BOTTEGA IT minds

Program szkolenia:

Implementacja Domain Driven Design - wzorce architektoniczne (część 2)

Informacje:

Nazwa:	**Implementacja Domain Driven Design - wzorce architektoniczne (część 2)**
Kod:	**ddd-workshop-DDD-impl**
Kategoria:	Warsztaty eksperckie DDD
Grupa docelowa:	architekci developerzy
Czas trwania:	2 dni
Forma:	50% wykłady / 50% warsztaty

Pragmatyczne podejście do implementacji DDD w wybranej technologii: Java, NET, PHP, RoR.

Zakres

Techniczne aspekty implementacji DDD

- architektury aplikacji: zarządzanie złożonością dzięki rozwarstwieniu logiki,
- architektury systemowej: integracja modułów, architektura zdarzeniowa zwiększająca responsywność i otwierająca na pluginy,
- skalowania systemu - architektura CqRS,
- konkretne techniki implementacji z wykorzystaniem popularnych stosów technologicznych,
- najlepsze praktyki Clean Code i dbanie o wysokie testability.

Techniki modelowania problemów biznesowych

Techniki modelowanie DDD (wzorce strategiczne i taktyczne oraz techniki lingwistyczne i wizualne) są omawiane na szkoleniu DDD-modelowanie, które powinno nastąpić w pierwszej kolejności, przed szkoleniem z zakresu implementacji.

Forma

Szkolenie bazuje na modelu stworzonym podczas szkolenia z zakresu modelowania, które poprzedza niniejsze szkolenie.

Podczas wykładów trener omawia najlepsze praktyki implementacji wzorców architektury aplikacyjnej i Building Blocks DDD
Podczas warsztatów implementujemy dwa moduły systemu klasy ERP. Kolejne zadania polegają na przyrostowym dodawaniu nowych funkcjonalności w sposób ilustrujący zagadnienia teoretyczne poznane podczas poprzedzającego je wykładu.
Podczas dyskusji uczestnicy mają dostęp do wiedzy technicznej trenera oraz mają możliwość zweryfikowania swoich rozwiązań z wypracowanymi przez innych uczestników szkolenia.

W czasie warsztatów uczestnicy rozwiązują postawione przed nimi problemy pracując w parach (Pair Programming), zmieniając po każdym zadaniu role: Pilot i Driver. Technika ta ma za na celu umożliwienie spojrzenia na zagadnienia z różnych perspektyw, aktywując jednocześnie więcej zasobów kognitywnych.

W ramach szkolenia omawiamy oraz ćwiczymy w praktyce zarówno podstawowe jak i zaawansowane techniki DDD, takie jak: wzorce Building Blocks, wypracowanie Ubiquitous Language oraz zestaw technik Strategic Design.

Projekt referencyjny

Sprawdź naszą implementację przykładowego projektu DDD+CqRS: Sample Leaven.

Zalety szkolenia:

- Omówienie najczęstszych błędów w implementacji Agregatów
- Najlepsze praktyki i pułapki podczas stosowania ORM i IoC
- Nowoczesne architektury: CqRS, Event Driven, Event Sourcing

BOTTEGA IT minds

Szczegółowy program:

1. Architektura aplikacji - praktyczna implementacja koncepcji architektonicznych

- 1.1. Wykorzystanie IoC przy projektowaniu z myślą o jakości - otwartość projektu na rozbudowę i testy
  - 1.1.1. Dependency Injection - pluginy zmieniające zachowanie systemu
  - 1.1.2. Zdarzenia - pluginy dodające nowe zachowanie
  - 1.1.3. AOP - aspekty ortogonalne (transakcje, bezpieczeństwo, audyty)
- 1.2. Podejście warstwowe - rozmieszczenie building blocks na warstwach
  - 1.2.1. Strukturyzacja systemu
    - 1.2.1.1. Moduły - budowanie
    - 1.2.1.2. Pakiety/namespaces
  - 1.2.2. Warstwa prezentacji
    - 1.2.2.1. Ochrona domeny przed wyciekaniem do UI
  - 1.2.3. Logika aplikacyjna
    - 1.2.3.1. Projektowanie API systemu
    - 1.2.3.2. Problem uczciwego API - wersjonowanie obiektów
  - 1.2.4. Logika domenowa (Building Blocks DDD)
    - 1.2.4.1. Widoczność klas - wykorzystanie zasięgu pakietowego
  - 1.2.5. Warstwa infrastruktury
    - 1.2.5.1. Wyzwania ORM
    - 1.2.5.2. Pułapki architektury zdarzeniowej

2. Architektura systemu

- 2.1. SOA
  - 2.1.1. Zasada jednego źródła prawdy
  - 2.1.2. Unikanie antywzorce "Corporate Model"

