

STORY 5 支持从所选 COMMAND 获得 DEFINITION 信息

在前一个 Story 中，我们已经可以从 `ArgsParsingResult` 获得 `ICommandDefinitionMetadata` 了，但是现在这个 `ICommandDefinitionMetadata` 还非常单薄。我们希望能够从这个对象中获得更多的命令定义信息。这个信息接下来可以帮助我们进行一系列的功能扩展（例如，输出非常漂亮的帮助信息）。

AC 1: 获得 Command 的基本信息:

为了接下来进行系统扩展，我们自然应当提供 Command 相关的信息，例如 Command 的名字、描述 (Description)、该 Command 支持的 Flags 等等。

首先我们扩展 `ICommandDefinitionMetadata` 接口:

```
public interface ICommandDefinitionMetadata
{
    string Symbol { get; }
    string Description { get; }
    IEnumerable<IOptionDefinitionMetadata>
    GetRegisteredOptionsMetadata();
}
```

其中，如果对于默认的 Command 而言。Description 永远为 `string.Empty`。我们着重介绍第三个方法:

```
IEnumerable<IOptionDefinitionMetadata>
GetRegisteredOptionsMetadata();
```

该方法将获得注册在当前 Command 上的所有的 Flag 的定义。如果该 Command 上并没有任何 Flags 定义，则返回空的集合（而不是 `null`）。其中 `IOptionDefinitionMetadata` 的定义如下:

```
public interface IOptionDefinitionMetadata
{
    IOptionSymbolMetadata SymbolMetadata { get; }
    string Description { get; }
}
```

其中对于创建时用 `null` 初始化的 `Description` 来说, 其返回值应当为 `string.Empty` 而不能为 `null`。而 `IOptionSymbolMetadata` 的定义为:

```
public interface IOptionSymbolMetadata
{
    char? Abbreviation { get; }
    string FullForm { get; }
}
```

这和添加 `Flag` 定义的时候是一致的。`Abbreviation` 是缩略形式, 如果 `Abbreviation` 不存在则返回 `null`; 相应的 `FullForm` 是完整形式, 若不存在 `FullForm` 则返回 `null`。需要注意的是这两个返回值都不包括前导划线。

以下是范例, 你可能需要补充更多的测试用例:

```
ArgsParser parser = new ArgsParserBuilder()
    .BeginDefaultCommand()
    .AddFlagOption("flag", 'f', "flag description")
    .EndCommand()
    .Build();

ArgsParsingResult result = parser.Parse(new [] {"--flag"});

Assert.True(result.IsSuccess);

IOptionDefinitionMetadata[] optionDefinitionMetadatas =
    result.Command.GetRegisteredOptionsMetadata().ToArray();
IOptionDefinitionMetadata flagMetadata =
    optionDefinitionMetadatas.Single(
        d => d.SymbolMetadata.FullForm.Equals("flag",
            StringComparison.OrdinalIgnoreCase));

Assert.Equal("flag", flagMetadata.SymbolMetadata.FullForm);
Assert.Equal('f', flagMetadata.SymbolMetadata.Abbreviation);
Assert.Equal("flag description", flagMetadata.Description);
```

AC 2: Non-Functional Integration

Please setup CI and Sonar for your project on Github. We will use the following free product:



Product Name	Description
Travis CI	The travis CI can test our library in linux environment.
Appveyor	The Appveyor is a CI tools that test our library in Windows environment
Sona Cloud Service	We will use sonar to do static code check.



Please make sure all three CI is in passed status:

Readme.md

Axe.Cli is a set of libraries to handle common tasks for command line application

Latest status

	Overall Quality	Maintainability Rating
Sonar Cloud Status	 quality gate passed	 maintainability A

	Testing on Linux	Testing on Windows
Testing Status	 build passing	 build passing