giganci programowania

Odczytywanie danych z plików

Celem lekcji jest nauczenie się metod oraz bibliotek w C# służących do odczytu danych z różnych formatów plików (PDF, JSON). W trakcie zajęć omówione zostaną aspekty obsługi wyjątków przy obsłudze uszkodzonych plików oraz praktyki walidacji poprawności danych.

Powtórzenie wiadomości

1. Czym są wzorce projektowe?
    - Zestawy sprawdzonych rozwiązań typowych problemów programistycznych.
    - **Dlaczego warto je znać?** Ułatwiają komunikację w zespole, przyspieszają proces tworzenia oprogramowania, pomagają pisać lepszy, bardziej zrozumiały kod.
2. Co to jest Singleton?

Singleton jest kreacyjnym wzorcem projektowym, który pozwala zapewnić istnienie wyłącznie jednej instancji danej klasy. Ponadto daje globalny punkt dostępowy do tejże instancji.

PDF

PDF (ang. Portable Document Format) to taki rodzaj pliku, w którym można przechować **tekst, obrazy i układ stron** w sposób, który wygląda tak samo na każdym komputerze czy telefonie.

Można go trochę porównać do **zdjęcia wydrukowanej kartki**, tylko w formie cyfrowej – dlatego jeśli wyślesz komuś plik PDF, to odbiorca zobaczy dokładnie to samo rozmieszczenie tekstu, obrazków i czcionek, jakie ustawiłeś.

Kiedy warto odczytywać pliki PDF

1. **Automatyczne przetwarzanie dokumentów**
   - Jeśli mamy dużo faktur, umów czy raportów w PDF, możemy w C# napisać program, który „przejrzy” je wszystkie i **wyciągnie** z nich najważniejsze informacje (np. nazwy produktów, ceny) do dalszej obróbki.
2. **Wyszukiwanie treści**
   - W aplikacji możemy zaimplementować **wyszukiwarkę**, która przeszukuje dokumenty PDF i zwraca konkretne fragmenty tekstu.
3. **Ekstrakcja danych do raportów**
   - Zamiast ręcznie przeklejać fragmenty, można w C# napisać kod, który samodzielnie ściągnie dane z PDF (np. wykresy, tabele) i wstawi je do innej aplikacji, np. do arkusza kalkulacyjnego.

JSON

JSON (z ang. JavaScript Object Notation) to sposób zapisu danych w formie tekstu. Możesz myśleć o nim jak o „liście zakupów” lub „kartotece danych”, gdzie każda informacja jest zapisana w parach klucz-wartość (jak słownik).

- Lekki i przejrzysty: Łatwo go odczytać nawet człowiekowi, a komputerowi – łatwo przetworzyć.

- Popularny format wymiany danych: Większość serwerów i aplikacji (np. strony internetowe, usługi w chmurze) wykorzystuje JSON do przesyłania informacji.

JSON

Przykład:

```
{
  "name": "Alice",
  "age": 25,
  "skills": ["C#", "HTML", "CSS"]
}
```

Tutaj „name”, „age” i „skills” to klucze, a „Alice”, 25 i lista [„C#”, „HTML”, „CSS”] to wartości.

JSON

Serializacja to proces, w którym przekształcamy obiekt (np. w C#) w tekst lub inny format, tak aby można go było przechować (np. w pliku) lub przesłać (np. przez sieć).

Na przykład, masz w programie obiekt Osoba z polami Imie i Wiek. Jeśli zrobisz z niego tekst JSON, żeby zapisać w pliku osoba.json, mówimy, że dokonujesz serializacji.

Deserializacja to proces odwrotny: bierzemy taki zapisany (zserializowany) tekst czy inne dane i odtwarzamy z nich z powrotem obiekt w programie.

Czyli jeśli plik osoba.json zawiera zapis { "Imie": "Ala", "Wiek": 10 }, to czytasz go w C# i tworzysz obiekt Osoba z ustawionym Imie = "Ala" i Wiek = 10. To właśnie deserializacja.

Walidacja danych

Dlaczego jest ważna?

- Nawet jeśli plik się poprawnie wczyta, może brakować ważnych pól, np. name produktu.

- Możemy mieć niepoprawne wartości (np. cena produktu = -100).

- Walidacja pozwala nam kontrolować poprawność danych, zanim zaczniemy je przetwarzać.