

CQRS 101

TDD Milano, 6 Giugno 2007

Origine

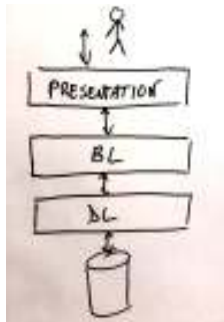
- CQS, Command Query Separation, Mayer, 1988
“Un *metodo* per la scrittura uno per la lettura”
- CQRS, Command Query Responsibility Segregation, Fowler, 2011
“Un *modello* per la scrittura, uno per la lettura”

Da Big Ball of Mud a CQRS

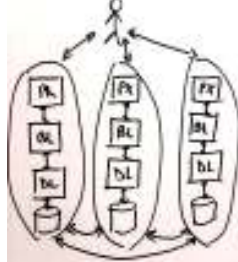
Big ball of mud



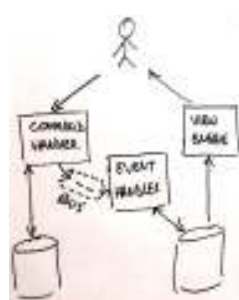
N-Tier



Microservices

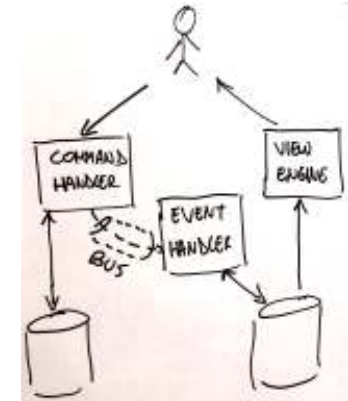


CQRS



CQRS in 2 minuti

- Command Handler
- Database principale
- Bus
- Event Handler
- Views Database
- View Engine



Activity Manager

Un semplice Timesheet – 1^parte

Premesse - Inversion of Control

- Johnson e Foote, 1988
- Gli oggetti ricevono il «flusso di controllo» da un framework
- Si estendono le funzionalità iniettando servizi
- Vengono disaccoppiate fisicamente le implementazioni dei servizi dalle interfacce
- Permette di gestire le dipendenze in modo automatico
- Facilita il mantenimento di una struttura «pulita»

Premesse - Dependency Injection

- Martin (Uncle Bob), 1996
- Hollywood Principle: «Don't call us, we'll call you»
- A compile time non si conoscono le implementazioni
- Gli oggetti concreti dipendono da interfacce
- Il ciclo di vita degli oggetti è gestito tramite un *Container*

Requisiti

- Azioni
 - Inserimento dell'inizio dell'attività
 - Completamento di un'attività
 - Possono essere inserite massimo 8 ore al giorno
- Visualizzazione
 - Attività completate
 - Attività da completare
- Tecnici
 - Ogni utente avrà la sua applicazione personale
 - Il tipo di database verrà scelto in seguito

Processo di sviluppo

- Identificazione dei Comandi (Use cases di scrittura) e write model
- Identificazione delle Viste (Use cases di lettura) e read model
- Test e implementazione dei Command Handler
- Implementazione degli Event Handlers
- Aggiunta dei servizi/API per accedere alle viste ed invocare i comandi
- Test, implementazione ed integrazione della validazione

Activity Manager

Aggiunta dei tipi di attività – 2^a parte

Premesse – Value Objects

- Invarianti, rispetto al contesto
- Sono utilizzati ma NON gestiti
- Idealmente sono forniti da servizi esterni non sotto il nostro controllo
- In generale, se possono essere astratti ad un valore numerico (o un id, possono essere qualificati come VO

Requisiti

- Azioni:
 - Assegnare ad ogni attività un tipo
 - Modificare il tipo dell'attività se non ancora completate
 - Gestione dei tipi di attività
- Visualizzazione
 - Rapporto di tutte le attività svolte per giornata/tipo

Activity Manager

Multi-tenancy – 3^a parte

Requisiti

- Azioni:
 - Inserimento, disattivazione e modifica di una società
 - Inserimento, disattivazione e modifica di un utente
 - Assegnazione di un utente ad una società
 - Assegnazione di un utente Admin
- Visualizzazione
 - Visualizzazione di tutte le attività per utente
 - Realizzazione di un rapporto per società dato un periodo di tempo

Conclusioni

- Soluzione semplicemente testabile
- Separazione precisa delle funzionalità
- Scalabilità
- Estendibilità
- Aderenza alla visione del business

Riferimenti

- Eric Evans, Domain-Driven Design, Addison-Wesley, 2003
- Jimmy Nilsson, Applying Domain-Driven Design and Patterns, Pearson, 2006
- Vaughn Vernon, Implementing Domain Driven Design, Pearson, 2013
- Martin Fowler, <https://martinfowler.com/>
- Jimmy Bogard, <https://jimmybogard.com/>

Grazie per l'attenzione

- Enrico Da Ros:
- E-mail: edr@kendar.org
- LinkedIn: <https://www.linkedin.com/in/enricodaros/>
- Github per slide e codice: <https://github.com/kendarorg/CQRS101>