

# FI-LOG

## ZADÁNÍ

Cílem tohoto 4. úkolu je procvičit si generické třídy a práci s textovými řetězci. Vyzkoušíte si jejich načítání ze souborů, extrakci údajů a následný export. Mimo jiné se naučíte i některé další užitečné věci, jako je práce s regulárními výrazy a reflexí.

Implementujte:

1. Implementujte třídu `class TextFileLogStorage : ILogStorage`, která je zodpovědná za načtení textu ze souboru a zapsání textu do souboru. Rozhraní `ILogStorage` předepisuje implementaci metod `string Load()` a `void Save(string content)`, které pouze načtou, nebo zapíší obsah textového souboru. Pokud soubor daného jména existuje, bude přepsán. Informaci o cestě k souboru s logem předejte v konstruktoru této třídy v podobě textového řetězce.

Protože práce se soubory nebyla zatím probírána na přednášce, dostudujte ji např. z následujícího zdroje: [http://msdn.microsoft.com/en-us/library/db5x7c0d\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/db5x7c0d(v=vs.110).aspx)

2. Implementujte třídu `class CsvLogImporter<T> : ILogImporter<T>`, která implementuje rozhraní `ILogImporter<T>`. Tato třída v konstruktoru přijme nakonfigurovanou instanci `ILogStorage`, mapování atributů návratového typu na sloupce v CSV souboru, informaci zda-li má CSV soubor v prvním řádku názvy sloupců, oddělovače sloupců a znak obalující textový řetězec. Konstruktor bude tedy vypadat následovně: `CsvLogImporter(ILogStorage logStorage, string[] columnMappings, bool hasHeaderRow, char columnDelimiter, char textQualifier)`

Mapování atributů bude fungovat tak, že pokud např. na 0. pozici v poli `columnMappings` je hodnota „EventTime“, systém se pokusí vždy hodnotu z 1. sloupce CSV vložit do vlastnosti `EventTime` třídy, která je určena jako typ `T`. Při načítání dané hodnoty bude aplikována konverze na datový typ cílové vlastnosti<sup>1</sup>. K dosažení tohoto chování bude nutné využít Reflection: <http://msdn.microsoft.com/en-us/library/ms173183.aspx>

Doporučuji ještě před samotným načítáním dat ze souboru ověřit, že typ `T` obsahuje všechny atributy, které jsou použity v mapování, pokud ne, vrátit vhodnou výjimku.

Vzhledem k tomu, že CSV soubor obsahuje kolekci záznamů, typickým návratovým typem metody `Import` bude `List<T>`, kdy `T` je typ řádku.

3. Implementujte třídu `class JsonLogImporter<T> : ILogImporter<T>`, která implementuje rozhraní `ILogImporter<T>`. Tato třída v konstruktoru přijme nakonfigurovanou instanci `ILogStorage`, což bude jediný parametr konstruktoru a po zavolání metody `Import` provede deserializaci načteného textu logu z formátu JSON. Pro práci s JSON použijte oblíbenou knihovnu `Json.NET` (<http://www.nuget.org/packages/Newtonsoft.Json/6.0.1>).
4. Implementujte třídu `class SQLServerFormatLogImporter : ILogImporter<SQLServerLogEvent>`, která implementuje rozhraní `ILogImporter<SQLServerLogEvent>`. Tato třída je navržena speciálně na jeden typ logu, který máte dodaný. Třída musí používat regulární výrazy pro parsování logu. Bližší informace o jejich použití naleznete zde: [http://msdn.microsoft.com/en-us/library/hs600312\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/hs600312(v=vs.110).aspx)

<sup>1</sup> Pro zjednodušení všechny logy pracují se stejným formátem `DateTime`

5. Implementujte třídu `class CsvLogExporter<T> : ILogExporter<T>`, která implementuje rozhraní `ILogExporter<T>`. Tato třída v konstruktoru přijme nakonfigurovanou instanci `ILogStorage`, mapování atributů typu `T` na sloupce v CSV souboru a informaci zda-li má CSV soubor v prvním řádku obsahovat názvy sloupců. Po zavolání metody dojde k vyexportování dat v zadaném formátu a jejich zápisu do souboru.
6. Implementujte třídu `class JsonLogExporter<T> : ILogExporter<T>`, která implementuje rozhraní `ILogExporter<T>`. Tato třída v konstruktoru přijme nakonfigurovanou instanci `ILogStorage` a po zavolání metody `Export` serializuje data do formátu JSON, který zapíše do souboru.

V kostře projektu máte připravena stejná vstupní data ve všech 3 formátech logů.<sup>2</sup>

Součástí Vašeho řešení bude aplikace demonstrující funkcionalitu naimplementovaných tříd.

### Podmínky kódování úkolů

- Bez dovolení nesmíte změnit zadání
  - Prakticky nesmíte přepsat dodané soubory - zejména měnit zadaná rozhraní a definované třídy (smíte doplnit chybějící implementace a rozšířit tyto třídy).
  - Nebráníme vám v definici vlastních rozhraní nebo použití dědičnosti, pokud to přispěje ke zpřehlednění kódu a odstranění duplicit
- Dodržujte principy a koncepty OOP, důsledně uplatňujte zapouzdření.
- Aplikace nesmí skončit neočekávanou chybou bez vnějšího vlivu, chybné vstupy musí být ošetřeny co nejdříve.
- Vyhýbejte se opakovanému psaní duplicitního kódu.
- Vhodně používejte konstrukce jazyka C#, např. vlastnosti a indexery.
- Jeden soubor ať obsahuje jednu třídu a název souboru odpovídá názvu třídy.
- Dodržujte jmenné konvence .NET:
  - Názvy parametrů metod, soukromých datových položek a proměnný pomocí camelCase (malé počáteční písmeno), vše ostatní PascalCase (velké počáteční písmeno)
  - Názvy rozhraní začínají velkým písmenem `I`, implementacím se nepřidává přípona `Impl`.
- Udržujte ve třídách pořádek, zvolte si jeden způsob řazení členů ve třídě a ten dodržujte,
  - v .NET je řazení v souboru obvykle: konstanty, dat. položky, vlastnosti, konstruktory, metody.
- Snažte se psát kód přehledně, volte vhodné názvy identifikátorů.
- Čleňte kód na vhodné celky (třídy/metody).

---

<sup>2</sup> Log ve specifickém formátu obsahuje 3 typy událostí. Protože pro každou musíte připravit samostatný regulární výraz, očekávejte, že funkčnost Vaší aplikace budeme testovat právě na těchto datech, ale budeme náhodně modifikovat jednotlivé hodnoty, u kterých můžete z logu odvodit, že se mění.