

# HELM Prompt Browser

0.1

Generated by Doxygen 1.13.2



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 HPB Namespace Reference	9
5.1.1 Enumeration Type Documentation	10
5.1.1.1 Vendor	10
5.1.2 Variable Documentation	11
5.1.2.1 DTColumnCount	11
5.1.2.2 DTDatasetNameColumn	11
5.1.2.3 DTLMListColumn	11
5.1.2.4 DTNumberOfModels	11
5.1.2.5 list_10	11
5.1.2.6 list_2	11
5.1.2.7 list_24	11
5.1.2.8 list_32	11
5.1.2.9 list_39	12
5.1.2.10 list_40	12
5.1.2.11 list_42	12
5.1.2.12 list_6	12
5.1.2.13 list_66a	12
5.1.2.14 list_66b	12
5.1.2.15 list_67	12
5.1.2.16 list_69	13
5.1.2.17 list_70	13
5.1.2.18 PTCIDColumn	13
5.1.2.19 PTCColumnCount	13
5.1.2.20 PTDatasetBaseColumn	13
5.1.2.21 PTDatasetSpecColumn	13
5.1.2.22 PTHasSpecificationsColumn	13
5.1.2.23 PTIsPromptColumn	13
5.1.2.24 PTIsSelectedColumn	13
5.1.2.25 PTNameIDColumn	14
5.1.2.26 PTPromptContentsColumn	14
5.1.2.27 PTReferencesColumn	14

5.2 Ui Namespace Reference . . . . .	14
<b>6 Class Documentation</b>	<b>15</b>
6.1 BooleanParser Class Reference . . . . .	15
6.1.1 Constructor & Destructor Documentation . . . . .	15
6.1.1.1 BooleanParser() . . . . .	15
6.1.2 Member Function Documentation . . . . .	15
6.1.2.1 check() . . . . .	15
6.1.2.2 parse() . . . . .	15
6.2 ExportOptionsDialog Class Reference . . . . .	16
6.2.1 Constructor & Destructor Documentation . . . . .	16
6.2.1.1 ExportOptionsDialog() . . . . .	16
6.2.1.2 ~ExportOptionsDialog() . . . . .	16
6.3 Expression Class Reference . . . . .	16
6.3.1 Constructor & Destructor Documentation . . . . .	17
6.3.1.1 Expression() [1/2] . . . . .	17
6.3.1.2 Expression() [2/2] . . . . .	17
6.3.2 Member Function Documentation . . . . .	17
6.3.2.1 addOperand() . . . . .	17
6.3.2.2 addOperands() . . . . .	17
6.3.2.3 clear() . . . . .	17
6.3.2.4 lhs() . . . . .	17
6.3.2.5 literal() . . . . .	18
6.3.2.6 op() . . . . .	18
6.3.2.7 rhs() . . . . .	18
6.3.2.8 scope() . . . . .	18
6.3.2.9 setOperator() . . . . .	18
6.4 LanguageModel Class Reference . . . . .	18
6.4.1 Constructor & Destructor Documentation . . . . .	18
6.4.1.1 LanguageModel() . . . . .	18
6.4.2 Member Function Documentation . . . . .	19
6.4.2.1 id() . . . . .	19
6.4.2.2 name() . . . . .	19
6.4.2.3 parameters() . . . . .	19
6.4.2.4 vendor() . . . . .	19
6.5 VendorDialog Class Reference . . . . .	19
6.5.1 Constructor & Destructor Documentation . . . . .	19
6.5.1.1 VendorDialog() . . . . .	19
6.5.1.2 ~VendorDialog() . . . . .	19
<b>7 File Documentation</b>	<b>21</b>
7.1 src/dialogs/exporthoptionsdialog.cpp File Reference . . . . .	21
7.2 src/dialogs/exporthoptionsdialog.hpp File Reference . . . . .	21

7.3 exportoptionsdialog.hpp . . . . .	21
7.4 src/dialogs/vendordialog.cpp File Reference . . . . .	22
7.5 src/dialogs/vendordialog.hpp File Reference . . . . .	22
7.6 vendordialog.hpp . . . . .	22
7.7 src/helperfunctions.cpp File Reference . . . . .	23
7.7.1 Function Documentation . . . . .	24
7.7.1.1 addPromptsToTree() . . . . .	24
7.7.1.2 Ask() . . . . .	25
7.7.1.3 deleteDatasetFromTree() . . . . .	25
7.7.1.4 generateCustomDataset() . . . . .	25
7.7.1.5 getCID() . . . . .	26
7.7.1.6 getDatasetBase() . . . . .	26
7.7.1.7 getDatasetSpec() . . . . .	26
7.7.1.8 getFiltersFromDatasetList() . . . . .	26
7.7.1.9 getHelmTaskDirs() . . . . .	26
7.7.1.10 getName() . . . . .	27
7.7.1.11 getPID() . . . . .	27
7.7.1.12 getPrompt() . . . . .	27
7.7.1.13 getPromptText() . . . . .	27
7.7.1.14 getReferences() . . . . .	27
7.7.1.15 getReferencesText() . . . . .	27
7.7.1.16 getSamples() . . . . .	28
7.7.1.17 getSelectedDatasetNames() . . . . .	28
7.7.1.18 getTaskInstances() . . . . .	28
7.7.1.19 hasSelectedPrompts() . . . . .	29
7.7.1.20 hasSpecifications() . . . . .	29
7.7.1.21 isPrompt() . . . . .	29
7.7.1.22 isSelected() . . . . .	29
7.7.1.23 loadHelmDataConfig() . . . . .	29
7.7.1.24 matches() . . . . .	30
7.7.1.25 PopUp() . . . . .	30
7.7.1.26 setCID() . . . . .	31
7.7.1.27 setSelectedStatus() . . . . .	31
7.7.1.28 splitDatasetName() . . . . .	31
7.7.1.29 transformDatasetTree() . . . . .	31
7.7.1.30 transformPromptTree() . . . . .	31
7.7.1.31 Warn() . . . . .	32
7.8 src/helperfunctions.hpp File Reference . . . . .	32
7.8.1 Function Documentation . . . . .	33
7.8.1.1 addPromptsToTree() . . . . .	33
7.8.1.2 Ask() . . . . .	34
7.8.1.3 deleteDatasetFromTree() . . . . .	34

7.8.1.4 generateCustomDataset()	34
7.8.1.5 getCID()	35
7.8.1.6 getDatasetBase()	35
7.8.1.7 getDatasetSpec()	35
7.8.1.8 getFiltersFromDatasetList()	35
7.8.1.9 getHelmTaskDirs()	35
7.8.1.10 getModelList()	36
7.8.1.11 getName()	36
7.8.1.12 getPID()	36
7.8.1.13 getPrompt()	36
7.8.1.14 getReferences()	36
7.8.1.15 getSamples()	36
7.8.1.16 getSelectedDatasetNames()	36
7.8.1.17 getTaskInstances()	37
7.8.1.18 hasSelectedPrompts()	37
7.8.1.19 hasSpecifications()	37
7.8.1.20 isPrompt()	38
7.8.1.21 isSelected()	38
7.8.1.22 loadHelmDataConfig()	38
7.8.1.23 matches()	38
7.8.1.24 PopUp()	39
7.8.1.25 prettyPrint()	39
7.8.1.26 setCID()	39
7.8.1.27 setSelectedStatus()	39
7.8.1.28 splitDatasetName()	39
7.8.1.29 transformDatasetTree()	39
7.8.1.30 transformPromptTree()	40
7.8.1.31 Warn()	40
7.8.2 Variable Documentation	40
7.8.2.1 _range	40
7.9 helperfunctions.hpp	41
7.10 src/hpb_globals.hpp File Reference	42
7.11 hpb_globals.hpp	43
7.12 src/languagemodel.cpp File Reference	44
7.13 src/languagemodel.hpp File Reference	44
7.14 languagemodel.hpp	44
7.15 src/parser/booleanparser.cpp File Reference	45
7.16 src/parser/booleanparser.hpp File Reference	45
7.16.1 Enumeration Type Documentation	45
7.16.1.1 TokenType	45
7.17 booleanparser.hpp	46
7.18 src/parser/expression.cpp File Reference	46

