

Program szkolenia:

Wzorce projektowe oraz efektywne techniki Object Oriented dla developerów aplikacji biznesowych

Informacje ogólne

Nazwa:	Wzorce projektowe oraz efektywne techniki Object Oriented dla developerów aplikacji biznesowych
Kod:	Patterns Biz
Kategoria:	Inżynieria oprogramowania
Grupa docelowa:	Programiści i projektanci aplikacji biznesowych
Czas trwania:	2-3 dni
Forma:	50% wykłady / 50% warsztaty

Szkolenie prezentuje wybrane Wzorce Projektowe w praktycznym i niepodręcznikowym ujęciu osadzonym w kontekście aplikacji biznesowych. Wszystkie wzorce są ilustrowane przykładami zastosowania w modelowaniu logiki aplikacji i logiki biznesowej aplikacji enterprise.

W odróżnieniu od książkowych przykładów nie omawiamy programowania maszyn lub edytorów tekstu a jedynie wzorce oraz techniki, które mają zastosowanie w modelowaniu aplikacji biznesowych.

Podczas szkolenia uczestnicy nabędą zintegrowaną wiedzę na temat zdobyczy nowoczesnej inżynierii oprogramowania, m.in. bazowe koncepcje, z których wynika zarówno konstrukcja jak i potrzeba wzorców.

Podczas warsztatów praktycznych łączymy wzorce projektowe i architektoniczne wraz z wbudowanymi mechanizmami frameworków Spring/Seam (do wyboru) aby stworzyć giętkie i otwarte na rozbudowę modele biznesowe cechujące się wysokim poziomem testowalności.

Szkolenie przeznaczone dla programistów, projektantów i architektów tworzących oprogramowanie klasy biznesowej, pragnących poszerzyć swe kompetencje w zakresie profesjonalnych technik programistycznych zwiększających jakość kodu i projektu.

Zdobyta wiedza przekłada się w praktyczny sposób na produktywność mierzoną w szerszej perspektywie czasu.

Zalety szkolenia:

- » Skupienie na kontekście aplikacji biznesowych
- » Wybór jedynie użytecznych wzorców oraz technik
- » Realne przykłady

Program szkolenia:

1. Techniki Object Oriented Design.

- 1.1. Analiza Paradygmatu Object oriented i jego poprawna interpretacja.
- 1.2. Ukierunkowanie myślenia w stylu OO.
- 1.3. Najlepsze praktyki i pułapki.
- 1.4. GRASP - General Responsibility Assignment Software Patterns.
- 1.5. SOLID - Single Responsibility Principle (SRP), the Open/Closed Principle (OCP), the Liskov Substitution Principle (LSP), the Dependency Inversion Principle (DIP), and the Interface Segregation Principle (ISP).

2. Antywzorce i typowe pułapki

- 2.1. Code smell – wykrywanie ok 20 zapachów kodu
- 2.2. Przegląd typowych błędów i pułapek
- 2.3. Techniki refaktoryzacji

3. Wzorce projektowe - praktyczne, nieksiążkowe przykłady oparte o rzeczywiste problemy w kontekście aplikacji Enterprise.

- 3.1. Command – hermetyzacja usług, autoryzacja dostępu.
- 3.2. Decorator – reużywalność logiki.
 - 3.2.1. Składanie złożonej logiki biznesowej o przyrostowym charakterze.
 - 3.2.2. Wrapper – odmiana wzorca użyteczna w modelowaniu zorientowanym na znaczenie.
 - 3.2.2.1. Opakowanie typów podstawowych wygodnymi obiektami.
 - 3.2.2.2. Alternatywa dla Utils.
 - 3.2.2.3. Archetyp Money.
 - 3.2.3. Połączenie Dekoratora ze Strategią w celu zbudowania odmian algorytmów przyrostowych.
- 3.3. Strategy – hermetyzacja logiki biznesowej.
 - 3.3.1. Wybór odmiany algorytmu bez ingerencji w core biznesowy.

3.3.2. Integracja z mechanizmami wstrzykiwania zależności.

3.3.3. Definiowanie konkretnej strategii biznesowej w kontenerze Injection of Control (XML lub metody fabrykujące).

3.4. Chain of Responsibility – dwie odmiany wzorca.

3.4.1. Dobór logiki biznesowej do aktualnych warunków.

3.4.2. Połączenie łańcucha ze Strategią w celu zbudowania odmian algorytmów warunkowych.

3.5. Abstract Factory – tworzenie artefaktów domenowych.

3.5.1. Spójny sposób na tworzenie rodzin obiektów biznesowych zależnych od konfiguracji wdrożeniowej systemu.

3.5.2. Produkowanie Strategii.

3.6. Builder – redukcja złożoności tworzenia struktur.

3.6.1. Zunifikowane eksportowanie obiektów domenowych.

3.6.2. Ukrywanie złożoności budowania zapytań.

3.7. Template Method – antywzorzec w przykładach.

3.7.1. Technika u Wspólniania logiki biznesowej.

3.7.2. Przykłady antywzorca.

3.8. Singleton – niebezpieczny wzorzec w przykładach.

3.8.1. Szczegóły implementacji Singletonów tworzonych z opóźnieniem, odpornych na współbieżny dostęp.

3.9. State – hermetyzacja procesu biznesowego.

3.9.1. Implementacja maszyny stanów reprezentującej złożony cykl życia obiektu biznesowego.

3.9.2. Maszyna Stanów jako Wrapper dodający nowe funkcjonalności.

3.10. Observer – redukcja zależności, wysokowydajne systemy Event Driven i asynchroniczne przetwarzanie.

3.11. Specification – hermetyzacja reguł biznesowych.

3.11.1. Redukcja złożoności systemów zawierających złożoną logikę decyzyjną.

3.11.2. Przypadek gdy istnieje wiele możliwych kryteriów logicznych.

3.11.3. Jednak w danym kontekście (wdrożenie, klient) używanym tylko podzbiorem reguł.

3.12. Facade – redukcja złożonej struktury pod wygodnym API.

4. Testability – wpływ użycia dobrych praktyk OOD i Wzorców na testowalność kodu.

4.1. Zagadnienia podatności architektury na testy: problemy i pułapki.

4.2. Techniki testowania jednostkowego: dummy, fake, stub, mock.

4.3. Narzędzia testowania jednostkowego i integracyjnego.

4.3.1. JUnit.

4.3.2. Mockito.