Paweł Rajba
pawel@cs.uni.wroc.pl
http://www.itcourses.eu/

CQRS

Agenda

- Motivations
- The common approach
- Nature of business applications
- CQS and CQRS
- Why to create separate models?
- Implementations
- A few more things about commands
- Synchronization
- Eventual consistency
- Benefits

Motivations: Collaboration

- Multiple actors want to operate on the same data
  - Actor can be human or system
- That implies concurrency problems with locking the data
- We can solve it on database level, we can handle exceptions
  - … but maybe we can do it more effective?

Motivations: Retrieving data

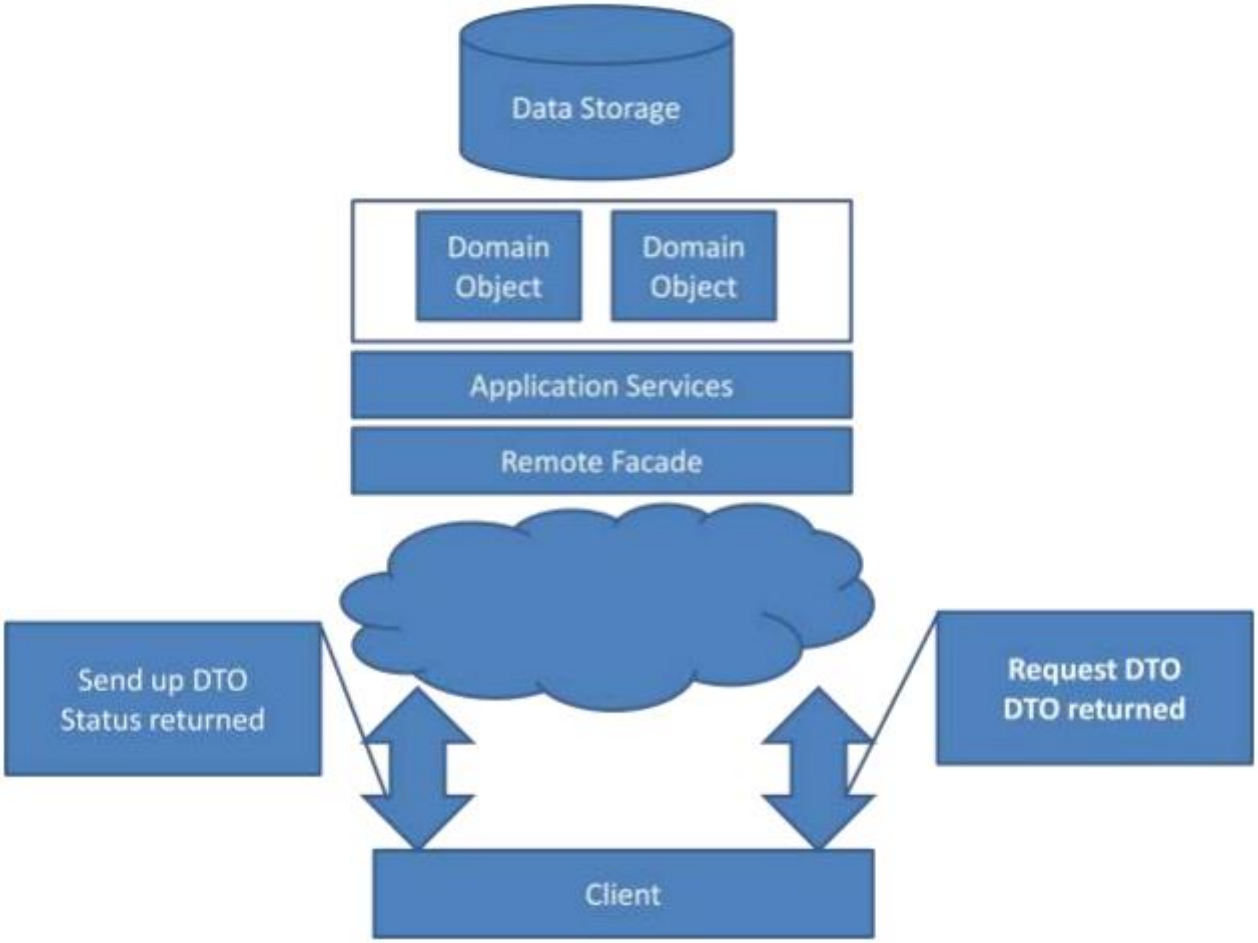
- In some cases, especially in collaboration, when you show data to a user, the same data may have been changed by another actor – it is stale
- If so, why all the time retrieve it from the master database?
- Why perform all the conversions between normalized structures and flat view model?
- Why do not have a separate datastore dedicated to queries? Maybe it could a different solution than a relational database? Maybe cache?
- Once we have such store, perhaps it would be easier to add more instance and get ease scaling?
- Finally, we can have a lot of and end up with…