BOTTEGA IT minds

BO·TT·EGA
IT minds

- 3.4. Serwisy Domenowe
  - 3.4.1. Zarządzanie przez IoC
- 3.5. Repozytoria
  - 3.5.1. Wstrzykiwanie zależności
  - 3.5.2. Aspekty ORM
    - 3.5.2.1. Odczyt EAGER Agregatów o dobrze zaprojektowanej granicy
    - 3.5.2.2. Kaskadowy zapis Agregatów o dobrze zaprojektowanej granicy
    - 3.5.2.3. 3 podejścia do optymistycznego blokowania
    - 3.5.2.4. Odczyt agregatów: blokowanie READ w celu zabezpieczenia obiektów tworzonych na podstawie odczytywanych agregatów
    - 3.5.2.5. Zapis agregatów: blokowanie WRITE korzenia agregaty w celi logicznej ochrony całego agregatu
  - 3.5.3. Izolacje transakcji
  - 3.5.4. Operacje na klasie bazowej agregatu
    - 3.5.4.1. Wersjonowanie
    - 3.5.4.2. Problem identyfikacji
    - 3.5.4.3. Audyty - wykorzystanie JPA: Callbacks i Listeners
- 3.6. Fabryki
  - 3.6.1. Fabryki zarządzane przez IoC
  - 3.6.2. Wstrzykiwanie zależności do Agregatów
  - 3.6.3. Wsparcie dla testability
  - 3.6.4. Zarządzanie Couplingiem Agregatów
  - 3.6.5. Teoria Couplingu C3
    - 3.6.5.1. Call
    - 3.6.5.2. Contain
    - 3.6.5.3. Create

BO·TT·EGA
IT minds

- 3.7. Polityki (strategie)
  - 3.7.1. Dekorowanie
  - 3.7.2. Wstrzykiwanie polityk jako pluginów
- 3.8. Zdarzenia biznesowe
  - 3.8.1. Podejście asynchroniczne
  - 3.8.2. Architektura pluginów
- 3.9. Specyfikacje
  - 3.9.1. Modelowanie złożonych warunków biznesowych
  - 3.9.2. Filtrowanie danych w repozytoriach
  - 3.9.3. Dyskusja nad wydajnością
- 3.10. Saga - model procesu biznesowego
  - 3.10.1. Projekt frameworka zarządzającego Sagą
  - 3.10.2. Pułapki techniczne
- 3.11. Role Object
  - 3.11.1. Model ról - gdy system powinien zachowywać się (nie tylko wyglądać) inaczej w zależności od roli domenowej użytkownika
- 3.12. Dodatkowe wzorce - rozszerzenia Building Blocks
  - 3.12.1. Dekoratory Polityk - Supple Design
  - 3.12.2. Agregat jako maszyna stanów
  - 3.12.3. Łańcuch odpowiedzialności
  - 3.12.4. Budowniczy

4. Praktyczne aspekty implementacji projektów opartych o DDD (implementacja w wybranych technologiach)

- 4.1. Building blocks – idea i implementacja w wybranych technologiach (Seam, Spring, EJB)

5. Architektura Command-query Responsibility Segregation

- 5.1. CqRS jako rozwinięcie architektury warstwowej

BO·TT·EGA IT minds

- 5.1.1. Koncepcja intencjonalnego GUI
- 5.1.2. Dwa stosy warstw
- 5.2. Stos Write
  - 5.2.1. Zakres odpowiedzialności
  - 5.2.2. Modelowanie scenariuszy (orkiestracja obiektów domenowych)
  - 5.2.3. Aspekty techniczne (zależności, transakcje)
  - 5.2.4. Implementacje
    - 5.2.4.1. Serwisy
    - 5.2.4.2. Command i CommandHandler
    - 5.2.4.3. Wykorzystanie noSQL - Bazy dokumentowe
- 5.3. Stos Read
  - 5.3.1. Problemy z wydajnością klasycznych podejść
  - 5.3.2. Modelowanie kwerend
  - 5.3.3. Projektowanie modelu do odczytu
  - 5.3.4. Implementacja
    - 5.3.4.1. SQL
    - 5.3.4.2. Widoki zmaterializowane
    - 5.3.4.3. Osobny model
    - 5.3.4.4. Uaktualnianie modelu do odczytu
    - 5.3.4.5. noSQL - bazy grafowe
- 5.4. Event Sourcing
  - 5.4.1. Sekwencja zdarzeń jako model behawioralny
  - 5.4.2. Projekcje modelu behawioralnego na model do odczytu
  - 5.4.3. Pojęcie Eventual Consistency
  - 5.4.4. Implementacja Agregatów sterowanych zdarzeniami

BOTTEGA IT minds

6. Testowanie automatyczne

- 6.1. Strategia testowania - mapowanie piramidy testów na warstwy aplikacji
  - 6.1.1. UI - end-to-end systemowe
  - 6.1.2. Logika aplikacyjna - testowanie end-to-end komponentowe
  - 6.1.3. Logika domenowa - testowanie jednostkowe
    - 6.1.3.1. Testowanie niezmienników modelu
- 6.2. Cel testowania
  - 6.2.1. Perfekcja i regresja - testy jednostkowe Building Blocks domenowych
  - 6.2.2. Akceptacje - testy API i UI
- 6.3. Techniki
  - 6.3.1. Zaślepki
    - 6.3.1.1. Mock - metody typu command
    - 6.3.1.2. Stub - metody typu query
  - 6.3.2. Konstruktory Agregatów umożliwiające wprowadzenie ich w dowolny stan
  - 6.3.3. Fabryki Agregatów zwiększające testability
  - 6.3.4. Zarządzanie Couplingiem C3
- 6.4. Wzorce
  - 6.4.1. Assembler - przygotowanie Agregatów
  - 6.4.2. Assert Object - hipotezy biznesowe Agregatów