7.19 src/parser/expression.hpp File Reference	47
7.19.1 Enumeration Type Documentation	47
7.19.1.1 Operator	47
7.20 expression.hpp	47
7.21 src/parser/logic.cpp File Reference	48
7.21.1 Function Documentation	48
7.21.1.1 isAtomic()	48
7.21.1.2 isConjunction()	48
7.21.1.3 isDisjunction()	48
7.21.1.4 isDNF()	48
7.21.1.5 isNegation()	48
7.21.1.6 isNNF()	49
7.21.1.7 NNFToDNF()	49
7.21.1.8 toDNF()	49
7.21.1.9 toNNF()	49
7.22 src/parser/logic.hpp File Reference	49
7.22.1 Function Documentation	49
7.22.1.1 isAtomic()	49
7.22.1.2 isConjunction()	50
7.22.1.3 isDisjunction()	50
7.22.1.4 isDNF()	50
7.22.1.5 isNegation()	50
7.22.1.6 isNNF()	50
7.22.1.7 NNFToDNF()	50
7.22.1.8 toDNF()	50
7.22.1.9 toNNF()	50
7.23 logic.hpp	51
7.24 src/parser/queryparser.cpp File Reference	51
7.24.1 Function Documentation	51
7.24.1.1 checkQuery()	51
7.24.1.2 getQueries()	51
7.25 src/parser/queryparser.hpp File Reference	51
7.25.1 Function Documentation	52
7.25.1.1 checkQuery()	52
7.25.1.2 getQueries()	52
7.26 queryparser.hpp	52
<b>Index</b>	<b>53</b>





# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">HPB</a>	.....	<a href="#">9</a>
<a href="#">Ui</a>	.....	<a href="#">14</a>



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

BooleanParser . . . . .	15
Expression . . . . .	16
LanguageModel . . . . .	18
QDialog	
ExportOptionsDialog . . . . .	16
VendorDialog . . . . .	19



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">BooleanParser</a>	15
<a href="#">ExportOptionsDialog</a>	16
<a href="#">Expression</a>	16
<a href="#">LanguageModel</a>	18
<a href="#">VendorDialog</a>	19



# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">src/helperfunctions.cpp</a>	23
<a href="#">src/helperfunctions.hpp</a>	32
<a href="#">src/hpb_globals.hpp</a>	42
<a href="#">src/languagemodel.cpp</a>	44
<a href="#">src/languagemodel.hpp</a>	44
<a href="#">src/dialogs/exporthoptionsdialog.cpp</a>	21
<a href="#">src/dialogs/exporthoptionsdialog.hpp</a>	21
<a href="#">src/dialogs/vendordialog.cpp</a>	22
<a href="#">src/dialogs/vendordialog.hpp</a>	22
<a href="#">src/parser/booleanparser.cpp</a>	45
<a href="#">src/parser/booleanparser.hpp</a>	45
<a href="#">src/parser/expression.cpp</a>	46
<a href="#">src/parser/expression.hpp</a>	47
<a href="#">src/parser/logic.cpp</a>	48
<a href="#">src/parser/logic.hpp</a>	49
<a href="#">src/parser/queryparser.cpp</a>	51
<a href="#">src/parser/queryparser.hpp</a>	51





## Chapter 5

# Namespace Documentation

### 5.1 HPB Namespace Reference

#### Enumerations

- enum class [Vendor](#) : uint8\_t {  
[AlephAlpha](#) = 0x0 , [ai21](#) = 0x1 , [anthropic](#) = 0x2 , [cohere](#) = 0x3 ,  
[eleutherai](#) = 0x4 , [lmsys](#) = 0x5 , [meta](#) = 0x6 , [microsoft](#) = 0x7 ,  
[mistralai](#) = 0x8 , [mosaicml](#) = 0x9 , [openai](#) = 0xA , [stanford](#) = 0xB ,  
[tiiuae](#) = 0xC , [together](#) = 0xD , [writer\\_palmyra](#) = 0xE }

#### Variables

- constexpr int [DTColumnCount](#) = 3
- constexpr int [DTDatasetNameColumn](#) = 0
- constexpr int [DTNumberOfModels](#) = 1
- constexpr int [DTLMListColumn](#) = 2
- constexpr int [PTColumnCount](#) = 9
- constexpr int [PTCIDColumn](#) = 0
- constexpr int [PTNameIDColumn](#) = 1
- constexpr int [PTDatasetBaseColumn](#) = 2
- constexpr int [PTDatasetSpecColumn](#) = 3
- constexpr int [PTIsPromptColumn](#) = 4
- constexpr int [PTPromptContentsColumn](#) = 5
- constexpr int [PTReferencesColumn](#) = 6
- constexpr int [PTHasSpecificationsColumn](#) = 7
- constexpr int [PTIsSelectedColumn](#) = 8
- const QList< int > [list\\_70](#) = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x41, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 }
- const QList< int > [list\\_69](#) = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 }

- `const QList< int > list_67 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 }`
- `const QList< int > list_66a = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0 }`
- `const QList< int > list_66b = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 }`
- `const QList< int > list_42 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 }`
- `const QList< int > list_40 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD }`
- `const QList< int > list_39 = { 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD }`
- `const QList< int > list_32 = { 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD2, 0xD3, 0xD4, 0xD5 }`
- `const QList< int > list_24 = { 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD2, 0xD3, 0xD4, 0xD5 }`
- `const QList< int > list_10 = { 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD }`
- `const QList< int > list_6 = { 0x20, 0xD0, 0xD2, 0xD3, 0xD4, 0xD5 }`
- `const QList< int > list_2 = { 0xA2, 0xA3 }`

## 5.1.1 Enumeration Type Documentation

### 5.1.1.1 Vendor

```
enum class HPB::Vendor : uint8_t [strong]
```

Enumerator

AlephAlpha	
ai21	
anthropic	
cohere	
eleutherai	
lmsys	
meta	
microsoft	
mistralai	
mosaicml	
openai	
stanford	
tiiuae	

together	
writer_palmyra	

## 5.1.2 Variable Documentation

### 5.1.2.1 DTColumnCount

```
int HPB::DTColumnCount = 3 [inline], [constexpr]
```

### 5.1.2.2 DTDatasetNameColumn

```
int HPB::DTDatasetNameColumn = 0 [inline], [constexpr]
```

### 5.1.2.3 DTLMListColumn

```
int HPB::DTLMListColumn = 2 [inline], [constexpr]
```

### 5.1.2.4 DTNumberOfModels

```
int HPB::DTNumberOfModels = 1 [inline], [constexpr]
```

### 5.1.2.5 list\_10

```
const QList<int> HPB::list_10 = { 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD } [inline]
```

### 5.1.2.6 list\_2

```
const QList<int> HPB::list_2 = { 0xA2, 0xA3 } [inline]
```

### 5.1.2.7 list\_24

```
const QList<int> HPB::list_24 = { 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD2, 0xD3, 0xD4, 0xD5 } [inline]
```

### 5.1.2.8 list\_32

```
const QList<int> HPB::list_32 = { 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD2, 0xD3, 0xD4, 0xD5 } [inline]
```

### 5.1.2.9 list\_39

```
const QList<int> HPB::list_39 = { 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31,
0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD } [inline]
```

### 5.1.2.10 list\_40

```
const QList<int> HPB::list_40 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16,
0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5,
0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD }
[inline]
```

### 5.1.2.11 list\_42

```
const QList<int> HPB::list_42 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16,
0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0,
0xE1 } [inline]
```

### 5.1.2.12 list\_6

```
const QList<int> HPB::list_6 = { 0x20, 0xD0, 0xD2, 0xD3, 0xD4, 0xD5 } [inline]
```

### 5.1.2.13 list\_66a

```
const QList<int> HPB::list_66a = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16,
0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61,
0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4,
0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0 } [inline]
```

### 5.1.2.14 list\_66b

```
const QList<int> HPB::list_66b = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x20,
0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62,
0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5,
0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 } [inline]
```

### 5.1.2.15 list\_67

```
const QList<int> HPB::list_67 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16,
0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61,
0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4,
0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 } [inline]
```

**5.1.2.16 list\_69**

```
const QList<int> HPB::list_69 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16,
0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61,
0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2,
0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 } [inline]
```

**5.1.2.17 list\_70**

```
const QList<int> HPB::list_70 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16,
0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x41, 0x42, 0x50, 0x51, 0x60,
0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1,
0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 } [inline]
```

