

# 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
- Contact: [oliver@oliversturm.com](mailto: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"

```
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"**

1. Objects are loaded into memory
2. Data is shown in UI
3. Changes are submitted
4. Loaded objects are modified
5. 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?

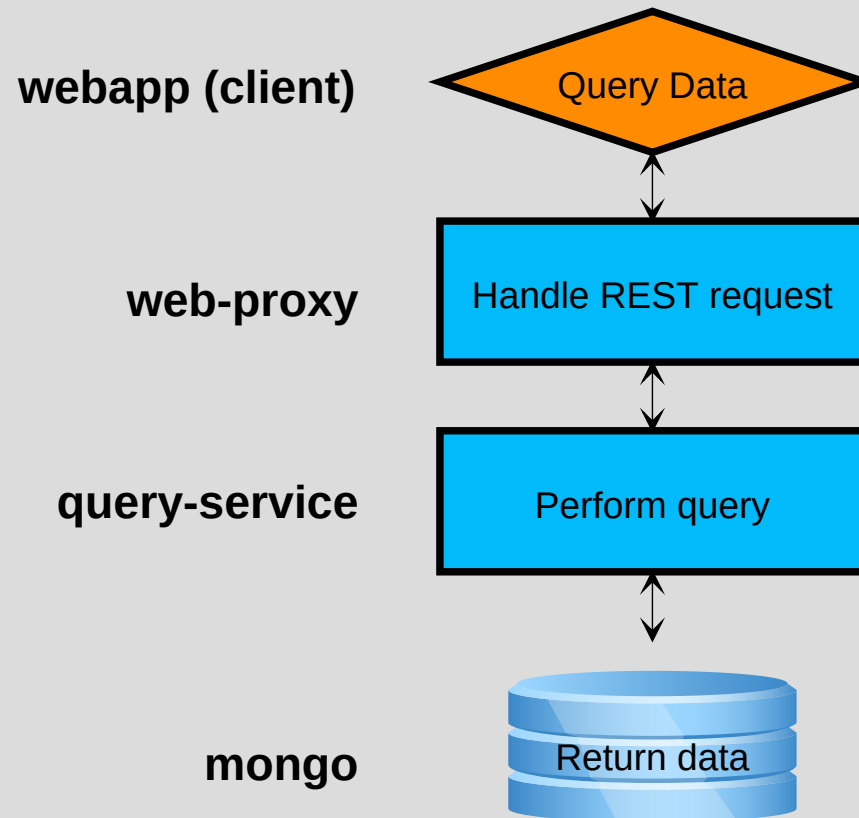
- Almost *anytime*!
- Typical doubts:
  - Pure client app — do I benefit?
  - More complex structure == more complicated maintenance work?
  - But what about ORM?
- Reality:
  - Structural advantages *benefit any architecture*
  - *Complex != complicated*, complex systems can have simple parts
