EDA in MVC using F#

Event Driven Architecture in MVC using F#

Agenda

- EDA
  - Concept
  - Why & When
- F#
  - Basic ideas
  - Benefits
  - Why & When
- Project overview
  - Command Pattern
  - Events
  - Event Sourcing (may be)

Why Event Driven Architecture

- EDA (Event Driven Architecture) is a methodology for building systems in which events flow between *decoupled components* and services
    - *Detect and consumption of events*
- There is no concept of Request/Response
- The event container is building up all the events for the consumer(s) which makes it easy to test (Dep. Inj. S in SOLID)
- Easy to implement Event Sourcing

  "Capture all changes to an application state as a sequence of events"

  (Martin Fowler)

  … "Capture all behaviors as a sequence of events"

When Event Driven Architecture

- EDA has Business value, you can easily extend the system with new things (events)
- Great decoupling make it easy to scale
- Component style Architecture
- Promote Single Responsibility
- Bring SOA to the next level
- Easy to implement Event Sourcing
  - **Used primarily to replay all past events to bring aggregate into current state**

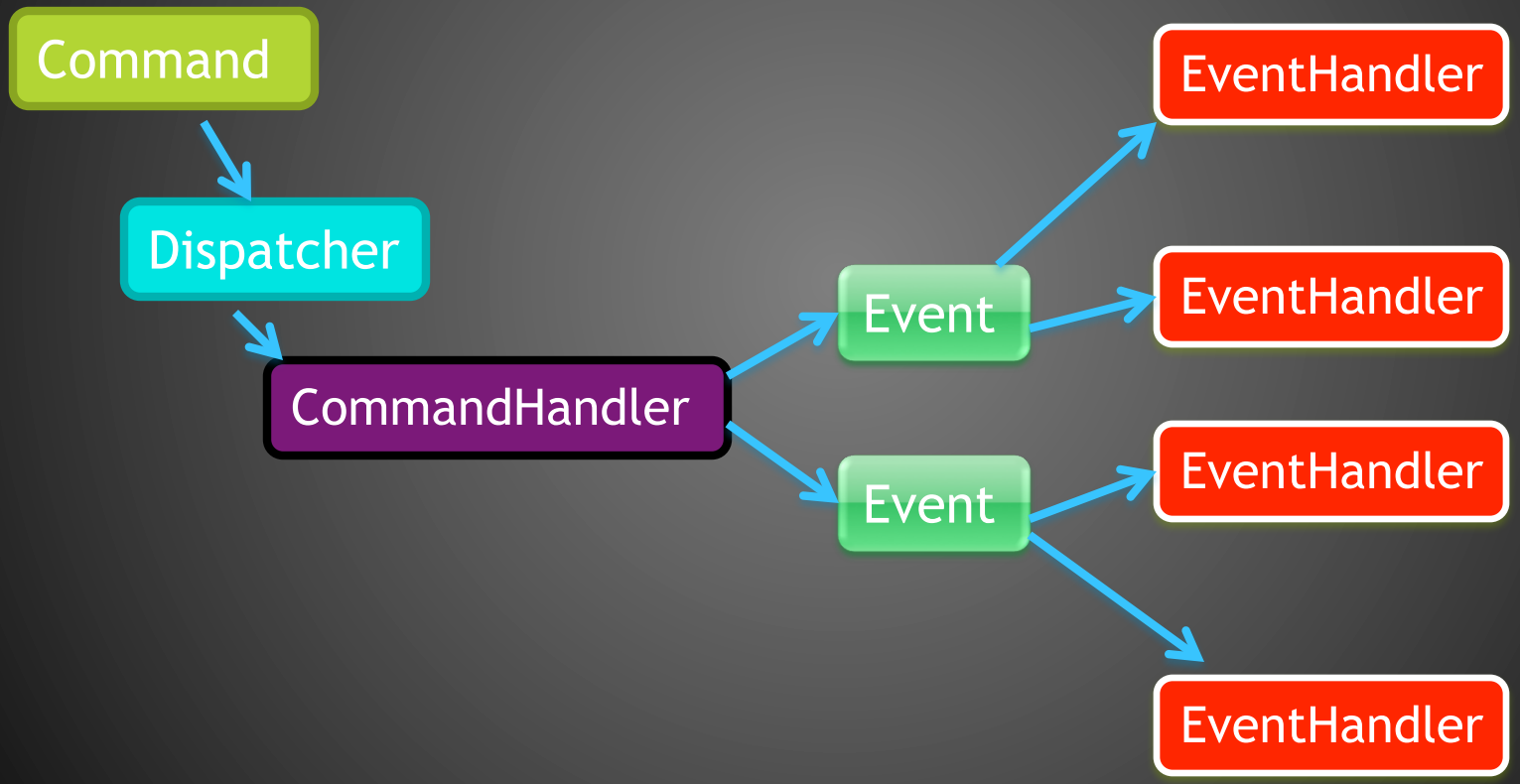
Commands

- Commands are directives to perform some action to the domain (Behaviors)
- Commands can be rejected by the domain (validation)
- Processing a Command will result in 0:n Events raised
- Commands are imperative and are components of the Ubiquitous Language to describe the domain (PlaceOrderCommand)

