STORY 4 支持 COMMAND

我们现在已经比较完善的支持了 Flags。但是，在许多的命令行应用程序中这些参数并非第一级的公民。第一级的往往被称为 command。决定了该命令的大方向。例如：

```
git commit -m "You say blah, blah, blah..."
```

中 `commit` 就是 command。command 之间是互相独立的，这意味着 command 可以定义自己独有的 flag 集合。

我们计划一步一步的支持 command。但是现在我们书写的程序中不支持 command。因此我们第一步将创建与其兼容的默认 command。默认 command 就是当不提供任何 command 信息的时候，parser 将自动选择的 command 定义。

AC 1：定义默认 Command

为了更好的定义 command，我们在 `ArgsParserBuilder` 中添加一个方法：

```
public CommandBuilder BeginDefaultCommand();
```

该方法将开始定义默认 command。但是在 command 定义完毕之前它并不会真正的保存 command 的定义。该方法将返回一个 `CommandBuilder` 用于将各种 Flags 的定义添加到默认 command 上。

`CommandBuilder` 中的方法则是之前我们在 `ArgsParserBuilder` 上定义的 `AddFlagOption` 方法。现在我们要把这个方法移动（意味着我们要删掉 `ArgsParserBuilder` 中的相应方法）到 `CommandBuilder` 中：

```
public class CommandBuilder
{
    ...
    public CommandBuilder AddFlagOption(
        string fullForm,
        char? abbreviation,
        string description);
}
```

这个方法的行为和之前的一模一样。只不过请注意，对于 Conflict 的判断的范围应该在 Command 之内，而并非整个 `ArgsParserBuilder` 范围。

当所有的默认 command 的 Flags 都定义完毕之后可以调用：`CommandBuilder` 上的 `EndCommand` 来结束定义：

```
public class CommandBuilder
{
    ...
    public ArgsParserBuilder EndCommand();
}
```

该方法结束对默认 command 的定义并返回原始的 `ArgsParserBuilder` 对象。并将整个 command 的定义添加到 `ArgsParserBuilder`。

该方法在如下的情况下会失败：

- 如果已经在 `ArgsParserBuilder` 中添加过一个默认 command 了，那么在 `EndCommand` 的时候会抛出 `InvalidOperationException`。

总结下来，我们定义 command 和 Flags 的方式变成了以下的形式：

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

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

result.AssertSuccess();
Assert.True(result.GetFlagValue("--flag"));
```

AC 2：由参数表确定 Command

由于我们接下来在一个命令行解析器中会支持若干的 command。因此我们必须确定最终应用程序的参数到底选择了哪一个 command。例如：

```
git commit -m "blah"
```

选择的 Command 为 `commit` 而不是 `push`。一次解析操作所选择的 command 只可能有一个。

由于我们现在仅支持默认 command。因此在解析成功的前提下，所选择的 command 也肯定是默认 command。

要想得到参数表选定的 command，我们需要为 `ArgsParsingResult` 类添加 `Command` 属性：

```
public class ArgsParsingResult
{
    ...
    public ICommandDefinitionMetadata Command { get; }
}
```

其中，`ICommandDefinitionMetadata` 接口目前只有一个属性就是 command 的名称。而默认 command 的名称为 `null`。

```
public interface ICommandDefinitionMetadata
{
    string Symbol { get; }
}
```

也即：

```
ArgsParser parser = new ArgsParserBuilder()
    .BeginDefaultCommand().EndCommand()
    .Build();

ArgsParsingResult result = parser.Parse(Array.Empty<string>());

Assert.True(result.IsSuccess);
Assert.NotNull(result.Command);
Assert.Null(result.Command.Symbol);
```

而对于失败的 `ArgsParsingResult` 其 `Command` 属性为 `null`。