

Program szkolenia:

Domain Driven Design - zwinne modelowanie złożonych domen

Informacje ogólne

Nazwa:	Domain Driven Design - zwinne modelowanie złożonych domen
Kod:	DDD
Kategoria:	Inżynieria oprogramowania
Grupa docelowa:	Analitycy, projektanci, architekci, programiści
Czas trwania:	1 dzień
Forma:	50% wykłady / 50% warsztaty

Szkolenie prowadzone jest w formie, która łączy następujące po sobie na przemian wykłady, warsztaty i dyskusje.

Podczas wykładów trener prezentuje kolejne rozdziały wiedzy merytorycznej, uzupełniając je o komentarze na podstawie własnych doświadczeń.

Podczas warsztatów tworzymy dwa moduły systemu lasy 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 eksperckiej 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.

Sprawdź naszą implementację przykładowego projektu DDD+CQRS: [Sample Leaven](#).

Zalety szkolenia:

- » Projektowanie zaawansowanych i krytycznych systemów
- » Realne sposoby implementacji wszystkich technik na platformie Java
- » Prezentacja alternatywnych podejść wraz z omówieniem konsekwencji

Program szkolenia:

1. Domain Driven Design - unifikacja analizy i projektowania

1.1. Zastosowanie i zalety

1.1.1. Kiedy NIE należy stosować DDD

1.1.2. DDD Lite - lekkie podejście dla małych projektów

1.1.3. Distributed DDD - zaawansowane podejście dla systemów skalowalnych

1.2. Wprowadzenie Ubiquitous Language

1.2.1. Wspólna płaszczyzna porozumienia pomiędzy Ekspertami Domenowymi i zespołem developerskim

1.3. Wzorce Building Blocks

1.3.1. Encje

1.3.2. Agregaty

1.3.2.1. Wstrzykiwanie zależności

1.3.2.2. Hermetyzacja - otwarcie na rozbudowę

1.3.3. Value objects

1.3.3.1. Zwiększenie siły wyrazu

1.3.3.2. Styl funkcyjny

1.3.3.3. Efektywne mapowanie w JPA

1.3.4. Serwisy Domenowe

1.3.4.1. Kiedy warto stosować

1.3.4.2. Transformatory danych

1.3.5. Repozytoria

1.3.5.1. Wstrzykiwanie zależności

1.3.5.2. Aspekty JPA

1.3.6. Fabryki

1.3.6.1. Walidacja

1.3.6.2. Logika biznesowa podczas składania obiektów

1.3.6.3. Wstrzykiwanie

1.3.6.4. Wsparcie dla testability

1.3.7. Polityki (strategie)

1.3.7.1. Supple Design

1.3.7.2. Dekorowanie

1.3.7.3. Wstrzykiwanie

1.3.8. Zdarzenia biznesowe

1.3.8.1. Decoupling Bounded Context

1.3.8.2. Anticorruption Layer

1.3.8.3. Podejście asynchroniczne

1.3.8.4. Architektura pluginów

1.3.9. Specyfikacje

1.3.9.1. Modelowanie złożonych warunków biznesowych

1.3.10. Rozszerzenia

1.3.10.1. Sagi - modelowanie złożonych procesów biznesowych

1.3.10.2. Dekoratory Polityk - Supple Design

1.3.10.3. Agregat jako maszyna stanów

1.3.11. Praktyczne przykłady modelowania biznesowego z wykorzystaniem Building Blocks

1.4. Techniki projektowe - uwspólnienie modelu analitycznego i projektowego

1.4.1. Podejście warstwowe

1.4.2. Przełożenie Use Case na warstwę aplikacji

1.4.3. Przełożenie modelu biznesowego na model building blocks

1.4.4. Modelowanie Bounded Context

1.4.4.1. Decoupling

1.4.4.2. Integracja

1.5. Strategiczne projektowanie

1.5.1. Określenie Core Domain

1.5.2. Integracja z domenami Generic i Supporting

1.6. Praktyczne aspekty implementacji projektów opartych o DDD (implementacja w wybranych technologiach: Seam, Spring, EJB)

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

1.6.2. Zwiększanie poziomu czytelności kodu poprzez Domain Specific Language

1.6.3. Architektura warstwowa - rozmieszczenie building blocks na warstwach

1.6.3.1. Warstwa aplikacji

1.6.3.2. Warstwa domenowa

1.6.3.3. Warstwa infrastruktury

1.6.4. Projektowanie z myślą o jakości - otwartość projektu na testy (Dependency Injection)

1.6.5. Projektowanie z myślą o rozbudowie - otwartość projektu na zmiany (Inversion of Control, Aspect Oriented)

2. Behavior Driven Development

2.1. Zintegrowany proces wytwórczy integrujący: DDD, TDD i Scrum

2.2. Techniki modelowania wymagań

2.2.1. User story, Scenario, Steps

2.2.2. Ewolucja testów w kierunku scenariuszy akceptacyjnych

2.3. Techniki tworzenia scenariuszy

2.3.1. Techniki budowania User Story

2.3.2. Używanie Ubiquitous Language