C# CMD PARSER

Richard Savčinský Lukáš Riedel

REQUIREMENTS

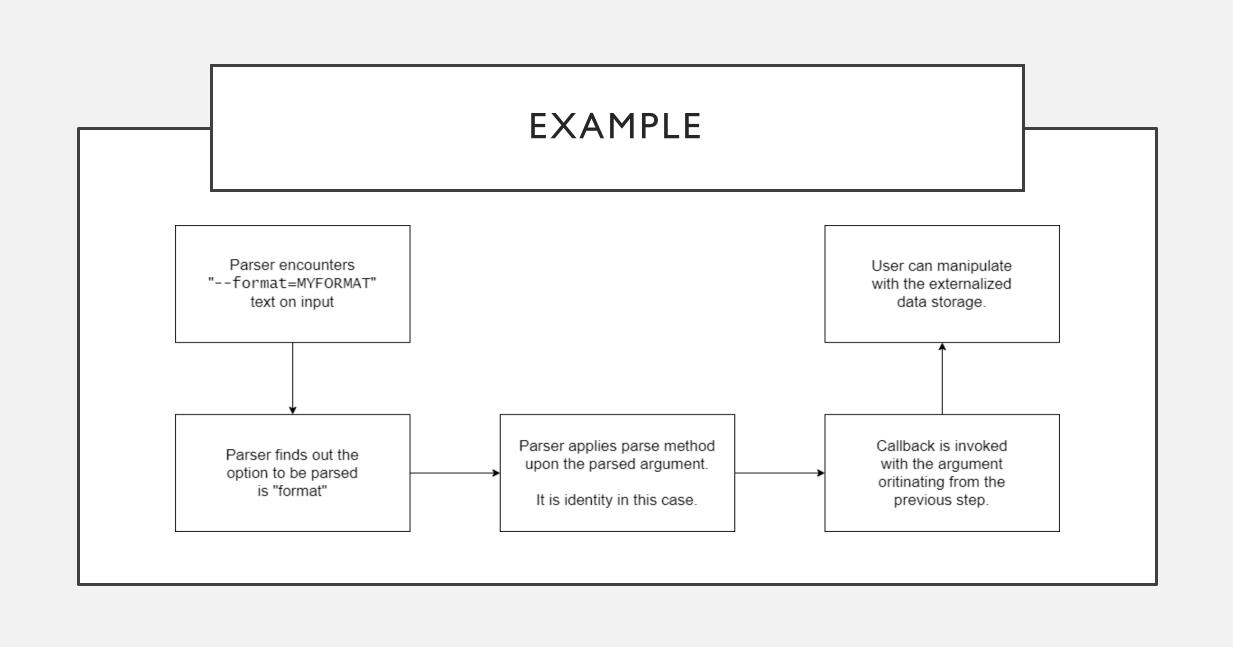
- Allow user to do whatever he wants to, do not limit him to the features implemented by the library
- Externalize storage of the parsed arguments
- Convenient set-up using builders

MAIN IDEA

- To fulfill the requirements, we use callback mechanism
 - The callback is called once the option is successfully 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();
```



MAIN IDEA

- We use builders to set-up the parser
- Parsed arguments are returned using IReadOnlyList<string> instance
- Help can be viewed for a single option, or as a structuralized text block

BENEFITS / DRAWBACKS

- Parser can be customized in any way the user wants to
 - Can be used even in complex workflow
- Due to callback mechanism, the library might not be beginner-friendly
- A lot of code needs to be implemented to set-up the parser
 - Even for naïve usage of the parser

REVIEW COMMENTS

- Basically positive, thanks for all of them
- We will write a tutorial for newcomers how to use the library
 - Until now, the usage had to be inferred from the provided example
- We will think about incorporating excluding options into the parser
 - That means, there should be several sets, each one allowing some options only