**5.1.2.18 PTCIDColumn**

```
int HPB::PTCIDColumn = 0 [inline], [constexpr]
```

**5.1.2.19 PTColumnCount**

```
int HPB::PTColumnCount = 9 [inline], [constexpr]
```

**5.1.2.20 PTDatasetBaseColumn**

```
int HPB::PTDatasetBaseColumn = 2 [inline], [constexpr]
```

**5.1.2.21 PTDatasetSpecColumn**

```
int HPB::PTDatasetSpecColumn = 3 [inline], [constexpr]
```

**5.1.2.22 PTHasSpecificationsColumn**

```
int HPB::PTHasSpecificationsColumn = 7 [inline], [constexpr]
```

**5.1.2.23 PTIsPromptColumn**

```
int HPB::PTIsPromptColumn = 4 [inline], [constexpr]
```

**5.1.2.24 PTIsSelectedColumn**

```
int HPB::PTIsSelectedColumn = 8 [inline], [constexpr]
```

#### 5.1.2.25 PTNameIDColumn

```
int HPB::PTNameIDColumn = 1 [inline], [constexpr]
```

#### 5.1.2.26 PTPromptContentsColumn

```
int HPB::PTPromptContentsColumn = 5 [inline], [constexpr]
```

#### 5.1.2.27 PTReferencesColumn

```
int HPB::PTReferencesColumn = 6 [inline], [constexpr]
```

## 5.2 Ui Namespace Reference

## Chapter 6

# Class Documentation

### 6.1 BooleanParser Class Reference

```
#include <booleanparser.hpp>
```

#### Public Member Functions

- [BooleanParser](#) ()
- bool [parse](#) (const QString &formula, [Expression](#) &expr)
- bool [check](#) (const QString &formula)

#### 6.1.1 Constructor & Destructor Documentation

##### 6.1.1.1 BooleanParser()

```
BooleanParser::BooleanParser ()
```

#### 6.1.2 Member Function Documentation

##### 6.1.2.1 check()

```
bool BooleanParser::check (  
    const QString & formula)
```

##### 6.1.2.2 parse()

```
bool BooleanParser::parse (  
    const QString & formula,  
    Expression & expr)
```

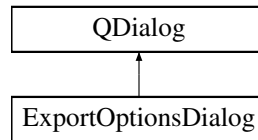
The documentation for this class was generated from the following files:

- [src/parser/booleanparser.hpp](#)
- [src/parser/booleanparser.cpp](#)

## 6.2 ExportOptionsDialog Class Reference

```
#include <exportoptionsdialog.hpp>
```

Inheritance diagram for ExportOptionsDialog:



### Public Member Functions

- [ExportOptionsDialog](#) (QString &outputPath, QString &outputFile, QString &compilationName, QString &helmDataJson, QWidget \*parent=nullptr)
- [~ExportOptionsDialog](#) () override

### 6.2.1 Constructor & Destructor Documentation

#### 6.2.1.1 ExportOptionsDialog()

```
ExportOptionsDialog::ExportOptionsDialog (  
    QString & outputPath,  
    QString & outputFile,  
    QString & compilationName,  
    QString & helmDataJson,  
    QWidget * parent = nullptr) [explicit]
```

#### 6.2.1.2 ~ExportOptionsDialog()

```
ExportOptionsDialog::~ExportOptionsDialog () [override]
```

The documentation for this class was generated from the following files:

- [src/dialogs/exportoptionsdialog.hpp](#)
- [src/dialogs/exportoptionsdialog.cpp](#)

## 6.3 Expression Class Reference

```
#include <expression.hpp>
```



## Public Member Functions

- `Expression()`=default
- `Expression (Operator op, QString literal="", const QList< Expression > &={})`
- `void setOperator (Operator op)`
- `void addOperand (const Expression &expr)`
- `void addOperands (const QList< Expression > &expressions)`
- `const Expression & lhs () const`
- `const Expression & rhs () const`
- `const Expression & scope () const`
- `const QString & literal () const`
- `Operator op () const`
- `void clear ()`

## 6.3.1 Constructor & Destructor Documentation

### 6.3.1.1 Expression() [1/2]

```
Expression::Expression () [default]
```

### 6.3.1.2 Expression() [2/2]

```
Expression::Expression (  
    Operator op,  
    QString literal = "",  
    const QList< Expression > & children = {})
```

## 6.3.2 Member Function Documentation

### 6.3.2.1 addOperand()

```
void Expression::addOperand (  
    const Expression & expr)
```

### 6.3.2.2 addOperands()

```
void Expression::addOperands (  
    const QList< Expression > & expressions)
```

### 6.3.2.3 clear()

```
void Expression::clear ()
```

### 6.3.2.4 lhs()

```
const Expression & Expression::lhs () const
```

#### 6.3.2.5 literal()

```
const QString & Expression::literal () const
```

#### 6.3.2.6 op()

```
Operator Expression::op () const
```

#### 6.3.2.7 rhs()

```
const Expression & Expression::rhs () const
```

#### 6.3.2.8 scope()

```
const Expression & Expression::scope () const
```

#### 6.3.2.9 setOperator()

```
void Expression::setOperator (  
    Operator op)
```

The documentation for this class was generated from the following files:

- [src/parser/expression.hpp](#)
- [src/parser/expression.cpp](#)

## 6.4 LanguageModel Class Reference

```
#include <languagemodel.hpp>
```

### Public Member Functions

- [LanguageModel](#) (int [id](#), QString [name](#), double [parameters](#))
- const QString & [name](#) () const
- double [parameters](#) () const
- [HPB::Vendor](#) [vendor](#) () const
- int [id](#) () const

### 6.4.1 Constructor & Destructor Documentation

#### 6.4.1.1 LanguageModel()

```
LanguageModel::LanguageModel (  
    int id,  
    QString name,  
    double parameters)
```

## 6.4.2 Member Function Documentation

### 6.4.2.1 id()

```
int LanguageModel::id () const
```

### 6.4.2.2 name()

```
const QString & LanguageModel::name () const
```

### 6.4.2.3 parameters()

```
double LanguageModel::parameters () const
```

### 6.4.2.4 vendor()

```
HPB::Vendor LanguageModel::vendor () const
```

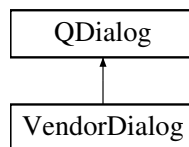
The documentation for this class was generated from the following files:

- [src/languagemodel.hpp](#)
- [src/languagemodel.cpp](#)

## 6.5 VendorDialog Class Reference

```
#include <vendordialog.hpp>
```

Inheritance diagram for VendorDialog:



### Public Member Functions

- [VendorDialog](#) (QList< int > &vendorList, QWidget \*parent=nullptr)
- [~VendorDialog](#) () override

## 6.5.1 Constructor & Destructor Documentation

### 6.5.1.1 VendorDialog()

```
VendorDialog::VendorDialog (
    QList< int > & vendorList,
    QWidget * parent = nullptr) [explicit]
```

### 6.5.1.2 ~VendorDialog()

```
VendorDialog::~VendorDialog () [override]
```

The documentation for this class was generated from the following files:

- [src/dialogs/vendordialog.hpp](#)
- [src/dialogs/vendordialog.cpp](#)



# Chapter 7

## File Documentation

### 7.1 src/dialogs/exportoptionsdialog.cpp File Reference

```
#include "exportoptionsdialog.hpp"
#include "ui_exportoptionsdialog.h"
#include <QFileDialog>
#include <QMessageBox>
#include <QStandardPaths>
```

### 7.2 src/dialogs/exportoptionsdialog.hpp File Reference

```
#include <QDialog>
```

#### Classes

- class [ExportOptionsDialog](#)

#### Namespaces

- namespace [Ui](#)

### 7.3 exportoptionsdialog.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <QDialog>
00004
00005 namespace Ui {
00006     class ExportOptionsDialog;
00007 } // namespace Ui
00008
00009 class ExportOptionsDialog : public QDialog
00010 {
```

```

00011     Q_OBJECT
00012
00013 public:
00014     explicit ExportOptionsDialog(QString& outputPath, QString& outputFile, QString& compilationName,
00015                                 QString& helmDataJson, QWidget *parent = nullptr);
00016     ~ExportOptionsDialog() override;
00017 private slots:
00018     void on_buttonBox_accepted();
00019     void on_export_path_pushButton_clicked();
00020     void on_helmDataJSON_pushButton_clicked();
00021
00022 private:
00023     Ui::ExportOptionsDialog *ui;
00024
00025     QString& m_outputPath;
00026     QString& m_outputFile;
00027     QString& m_compilationName;
00028     QString& m_helmDataJSON;
00029 };

```

## 7.4 src/dialogs/vendordialog.cpp File Reference

