SZKOLENIE ŚREDNIO ZAAWANSOWANE

Test-Driven Development w języku C#

CH/TDD

Czas trwania: 3 dni

Szkolenie Test-Driven Development w języku C# przedstawia jedną ze zwinnych metodologii tworzenia oprogramowania zorientowana na bezpośrednie odwzorowanie wymagań biznesowych. Opiera na się o krótką sekwencje kroków powtarzaną cyklicznie do momentu osiągnięcia oczekiwanego rezultatu.

Cele szkolenia

- Wyrobienie w uczestnikach pewności w stosowaniu TDD na co dzień
- Zaznajomienie z technikami tworzenie wysokiej jakości kodu, który wygodnie poddaje się utrzymaniu i rozwojowi

Zalety

- Przewaga praktycznych warsztatów ponad akademickie przekazywanie wiedzy w formie wykładu
- Praktykowanie TDD poprzez serię ćwiczeń mających na celu poznanie i zrozumienie czym jest TDD
- Praktyka przed teorią - wszystkie szkolenia technologiczne prowadzone są w formie warsztatowej. Konieczna teoria jest wyjaśniana na przykładzie praktycznych zadań
- Konkretne umiejętności - w ramach każdego szkolenia rozwijamy praktyczne umiejętności związane z daną technologią i tematyką
- Nauka z praktykami - wszyscy trenerzy na co dzień pracują w projektach, gwarantuje to dostęp do eksperckiej wiedzy i praktycznego know-how

Dla kogo?

- Tematyka szkolenia Test-Driven Development w języku C# koncentruje się wokół organizacji oraz jakości kodu. Test-Driven Development to warsztaty skierowane do programistów oraz testerów chcących tworzyć testowalny kod odpowiadający zapotrzebowaniu zamawiającego.

Wymagania

- Praktyczna znajomość języka programowania C#
- Mile widziane podstawy programowania obiektowego oraz testowania oprogramowania

Program

1. TDD od kuchni
    a. Historia
    b. Idea
    c. Ciemna strona TDD
    d. Co i jak testować?
2. Niezbędnik do pisania dobrych testów
    a. Nazwa testu, klucz do sukcesu
    b. Red-green-refactor
    c. F.I.R.S.T
    d. Dubler - mock, fake, stub
    e. Pułapki w stosowaniu TDD
    f. Generatory danych testowych
3. Narzędzia developerskie
    a. Visual Studio i jego możliwości
    b. ReSharper
    c. Silnik testów jednostkowych (np.: MsTest, NUnit, xUnit)
    d. Generator dublerów (np.: Moq, NSubstitution, RhinoMocks, FakeItEasy)
    e. Weryfikacja poprawności (np.: FluentAsserions)
    f. Pokrycie kodu testami (Code Coverage)
4. Testowalny kod - praktyczna odsłona
    a. Trudno testowalny kod - co to znaczy?
    b. Oczekuj zamiast tworzyć (DI)
    c. SOLID - fundament dobrego kodu
    d. Techniki dublowania zależności
5. Testowanie funkcjonalne (regresja/end-2-end)
    a. Testowanie adaptacyjne
    b. Testowanie funkcjonalne (end-to-end)
    c. Automatyzacja testów funkcjonalnych
6. Refaktoryzacja
    a. Techniki przydatne w TDD
    b. Usuwanie redundancji w kodzie
    c. Code Smell - indentyfikacja, usuwanie
    d. Refaktoryzacja do wzorców projektowych
    e. Praca z kodem odziedziczonym (legacy code)