Unit Testing a kvalita kódu

Unit Testing je proces automatizovaného ověřování funkčnosti jednotlivých komponent programu v izolaci. Cílem není najít všechny chyby v systému, ale zajistit, že každá jednotlivá část kódu (jednotka) dělá přesně to, co se od ní očekává.

1. Co je to "jednotka"?

Jednotkou se v objektově orientovaném programování rozumí obvykle jedna metoda třídy. Test by měl být:

- **Atomický:** Testuje pouze jednu konkrétní věc.
- **Izolovaný:** Testovaná metoda by neměla záviset na databázi, síti nebo souborovém systému (k tomu se používají tzv. **Mocks** – simulované objekty).
- **Rychlý:** Tisíce testů by měly proběhnout v řádu sekund.

—

2. Proč psát unit testy?

Unit testing není „práce navíc“, ale investice, která se vrací v podobě:

- **Prevence regresních chyb:** Máte jistotu, že nová úprava kódu nerozbila nic, co dříve fungovalo.
- **Odvážnější refaktoring:** Pokud máte kód pokrytý testy, můžete jej čistit a měnit s jistotou, že zachováte funkčnost.
- **Dokumentace kódu:** Testy slouží jako praktické příklady toho, jak se má daná metoda používat a co má vracet.
- **Lepší design:** Kód, který se špatně testuje, je obvykle špatně navržen (porušuje principy SOLID).

—

3. Životní cyklus testu: AAA vzor

Většina unit testů se píše podle struktury **AAA** (Arrange, Act, Assert):

```
1. **Arrange (Příprava):** Nastavení testovacího prostředí, inicializace objektů a příprava vstupních dat.
2. **Act (Akce):** Zavolání testované metody.
3. **Assert (Ověření):** Kontrola, zda je skutečný výsledek shodný s očekávaným.
```

```
// Příklad v jazyce C# (NUnit)
```

```
[Test]
public void Add_TwoNumbers_ReturnsSum() {
    // Arrange
    var calculator = new Calculator();
    // Act
    var result = calculator.Add(2, 3);
    // Assert
    Assert.AreEqual(5, result);
}
```

—

4. TDD – Test Driven Development

TDD je metodika, kde **test píšete dříve než samotný kód**. Probíhá v cyklu:

```
1. **Red (Červená):** Napíšete test na funkci, která ještě neexistuje. Test selže.
2. **Green (Zelená):** Napíšete minimální množství kódu nutné k tomu, aby test prošel.
3. **Refactor (Refaktoring):** Vyčistíte kód, zatímco testy hlídají jeho funkčnost.
```

—

5. Metriky kvality: Code Coverage

Code Coverage (pokrytí kódu) udává, kolik procent řádků vašeho kódu je vykonáno během testů.

- Vysoké pokrytí (např. 80 %+) dává vyšší jistotu stability.
- **Pozor:** 100% pokrytí neznamená, že kód je bez chyb. Znamená to jen, že všechny řádky byly spuštěny, nikoliv že byly otestovány všechny logické kombinace.

—

6. Oblíbené nástroje (Frameworky)

Jazyk	**Frameworky**
Java	JUnit, TestNG
C# / .NET	xUnit, NUnit, MSTest
Python	PyTest, unittest
JavaScript	Jest, Mocha
PHP	PHPUnit

Související články:

- Refaktoring kódu
- Principy SOLID
- DevOps a CI/CD (Automatizace testů)

Tagy: programming testing unit-testing tdd software-quality clean-code