```

#include "vendordialog.hpp"
#include "ui_vendordialog.h"
#include <QListWidget>
#include <QListWidgetItem>

```

## 7.5 src/dialogs/vendordialog.hpp File Reference

```

#include <QDialog>
#include <QList>

```

### Classes

- class [VendorDialog](#)

### Namespaces

- namespace [Ui](#)

## 7.6 vendordialog.hpp

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <QDialog>
00004 #include <QList>
00005
00006 namespace Ui {
00007     class VendorDialog;
00008 } // namespace Ui
00009
00010 class VendorDialog : public QDialog
00011 {
00012     Q_OBJECT

```

```

00013
00014 public:
00015     explicit VendorDialog(QList<int>& vendorList, QWidget *parent = nullptr);
00016     ~VendorDialog() override;
00017
00018 private slots:
00019     void on_buttonBox_accepted();
00020
00021     void on_clearAll_pushButton_clicked();
00022
00023     void on_selectAll_pushButton_clicked();
00024
00025 private:
00026     Ui::VendorDialog *ui;
00027     QList<int>& m_VendorList;
00028 };

```

## 7.7 src/helperfunctions.cpp File Reference

```

#include "helperfunctions.hpp"
#include <QCheckBox>
#include <QDir>
#include <QFile>
#include <QJsonArray>
#include <QJsonDocument>
#include <QList>
#include <QMessageBox>
#include <QRegularExpression>
#include <QString>
#include <QSysInfo>
#include <QTimer>
#include <QTreeWidgetItem>
#include "hpb_globals.hpp"

```

### Functions

- int [Ask](#) (const QString &text, const QString &informativeText, bool &dontShowAgain)  
*Displays a message box with Yes/No options and an optional "Don't show again" checkbox.*
- void [PopUp](#) (const QString &message)  
*Displays a popup message box that automatically closes after 1.5 seconds.*
- void [Warn](#) (const QString &message)  
*Displays a warning message box.*
- QJsonObject [generateCustomDataset](#) (const QTreeWidgetItem \*item, const QString &datasetBase, const QString &datasetSpec, const QJsonObject &helmDataJson)  
*Generates a custom dataset JSON object based on tree widget item and dataset specifications.*
- QJsonObject [getSamples](#) (const QTreeWidgetItem \*item)  
*Extracts sample data from a QTreeWidgetItem and returns it as a JSON object.*
- QJsonDocument [getTaskInstances](#) (const QString &taskDir, const QString &helmDataPath)  
*Loads task instances from a JSON file within the specified directory.*
- QJsonObject [loadHelmDataConfig](#) (const QString &helmDataJson)  
*Loads the configuration for Helm dataset from a JSON file.*
- QString [getPromptText](#) (const QJsonObject &obj, const QString &dataset)  
*Constructs a formatted prompt text from a JSON object.*
- QString [getReferencesText](#) (const QJsonObject &obj, const QString &dataset)  
*Retrieves formatted references text from a JSON object.*

- void [addPromptsToTree](#) (const QString &dataset, const QJsonDocument &instances, const QList< QPair< QStringList, QStringList > > &queries, const bool searchIsCaseSensitive, const bool searchIsRegex, QTreeWidget \*tree)  
*Adds prompts matching search criteria to a QTreeWidget.*
- void [deleteDatasetFromTree](#) (const QString &datasetName, QTreeWidget \*tree)  
*Deletes a dataset and its prompts from the QTreeWidget.*
- QStringList [getFiltersFromDatasetList](#) (const QStringList &datasetNames)  
*Generates a list of file filters for dataset directories based on the OS.*
- QStringList [getHelmTaskDirs](#) (const QStringList &datasets, const QString &helmDataPath)  
*Retrieves Helm task directories based on dataset names and path.*
- QStringList [getSelectedDatasetNames](#) (const QTreeWidget \*tree)  
*Retrieves the list of selected datasets from a QTreeWidget.*
- bool [hasSelectedPrompts](#) (const QTreeWidgetItem \*item)  
*Checks if a QTreeWidgetItem has any selected prompts.*
- bool [matches](#) (const QString &prompt, const QList< QPair< QStringList, QStringList > > &queries, bool searchIsCaseSensitive, bool searchIsRegex)  
*Determines if a given prompt matches any query based on inclusion and exclusion terms.*
- QPair< QString, QString > [splitDatasetName](#) (const QString &dataset)  
*Splits a dataset name into base and specification parts.*
- void [transformDatasetTree](#) (QTreeWidget \*datasetTree, const std::function< void(QTreeWidgetItem \*)> &transformation)  
*Transforms all dataset entries in a QTreeWidget using a given transformation function.*
- void [transformPromptTree](#) (QTreeWidget \*promptTree, const std::function< void(QTreeWidgetItem \*)> &transformation)  
*Transforms all prompt entries in a QTreeWidget using a given transformation function.*
- QString [getCID](#) (const QTreeWidgetItem \*item)
- QString [getDatasetBase](#) (const QTreeWidgetItem \*item)
- QString [getDatasetSpec](#) (const QTreeWidgetItem \*item)
- QString [getName](#) (const QTreeWidgetItem \*item)
- QString [getPID](#) (const QTreeWidgetItem \*item)
- QString [getPrompt](#) (const QTreeWidgetItem \*item)
- QString [getReferences](#) (const QTreeWidgetItem \*item)
- bool [hasSpecifications](#) (const QTreeWidgetItem \*item)
- bool [isPrompt](#) (const QTreeWidgetItem \*item)
- bool [isSelected](#) (const QTreeWidgetItem \*item)
- void [setCID](#) (QTreeWidgetItem \*item, const QString &cid)
- void [setSelectedStatus](#) (QTreeWidgetItem \*item, bool status)

## 7.7.1 Function Documentation

### 7.7.1.1 addPromptsToTree()

```
void addPromptsToTree (
    const QString & dataset,
    const QJsonDocument & instances,
    const QList< QPair< QStringList, QStringList > > & queries,
    const bool searchIsCaseSensitive,
    const bool searchIsRegex,
    QTreeWidget * tree)
```

Adds prompts matching search criteria to a QTreeWidget.



## Parameters

<i>dataset</i>	The dataset name.
<i>instances</i>	The JSON document containing instance data.
<i>queries</i>	List of query pairs (inclusions and exclusions).
<i>searchIsCaseSensitive</i>	Boolean flag indicating case-sensitive search.
<i>searchIsRegex</i>	Boolean flag indicating if search terms are regular expressions.
<i>tree</i>	The QTreeWidget to populate with matched prompts.

## 7.7.1.2 Ask()

```
int Ask (
    const QString & text,
    const QString & informativeText,
    bool & dontShowAgain)
```

Displays a message box with Yes/No options and an optional "Don't show again" checkbox.

## Parameters

<i>text</i>	The main text of the message box.
<i>informativeText</i>	Additional information displayed in the message box.
<i>dontShowAgain</i>	Reference to a boolean variable indicating if the "Don't show again" checkbox was checked.

## Returns

int The button pressed by the user (QMessageBox::Yes or QMessageBox::No).

## 7.7.1.3 deleteDatasetFromTree()

```
void deleteDatasetFromTree (
    const QString & datasetName,
    QTreeWidget * tree)
```

Deletes a dataset and its prompts from the QTreeWidget.

## Parameters

<i>datasetName</i>	The name of the dataset to delete.
<i>tree</i>	The QTreeWidget containing the dataset structure.

## 7.7.1.4 generateCustomDataset()

```
QJsonObject generateCustomDataset (
    const QTreeWidgetItem * item,
    const QString & datasetBase,
    const QString & datasetSpec,
    const QJsonObject & helmDataJson)
```

Generates a custom dataset JSON object based on tree widget item and dataset specifications.

**Parameters**

<i>item</i>	The tree widget item representing a dataset entry.
<i>datasetBase</i>	The base name of the dataset.
<i>datasetSpec</i>	The dataset specification (optional).
<i>helmDataJson</i>	The JSON object containing dataset metadata.

**Returns**

QJsonObject The generated dataset JSON object.