Query of despair

Motivations: Data modifications

- A common approach is that we have an update form to change a business object
- Let's assume that we have a bookstore and we want to correct a responsible person for specimen together with reserving the one
- But in the meantime, someone else reserved the specimen and in most common scenario we reject all changes although changing person is still valid
- Why don't make provide a right level of granularity and allow changing the person?

The common approach

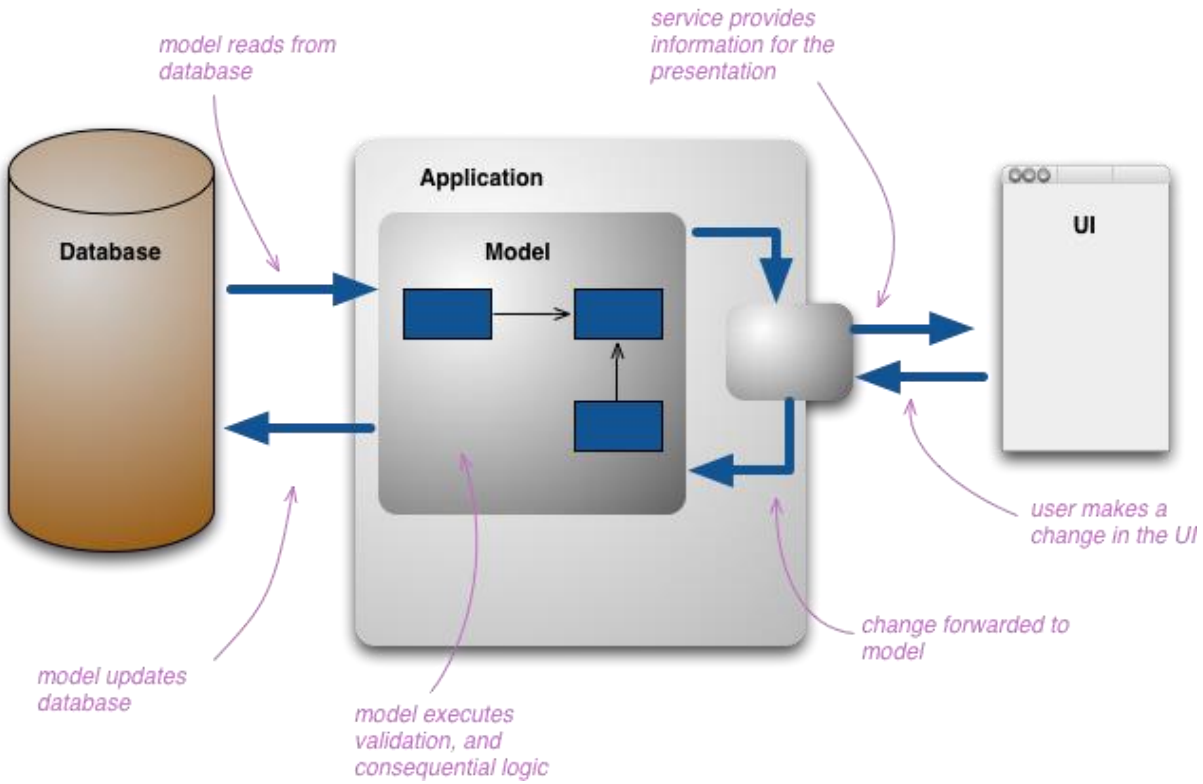


The common approach

- What is specific in that approach? If we focus a specific bounded context, then we have…
  One model for all types of operations
- If we focus a specific part of UI, then again we have…
  One model for all types of operations
- Although in most cases it is ok, sometimes it doesn't…

The common approach

- From a flow perspective



Nature of business applications

- Business requirements usually can be represented by use cases
- Uses cases can be generally split into:
  - Ones in which user want to **modify** data
  - Ones in which user want to **search** and **read** data

Let’s introduce some definitions…

CQS

- First pattern which applies that observation is CQS Command-Query Separation

> *It states that every method should either be a command that performs an action, or a query that returns data to the caller, but not both. In other words, asking a question should not change the answer. (Wikipedia, CQS)*

