

EVENT SOURCING

WHEN CRUD NICHT REICHT

*“It ain't what you don't know
that gets you into trouble, it's
what you know for sure that
ain't so” -- Marc Twain*

ÜBER UNS

Jörg Herbst
@terrible_herbst



Jan Sauer
@jansauer



ÜBER UNS

10m GmbH
Koblenz, Düsseldorf,
Hannover
www.10m.de



3 THESEN ZU DB MODELLEN

1. Unsere Datenmodelle leiden unter **Datenverlust**
2. ERM für heutige Anforderungen ist **komplex**
3. Keiner kennt die Anforderungen von **morgen**

SZENARIEN

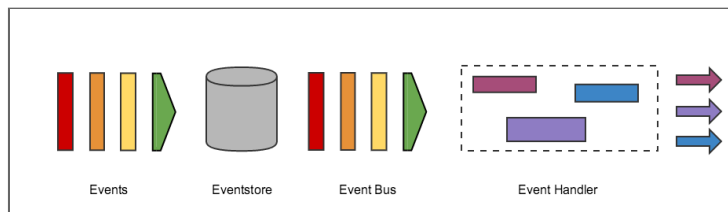
1. Fehler sind nicht reproduzierbar
2. Business Anforderungen ändern sich
3. Der Zustand der Daten lässt sich nicht erklären
4. Das ist zu komplex zum Testen

DER LÖSUNGSANSATZ

Anstelle des **Zustandes** speichern wir **Ereignisse**

EVENTFLOW

- Trennung zwischen Schreib-/Leselogik
- Ein Modell das nur Methoden hat die den Zustand ändern
- Ein oder mehrere Modell zum Lesen



EIGENSCHAFTEN VON EVENTS

- Events **haben** bereits stattgefunden (Vergangenheit)
- Events dürfen sich **nie ändern** (Immutable)
- Events haben eine **feste Reihenfolge**

WAS HABEN WIR JETZT GEWONNEN

- kein Informationsverlust
- Audit Log 4 free
- flexibeles Datenmodel
- einfache Fehlersuche
- historischer Zustand verfügbar
- einfache Testbarkeit

DOMAIN MODEL (STANDARD)

```
class ShoppingCart {  
  
    BigDecimal total = new BigDecimal(100);  
    public add(Item item) {  
        if (!item.isAvailable()) throw new BasketException  
            "Item must not be sold")  
        BigDecimal newTotal = total.add(item.getItemTotal)  
        ...  
  
        this.total = newTotal  
    }  
}
```

DOMAIN MODEL (ERWEITERT)

```
class ShoppingCart {
    List events = new LinkedList()
    BigDecimal total = new BigDecimal(100);
    public add(Item item) {
        if (!item.isAvailable()) throw new BasketException(
            "Item must not be sold")
        BigDecimal newTotal = total.add(item.getItemTotal())
        event = new ItemAddedEvent(item.articleno, item.getItemTotal())
        events.add(event)
        ...

        this.total = newTotal
    }
}
```

PSEUDOCODE

```
public void action() {  
    Read allItemsFromEventStore  
    Create new Basket  
    applyEvents(pastEvents)  
    newEvents = processCmd(theCommand)  
    saveEvents(newEvents)  
}
```

WEITERE VORTEILE VON EVENT SOURCING

- Skalierbarkeit (physische Trennung)
- Tests mit Daten vom Zeitpunkt X
- Ändern der Vergangenheit
- Snapshots

NACHTEILE VON EVENT SOURCING

- Aufwändig
- Ungewohnt und kein JEE Standard
- Schlechte Events bleiben für immer
- Transaktionshandling ist komplex
- Suche nur über Primärschlüssel

Big Data CQRS Docker MQTT Domain

Driven Design Non-blocking

MapReduce NoSQL Cloud

Microservices

Skalierbarkeit Internet of Things

EventBus

WEITERE LINKS

- Greg Young **Event Sourcing**
- **AxonFramework**
- Buch **CQRS and Event Sourcing** von MS