉ có 1 tham số duy nhất.

@Component  
public class ActionsListener {  
  
 @Autowired  
 private QuanaqtConfig quanaqtConfig;  
  
 @EventListener  
 public void goFishing(GoFishingEvt fishingEvt) {  
 System.*out*.println(Thread.*currentThread*().getName() + "-" +fishingEvt);  
 }  
  
 @Async  
 @EventListener  
 public void playSoccer(Soccer soccer) {  
 System.*out*.println(Thread.*currentThread*().getName() + "-" + soccer + " play at " + soccer.getLocation() + " with " + quanaqtConfig);  
  
 }  
}

.

**Các luồng phát sự kiện -> lắng nghe -> xử lý sk là mặc định đồng bộ . Nếu muốn các sự kiện được sử ly bất đồng bộ so với luồng chính, cần sử dụng @Async**