Spring ApplicationEvents:

Wat is Spring ApplicationEvents? Een ApplicationEvent in Spring is een mechanisme waarmee onderdelen binnen een applicatie met elkaar kunnen communiceren door gebeurtenissen (events) te verzenden en te beluisteren, zonder directe afhankelijkheden. Dit stelt een component in staat een event te publiceren wanneer er een bepaalde actie plaatsvindt (bijvoorbeeld het aanmaken van een bestelling), terwijl andere componenten op dat event kunnen reageren door bijvoorbeeld een e-mail te sturen of een log aan te maken.

Spring ApplicationEvents maakt het mogelijk om te reageren op gebeurtenissen zonder harde koppelingen tussen componenten. Dit bevordert loose coupling, vooral wanneer je nieuwe functionaliteit wilt toevoegen die slechts een reactie is op bestaande functionaliteit. In ons project zou dit goed van pas zijn gekomen.

Een voorbeeld: nadat een nieuwe lading is verwerkt in de StockPortionDeliveryService, moet de UnfulfilledOrderLineService controleren of er nog orderregels van de inkooporders zijn die nog niet zijn opgehaald (voor die specifieke grondstof). Momenteel zijn deze services echter hard gekoppeld, terwijl de logica eigenlijk los van elkaar zou moeten staan. Door deze koppeling moest ik wijzigingen aanbrengen in StockPortionDeliveryService, wat weer leidde tot aanpassingen elders in de code—een omslachtige aanpak.

Hoe zou Spring ApplicationEvents dit oplossen? In dit voorbeeld zou StockPortionDeliveryService een event kunnen publiceren bij een nieuwe lading, terwijl UnfulfilledOrderLineService als listener optreedt en reageert op dit specifieke event (de levering van de lading). Dit zou de harde koppeling tussen beide services verminderen. Dit was niet de enige plek waar Spring ApplicationEvents de architectuur zou hebben verbeterd. Ook zou dit heel handig zijn geweest bij het implementeren van het starten van de operaties, vaak werden hun triggers veroorzaakt door andere acties. Bijvoorbeeld het einde van de bunkeringoperatie zorgde voor de start van de volgende bunkeringoperatie.

Er zijn echter enkele nadelen. Spring ApplicationEvents kunnen de complexiteit verhogen, omdat het moeilijker wordt te volgen welke componenten op welke events reageren, vooral bij asynchrone verwerking. Daarnaast kan intensief gebruik leiden tot prestatieproblemen, doordat alle listeners bij elk event (waaraan de listener geregistreerd is) moeten reageren. Ook ontstaan er onzichtbare afhankelijkheden tussen componenten, wat het onderhoud bemoeilijkt. Tot slot maakt het los verwerken van events het lastig om transactionele logica te garanderen, waardoor de consistentie in complexe processen onder druk kan komen te staan.

Met deze nadelen in gedachten is het belangrijk om Spring ApplicationEvents toe te passen in situaties waar loskoppeling een duidelijke meerwaarde heeft en de voordelen zwaarder wegen dan de nadelen.