- Godfather: Bertrand Meyer
- To put it simply, it separates methods for

Commands Queries

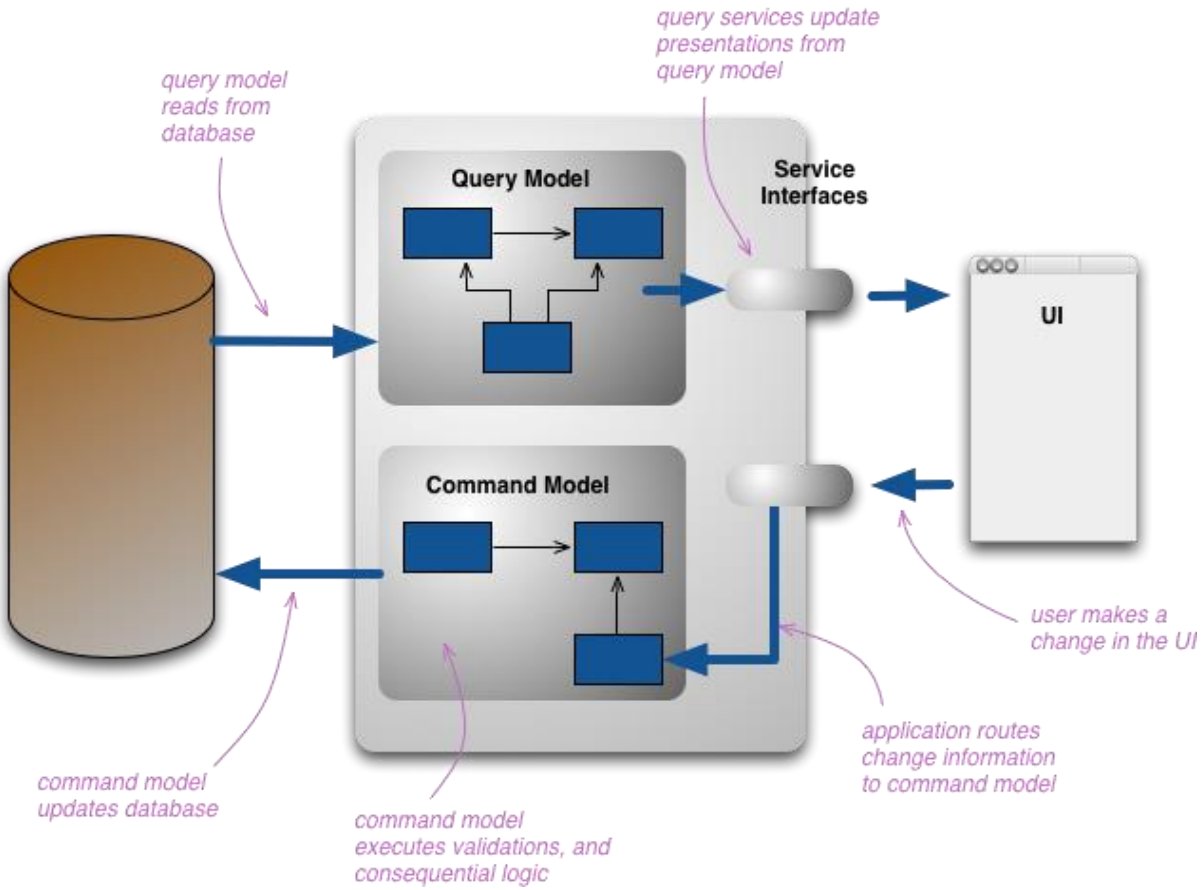
CQRS

- CQRS: Command-Query Responsiblity Segregation
- How to put it in a shorter way?

> *CQRS is simply the creation of two objects where there was previously only one*
>
> *CQRS by Greg Young*

- Godfathers: Greg Young and Udi Dahan

CQRS


query services update
presentations from
query model
query model
reads from
database
Query Model
Service
Interfaces
UI
Command Model
user makes a
change in the UI
application routes
change information
to command model
command model
updates database
command model
executes validations, and
consequential logic

Why to create separate models?

