How to avoid your business application screwing up

masak

YAPC::EU 2011
airports
(hi, I'm masak)
Airport

Check in
Airport

Check in

Drop Luggage
Airport

Check in

Drop Luggage

Passport control
Airport

- Check in
- Luggage
- Passport control
- Boarding

Diagram of airport flow.
post-hypnotic suggestion
it's ok to have more than one model
traditionally

traditionally
data
nouns
<table>
<thead>
<tr>
<th>Passenger</th>
<th>Flight</th>
<th>Luggage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
normalized
domain model
focus on the verbs
aggregate
bounded context
Passenger tracking

Luggage tracking
so, traditionally
PassangerService

void PutPassengerInFirstClass(PassengerId)
Passenger GetPassenger(PassengerId)
ArrayRef[Passenger] GetPassengersWithName(Name)
ArrayRef[Passenger] GetFirstClassPassengers()
void ChangePassengerLocale(PassengerId, NewLocale)
void RegisterPassenger(Name, SSN, FlightId)
void EditPassengerDetails(PassengerDetails)
PassangerWriteService

void PutPassengerInFirstClass(PassengerId)
void ChangePassengerLocale(PassengerId, NewLocale)
void RegisterPassenger(Name, SSN, FlightId)
void EditPassengerDetails(PassengerDetails)

PassangerReadService

Passenger GetPassenger(PassengerId)
ArrayRef[Passenger] GetPassengersWithName(Name)
ArrayRef[Passenger] GetFirstClassPassengers()
the end
the end?
hm...
read-side/write-side
be normal
why?
Databases intended for online transaction processing (OLTP) are typically more normalized than databases intended for online analytical processing (OLAP).

- Wikipedia
reads are common
optimize for reads
Passenger booked
Passenger Checked In
Passenger cleared security
Passenger boarded
sum = foldl (+) 0
state = foldl apply empty
customize your read-side
current state isn't always enough
need prediction FAIL
CommandHandler can rebuild an aggregate from events. It can call methods on an aggregate. Basically, it holds your business logic. It can rebuild an aggregate from events. Data Storage fetches and stores data.

Diagram:
- Command
- CommandHandler
- Aggregate
- Data Storage
Events table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Column type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AggregateId</td>
<td>Guid</td>
</tr>
<tr>
<td>Data</td>
<td>Blob</td>
</tr>
<tr>
<td>Version</td>
<td>Int</td>
</tr>
</tbody>
</table>
## Aggregates table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Column type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AggregateldId</td>
<td>Guid</td>
</tr>
<tr>
<td>Type</td>
<td>Varchar</td>
</tr>
<tr>
<td>Version</td>
<td>Int</td>
</tr>
</tbody>
</table>
Passenger
BC
Luggage
BC
Flight
BC
Check-in
Drop luggage
Passport control
Board plane
problem
consistency
saga
testing
Given

an aggregate in a certain state

When

an action performed on the aggregate

Then

a number of consequences
Given

a number of events

When

an action performed on the aggregate

Then

a number of consequences
Given

ArrayRef[Event]

When

an action performed on the aggregate

Then

a number of consequences
Given

ArrayRef[Event]

When

a command performed on the aggregate

Then

a number of consequences
Given

ArrayRef[Event]

When

Command

Then

a number of consequences
Given

ArrayRef[Event]

When

Command

Then

a number of events
Given

ArrayRef[Event]

When

Command

Then

ArrayRef[Event]
Given

ArrayRef[Event]

When

Command

Then

ArrayRef[Event] | Exception
team independence
agile
outsourcing
summary
more than one model
aggregates
CQRS
read side/write side
event sourcing
thank you