**Event Sourcing**

As I mentioned above, CQRS often goes hand in hand with event sourcing. This is a pattern where you don’t store the current state of your model in the database, but rather the events that happened to the model. So when the name of a customer changes, you won’t store the value in a “Name” column. You will store a “NameChanged” event with the new value (and possibly the old one too).

event sourcing

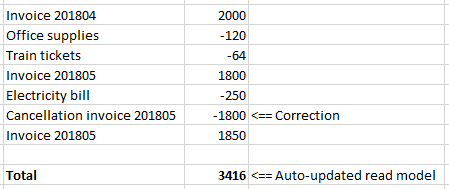
همانطور که میدانید ، CQRS اغلب با کمک event sourcing اتفاق می افتد. این الگویی است که شما وضعیت فعلی مدل خود را در دیتابیس ذخیره نمی کنید ، بلکه اتفاقاتی است که برای مدل افتاده است. بنابراین وقتی نام مشتری تغییر می کند ، شما ارزش آن را در ستون "نام" ذخیره نمی کنید. شما یک رویداد ""NameChanged را با مقدار جدید (و احتمالاً قدیمی) نیز ذخیره خواهید کرد.

When you need to retrieve a model, you retrieve all its stored events and reapply them on a new object. We call this rehydrating an object.

هنگامی که شما نیاز به بازیابی یک مدل دارید ، تمام رویدادهای ذخیره شده آن را بازیابی می کنید و آنها را مجدداً بر روی یک شی جدید قرار می دهید. ما این کار را دوباره زنده کردن یک شیء می نامیم.

A real-life analogy of event sourcing is accounting. When you add an expense, you don’t change the value of the total. In accounting, a new line is added with the operation to be performed. If an error was made, you simply add a new line. To make your life easier, you could calculate the total every time you add a line. This total can be regarded as the read model. The example below should make it more clear.

قیاس در زندگی واقعی از event sourcing ، حسابداری است. وقتی هزینه اضافه می کنید ، مقدار کل را تغییر نمی دهید. در حسابداری ، خط جدیدی با عملیاتی که باید انجام شود اضافه می شود. اگر خطایی ایجاد شد ، شما به سادگی یک خط جدید اضافه می کنید. برای آسانتر کردن زندگی ، می توانید هر بار اضافه کردن یک خط ، کل را محاسبه کنید. این جمع را می توان به عنوان مدل خوانده شده در نظر گرفت. مثال زیر باید آن را روشن تر کند.

[](https://blog.ndepend.com/wp-content/uploads/layered-5.png)

You can see that we made an error when adding Invoice 201805. Instead of changing the line, we added two new lines: first, one to cancel the wrong line, then a new and correct line. This is how event sourcing works. You never remove events, because they have undeniably happened in the past. To correct situations, we add new events.

می بینید که هنگام اضافه کردن فاکتور 201805 خطایی مرتکب شدیم. به جای اینکه خط را تغییر دهیم ، دو خط جدید اضافه کردیم: اول یکی برای لغو خط اشتباه ، سپس یک خط جدید و صحیح. این چگونگی عملکرد event sourcing می باشد. شما هرگز وقایع را حذف نمی کنید ، زیرا آنها در گذشته غیرقابل انکار اتفاق افتاده اند. برای تصحیح موقعیت ها ، ما رویدادهای جدیدی اضافه می کنیم.

Also, note how we have a cell with the total value. This is simply a sum of all values in the cells above. In Excel, it automatically updates so you could say it synchronizes with the other cells. It is the read model, providing an easy view for the user.

همچنین توجه داشته باشید که چگونه یک سلول با مقدار کل داریم. این فقط مجموعه ای از کل مقادیر موجود در سلول های فوق است. در اکسل به طور خودکار بروزرسانی می شود ، بنابراین می توانید بگویید که آن را با سلولهای دیگر همگام می کند. این مدل خوانده شده است ، یک نمای آسان برای کاربر فراهم می کند.

Event sourcing is often combined with CQRS because rehydrating an object can have a performance impact, especially when there are a lot of events for the instance. A fast read model can significantly improve the response time of the application.

معماری event sourcing معمولاً با معماری CQRS ترکیب می شود زیرا آب دادن مجدد به یک جسم می تواند تأثیر عملکردی داشته باشد ، به ویژه هنگامی که برای مثال تعداد زیادی واقعه اتفاق می افتد. یک مدل سریع خواندنی می تواند زمان پاسخ برنامه را به میزان قابل توجهی بهبود بخشد.

**Advantages**

* This software architecture pattern can provide an audit log out of the box. Each event represents a manipulation of the data at a certain point in time.

مزایا

• این الگوی معماری نرم افزار می تواند یک حسابرسی خارج از کادر ارائه دهد. هر رویداد نمایانگر دستکاری داده ها در یک نقطه خاص از زمان است.

**Disadvantages**

* It requires some discipline because you can’t just fix wrong data with a simple edit in the database.
* It’s not a trivial task to change the structure of an event. For example, if you add a property, the database still contains events without that data. Your code will need to handle this missing data graciously.

معایب

• به بعضی از نظم ها احتیاج دارد زیرا نمی توانید فقط با یک ویرایش ساده در دیتابیس داده های اشتباه را اصلاح کنید.

• تغییر ساختار یک رویداد یک کار ناچیز نیست. به عنوان مثال ، اگر یک ویژگی اضافه کنید ، بانک اطلاعاتی هنوز حاوی وقایع بدون آن داده است. کد شما باید با مهربانی به این داده های از دست رفته نیاز داشته باشد.

**Ideal for applications that**

* Need to publish events to external systems
* Will be built with CQRS
* Have complex domains
* Need an audit log of changes to the data

ایده آل برای برنامه های کاربردی که

• نیاز به انتشار وقایع در سیستم های خارجی دارند

• با CQRS ساخته خواهد شد

• دامنه های پیچیده ای دارند

• به یک حسابرسی از تغییرات داده ها نیاز دارند

<https://dzone.com/articles/software-architecture-the-5-patterns-you-need-to-k>