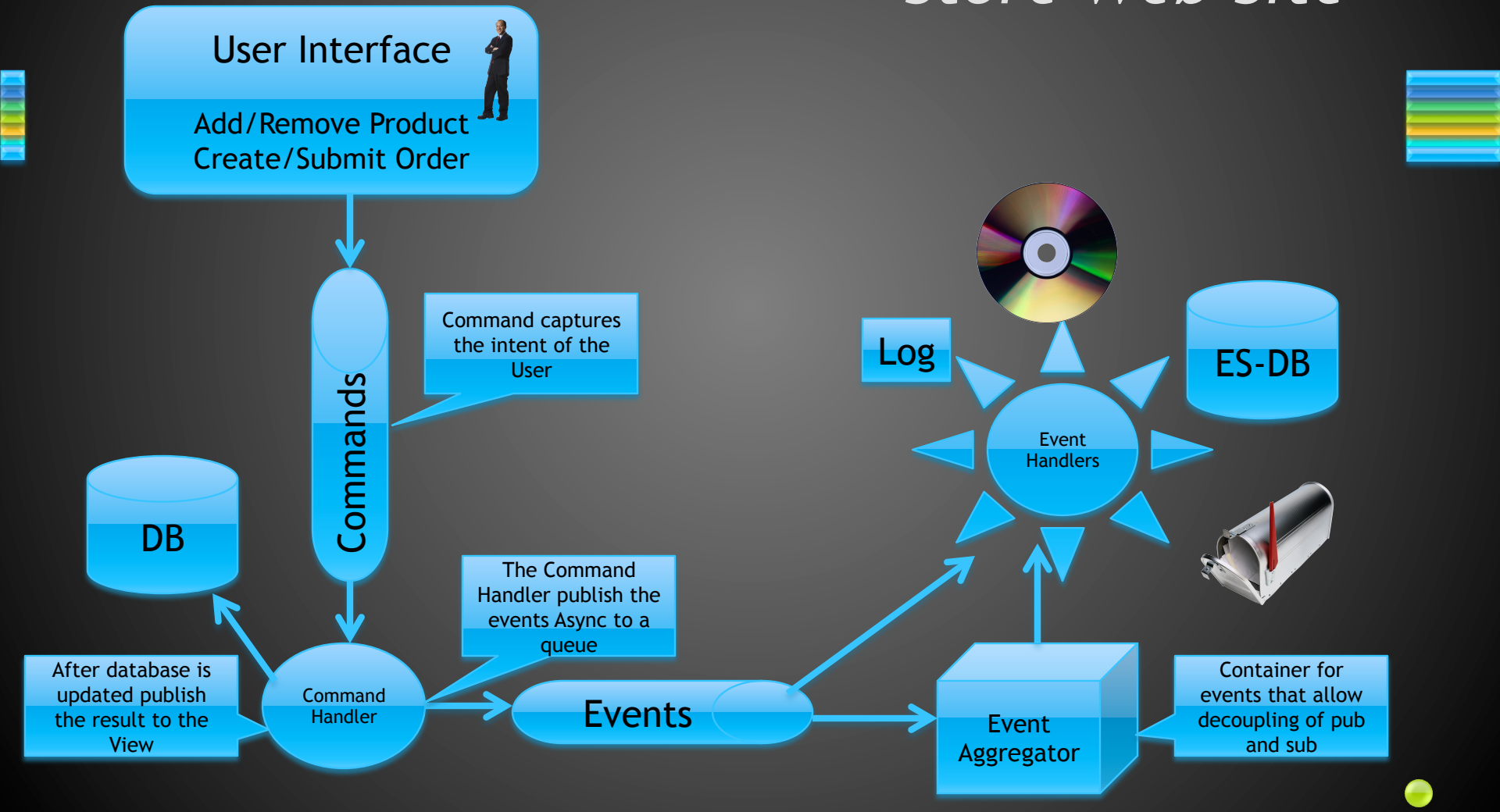
Events

- Events are the result of some action already happened
- Events may never be rejected
- Events are in past-tense and are components of the Ubiquitous Language to describe the domain (OrderPlacedEvent)

Commands & Events



Project Overview
Store Web Site



Why F#

- Powerful type inference
- Support REPL
- Support for OOP
- Higher Oder Function
- Avoid nulls
- Record Type
  - Immutable
  - No inheritance
  - Structural equality, comparison
- Discrimination Union
- Object Expression
- Full .Net Library support and interoperability with other .Net languages
- Active Patterns
- Pattern Matching
- Pipe operator |>
- Type Providers
- Support Actor model

When F#

- F# is not a silver bullet, is great tool to know!
- Fast and easy prototyping
- Fast and easy development
- Data Modeling
- Need concurrency (immutable as default)
- Scalability (Async I/O)
- Big Data

github.com/DCFsharp

meetup.com/DC-fsharp/

@DCFsharp

riccardo@teknology360.com