**7.7.1.5 getCID()**

```
QString getCID (
    const QTreeWidgetItem * item)
```

**7.7.1.6 getDatasetBase()**

```
QString getDatasetBase (
    const QTreeWidgetItem * item)
```

**7.7.1.7 getDatasetSpec()**

```
QString getDatasetSpec (
    const QTreeWidgetItem * item)
```

**7.7.1.8 getFiltersFromDatasetList()**

```
QStringList getFiltersFromDatasetList (
    const QStringList & datasetNames)
```

Generates a list of file filters for dataset directories based on the OS.

**Parameters**

<i>datasetNames</i>	The list of dataset names.
---------------------	----------------------------

**Returns**

QStringList The list of formatted dataset filters.

**7.7.1.9 getHelmTaskDirs()**

```
QStringList getHelmTaskDirs (
    const QStringList & datasets,
    const QString & helmDataPath)
```

Retrieves Helm task directories based on dataset names and path.

## Parameters

<i>datasets</i>	The list of dataset names.
<i>helmDataPath</i>	The base path for Helm data.

## Returns

QStringList The list of task directories.

**7.7.1.10 getName()**

```
QString getName (  
    const QTreeWidgetItem * item)
```

**7.7.1.11 getPID()**

```
QString getPID (  
    const QTreeWidgetItem * item)
```

**7.7.1.12 getPrompt()**

```
QString getPrompt (  
    const QTreeWidgetItem * item)
```

**7.7.1.13 getPromptText()**

```
QString getPromptText (  
    const QJsonObject & obj,  
    const QString & dataset)
```

Constructs a formatted prompt text from a JSON object.

## Parameters

<i>obj</i>	The JSON object containing prompt details.
<i>dataset</i>	The dataset name associated with the prompt.

## Returns

QString The formatted prompt text.

**7.7.1.14 getReferences()**

```
QString getReferences (  
    const QTreeWidgetItem * item)
```

**7.7.1.15 getReferencesText()**

```
QString getReferencesText (  
    const QJsonObject & obj,  
    const QString & dataset)
```

Retrieves formatted references text from a JSON object.

**Parameters**

<i>obj</i>	The JSON object containing reference details.
<i>dataset</i>	The dataset name associated with the references.

**Returns**

QString The formatted references text.

**7.7.1.16 getSamples()**

```
QJsonObject getSamples (  
    const QTreeWidgetItem * item)
```

Extracts sample data from a QTreeWidgetItem and returns it as a JSON object.

**Parameters**

<i>item</i>	The tree widget item representing a dataset entry.
-------------	--

**Returns**

QJsonObject The extracted samples.

**7.7.1.17 getSelectedDatasetNames()**

```
QStringList getSelectedDatasetNames (  
    const QTreeWidget * tree)
```

Retrieves the list of selected datasets from a QTreeWidget.

**Parameters**

<i>tree</i>	The QTreeWidget representing the dataset structure.
-------------	---

**Returns**

QStringList The list of selected dataset names.

**7.7.1.18 getTaskInstances()**

```
QJsonDocument getTaskInstances (  
    const QString & taskDir,  
    const QString & helmDataPath)
```

Loads task instances from a JSON file within the specified directory.

**Parameters**

<i>taskDir</i>	The directory containing the instances file.
<i>helmDataPath</i>	The base path for the dataset.

**Returns**

QJsonDocument The loaded task instances as a JSON document.

**7.7.1.19 hasSelectedPrompts()**

```
bool hasSelectedPrompts (  
    const QTreeWidgetItem * item)
```

Checks if a QTreeWidgetItem has any selected prompts.

**Parameters**

<i>item</i>	The tree widget item to check.
-------------	--------------------------------

**Returns**

bool True if the item has selected prompts, false otherwise.

**7.7.1.20 hasSpecifications()**

```
bool hasSpecifications (  
    const QTreeWidgetItem * item)
```

**7.7.1.21 isPrompt()**

```
bool isPrompt (  
    const QTreeWidgetItem * item)
```

**7.7.1.22 isSelected()**

```
bool isSelected (  
    const QTreeWidgetItem * item)
```

**7.7.1.23 loadHelmDataConfig()**

```
QJsonObject loadHelmDataConfig (  
    const QString & helmDataJson)
```

Loads the configuration for Helm dataset from a JSON file.

**Parameters**

<i>helmDataJson</i>	Path to the JSON file containing the Helm dataset configuration.
---------------------	--

**Returns**

QJsonObject The parsed JSON object containing the dataset configuration.

**7.7.1.24 matches()**

```
bool matches (
    const QString & prompt,
    const QList< QPair< QStringList, QStringList > > & queries,
    bool searchIsCaseSensitive,
    bool searchIsRegex)
```

Determines if a given prompt matches any query based on inclusion and exclusion terms.

This function checks if the provided `prompt` satisfies at least one query from `queries`. Each query consists of inclusion and exclusion term lists:

- The prompt must contain all inclusion terms.
- The prompt must not contain any exclusion terms.

The function supports both case-sensitive and case-insensitive searches, as well as regular expression matching.

**Parameters**

<i>prompt</i>	The text to be matched against the queries.
<i>queries</i>	A list of queries, where each query contains a pair of: <ul style="list-style-type: none"> <li>• A list of inclusion terms (all must be present).</li> <li>• A list of exclusion terms (none must be present).</li> </ul>
<i>searchIsCaseSensitive</i>	If true, the search is case-sensitive; otherwise, it's case-insensitive.
<i>searchIsRegex</i>	If true, terms are treated as regular expressions; otherwise, they are treated as plain text.

**Returns**

True if the prompt matches at least one query (meeting all inclusions and avoiding all exclusions); otherwise, false.

**7.7.1.25 PopUp()**

```
void PopUp (
    const QString & message)
```

Displays a popup message box that automatically closes after 1.5 seconds.

## Parameters

<i>message</i>	The message to be displayed in the popup.
----------------	---

**7.7.1.26 setCID()**

```
void setCID (
    QTreeWidgetItem * item,
    const QString & cid)
```

**7.7.1.27 setSelectedStatus()**

```
void setSelectedStatus (
    QTreeWidgetItem * item,
    bool status)
```

**7.7.1.28 splitDatasetName()**

```
QPair< QString, QString > splitDatasetName (
    const QString & dataset)
```

Splits a dataset name into base and specification parts.

## Parameters

<i>dataset</i>	The dataset name.
----------------	-------------------

## Returns

QPair<QString, QString> The separated base name and specification.

**7.7.1.29 transformDatasetTree()**

```
void transformDatasetTree (
    QTreeWidget * datasetTree,
    const std::function< void(QTreeWidgetItem *)> & transformation)
```

Transforms all dataset entries in a QTreeWidget using a given transformation function.

## Parameters

<i>datasetTree</i>	The QTreeWidget representing datasets.
<i>transformation</i>	The transformation function to apply.

**7.7.1.30 transformPromptTree()**

```
void transformPromptTree (
    QTreeWidget * promptTree,
    const std::function< void(QTreeWidgetItem *)> & transformation)
```

Transforms all prompt entries in a QTreeWidget using a given transformation function.

## Parameters

<i>promptTree</i>	The QTreeWidgetItem representing prompts.
<i>transformation</i>	The transformation function to apply.

**7.7.1.31 Warn()**

```
void Warn (
    const QString & message)
```

Displays a warning message box.

