C# CMD PARSER

Richard Savčinský

Lukáš Riedel

REQUIREMENTS

- Allow user to do whatever he wants to, do not limit him to the features implemented by the library
- Externalize storage of the parsed arguments
- Convenient set-up using builders

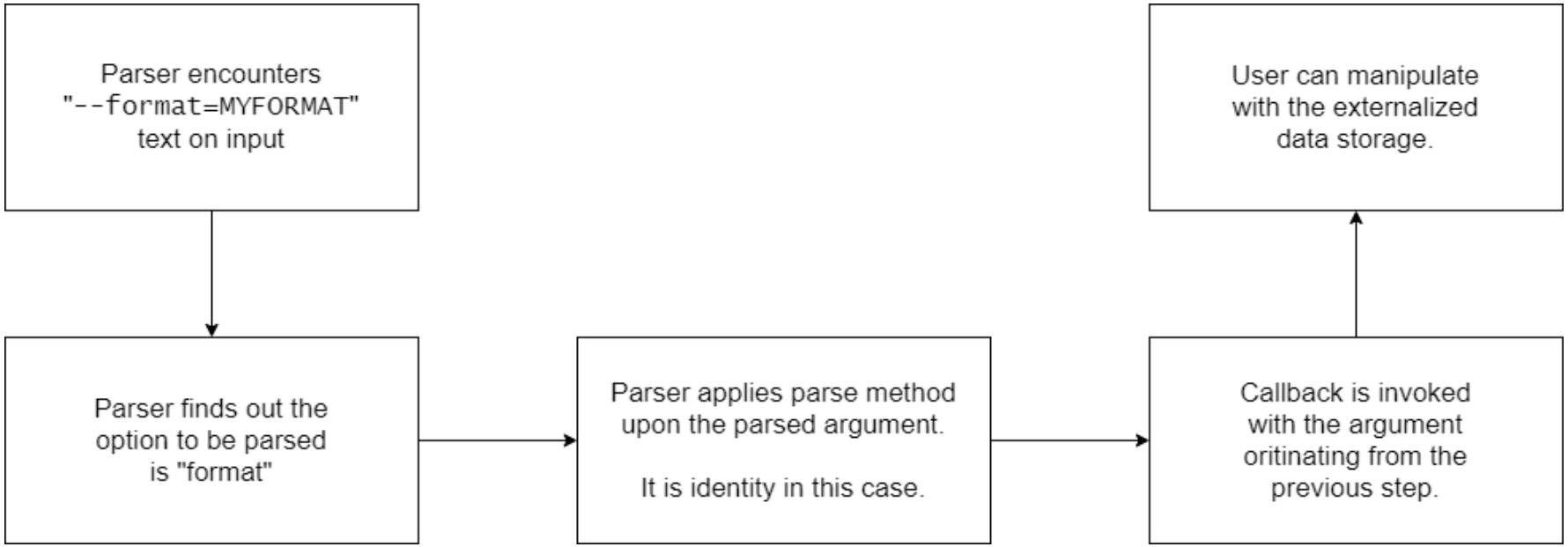
MAIN IDEA

- To fulfill the requirements, we use callback mechanism
  - The callback is called once the option is succesfully parsed
- Parameters for options are strongly-typed
  - The library predefines parse methods for common types (string, int...)
  - However, parse method for user-defined type can be supplied
  - In both cases, callback is invoked with argument corresponding to result of the parse method

EXAMPLE

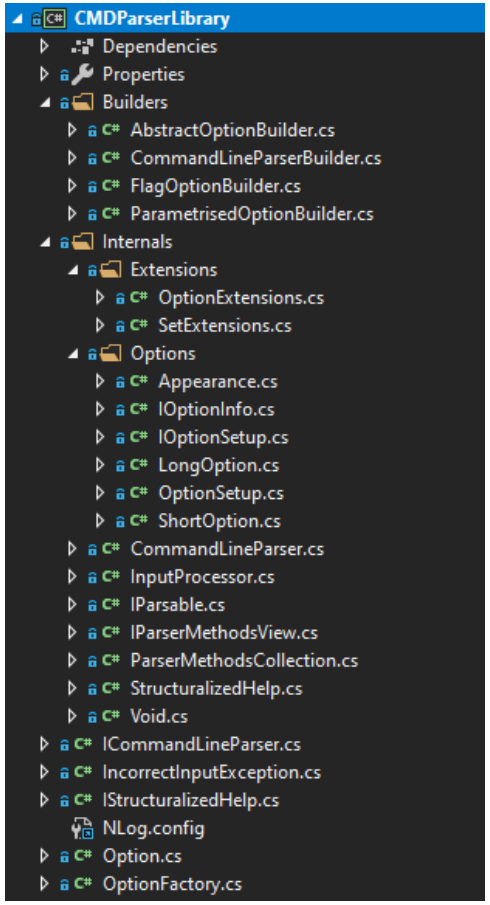
```
parserBuilder.SetupOption<string>(Short('f'), Long("format"))
    .WithDescription("Description of the option")
    .Callback(format => output.OutputFormat = format)
    .ParameterRequired();
```

EXAMPLE


Parser encounters "--format=MYFORMAT" text on input
Parser finds out the option to be parsed is "format"
Parser applies parse method upon the parsed argument.
It is identity in this case.
Callback is invoked with the argument oritinating from the previous step.
User can manipulate with the externalized data storage.

