**What is a domain event?**

An event is something that has happened in the past. A domain event is, logically, something that happened in a particular domain, and something you want other parts of the same domain (in-process) to be aware of and potentially react to.

یک رخداد، چیزی است که در گذشته اتفاق افتاده است. یک رخداد دامنه، به طور منطقی، چیزی است که در یک دامنه خاص اتفاق افتاده است و چیزی است که شما می خواهید بخش هایی از همان دامنه (دامنه موجود در یک پروسس) از آن خبر داشته باشند و به طور بالقوه به آن واکنش نشان دهند.

An important benefit of domain events is that side effects after something happened in a domain can be expressed explicitly instead of implicitly. Those side effects must be consistent so either all the operations related to the business task happen, or none of them. In addition, domain events enable a better separation of concerns among classes within the same domain.

مهترین نکته در مورد رخ دادهای دامنه این است: اثرات جانبی که بعد از یک اتفاق رخ می دهند، می توانند به صراحت،نه به طور ضمنی، بیان شوند. این اثرات جانبی باید استوار یا اصطلاحا نامتناقض باشند، به این معنی که یا همه عملیات ها مرتبط به با یک وظیفه کسب و کار ، یا هیچ عملیاتی، باید انجام شود. علاوه بر آن، رخدادهای دامنه باعث جدایی بهتر وابستگی بین کلاس ها درون یک دامنه می شود که این از نظر مهندسی نرم افزار یک امتیاز حساب می شود.

For example, if you're just using Entity Framework and entities or even aggregates, if there have to be side effects provoked by a use case, those will be implemented as an implicit concept in the coupled code after something happened. But, if you just see that code, you might not know if that code (the side effect) is part of the main operation or if it really is a side effect. On the other hand, using domain events makes the concept explicit and part of the ubiquitous language. For example, in the eShopOnContainers application, creating an order is not just about the order; it updates or creates a buyer aggregate based on the original user, because the user is not a buyer until there is an order in place. If you use domain events, you can explicitly express that domain rule based in the ubiquitous language provided by the domain experts.

برای مثال، اگر در یک برنامه شما فقط از Entity Framework و entity ها یا حتی از aggregate ها استفاده می کنید، به شما مورد کاربری دهند که یک اثر جانبی را پیاده سازی کنید، شما آن را با به طور ضمنی با نوشتن پند خط کد بعد از اتفاقی که رخ می دهد پیاده سازی می کنید. اما، اگر به آن کد نگاه کنید، شما ممکن متوجه نشوید که آیا آن کد (اثر جانبی) بخشی از عملیات اصلی است یا در واقع یک اثر جانبی است. به عبارت دیگر، استفاده از رخ دادهای دامنه باعث ایجاد مفهوم صراحتی می شود که بخش از زبان ubiquitous است.

برای مثال، در یک اپلیکیشن فروشگاه، ایجاد یک سفارش فقط منحصر به خود سفارش نیست؛ بلکه باعث می شود که aggregate خریدار ، بروز رسانی یا ایجاد شود. به این معنی که یک کاربر تا زمانی سفارش نداشته باشد یک خریدار نیست. اگر از مفهوم رخداد دامنه استفاده کنید، شما می توانید به طور صریح بیان کنید که قاعده دامنه مبتنی بر زبان ubiquitous توسط متخصص دامنه فراهم گردیده است.

Domain events are somewhat similar to messaging-style events, with one important difference. With real messaging, message queuing, message brokers, or a service bus using AMPQ, a message is always sent asynchronously and communicated across processes and machines. This is useful for integrating multiple Bounded Contexts, microservices, or even different applications. However, with domain events, you want to raise an event from the domain operation you are currently running, but you want any side effects to occur within the same domain.

The domain events and their side effects (the actions triggered afterwards that are managed by event handlers) should occur almost immediately, usually in-process, and within the same domain. Thus, domain events could be synchronous or asynchronous. Integration events, however, should always be asynchronous.

رخ دادهای دامنه چیزی شبیه رخدادهای با ساختار پیام رسانی هستند، البته با یک تفاوت مهم. در پیغام رسانی واقعی، صف بندی پیغام، کارگزاران پیام، یا یک سرویس باس AMPQ، یک پیغام همیشه به صورت غیر همزمان فرستاده می شود و با چندین پروسس و ماشین ارتباط برقرار می کند. این ساختار برای پیکپارچه سازی چندین bounded context یا میکرسرویس یا حتی چندین برنامه کاربردی مفید است. اگرچه، با رخ داد های دامنه، شما می خواهید توانید یک رخداد را از عملیات دامنه ای که در حال اجرا است، صادر کنید. اما همچنان می خواهید که اثرات جانبی درون همان دامنه، نه خارج از آن، اتفاق بیافتد.

رخدادهای دامنه و اثرات جانبی (عملیات های که بعد از صادر شدن رخ دادها باید توسط هندلرهای رخداد مدیریت شود) باید به صورت تقریبا فوری انجام گیرد، معمولا در پروسس، و درون همان دامنه. بنابراین، رخ دادهای دامنه می توانند باید همزمان، نه غیرهمزمان، باشند. رخ دادهای یکپارچه، می توانند، همیشه به طور غیر همزان رخ دهند.