Using Keyword Parameter Parser

Many keywords in Odysseus Scripts supports or needs the definition of parameters. There is an existing parser class **PreParserKeywordParameterHelper** , that supports an easy definition and parsing of these parameters. The following code example shows, how define these parameters. There are two possibilities (with the name of the parameter or a short form without names). Note that if there are no names given, the position of the parameter is relevant. If the parameters are defined with names, the position is arbitrary.

```
/// keyword with parameters and their names
#PARALLELIZATION (type=INTER_OPERATOR) (degree=2) (buffersize=10000000) (optimization=true)
(threadedbuffer=true)

/// or definition without names
#PARALLELIZATION INTER_OPERATOR 2 10000000 true true
```

The following steps are needed, to use this parser.

1. Creating the parameter enum. This enum needs to implement the Interface IKeywordParameter. The enum provides the following informations:

- name - the name, which can be used in Odysseus Script, need to be unique
- position - if no parameter name is set in odysseus script these position identifies the parameter
- isOptional - the parameter is not mandatory

The following code shows an example for implementing these enums.

```
package de.uniol.inf.is.odysseus.parallelization.parameter;


import de.uniol.inf.is.odysseus.script.parser.parameter.IKeywordParameter;

public enum ParallelizationKeywordParameter implements IKeywordParameter{
	PARALLELIZATION_TYPE("type", 0, false),
	DEGREE_OF_PARALLELIZATION("degree", 1, false),
	BUFFERSIZE("buffersize", 2, false),
	OPTIMIZATION("optimization", 3, true),
	THREADEDBUFFER("threadedbuffer", 4, true);

	private ParallelizationKeywordParameter(String name, int position, boolean isOptional){
		this.name = name;
		this.position = position;
		this.isOptional = isOptional;
	}

	private String name;
	private Integer position;
	private boolean isOptional;

	@Override
	public String getName() {
		return this.name;
	}

	@Override
	public int getPosition() {
		return this.position;
	}

	@Override
	public boolean isOptional() {
		return this.isOptional;
	}
}
```

2. Instantiating the parser (a useful place is inside the validate method of the keyword-class)

```
parameterHelper = PreParserKeywordParameterHelper
                                .newInstance(ParallelizationKeywordParameter.class);
```

3. Validating the given String (a useful place is inside the validate method of the keyword-class)

```
parameterHelper.validateParameterString(parameterString);
```

4. Do the parsing (a useful place is inside the execute method of the keyword-class) and getting the values of the result. The parsing returns a map with the above created enum as key and the parsed value as the value of the map. Because of this design it is easy to get the values from the resulting map, as shown in the following code example.

```
// do parsing
Map<IKeywordParameter, String> result = parameterHelper.parse(parameterString);

// getting values from result
String value = result.get(ParallelizationKeywordParameter.DEGREE_OF_PARALLELIZATION);
```

Info

By default the structure of the parameter string is validated through and regular expression. In this case the value of the parameter need to contain only a-z0-9 and underscores. If the value has a more complex structure it is possible to define a custom regular expression. In this case the instantiation of the parser is shown in the following code example. In this example it is needed to change the expression for the constant PATTERN_PAIRS.

```
private static final String PATTERN_PAIRS = "((([a-zA-Z0-9_]+)|([(][a-zA-Z0-9_]+([:][a-zA-Z0-9_]+)+[)]))
[,])"
                        + "*(([a-zA-Z0-9_]+)|([(][a-zA-Z0-9_]+([:][a-zA-Z0-9_]+)+[)]))";
private static final String PATTERN_WITH_ATTRIBUTESNAMES = "([(][a-zA-Z0-9_]+[=]"
                        + PATTERN_PAIRS
                        + "[)])([\\s][(][a-zA-Z0-9_]+[=]"
                        + PATTERN_PAIRS
                        + "[)])*";
private static final String PATTERN_WITHOUT_ATTRIBUTENAMES = "("
                        + PATTERN_PAIRS + ")([\\s]" + PATTERN_PAIRS + ")*";
private static final String PATTERN_KEYWORD = PATTERN_WITH_ATTRIBUTESNAMES
                        + "|" + PATTERN_WITHOUT_ATTRIBUTENAMES;

...

parameterHelper = PreParserKeywordParameterHelper.newInstance
(InterOperatorParallelizationKeywordParameter.class, PATTERN_KEYWORD);
```