CQRS and Event Sourcing

Oliver Sturm • @olivers • oliver@oliversturm.com

Oliver Sturm

- Training Director at DevExpress

- Consultant, trainer, author, software architect and developer for over 25 years

- Microsoft C# MVP

- Contact: oliver@oliversturm.com

Agenda

- CQRS - Why? When? How?

  - Sometimes there are choices
  - Sometimes the decision is natural
  - Consequences

- Event Sourcing

  - Again: Why? When? How?

- Eventual consistency

Data access, "traditionally"

```csharp
ImportantData editObject;

protected override void OnInit(EventArgs e) {
  editObject = LoadEditObject();
  control.DataSource = editObject;
  control.DataBind();
}

protected void Page_Load(object sender, EventArgs e) {
  if (IsPostBack) {
    MergeEditorChanges(editObject);
    SaveObject(editObject);
  } ...
}
```

Data access, "traditionally"

- Objects are loaded into memory
- Data is shown in UI
- Changes are submitted
- Loaded objects are modified
- Local change detection optimizes process of persistence

CQRS — Why?

- Because "loading data for visualization" doesn't have the same requirements as "persisting data"
- Because one loading process can be different from another
- Because one persistence process can be different from another
- Because we can save time in "page cycle" environments
- Because separate execution paths are easier to test and maintain

CQRS — When?

- Almost anytime!
- Typical doubts:

  - Pure client app — do I benefit?
  - More complex structure == more complicated maintenance work?
  - But what about ORM?

CQRS — How?

- Separate execution paths for data reading and writing
- Consider modeling changes as commands
- Consider efficient data models to support business operations
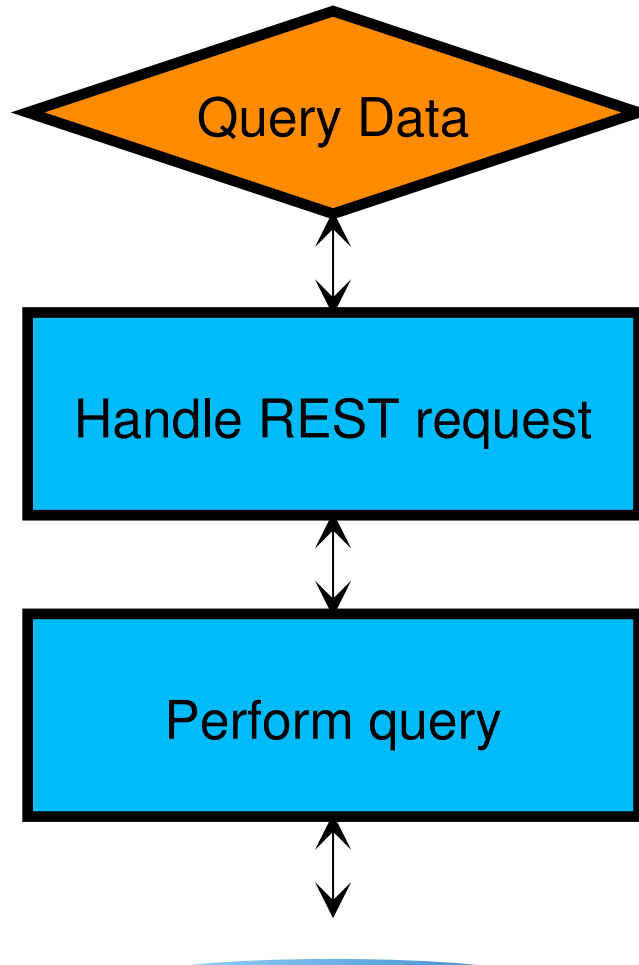
Querying data