## Parameters

<i>message</i>	The warning message to be displayed.
----------------	--------------------------------------

**7.8 src/helperfunctions.hpp File Reference**

```
#include <functional>
#include <ranges>
#include <QJsonObject>
#include <QString>
#include <QStringList>
#include <QTreeWidgetItem>
#include <QTreeWidgetItem>
```

**Functions**

- int [Ask](#) (const QString &text, const QString &informativeText, bool &dontShowAgain)  
*Displays a message box with Yes/No options and an optional "Don't show again" checkbox.*
- void [PopUp](#) (const QString &message)  
*Displays a popup message box that automatically closes after 1.5 seconds.*
- void [Warn](#) (const QString &message)  
*Displays a warning message box.*
- QJsonObject [generateCustomDataset](#) (const QTreeWidgetItem \*item, const QString &datasetBase, const QString &datasetSpec, const QJsonObject &helmDataJson)  
*Generates a custom dataset JSON object based on tree widget item and dataset specifications.*
- QJsonObject [getSamples](#) (const QTreeWidgetItem \*item)  
*Extracts sample data from a QTreeWidgetItem and returns it as a JSON object.*
- QJsonDocument [getTaskInstances](#) (const QString &taskDir, const QString &helmDataPath)  
*Loads task instances from a JSON file within the specified directory.*
- QJsonObject [loadHelmDataConfig](#) (const QString &helmDataJson)  
*Loads the configuration for Helm dataset from a JSON file.*
- QString [prettyPrint](#) (const QJsonObject &obj, const QString &dataset)
- QStringList [getFiltersFromDatasetList](#) (const QStringList &datasetNames)  
*Generates a list of file filters for dataset directories based on the OS.*



- const QList< int > & [getModelList](#) (const QTreeWidgetItem \*)
- QStringList [getSelectedDatasetNames](#) (const QTreeWidgetItem \*tree)  
*Retrieves the list of selected datasets from a QTreeWidgetItem.*
- void [transformDatasetTree](#) (QTreeWidgetItem \*datasetTree, const std::function< void(QTreeWidgetItem \*)> &transformation)  
*Transforms all dataset entries in a QTreeWidgetItem using a given transformation function.*
- void [addPromptsToTree](#) (const QString &dataset, const QJsonDocument &instances, const QList< QPair< QStringList, QStringList > > &queries, bool searchIsCaseSensitive, bool searchIsRegex, QTreeWidgetItem \*tree)  
*Adds prompts matching search criteria to a QTreeWidgetItem.*
- void [deleteDatasetFromTree](#) (const QString &datasetName, QTreeWidgetItem \*tree)  
*Deletes a dataset and its prompts from the QTreeWidgetItem.*
- bool [hasSelectedPrompts](#) (const QTreeWidgetItem \*item)  
*Checks if a QTreeWidgetItem has any selected prompts.*
- void [transformPromptTree](#) (QTreeWidgetItem \*promptTree, const std::function< void(QTreeWidgetItem \*)> &transformation)  
*Transforms all prompt entries in a QTreeWidgetItem using a given transformation function.*
- QStringList [getHelmTaskDirs](#) (const QStringList &datasets, const QString &helmDataPath)  
*Retrieves Helm task directories based on dataset names and path.*
- QPair< QString, QString > [splitDatasetName](#) (const QString &dataset)  
*Splits a dataset name into base and specification parts.*
- QString [getCID](#) (const QTreeWidgetItem \*item)
- QString [getDatasetBase](#) (const QTreeWidgetItem \*item)
- QString [getDatasetSpec](#) (const QTreeWidgetItem \*item)
- QString [getName](#) (const QTreeWidgetItem \*item)
- QString [getPID](#) (const QTreeWidgetItem \*item)
- QString [getPrompt](#) (const QTreeWidgetItem \*item)
- QString [getReferences](#) (const QTreeWidgetItem \*item)
- bool [hasSpecifications](#) (const QTreeWidgetItem \*item)
- bool [isPrompt](#) (const QTreeWidgetItem \*item)
- bool [isSelected](#) (const QTreeWidgetItem \*item)
- bool [matches](#) (const QString &prompt, const QList< QPair< QStringList, QStringList > > &queries, bool searchIsCaseSensitive, bool searchIsRegex)  
*Determines if a given prompt matches any query based on inclusion and exclusion terms.*
- void [setCID](#) (QTreeWidgetItem \*item, const QString &cid)
- void [setSelectedStatus](#) (QTreeWidgetItem \*item, bool status)

## Variables

- auto [\\_range](#) = [] (auto min, auto max) { return std::views::iota(min, max); }

## 7.8.1 Function Documentation

### 7.8.1.1 addPromptsToTree()

```
void addPromptsToTree (
    const QString & dataset,
    const QJsonDocument & instances,
    const QList< QPair< QStringList, QStringList > > & queries,
    const bool searchIsCaseSensitive,
    const bool searchIsRegex,
    QTreeWidgetItem * tree)
```

Adds prompts matching search criteria to a QTreeWidgetItem.

**Parameters**

<i>dataset</i>	The dataset name.
<i>instances</i>	The JSON document containing instance data.
<i>queries</i>	List of query pairs (inclusions and exclusions).
<i>searchIsCaseSensitive</i>	Boolean flag indicating case-sensitive search.
<i>searchIsRegex</i>	Boolean flag indicating if search terms are regular expressions.
<i>tree</i>	The QTreeWidget to populate with matched prompts.

**7.8.1.2 Ask()**

```
int Ask (
    const QString & text,
    const QString & informativeText,
    bool & dontShowAgain)
```

Displays a message box with Yes/No options and an optional "Don't show again" checkbox.

**Parameters**

<i>text</i>	The main text of the message box.
<i>informativeText</i>	Additional information displayed in the message box.
<i>dontShowAgain</i>	Reference to a boolean variable indicating if the "Don't show again" checkbox was checked.

**Returns**

int The button pressed by the user (QMessageBox::Yes or QMessageBox::No).

**7.8.1.3 deleteDatasetFromTree()**

```
void deleteDatasetFromTree (
    const QString & datasetName,
    QTreeWidget * tree)
```

Deletes a dataset and its prompts from the QTreeWidget.

**Parameters**

<i>datasetName</i>	The name of the dataset to delete.
<i>tree</i>	The QTreeWidget containing the dataset structure.

**7.8.1.4 generateCustomDataset()**

```
QJsonObject generateCustomDataset (
    const QTreeWidgetItem * item,
    const QString & datasetBase,
    const QString & datasetSpec,
    const QJsonObject & helmDataJson)
```

Generates a custom dataset JSON object based on tree widget item and dataset specifications.

## Parameters

<i>item</i>	The tree widget item representing a dataset entry.
<i>datasetBase</i>	The base name of the dataset.
<i>datasetSpec</i>	The dataset specification (optional).
<i>helmDataJson</i>	The JSON object containing dataset metadata.

## Returns

QJsonObject The generated dataset JSON object.

## 7.8.1.5 getCID()

```
QString getCID (
    const QTreeWidgetItem * item)
```

## 7.8.1.6 getDatasetBase()

```
QString getDatasetBase (
    const QTreeWidgetItem * item)
```

## 7.8.1.7 getDatasetSpec()

```
QString getDatasetSpec (
    const QTreeWidgetItem * item)
```

## 7.8.1.8 getFiltersFromDatasetList()

```
QStringList getFiltersFromDatasetList (
    const QStringList & datasetNames)
```

Generates a list of file filters for dataset directories based on the OS.

## Parameters

<i>datasetNames</i>	The list of dataset names.
---------------------	----------------------------

## Returns

QStringList The list of formatted dataset filters.

## 7.8.1.9 getHelmTaskDirs()

```
QStringList getHelmTaskDirs (
    const QStringList & datasets,
    const QString & helmDataPath)
```

Retrieves Helm task directories based on dataset names and path.

**Parameters**

<i>datasets</i>	The list of dataset names.
<i>helmDataPath</i>	The base path for Helm data.

**Returns**

QStringList The list of task directories.

**7.8.1.10 getModelList()**

```
const QList< int > & getModelList (
    const QTreeWidgetItem * )
```

**7.8.1.11 getName()**

```
QString getName (
    const QTreeWidgetItem * item)
```

**7.8.1.12 getPID()**

```
QString getPID (
    const QTreeWidgetItem * item)
```

**7.8.1.13 getPrompt()**

```
QString getPrompt (
    const QTreeWidgetItem * item)
```

**7.8.1.14 getReferences()**

```
QString getReferences (
    const QTreeWidgetItem * item)
```

**7.8.1.15 getSamples()**

```
QJsonObject getSamples (
    const QTreeWidgetItem * item)
```

Extracts sample data from a QTreeWidgetItem and returns it as a JSON object.

**Parameters**

<i>item</i>	The tree widget item representing a dataset entry.
-------------	--

**Returns**

QJsonObject The extracted samples.

**7.8.1.16 getSelectedDatasetNames()**

```
QStringList getSelectedDatasetNames (
    const QTreeWidget * tree)
```

Retrieves the list of selected datasets from a QTreeWidget.

**Parameters**

<i>tree</i>	The QTreeWidget representing the dataset structure.
-------------	---

**Returns**

QStringList The list of selected dataset names.

**7.8.1.17 getTaskInstances()**

```
QJsonDocument getTaskInstances (  
    const QString & taskDir,  
    const QString & helmDataPath)
```

Loads task instances from a JSON file within the specified directory.

**Parameters**

<i>taskDir</i>	The directory containing the instances file.
<i>helmDataPath</i>	The base path for the dataset.

