Event bus API

- Overview
- Event bus API
- Event bus Implementation
- Event classes
- Example of creating and pushing an event

Overview

The Event bus API is used for communication and notifications about actions that occur in the components of the system. In order not to have tight coupling between modules and their API, the Event API was introduced. The API is also used for handling events with custom rules that the user can implement in Groovy.

Event bus API

The Event bus is represented by the `com.epam.fej.event.EventBus` interface, which can be injected in any required module and has the following methods:

Method	Description
<T extends AppEvent> void publish(T appEvent)	Publishes the event provided to subscribers.
<T extends AppEvent> void publish(T appEvent, boolean sync)	The same as the previous method but with an ability to specify the mode of publishing.
<T extends AppEvent> void publish(T appEvent, boolean sync, Predicate<T> afterRuleCallback, Consumer<T> afterPublishingCallback)	`afterRuleCallback` - callback that decides if further rule processing is executed `afterPublishingCallback` - callback that is called either after processing all applicable rules or at the end of the method if there are not any rules for the event
<T extends AppEvent> void subscribe(Consumer<T> consumer, Class<T> type)	Subscribes to the specified event type.

It is also possible to register an event listener by marking it with the `com.epam.fej.event.annotation.EventSubscriber` annotation. The marked class should implement the `java.util.function.Consumer` interface and be registered as a Spring bean.

Event bus Implementation

The Event API is based on the custom implementation of the event bus interface because none of the analyzed libraries suit the requirement of garbage free. It is the `com.epam.fej.event.EventBusImpl` class that uses the pool of events to avoid garbage. The declaration is in `com.epam.fej.event.SpringConfiguration`. The implementation notifies subscribers asynchronously using a specified executor with one thread in its pool to have the event ordered.

Event classes

The `com.epam.fej.event.AppEvent` interface implementations are expected as events for the event bus. It implies the re-usability of instances and creates them only from a specific pool.

`com.epam.fej.event.EventPool` is a static class that can create an instance of an event or retrieve it from the pool with the getEvent(Class<T> type) method.
Note: You do not need to release the objects. The event bus does it automatically after notifying all subscribers.

Example of creating and pushing an event

```
SchedulerEvent schedulerEvent = EventPool.getEvent(SchedulerEvent.class);
schedulerEvent.setId(id);
eventBus.publish(schedulerEvent);
```