eng.event_or_not_event

Alexandr Kirilov (https://github.com/alexandrkirilov)

Event or not event

Event driven systems became popular among developers. This architectural approach is very powerful, but not always it might be well suited for the cases or situations.

All of this article based on meaning term "event" explained in Collins Cobuild Dictionary:

An event is something that happens, especially when it is unusual or important. You can use events to describe all the things that are happening in a particular situation.

The key point for this article is "... to describe the all things ...". In accordance with it - if you have no necessity to store massive amount of the different kinds of information about events within ability to manage this kinds dynamically, then no necessity to use this approach.

Any kind of event system is always very complex system that is trying to unify different natures of events under one engine covering. Complexity of the system will require to care about level of knowledge of developers that starting it, the developers have to have enough experience in abstraction, because there always be an issue of solving problem what is "general" and what is "special". The developing unified engine will possibly take more time then any plain solution, directly based on information without splitting them up on 3W parts (When? Where? What?) and interpreting or organizing them like events. And more, and more ...

In contrast to the previously described complexity, there are a lot of advantages. If you've been paying enough attention to careful examination and analyzing of events, if you'd done good enough abstraction - you have no necessity to care about how to deliver events to the system, you don't need to worry about storing data and other, all of it related to the "general" part and developing one time at the begin. You need to consider about event's model only. It might be super useful and cheap in case of maintaining huge system. Well projected and developed event system is allowing to make research and analyze more lightly for the case of big-data. Unified model of getting events is allowing to attach any kind of triggering system (this point might be very critical in some cases) within rules constructor. And other, and other ...

The conclusion about necessity of event driven system: if you need to care about huge list of kinds of events from different sources, if there required ability to manage the event's kinds and data models (adding new events, blocking and adjusting previously developed) and you don't know the next kind of event that going to be attached to the system, if you have previously gathered information in different manners of storing or formatting (some kind of old-ages data inheritance from applications), if you have enough resources for developing and organizing test-lab, if you have enough patience for gathering information about possible procedures that going to be performed for events - then there are obvious solution, event driven system.

Some examples when the event driven good and when not.

Just imagine some kind of library where stored articles, novels, poems, etc. And this content might be used inside of restricted population of users - then for this solution is really bad idea to use events. But! When you need to care about logging of user activities and make a prospective analyze of content usage of billions articles by billions users - then use event's approach. It will save a lot of money at time of maintaining.

Follow author updates on **Linkedin**

Follow AR|BO|RE|US updates on **Twitter** and **Linkedin**