Arpar – argument parser library

# Návrh

Pro zadávání akceptovaných argumentů aplikací jsme se rozhodli použít variantu takovou, kde uživatel si vytvoří vlastní objekt – třídu – ve kterém si následně naspecifikuje jaké argumenty mají být knihovnou rozeznávány. Tato specifikace se provede pomocí proměnných, které se odpovídajícím způsobem oanotují pomocí atributů.

Pro definici v rámci této konfigurační třídy se načítají pouze proměnné, které mají viditelnost nastavenou na public. Pokud uživatel nechce pracovat s proměnnými ale s vlastnostmi, může si je jednoduše ručně vytvořit nad těmito proměnnými.

Poté, co si uživatel knihovny takto připraví objekt s požadovanými argumenty, tak jej předá knihovně, ta si pomocí reflexe zjistí jaké argumenty s jakými parametry jsou očekávány a připraví si vnitřní struktury. Poté se zavoláním metody Parse provede načtení hodnot a zjištěné hodnoty argumentů jsou následně zapsány do uživatelského objektu, který byl předán do konstruktoru knihovny.

# Implementace

## Anotace

Pro anotaci jsou k dispozici tyto definované atributy:

|  |  |
| --- | --- |
| Atribut | Popis |
| Argument | Základní atribut pro označení že daná proměnná má být považována za argument. |
| BoundedArgument | Specializovaný atribut, který dědí ze základní Argument a navíc umožňuje nastavit meze na hodnotu argumentu, smí být použit pouze pro anotování proměnných typu int (knihovna toto kontroluje). |
| ChoicesArgument | Specializovaný atribut, který taktéž dědí ze základní Argument a navíc umožňuje nastavit enumeraci stringových hodnot, kterých argument může nabývat. Tento atribut smí být použit pouze pro anotování proměnných typu string (knihovna toto také kontroluje). |
| ArgumentAlias | Atribut pro přidávání dalších aliasů daného argumentu. Jejich počet není omezen (pouze ze kontroluje unikátnost jména napříč všemi argumenty). |

U každého atributu je nutné specifikovat jeho jméno, pod kterým bude rozpoznán mezi argumenty, kromě tohoto jména může mít argument libovolné množství dalších jmen – aliasů.

Každé jméno může být dvojího typu: Long nebo Short. Toto rozdělení se používá pro rozlišení, jestli se jedná o krátkou nebo dlouhou volbu (a tyto typy se liší svým prefixem před jménem). Pro snazší používání není nutné typ jména určovat a jako výchozí hodnota se použije dlouhá varianta.

Výchozí nastavení knihovny obsahuje takto nastavené prefixy a oddělovače, které jdou změnit pomocí statických vlastností definovaných na hlavní třídě ArgumentParser.

|  |  |  |
| --- | --- | --- |
| Jméno vlastnosti | Popis vlastnosti | Výchozí hodnota |
| ShortOptionPrefix | Prefix krátké volby | -- |
| LongOptionPrefix | Prefix dlouhé volby | - |
| Splitter | Oddělovač pro ukončení argumentů | -- |
| ValueDelimiter | Oddělovač hodnoty a názvu argumentu | = |

U všech typů atributů kromě AliasArgument lze dále specifikovat tyto volby pro ovlivnění funkcionality:

|  |  |
| --- | --- |
| Volba | Popis |
| Description | Popisek, který se používá pro generování dokumentace argumentů. |
| IsMandatory | Pokud je nastaveno na true, tak pokud tento argument nebude použit, tak parser vyvolá vyjímku. |
| ValueRequirements | Určuje, jestli argument hodnotu vyžaduje (ParameterRequirements.Mandatory), je volitelná (ParameterRequirements.Optional) nebo je zakázaná (ParameterRequirements.Denied). |

U BoundedArgument, který lze aplikovat na proměnné typu int, lze navíc specifikovat:

|  |  |
| --- | --- |
| Volba | Popis |
| LowBound | Dolní mez pro celočíselné hodnoty. |
| HighBound | Horní mey pro celočíselné hodnoty. |