webapp (client) Query Data

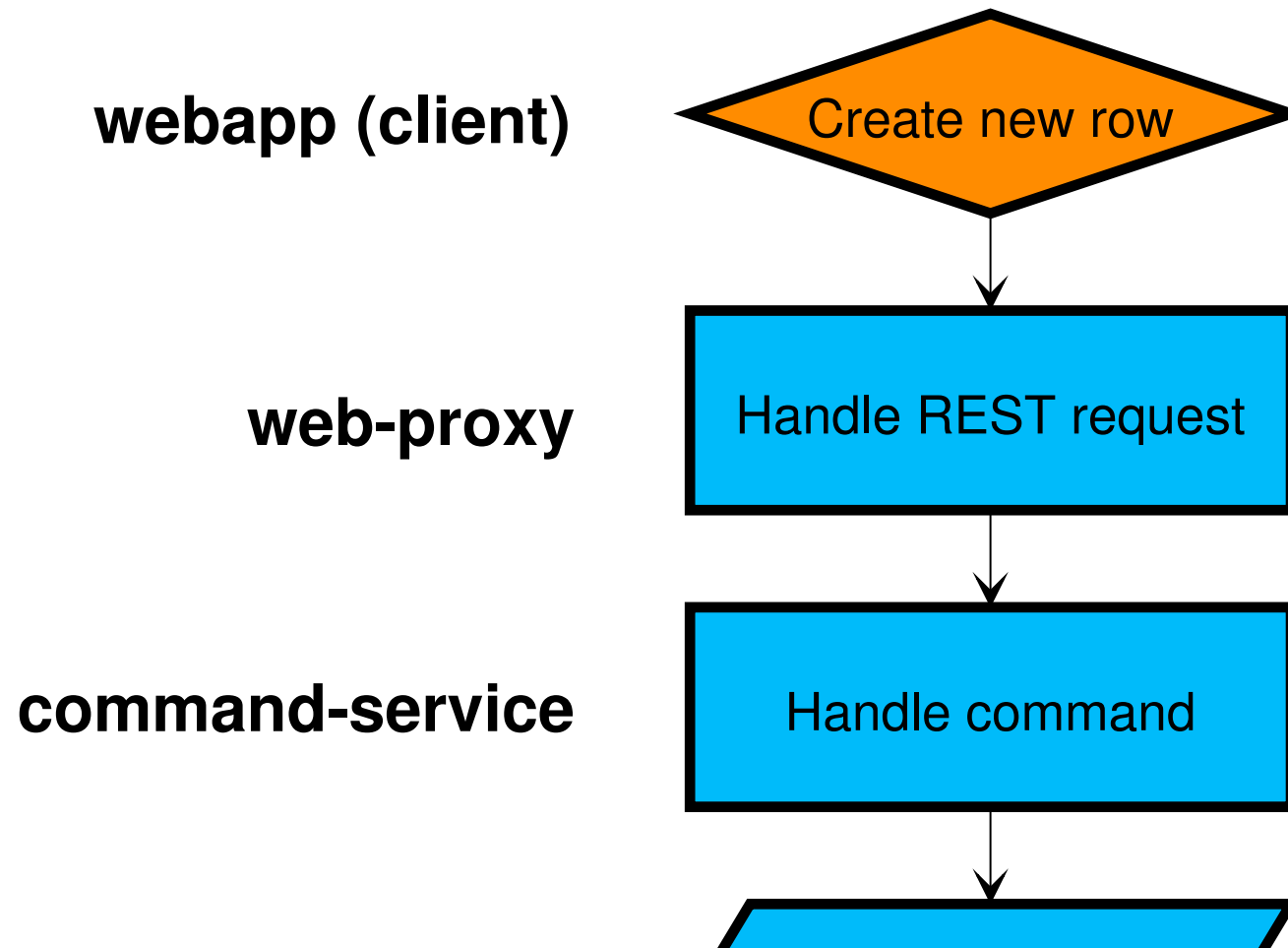**web-proxy** Handle REST request

query-service Perform query

mongo

Return data

Creating a new row

webapp (client) — Create new row

web-proxy — Handle REST request

command-service — Handle command

mongo New Row

Event Sourcing

- Starting from *command* idea

  - Primarily persist events, instead of data
  - Append-only event log
  - Derive entity state at any time, for any point in time

- Entities/Aggregates/domain objects
- Optimizations: snapshots, projections, (persistent) read models

Event Sourcing — Why?