**Returns**

QJsonDocument The loaded task instances as a JSON document.

**7.8.1.18 hasSelectedPrompts()**

```
bool hasSelectedPrompts (  
    const QTreeWidgetItem * item)
```

Checks if a QTreeWidgetItem has any selected prompts.

**Parameters**

<i>item</i>	The tree widget item to check.
-------------	--------------------------------

**Returns**

bool True if the item has selected prompts, false otherwise.

**7.8.1.19 hasSpecifications()**

```
bool hasSpecifications (  
    const QTreeWidgetItem * item)
```

### 7.8.1.20 isPrompt()

```
bool isPrompt (
    const QTreeWidgetItem * item)
```

### 7.8.1.21 isSelected()

```
bool isSelected (
    const QTreeWidgetItem * item)
```

### 7.8.1.22 loadHelmDataConfig()

```
QJsonObject loadHelmDataConfig (
    const QString & helmDataJson)
```

Loads the configuration for Helm dataset from a JSON file.

#### Parameters

<i>helmDataJson</i>	Path to the JSON file containing the Helm dataset configuration.
---------------------	--

#### Returns

QJsonObject The parsed JSON object containing the dataset configuration.

### 7.8.1.23 matches()

```
bool matches (
    const QString & prompt,
    const QList< QPair< QStringList, QStringList > > & queries,
    bool searchIsCaseSensitive,
    bool searchIsRegex)
```

Determines if a given prompt matches any query based on inclusion and exclusion terms.

This function checks if the provided `prompt` satisfies at least one query from `queries`. Each query consists of inclusion and exclusion term lists:

- The prompt must contain all inclusion terms.
- The prompt must not contain any exclusion terms.

The function supports both case-sensitive and case-insensitive searches, as well as regular expression matching.

#### Parameters

<i>prompt</i>	The text to be matched against the queries.
<i>queries</i>	A list of queries, where each query contains a pair of: <ul style="list-style-type: none"> <li>• A list of inclusion terms (all must be present).</li> <li>• A list of exclusion terms (none must be present).</li> </ul>
<i>searchIsCaseSensitive</i>	If true, the search is case-sensitive; otherwise, it's case-insensitive.
<i>searchIsRegex</i>	If true, terms are treated as regular expressions; otherwise, they are treated as plain text.

#### Returns

True if the prompt matches at least one query (meeting all inclusions and avoiding all exclusions); otherwise, false.

**7.8.1.24 PopUp()**

```
void PopUp (
    const QString & message)
```

Displays a popup message box that automatically closes after 1.5 seconds.

**Parameters**

<i>message</i>	The message to be displayed in the popup.
----------------	---

**7.8.1.25 prettyPrint()**

```
QString prettyPrint (
    const QJsonObject & obj,
    const QString & dataset)
```

**7.8.1.26 setCID()**

```
void setCID (
    QTreeWidgetItem * item,
    const QString & cid)
```

**7.8.1.27 setSelectedStatus()**

```
void setSelectedStatus (
    QTreeWidgetItem * item,
    bool status)
```

**7.8.1.28 splitDatasetName()**

```
QPair< QString, QString > splitDatasetName (
    const QString & dataset)
```

Splits a dataset name into base and specification parts.

**Parameters**

<i>dataset</i>	The dataset name.
----------------	-------------------

**Returns**

QPair<QString, QString> The separated base name and specification.

**7.8.1.29 transformDatasetTree()**

```
void transformDatasetTree (
    QTreeWidgetItem * datasetTree,
    const std::function< void(QTreeWidgetItem *)> & transformation)
```

Transforms all dataset entries in a QTreeWidgetItem using a given transformation function.

## Parameters

<i>datasetTree</i>	The QTreeWidget representing datasets.
<i>transformation</i>	The transformation function to apply.

**7.8.1.30 transformPromptTree()**

```
void transformPromptTree (
    QTreeWidget * promptTree,
    const std::function< void(QTreeWidgetItem *)> & transformation)
```

Transforms all prompt entries in a QTreeWidget using a given transformation function.

## Parameters

<i>promptTree</i>	The QTreeWidget representing prompts.
<i>transformation</i>	The transformation function to apply.

**7.8.1.31 Warn()**

```
void Warn (
    const QString & message)
```

Displays a warning message box.

## Parameters

<i>message</i>	The warning message to be displayed.
----------------	--------------------------------------

**7.8.2 Variable Documentation****7.8.2.1 \_range**

```
auto _range = [] (auto min, auto max) { return std::views::iota(min, max); } [inline]
```



## 7.9 helperfunctions.hpp

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <functional>
00004 #include <ranges>
00005
00006 #include <QJsonObject>
00007 #include <QString>
00008 #include <QStringList>
00009 #include <QTreeWidget>
00010 #include <QTreeWidgetItem>
00011
00012 inline auto _range = [] (auto min, auto max) { return std::views::iota(min, max); };
00013
00014 /*****
00015  * QMessageBoxes *
00016  *****/
00017
00018 int Ask(const QString& text, const QString& informativeText, bool& dontShowAgain);
00019 void PopUp(const QString& message);
00020 void Warn(const QString& message);
00021
00022 /*****
00023  * QJson convenience functions *
00024  *****/
00025
00026 QJsonObject generateCustomDataset(const QTreeWidgetItem* item, const QString& datasetBase, const
  QString& datasetSpec, const QJsonObject& helmDataJson);
00027 QJsonObject getSamples(const QTreeWidgetItem* item);
00028 QJsonDocument getTaskInstances(const QString& taskDir, const QString& helmDataPath);
00029 QJsonObject loadHelmDataConfig(const QString& helmDataJson);
00030 QString prettyPrint(const QJsonObject& obj, const QString& dataset);
00031
00032 /*****
00033  * Dataset tree convenience functions *
00034  *****/
00035
00036 QStringList getFiltersFromDatasetList(const QStringList& datasetNames);
00037 const QList<int>& getModelList(const QTreeWidgetItem*);
00038 QStringList getSelectedDatasetNames(const QTreeWidgetItem* tree);
00039 void transformDatasetTree(QTreeWidgetItem* datasetTree, const std::function<void(QTreeWidgetItem*)>&
  transformation);
00040
00041 /*****
00042  * Prompt and prompt tree convenience functions *
00043  *****/
00044
00045 void addPromptsToTree(const QString& dataset,
00046                      const QJsonDocument& instances,
00047                      const QList<QPair<QStringList, QStringList>>& queries,
00048                      bool searchIsCaseSensitive,
00049                      bool searchIsRegex,
00050                      QTreeWidgetItem* tree);
00051 void deleteDatasetFromTree(const QString& datasetName, QTreeWidgetItem* tree);
00052 bool hasSelectedPrompts(const QTreeWidgetItem* item);
00053 void transformPromptTree(QTreeWidgetItem* promptTree, const std::function<void(QTreeWidgetItem*)>&
  transformation);
00054
00055 QStringList getHelmTaskDirs(const QStringList& datasets, const QString& helmDataPath);
00056 QPair<QString, QString> splitDatasetName(const QString& dataset);
00057
00058
00059 /*****
00060  * Prompt-related functions *
00061  *****/
00062
00063 QString getCID(const QTreeWidgetItem* item);
00064 QString getDatasetBase(const QTreeWidgetItem* item);
00065 QString getDatasetSpec(const QTreeWidgetItem* item);
00066 QString getName(const QTreeWidgetItem* item);
00067 QString getPID(const QTreeWidgetItem* item);
00068 QString getPrompt(const QTreeWidgetItem* item);
00069 QString getReferences(const QTreeWidgetItem* item);
00070 bool hasSpecifications(const QTreeWidgetItem* item);
00071 bool isPrompt(const QTreeWidgetItem* item);
00072 bool isSelected(const QTreeWidgetItem* item);
00073 bool matches(const QString& prompt,
00074             const QList<QPair<QStringList, QStringList>>& queries,
00075             bool searchIsCaseSensitive,
00076             bool searchIsRegex);
00077 void setCID(QTreeWidgetItem* item, const QString& cid);
00078 void setSelectedStatus(QTreeWidgetItem* item, bool status);

```

## 7.10 src/hpb\_globals.hpp File Reference

