ApplicationEvents in Spring

Een application event in Spring is een mechanisme die de communicatie binnen een Spring-applicatie mogelijk maakt. Het mechanisme stelt de applicatie in staat om asynchroon en losgekoppeld van elkaar te communiceren en eventueel code uit te voeren als er veranderingen gebeuren. Deze technologie wordt vaak vergeleken met het Publisher-Subscriber Patroon of het Observer Patroon dat zeer vaak gebruikt wordt. Application events in Spring heeft hiermee verschillende overeenkomsten.

Implementatie:

Tijdens het maken van deze opdracht kwam ik erachter dat we in ons project dit mechanisme hadden gebruikt.
We hebben dit mechanisme toegepast voor het toevoegen van loyalty-points tijdens het bevestigingsproces van een order. Als de klant de order bevestigd dan wordt er op een soort “queue” een message geplaatst met in ons geval een LoyaltyPointEvent. De code is als volgt:

```
publisher.publishEvent(new
LoyaltyPointEvent(order.getTotalPrice(), accountId));
```

In onze LoyaltyService bevindt zich de code die de message van de queue kan lezen. Als die bepaalde message met een LoyaltyPointEvent op de “queue” wordt gezet, worden automatisch de bijbehorende subscribers verwittigd. Dat gebeurt op volgende manier:

```
@EventListener
private void addLoyaltyPoints(LoyaltyPointEvent loyaltyPointEvent)
{
   // uit te voeren code
}
```

Aan de hand van de annotatie `@EventListener` wordt de code automatisch uitgevoerd als de message op de “queue” wordt gezet. We kunnen dit nog op verschillende plaatsen implementeren zoals bij een message van applicatie naar applicatie. In ons voorbeeld is dat van de client applicatie naar de bakery applicatie. Wanneer de bakery applicatie een message ontvangt, wordt dit via het mechanisme hierboven beschreven verstuurd naar een methode die de order in ons geval gaat loggen. Dit hebben we ook zo toegepast in de code.

Voordelen:

- **Low-Coupling**: Verschillende klassen hoeven geen directe referenties naar elkaar. Wat leidt tot minder afhankelijkheid.
- **Geen externe of extra packages nodig**: Dit mechanisme zit ingebouwd in Spring waardoor er geen externe code moet worden geïmplementeerd of geïnstalleerd.
- **Gemakkelijke code**: Spring automatiseert heel veel en dat is hierbij ook het geval. Je hoeft niet veel code toe te voegen om dit te kunnen realiseren.

Nadelen

De nadelen van het gebruik van application events in Spring is beperkt. De gevaren van dit gebruik zou ik als volgt omschrijven:

- **Te veel gebruik**: Je kunt dit te veel gaan toepassen waardoor je weinig koppeling krijgt, het gevaar zit hierin dat je code niet meer leesbaar is en niet duidelijk is voor een eventuele opvolger van je project.

- **Complex**: Dit hangt samen met te veel gebruik, de code wordt veel complexer als je dit te veel gaat gebruiken.