- Events describe what the system was asked to do, any technical consequences of an event are not set in stone. Fantastic for long-term maintenance!
- Clean, extensible and scalable structure, supporting strict separations of concerns.
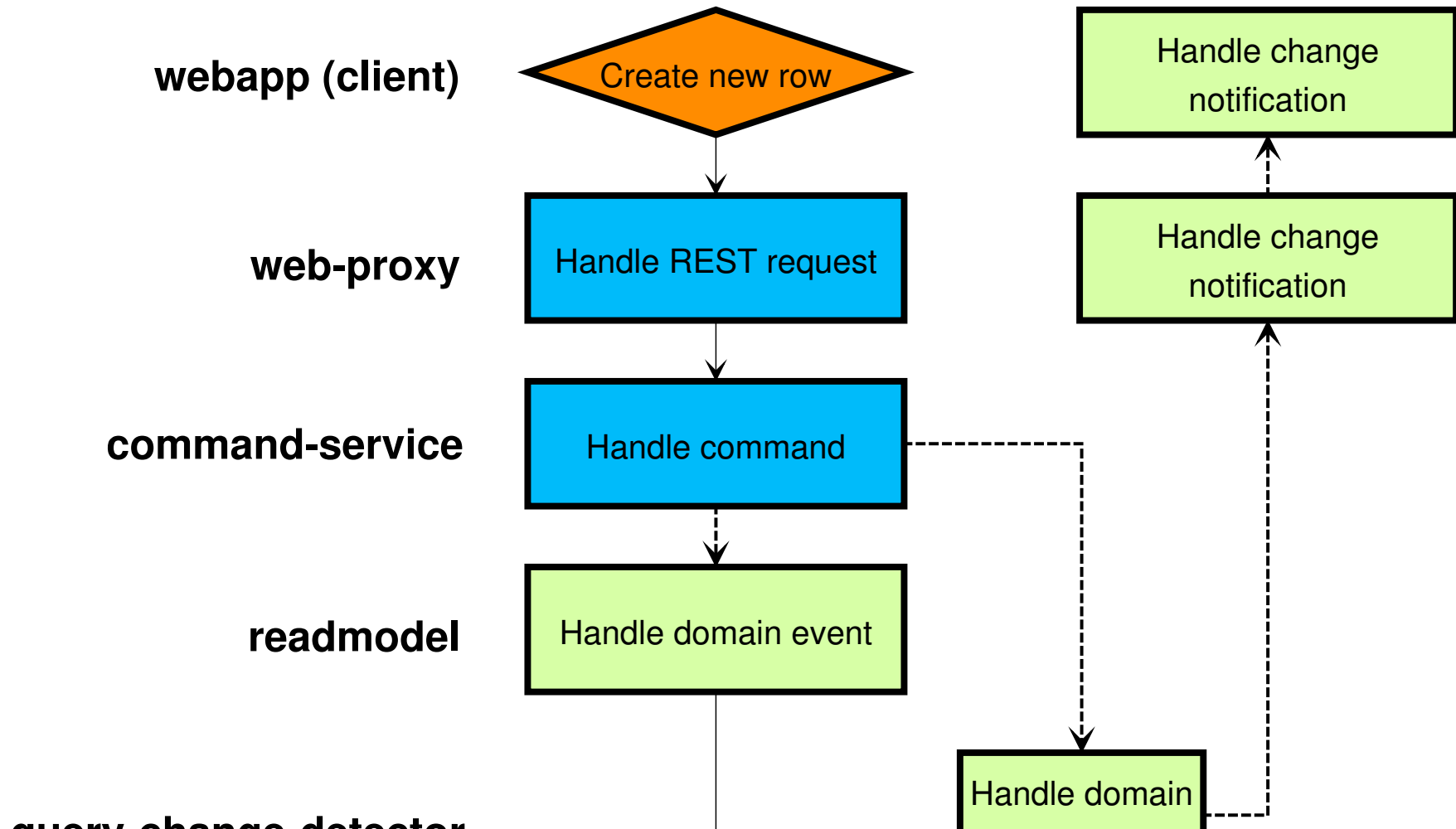- Event Storming — very practical planning method

Event Sourcing — When?

- Tempting pattern for many applications, but with structural consequences

    - In-process? Possible…

Event Sourcing — How?

