BO·TT·EGA
IT minds

Program szkolenia:

Wzorce projektowe oraz efektywne techniki Object Oriented dla developerów aplikacji biznesowych

Informacje:

Nazwa:	**Wzorce projektowe oraz efektywne techniki Object Oriented dla developerów aplikacji biznesowych**
Kod:	**craft-patterns-Patterns Biz**
Kategoria:	Wzorce projektowe
Grupa docelowa:	
Czas trwania:	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.

Program jest ogólną ramą merytoryczną. Jego realizacja wykracza poza 3 dni. W trakcie analizy przedszkoleniowej wybieramy wzorce, które będą przydatne dla zespołu i skupiamy się tylko na nich w trakcie szkolenie. W ciągu 3 dni zwykle jesteśmy w stanie zaadresować ok 70% wymienionych wzorców.

Zalety szkolenia:

- Skupienie na kontekście aplikacji biznesowych
- Wybór jedynie użytecznych wzorców oraz technik
- Realne przykłady
- Clen Code i testability

BO·TT·EGA
IT minds

Szczegółowy program:

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.

1.3.1. Wady naiwnego modelowania rzeczowniki-czasowniki

1.3.2. Modelowanie hermetycznych agregatów - model behawioralny

1.3.3. Modelowanie niezmienników.

1.3.4. Modelowanie czasu jako jednego z głównych czynników złożoności esencjonalnej

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. Strategy – hermetyzacja logiki biznesowej.

3.1.1. Wybór odmiany algorytmu bez ingerencji w core biznesowy.

3.1.2. Integracja z mechanizmami wstrzykiwania zależności.

3.1.3. Definiowane konkretnej strategii biznesowej w kontenerze Incersion of Control (XML lub metody fabrykujące).

3.2. Decorator – reużywalnośc logiki.

3.2.1. Składanie złożonej logiki biznesowej o przyrostowym charakterze.

BO·TT·EGA IT minds

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. Chain of Responsibity – dwie odmiany wzorca.

3.3.1. Dobór logiki biznesowej do aktualnych warunków.

3.3.2. Połączenie Łańcucha ze Strategią w celu zbudowania odmian algorytmów warunkowych.

3.4. Abstract Factory – tworzenie artefaktów domenowych.

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

3.4.2. Produkowanie Strategii.

3.5. Builder – redukcja złożoności tworzenia struktur.

3.5.1. Zunifikowane eksportowanie obiektów domenowych.

3.5.2. Ukrywanie złożoności budowania zapytań.

3.6. Template Method – szablony procesów.

3.6.1. Technika uwspólniania logiki biznesowej.

3.6.2. Szablonowe Strategie

3.6.3. Przykłady antywzorca.

3.7. Singleton – niebezpieczny wzorzec w przykładach.

3.7.1. Szczegóły implementacji Singeltonów tworzonych z opóźnieniem, odpornych na współbieżny dostęp.

3.8. State – hermetyzacja procesu biznesowego.

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

3.8.2. Maszyna Stanów jako Wrapper dodający nowe funkcjonalności.

3.8.3. Pułapki zbytniego uogólniania Maszyn Stanów

3.9. Specification – hermetyzacja reguł biznesowych.

3.9.1. Redukcja złożoności systemów zawierających złożoną logikę decyzyjną.

3.9.2. Przypadek gdy istnieje wiele możliwych kryteriów logicznych.

3.9.3. Jednak w danym kontekście (wdrożenie, klient) używanym tylko podzbioru reguł.

3.10. Zdarzenia

3.10.1. Observer

3.11. Event Broker

3.12. Event Bus

3.13. Wysokowydajne systemy Event Driven i asynchroniczne przetwarzanie.

3.14. Saga - orkiestracja wielu zdarzeń w czasie

3.15. Role Object

3.15.1. Modelowanie ról w systemie.

3.15.2. Alternatywa dla dziedziczenia.

3.15.3. Odmiana wzorca Bridge.

3.16. Extension Object

3.16.1. Uogólnienie Role Object dla systemów otwartych na rozbudowę.

3.17. Facade – redukcja złożonej struktury pod wygodnym API.

3.18. Command – hermetyzacja usług, autoryzacja dostępu.

3.18.1. Command-CommandHandler - wariant wzorca w architekturze klient-serwer

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.