Command

- Targets a single Aggregate
- Validation rules
- Examples:
  - CreatePost, AddComment, SubmitOrder, LockUser
- Optimization for update

Query

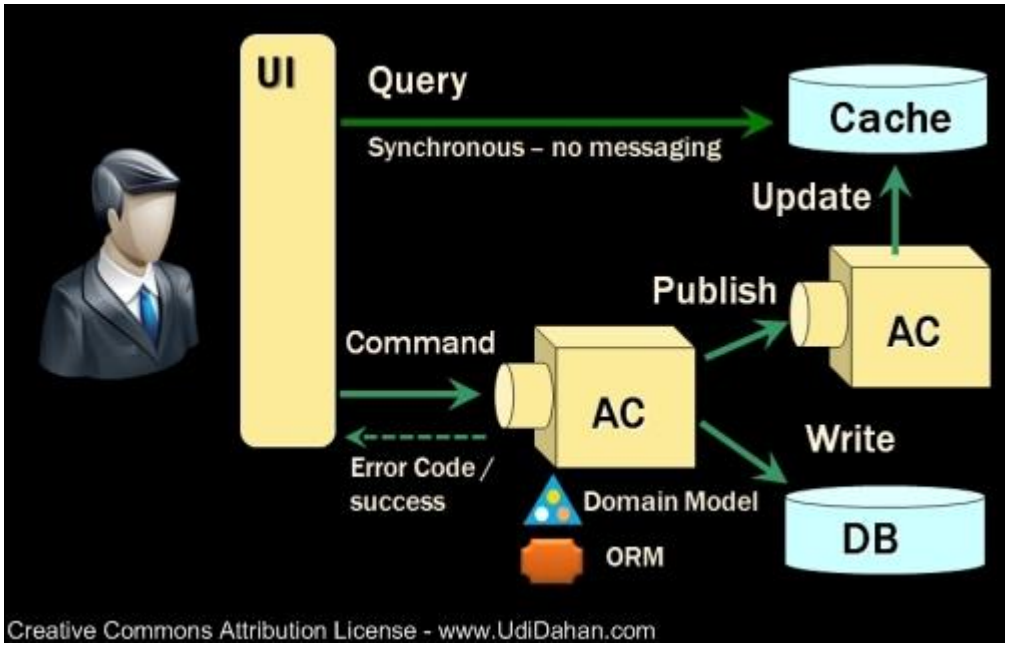
- Collapsing multiple records into one
- Forming virtual records by combining information for different places
- No validation needed
- Optimization for searching

Implementation

- Different models
  - Only in presentation layer
  - Both in domain layer and then presentation layer
  - On database layer with 2 different databases
- It depends on the scenario and **WHY** we need CQRS
- Vaugn remind to apply that segregation on the repository level as well
- Introducing a Command Handler (Agent) to avoid problems with concurrency

Implementation

- Another picture to see more details



A few more things about commands

- What is command?
  - Actually is a request for change, a simple object
- Sometimes we expect that commands should be queued
- Another thing is that command should be able to process without quering for a data

But this is not only about models
– a new mindset is needed in creating UI

Synchronization

- Depending on implementation, we would need synchronization.
  - If datastore is single – we need to decide if we need synch models
  - If there are 2 datastores – definitely synchronization is needed
- Usually it is done by publishing events
- However, an old/new term occurs in that context…

Eventual consistency

- An oposite to the transactional consistency
- It guaranties that model become consistent

  at some point in time…

  … but not immediately.
- So, what are the delayes?
  - It depends: seconds, minutes, days…
- Very often encountered in aggregates considerations, but…
  - Can be found in other considerations, e.g. databases
- What if something go wrong?
  - Well, then sometimes we may have a big problem

Eventual consistency

Now, we can apply that technique to models

Implementation: usually based on domain events

Benefits

- If we have collaborative environment, we can reduce concurrency problems by applying CQRS
- If we want to cover both queries and modifications, model may become quite complicated. Separation following CQRS pattern can reduce complexity.
- If we expect high performance solution, then separate models allows to apply different optimizations' strategies for every type of operation

References

- Article by Martin Fowler
  http://martinfowler.com/bliki/CQRS.html
- Great introduction by Udi Dahan
  http://www.udidahan.com/2009/12/09/clarified-cqrs/
- Good presentations
  http://www.slideshare.net/jeppec/cqrs-why-what-how
  http://www.slideshare.net/brianritchie1/cqrs-command-query-responsibility-segregation
- Nice introduction
  http://cqrs.nu/

References

- CQRS document by Greg Young
  http://cqrs.files.wordpress.com/2010/11/cqrs_documents.pdf
- Article by Microsoft
  http://msdn.microsoft.com/en-us/library/dn568103.aspx
- DDD/CQRS sample application
  http://cqrssample.codeplex.com/
- REST API and CQRS
  http://www.infoq.com/articles/rest-api-on-cqrs