- Any service can receive commands
- Raising domain events across service boundaries requires communication infrastructure
- Persisting events and possibly read models requires a persistence layer
- Structural maintenance of aggregates and projections is a bit fiddly
- Recommended: use libraries existing for all platforms
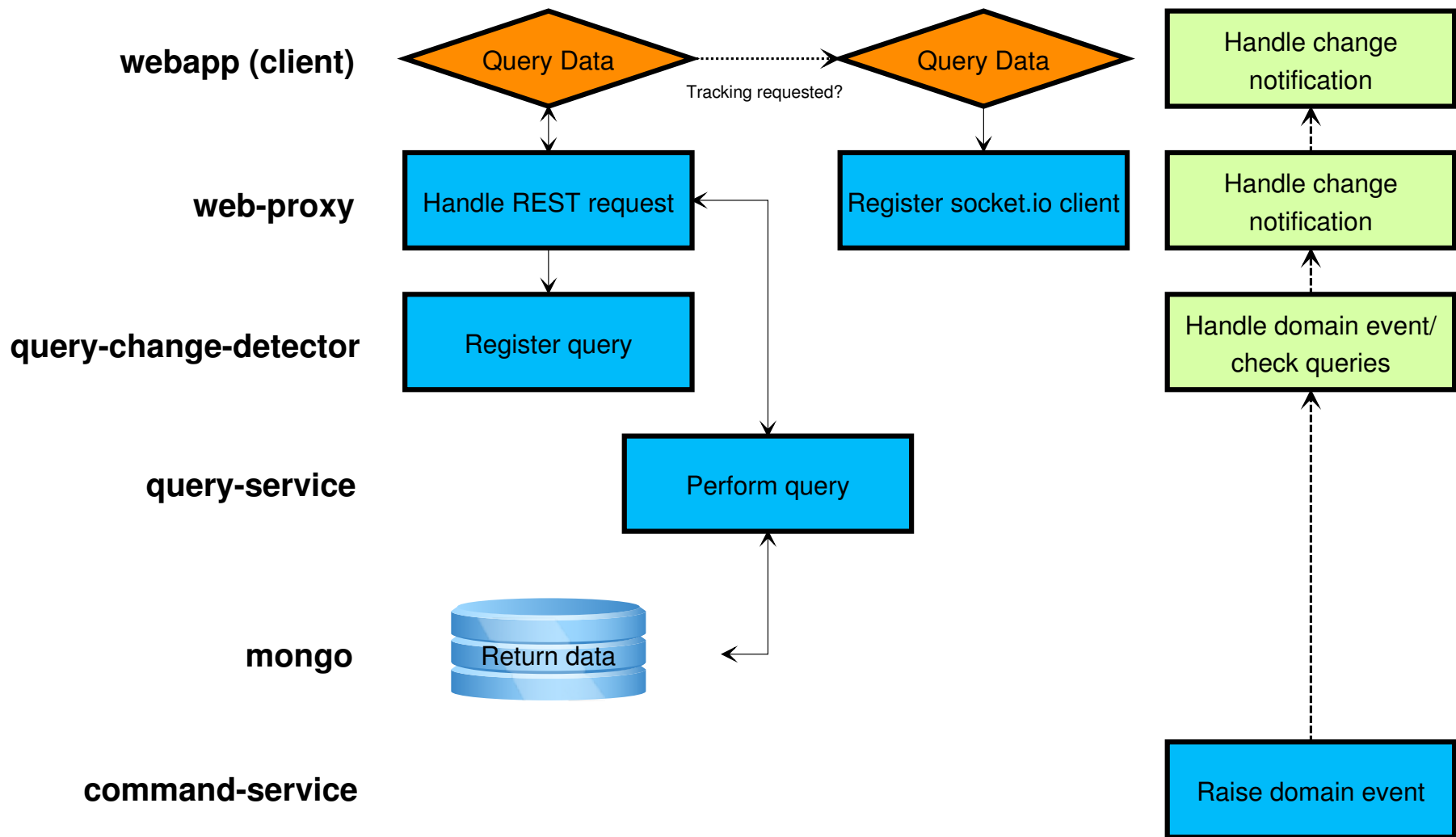
Creating a new row with CQRS/ES

webapp (client)

Create new row

web-proxy

Handle REST request

Handle change notification

command-service

Handle command

Handle change notification

readmodel

Handle domain event

Handle domain

query change detector

query-change-detector

mongo

New Row

event

Querying data with CQRS/ES

webapp (client)	Query Data	Query Data	Handle change notification
web-proxy	Handle REST request	Register socket.io client	Handle change notification
query-change-detector	Register query		Handle domain event/ check queries
query-service	Perform query		
mongo	Return data		
command-service			Raise domain event

Tracking requested?

Eventual consistency

Consistency is over-rated (Greg Young, Mr CQRS)

- General issue in distributed systems - CAP theorem
- Eventual consistency exists in the real world. Starbucks?
- How eventual are things in your system?
- Business logic needs to deal with issues resulting from eventual consistency

    - Compensation
    - Special programming tactics
    - Check this out: http://queue.acm.org/detail.cfm?id=2462076

Sources

- This presentation:

  - https://oliversturm.github.io/cqrs-event-sourcing
  - Deprettified content in pdf format: https://oliversturm.github.io/cqrs-event-sourcing/slidecontent.pdf

- Demo code:

  - https://github.com/oliversturm/cqrs-grid-demo (check *event-sourcing* branch)

- Talk to Seneca

  - https://github.com/oliversturm/talk-to-seneca

Thank You

Please feel free to contact me about the content anytime.

oliver@oliversturm.com