U ChoicesArgument, který lze aplikovat na proměnné typu string, lze navíc specifikovat:

|  |  |
| --- | --- |
| Volba | Popis |
| Choices | Pole stringů hodnot, kterých smí argument nabývat. |

Kromě přímé definice tohoto pole lze využít i upraveného konstruktoru u atributu ChoicesArgument, který umožňuje tento pohodlnější zápis:

[ChoicesArgument("gender", "Male", "Female")]

public string Gender = "Male";

Tento zápis je ekvivalentní zápisu:

[ChoicesArgument("gender", Choices = new string[] { "Male", "Female" })]

public string Gender = "Male";

Výchozí hodnoty argumentů (pokud nebude jejich hodnota předána přímo) lze snadno specifikovat takto:

[ArgumentAlias("maxlen")]

public string Maxlen = 16;

## Zápis na příkazové řádce

Hodnota parametru na příkazové řádce může být zadána těmito dvěma způsoby:

-len 5

-len=5

Oba tyto způsoby zápisu jsou rovnocenné.

Pokud bude mít parametr typu string definovanou hodnotu volitelně a hodnota bude moci začínat pomlčkou, pak argumenty zapsane takto:

-str -opt

budou považovány za dva různé argumenty a ne za argument -str a jeho hodnotu -opt. Bude-li potřeba aby stringový argument měl hodnotu začínající pomlčkou je třeba tento zápis:

-str=-opt

Argumenty, které mají specifikování hodnoty zakázáno musejí být typu bool.

## Vyjímky

Všechny tyto vyjimky obsahují zprávu o podrobnostech, kterou je vhodné vypsat uživateli programu.

|  |  |
| --- | --- |
| Vyjímka | Situace |
| ArgumentException | Pokud BoundedArgument bude definován u proměnné, která není typu int. |
| Pokud ChoicesArgument bude definován u proměnné, která není typu string. |
| Pokud při parsování nejsou naplněné všechny proměnné, které jsou vyžadovanou volbou. |
| Pokud se při parsování vyskytne parametr, který nebyl definovaný a nejedná se o obyčejný parametr. |
| Pokud při parsování bude jeden argument zadán vícekrát. |
| Pokud promenna atributu bude nepodporovaneho typu. |
| Pokud bude specifikována hodnota atributu, který to má zakázané. |
| Pokud nebude specifikována hodnota atributy, který to vyžaduje. |
| Pokud specifikace hodnoty atributu, který má daný seznam hodnot, není ze seznamu. |
| ArgumentOutOfRangeException | Pokud výchozí hodnota proměnné, která má definované meze, není v těchto mezích |
| Pokud výchozí hodnota proměnné, která má definované možnosti, není jedna z těchto možností |
| Pokud specifikovaná hodnota atributu, který má dané hranice není v rozsahu |
| DuplicateArgumentException | Pokud bude definováno více voleb stejného jména. |

## Obyčejné parametry

Obyčejné parametry jsou nepojmenované stringové hodnoty, které se programu předávají v daném pořadí. Můžou být povinné i nepovinné. Po rozparsování dostane Programátor jejich seznam. Přes obyčejné parametry lze jednoduše specifikovat například vstupní a výstupní soubor, nebo internetová adresa. Díky pevnému pořadí těchto parametrů není třeba k jejich rozlišení speciálních voleb.

Seznam obyčejných parametrů lze definovat anotací atributem CommonArgument, ve které se specifikuje popis parametru a to, zda je povinný. Pořadí obyčejných parametrů je dáno pořadím jejich anotací. Takto také budou vypsány požadavky na obyčejné parametry v dokumentaci voleb programu.

Seznam obyčejných parametrů lze po rozparsování získat z parseru z vlastnosti CommonArguments na instanci třídy ArgumentParser.

Příklad použití:

[CommonArgument("InputFile", IsMandatory = true)]

class ExampleSettings2

{

// definition of arguments

}

# Spuštění

Pro spuštění je dodán Makefile pro GNU make. V kořenovém adresáři projektu je Makefile, který volá Makefile soubory jak pro testovací aplikaci, tak pro samotnou knihovnu. Pro snadné spuštění testovací aplikace je v kořenovém adresáři s Makefile souborem také shell skript run.sh, který testovací aplikaci spustí.