Platforma .NET

3. seminář

Radek Janoštík

Univerzita Palackého v Olomouci

9. 10. 2025

Reakce na úkoly

- Jména návěstí
- Četnost návětí
- Jinak více méně v pořádku

Testování software (1 / 2)

- Co znáte z ostatních předmětů?

Testování software (1 / 2)

- Co znáte z ostatních předmětů?
- Proces zjišťování, zda je SW v souladu se specifikací
- Manuální vs. automatické testy
- Testování se znalostí interní struktury (= „bílá skříňka“)
- Testování bez znalosti interní struktury (= „černá skříňka“)
- Typy automatických testů
  - Unit testy (Testy jednotek, malé funkční celky)
  - Integrační testy (provázanost komponent, s OS + ostatní systémy)
  - UI testy
  - Performance testy

Testování software (2 / 2)

- Kdo navrhuje testy
  - Programátor (autor) sám
  - Někdo jiný (tester)
  - Pozor na autorský bias – „hýčkání si svého dítěte“
- Continuous Integration/Continuous delivery pipeline (CI/CD) – spolupráce se správou verzí
- Problém rozdílných architektur (Monolitická vs. Microservice)
- Dnes si ukážeme jen Unit testy v .NET

Unit testy

- (Automatizované) testy nejmenších funkčních celků (Metody, třídy)
- Specifikování jaký je očekávaný výsledek metody pro daný vstup
- Jak vybrat vstupní data?
  - Mezní hodnoty – `null`, prázdné pole, prázdný řetězec, první prvek, ...
  - Nesmyslné hodnoty – záporné číslo, ...
  - Smysluplné hodnoty
- Odstínění závislostí – náhradní data, *Mock* objekty
- Regrese – při objevení chyby dopsat testy, které ji vystihnou $\Rightarrow$ v budoucnu už bude chyba odhalena
- Problém Unit testů: Píšeme kód na testování kódu
  - Testy na testy?

Testování v .NET

- Velká podpora všech druhů testů
- Např.: Testování UI `https://docs.microsoft.com/en-us/visualstudio/test/use-ui-automation-to-test-your-code?view=vs-2019`
- Performance testy: `https://docs.microsoft.com/en-us/visualstudio/test/quickstart-create-a-load-test-project?view=vs-2019`
- Dnes probereme Unit testy pomocí základních knihoven `MSTest, NUnit, xUnit`

MSTest

- Základní knihovna pro unit testy s podporou ve VS
- Flow testů řízeno pomocí atributů tříd - „anotací“
- Vytvoření testovacích scénářů
- Statické metody třídy `Asert`
- Testy defaultně spouštěny v abecedním pořadí
  - Na pořadí testů by však z podstaty nemělo záležet
  - Můžeme vytvářet playlisty testů
- Možno vynutit paralelní spouštění

```
[assembly: Parallelize(Workers = 0, Scope = ExecutionScope.MethodLevel)]
```

Flow spouštění testů

- Pomocí atributů můžeme označit metody, které se vykonají jen někdy
- Důležité pro globální nastavení (databáze, soubory, . . .)
- `[AssemblyInitialize]` a `[AssemblyCleanup]`
  - Takto označené metody musí být statické
  - Vykonají se před všemi/po všech testech
  - `[AssemblyInitialize]` potřebuje parametr `TestContext`
- `[ClassInitialize]` a `[ClassCleanup]`
  - Obdobné, ale s testovací třídou
- `[TestInitialize]`, `[TestCleanup]`
  - Nejsou statické, bez parametru
  - Spustí se před/po každém testu

NUnit

- Původně port **JUnit**, od verze 3 kompletně přepsán
- `https://nunit.org/` – bohatá dokumentace
- Součástí .NET Foundation
- V základu velmi podobné jako `MSTest`
- Řízení pomocí atributů (velké množství)
- Ukázka

xUnit

- Podle mě nejjednodušší testovací framework
- Nakonec nebudeme ukazovat, nechám na vás
- Porovnání: `https://xunit.net/docs/comparisons`

Test-driven development (TDD)

- Způsob vývoje programu, kdy nejprve napíšeme testy a pak až implementaci
- Potřebujeme přidat novou vlastnost do programu
  - Nejprve přidáme testy na možné případy pro novou vlastnost
  - Testy spustíme – všechny nové by neměly projít
  - Pokusíme napsat co nejjednodušší kód, který splní všechny naše testy
  - Pak může přijít do úvahy refactoring (= kontrola funkčnosti pomocí testů)
- Toto můžeme dál opakovat

Úkol (1 / 2)

- Vyberte si testovací framework, vytvořte knihovnu `MathLib` a v ní implementujte: Třídu se statickými metodami:
  - `int? MaxItem(int[] arr)` – najde maximální prvek v poli
  - `bool IsPowOf(int num, int base)` – je `num` mocninou `base`?
  - `string DecToBin(uint decNum)` – převod do binárního řetězce (bez použití vestavěných metod
- Implementace doplňte vhodnými testy (množství, vhodně zvolené případy, . . .)

Úkol (2 / 2)

- Třídu `Set` reprezentující množinu nad celými čísly s metodami:
  - `void Add(int i)` – přidání prvku i
  - `bool Contains(int i)` – obsahuje prvek i?
  - `void Remove(int i)` – odebrání prvku i
  - `int Size()` – velikost množiny
  - `Set Union(Set other)` vytvoří novou obsahující sjednocení
  - `Set Intersection(Set other)` – vytovoří novou obsahující průnik
  - `Set Subtract(Set other)` – vytvoří novou obsahující množ. rozdíl
  - `bool IsSubsetOf(Set other)` – je podmnožinou?
- Třídu vhodně doplňte testy
- Volitelné: Zkuste (si) to udělat pomocí TDD