

Program szkolenia:

Wzorce projektowe i ich implementacja w C# oraz testowanie automatyczne

Informacje:

Nazwa:	Wzorce projektowe i ich implementacja w C# oraz testowanie automatyczne
Kod:	NET-.NET patterns
Kategoria:	.NET
Grupa docelowa:	developerzy architekci
Czas trwania:	3 dni
Forma:	50% wykłady / 50% ćwiczenia

Szkolenie prezentuje poprawną implementację popularnych wzorców projektowych.

Szkolenie jest przeznaczone dla zaawansowanych programistów, projektantów i architektów poszukujących rozwiązań dla systemów o złożonej logice.

Podczas szkolenia stopniowo rozwijamy istniejący system o nowe funkcjonalności korzystając z nabytej wiedzy o wzorcach. Z uwagi na ilość wzorców jaka jest przedstawiana ćwiczenia polegają często na małych (ale krytycznych) zmianach w istniejącym kodzie.

W trakcie szkolenia poruszamy problemy testowania automatycznego: wpływ stosowania wzorców na testowalność kodu oraz wzorce dla kodu testów.

Zalety szkolenia:

- Niwelowanie długu technicznego na wczesnym etapie
- Architektura podatna na rozwój, utrzymanie i testowanie
- Praktyczne, nieakademickie przykłady

Szczegółowy program:

1. Wprowadzenie

- 1.1. OO - poprawna interpretacja
- 1.2. GRASP
- 1.3. SOLID
- 1.4. Dlaczego warto pamiętać o KISS i DRY

2. Wzorce

- 2.1. Strategy
 - 2.1.1. Pluginy
 - 2.1.2. Kustomizacja logiki per wdrożenie
 - 2.1.3. Wstrzykiwanie strategii
 - 2.1.4. Modelowanie w stylu funkcyjnym
- 2.2. Adapter
 - 2.2.1. Trywializacja do postaci Wrappera
 - 2.2.2. Value Objects - odpowiedni poziom abstrakcji dla modeli biznesowych
- 2.3. Bridge
 - 2.3.1. Otwarcie na zmiany poprzez podwójną abstrakcję
 - 2.3.2. Nawiązanie do Role Object i Extension Object
- 2.4. Builder
 - 2.4.1. Dostęp do hermetycznych obiektów
 - 2.4.2. Assembler i Object Mother - zastosowanie w testach jednostkowych
- 2.5. Chain of Responsibility
 - 2.5.1. Manager łańcucha - zastosowanie do walidacji
- 2.6. Command

2.6.1. Anti-Command - zastodowanie do operacji undo

2.6.2. Command Handler w architekturach klient-serwer

2.6.2.1. Wersjonowanie API

2.6.2.2. Własny framework AOP w 50 liniach kodu

2.7. Composite

2.7.1. Specification w kontekście dynamicznych reguł biznesowych

2.8. Decorator

2.8.1. Dynamiczne dziedziczenie

2.9. Event Aggregator

2.10. Facade

2.10.1. Zmiana API

2.10.2. Redukacja interakcji z serwerem

2.11. Factory idiom

2.12. Mediator

2.12.1. Orkiestracja skomplikowanego UI

2.13. Memento

2.14. MVP

2.15. MVVM

2.16. NULL Object

2.17. Observer

2.17.1. Pluginy

2.17.2. Event Broker i Event Bus w architekturze serwerowej

2.18. Proxy

2.19. Repository

2.20. Role Object

2.20.1. Uzależnienie logiki od roli zalogowanego użytkownika

2.20.2. Bezpieczeństwo

2.20.3. Integracja z archetypem biznesowym Party

2.21. Singleton

2.21.1. Antywzorzec w przykładach

2.22. Service Locator

2.23. State

2.23.1. Alternatywne podejścia do konstruowania maszyn stanów

2.24. Template Method

2.24.1. Otarcie się o antywzorzec

2.25. Unit of Work

2.26. Visitor

2.26.1. Wygodny sposób na obsługę typów wyliczeniowych

2.27. Rules

3. Testowanie automatyczne

3.1. Wpływ dobrego projektu obiektowego na testowalność kodu

3.1.1. Wysoka kohezja

3.1.1.1. Zastosowanie SOLID i GRASP

3.1.2. Niski coupling

3.1.2.1. Zastosowanie Fabryk, Zdarzeń i Strategii

3.2. TDD

3.2.1. Podejście Specification First

3.2.2. Cykl Red, Green, Refactor

3.3. Narzędzia

3.3.1. Uruchamianie testów

3.3.2. Zaśleпки: Mock, Stub, Fake, Spy

3.3.3. Automatyczne uruchamianie testów w tle w Visual Studio

3.4. Wzorce tworzenia testów

3.4.1. Testujemy zachowania a nie metody

3.4.2. Arrange, Act, Assert

3.4.3. Przygotowanie stanu

3.4.3.1. Object Mother

3.4.3.2. Assembler (Builder)

3.4.4. Sprawdzenie

3.4.4.1. Assert Object

3.4.4.2. Matcher