WHAT HAVE WE DONE SO FAR

- We implemented PoC solution a month ago
  - Right now, we are just tuning some things



WHAT HAVE WE DONE SO FAR

- We integrated provided tests
  - Most of them are very good
  - Helped us to discover problems in our PoC version

CMDParserLibrary.Tests (1)	27 ms
CMDParser.Tests (1)	27 ms
CommandLineParserIntegrationTests (1)	27 ms
api-testing-s08 (19)	78 ms
CMDParserLibrary.BlackBoxTests (19)	78 ms
CommandLineParserBuilderTests (5)	56 ms
CommandLineParserTests (1)	10 ms
OptionBuilderTests (3)	2 ms
OptionFactoryTests (6)	2 ms
OptionTests (2)	< 1 ms
StructuralizedHelpTests (2)	8 ms
api-testing-s45 (19)	86 ms
CMDParser.Tests (19)	86 ms
CommandLineParserUnitTests+ArgumentTests (4)	27 ms
CommandLineParserUnitTests+HelpPageTests (1)	19 ms
CommandLineParserUnitTests+OptionTests (10)	21 ms
CommandLineParserUnitTests+TypeParserTests (4)	19 ms

WHAT HAVE WE DONE SO FAR

- We wrote a documentation
  - Overview
  - Step-by-step tutorial
  - Some technical details

Set-up

Step 1: The parser is meant to be set-up using several builder instar
`CommandLineParserBuilder`. Therefore, it is enough to call the para

```
var parserBuilder = new CommandLineParserBuilder();
```

Note that there is also a constructor accepting a `string` argument,
accessible from help text.

```
var parserBuilder = new CommandLineParserBuilder("MyComm
```

Step 2: Now it is time to set-up parse methods. Parse methods for a
example, `string`, `int`, `double` or `bool`. All the predefined parse

However, having these types only would have been too limiting. Th
following class:

```
class WorkingDays
{
    public bool Monday { get; }
    public bool Tuesday { get; }
    ...
}
```

Weekdays can be coded into some string instance somehow. For ex
string `1001000` would mean that the company works on Monday a

WHAT HAVE WE DONE SO FAR

- We wrote API documentation

```
/// <summary>
/// Creates a new setup for the parametrised option.
/// </summary>
/// <typeparam name="TParsedType">Type of the parsed argument.</typeparam>
/// <param name="option">A collection of <see cref="Option"/> instances.</param>
/// <returns>A builder that can set-up the options.</returns>
/// <remarks>
/// The <paramref name="option"/> collection defines aliases (synonyms) for the option.
/// </remarks>
/// <exception cref="ArgumentException">Thrown when <paramref name="option"/> is empty.</exception>
19 references | 7/7 passing | Lukáš Riedel, 6 days ago | 1 author, 3 changes
public ParametrisedOptionBuilder<TParsedType> SetupOption<TParsedType>(params Option[] option)
```

WHAT ARE WE GOING TO DO

- Write more tests
- Clean-up the code
- Fix bugs that have not been discovered yet

SOME TECHNICAL DETAILS

```
/// <summary> A collection of parse methods.
4 references | Lukáš Riedel, 6 days ago | 1 author, 4 changes
internal class ParserMethodsCollection : IParserMethodsView
{
    private readonly IDictionary<Type, Func<string, object>> _dic = new Dictionary<Type, Func<string, object>>();

    /// <summary> Creates a new ParserMethodsCollection and registers common parse m ...
    1 reference | Lukáš Riedel, 6 days ago | 1 author, 2 changes
    public ParserMethodsCollection()...

    /// <summary> Registers parser method.
    9 references | Lukáš Riedel, 6 days ago | 1 author, 3 changes
    public void RegisterParseMethod<TParsedType>(Func<string, TParsedType> parser)
    {
        if (!_dic.ContainsKey(typeof(TParsedType)))
            _dic.Add(typeof(TParsedType), x => parser(x)!);
        else
            _dic[typeof(TParsedType)] = x => parser(x)!; // If contained yet, overwrite it and start using the new parse method.
    }

    /// <inheritdoc/> ...
    2 references | Lukáš Riedel, 6 days ago | 1 author, 2 changes
    public TParsedType Parse<TParsedType>(string input)
    {
        if (!_dic.ContainsKey(typeof(TParsedType)))
            throw new NotSupportedException($"The collection does not support parsing of { typeof(TParsedType) } type.");

        return (TParsedType)_dic[typeof(TParsedType)](input);
    }
}
```