  - Maybe we don't always need ORM...

# CQRS — How?

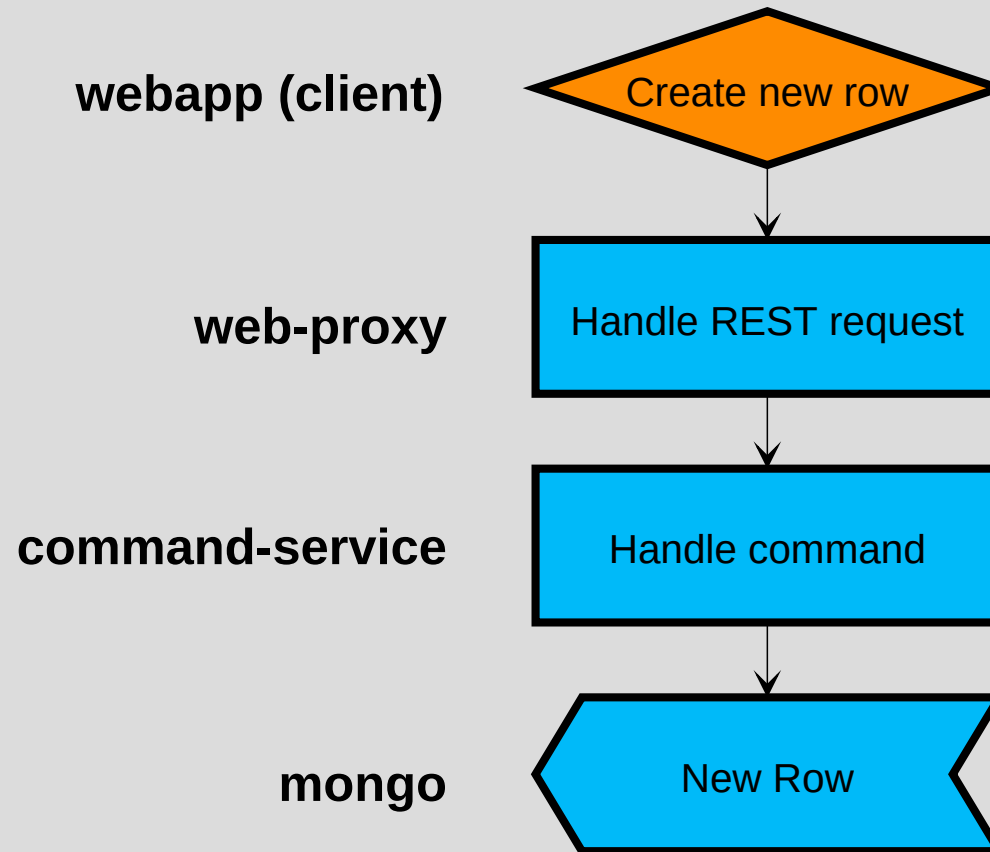
- *Separate execution paths* for data reading and writing
- Consider modeling *changes as commands*
- Consider *efficient data models* to support business operations



# QUERYING DATA



# CREATING A 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 supports *strict separations of concerns*
- Event Storming — very practical planning method

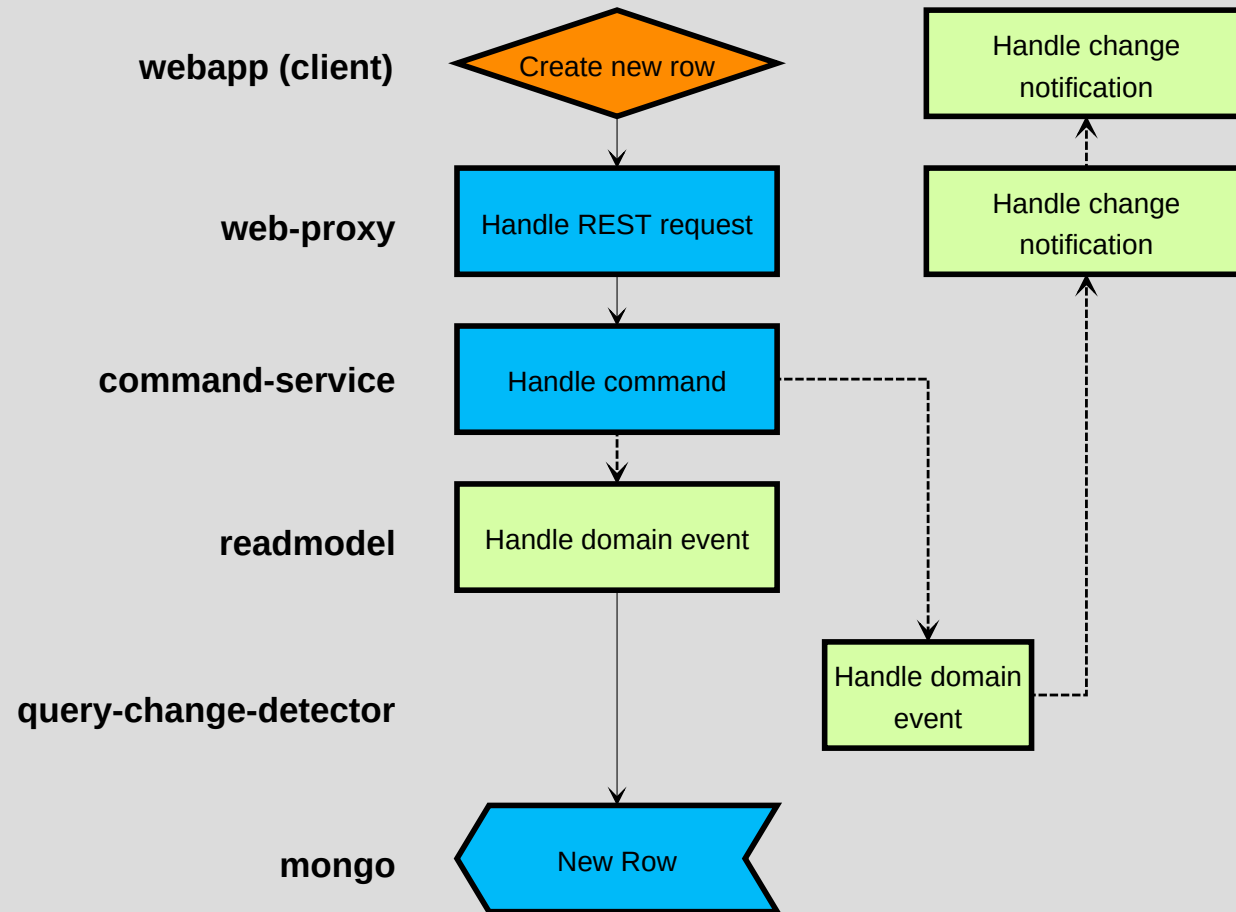
# EVENT SOURCING — WHEN?

- Tempting pattern for many applications, but with structural consequences (complexity)
- Very "clean complexity"
- *In real-world well structured service based apps generally a good recommendation*
- In-process, in full-fat clients? Possible...

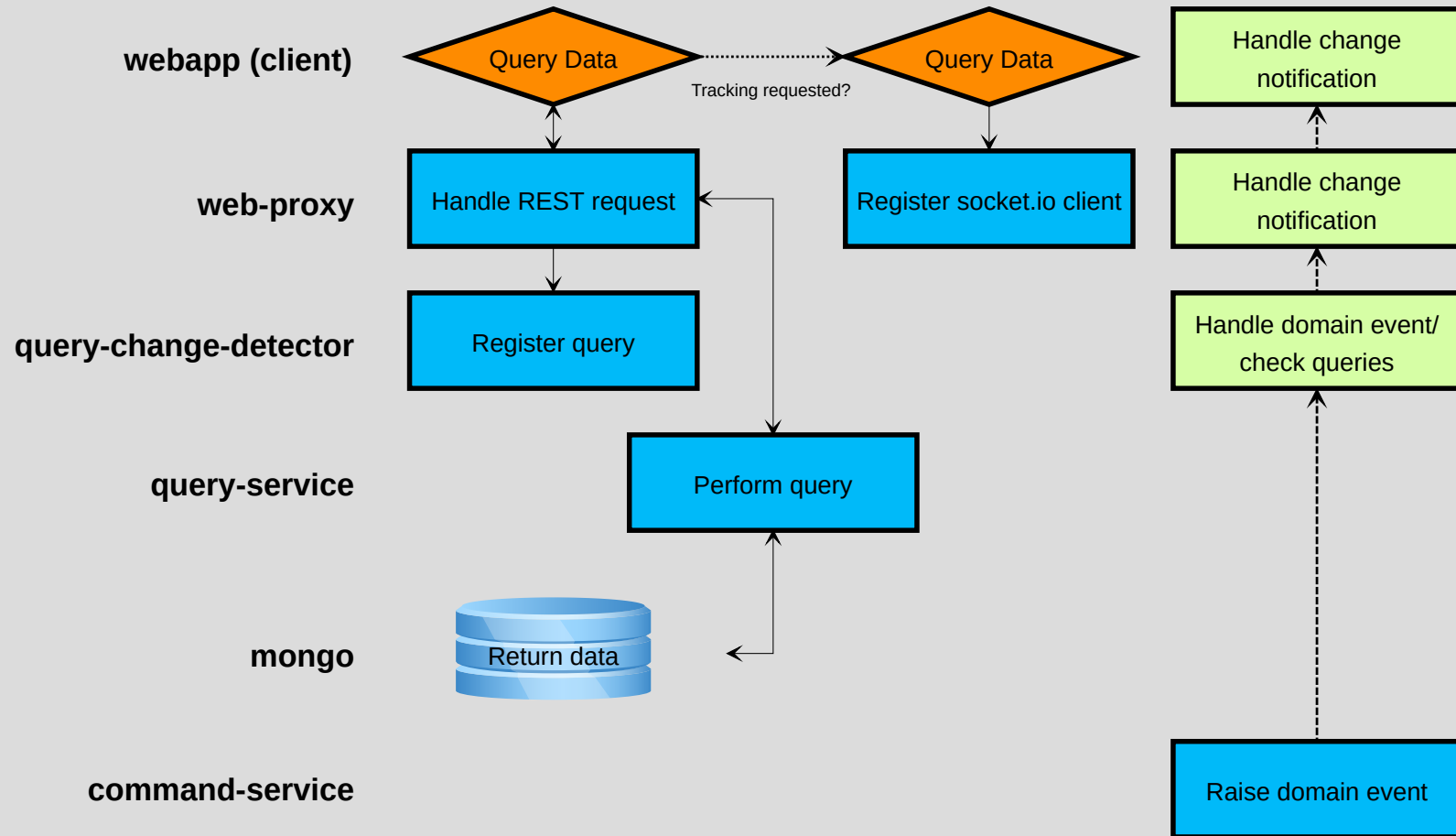
# EVENT SOURCING — How?

- Easy part: *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, especially in typed languages
- Recommended: *use libraries existing for all platforms*

# CREATING A NEW ROW WITH CQRS/ES



# QUERYING DATA WITH CQRS/ES





# 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>
  - PDF download: <https://oliversturm.github.io/cqrs-event-sourcing/slides.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 Sturm • @olivers • oliver@oliversturm.com