```
#include <QList>
```

### Namespaces

- namespace [HPB](#)

### Enumerations

- enum class [HPB::Vendor](#) : uint8\_t {  
[HPB::AlephAlpha](#) = 0x0 , [HPB::ai21](#) = 0x1 , [HPB::anthropic](#) = 0x2 , [HPB::cohere](#) = 0x3 ,  
[HPB::eleutherai](#) = 0x4 , [HPB::lmsys](#) = 0x5 , [HPB::meta](#) = 0x6 , [HPB::microsoft](#) = 0x7 ,  
[HPB::mistralai](#) = 0x8 , [HPB::mosaicml](#) = 0x9 , [HPB::openai](#) = 0xA , [HPB::stanford](#) = 0xB ,  
[HPB::tiiuae](#) = 0xC , [HPB::together](#) = 0xD , [HPB::writer\\_palmyra](#) = 0xE }

### Variables

- constexpr int [HPB::DTColumnCount](#) = 3
- constexpr int [HPB::DTDatasetNameColumn](#) = 0
- constexpr int [HPB::DTNumberOfModels](#) = 1
- constexpr int [HPB::DTLMListColumn](#) = 2
- constexpr int [HPB::PTColumnCount](#) = 9
- constexpr int [HPB::PTCIDColumn](#) = 0
- constexpr int [HPB::PTNameIDColumn](#) = 1
- constexpr int [HPB::PTDatasetBaseColumn](#) = 2
- constexpr int [HPB::PTDatasetSpecColumn](#) = 3
- constexpr int [HPB::PTIsPromptColumn](#) = 4
- constexpr int [HPB::PTPromptContentsColumn](#) = 5
- constexpr int [HPB::PTReferencesColumn](#) = 6
- constexpr int [HPB::PTHasSpecificationsColumn](#) = 7
- constexpr int [HPB::PTIsSelectedColumn](#) = 8
- const QList< int > [HPB::list\\_70](#) = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x41, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 }
- const QList< int > [HPB::list\\_69](#) = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 }
- const QList< int > [HPB::list\\_67](#) = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 }
- const QList< int > [HPB::list\\_66a](#) = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0 }

- `const QList< int > HPB::list_66b = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 }`
- `const QList< int > HPB::list_42 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 }`
- `const QList< int > HPB::list_40 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD }`
- `const QList< int > HPB::list_39 = { 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD }`
- `const QList< int > HPB::list_32 = { 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD2, 0xD3, 0xD4, 0xD5 }`
- `const QList< int > HPB::list_24 = { 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD2, 0xD3, 0xD4, 0xD5 }`
- `const QList< int > HPB::list_10 = { 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD }`
- `const QList< int > HPB::list_6 = { 0x20, 0xD0, 0xD2, 0xD3, 0xD4, 0xD5 }`
- `const QList< int > HPB::list_2 = { 0xA2, 0xA3 }`

## 7.11 hpb\_globals.hpp

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <QList>
00004
00005 namespace HPB {
00006     inline constexpr int DTColumnCount = 3;
00007     inline constexpr int DTDatasetNameColumn = 0;
00008     inline constexpr int DTNumberOfModels = 1;
00009     inline constexpr int DTLMListColumn = 2;
00010
00011     inline constexpr int PTColumnCount = 9;
00012     inline constexpr int PTCIDColumn = 0;
00013     inline constexpr int PTNameIDColumn = 1;
00014     inline constexpr int PTDatasetBaseColumn = 2;
00015     inline constexpr int PTDatasetSpecColumn = 3;
00016     inline constexpr int PTIsPromptColumn = 4;
00017     inline constexpr int PTPromptContentsColumn = 5;
00018     inline constexpr int PTReferencesColumn = 6;
00019     inline constexpr int PTHasSpecificationsColumn = 7;
00020     inline constexpr int PTIsSelectedColumn = 8;
00021
00022     inline const QList<int> list_70 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16,
0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x41, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62,
0x63, 0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7,
0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6,
0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 };
00023     inline const QList<int> list_69 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16,
0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63,
0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8,
0xA9, 0xAA, 0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7,
0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 };
00024     inline const QList<int> list_67 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16,
0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63,
0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA,
0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9,
0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 };
00025     inline const QList<int> list_66a = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16,
0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63,
0x64, 0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA,
0xAB, 0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9,
0xDA, 0xDB, 0xDC, 0xDD, 0xE0 };
00026     inline const QList<int> list_66b = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x20,
0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x40, 0x42, 0x50, 0x51, 0x60, 0x61, 0x62, 0x63, 0x64,

```

```

0x65, 0x66, 0x70, 0x71, 0x80, 0x90, 0x91, 0xA0, 0xA1, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB,
0xAC, 0xB0, 0xC0, 0xC1, 0xC2, 0xC3, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA,
0xDB, 0xDC, 0xDD, 0xE0, 0xE1 };
00027     inline const QList<int> list_42 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16,
0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9,
0xAA, 0xAB, 0xAC, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD, 0xE0, 0xE1 };
00028     inline const QList<int> list_40 = { 0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16,
0x20, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9,
0xAA, 0xAB, 0xAC, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD };
00029     inline const QList<int> list_39 = { 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31,
0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA,
0xAB, 0xAC, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD };
00030     inline const QList<int> list_32 = { 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x30, 0x31,
0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x70, 0x71, 0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC,
0xD0, 0xD2, 0xD3, 0xD4, 0xD5 };
00031     inline const QList<int> list_24 = { 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x20, 0x70, 0x71,
0xA0, 0xA1, 0xA4, 0xA5, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xD0, 0xD2, 0xD3, 0xD4, 0xD5 };
00032     inline const QList<int> list_10 = { 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xDA, 0xDB, 0xDC, 0xDD };
00033     inline const QList<int> list_6 = { 0x20, 0xD0, 0xD2, 0xD3, 0xD4, 0xD5 };
00034     inline const QList<int> list_2 = { 0xA2, 0xA3 };
00035
00036     enum class Vendor : uint8_t {
00037         AlephAlpha = 0x0,
00038         ai21 = 0x1,
00039         anthropic = 0x2,
00040         cohere = 0x3,
00041         eleutherai = 0x4,
00042         lmsys = 0x5,
00043         meta = 0x6,
00044         microsoft = 0x7,
00045         mistralai = 0x8,
00046         mosaicml = 0x9,
00047         openai = 0xA,
00048         stanford = 0xB,
00049         tiuae = 0xC,
00050         together = 0xD,
00051         writer_palmyra = 0xE,
00052     };
00053
00054 } // namespace HPB

```

## 7.12 src/languagemodel.cpp File Reference

```
#include "languagemodel.hpp"
```

## 7.13 src/languagemodel.hpp File Reference

```
#include <QString>
#include "hpb_globals.hpp"
```

### Classes

- class [LanguageModel](#)

## 7.14 languagemodel.hpp

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <QString>
00004

```

```

00005 #include "hpb_globals.hpp"
00006
00007 class LanguageModel {
00008 public:
00009     LanguageModel(int id, QString name, double parameters);
00010     const QString& name() const;
00011     double parameters() const;
00012     HPB::Vendor vendor() const;
00013     int id() const;
00014
00015 private:
00016     int m_Id;
00017     QString m_Name;
00018     double m_Parameters;
00019 };

```

## 7.15 src/parser/booleanparser.cpp File Reference

```

#include "booleanparser.hpp"
#include <QChar>
#include <QStack>

```

## 7.16 src/parser/booleanparser.hpp File Reference

```

#include <QList>
#include <QMap>
#include <QPair>
#include <QString>
#include "expression.hpp"

```

### Classes

- class [BooleanParser](#)

### Enumerations

- enum class [TokenType](#) : uint8\_t {  
[START\\_SYMBOL](#) , [END\\_SYMBOL](#) , [LPAREN](#) , [RPAREN](#) ,  
[AND](#) , [OR](#) , [NOT](#) , [IDENTIFIER](#) ,  
[ILLEGAL](#) }

### 7.16.1 Enumeration Type Documentation

#### 7.16.1.1 TokenType

```
enum class TokenType : uint8_t [strong]
```

#### Enumerator

<a href="#">START_SYMBOL</a>	
<a href="#">END_SYMBOL</a>	

## Enumerator

LPAREN	
RPAREN	
AND	
OR	
NOT	
IDENTIFIER	
ILLEGAL	

## 7.17 booleanparser.hpp

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <QList>
00004 #include <QMap>
00005 #include <QPair>
00006 #include <QString>
00007
00008 #include "expression.hpp"
00009
00010 enum class TokenType : uint8_t { START_SYMBOL, END_SYMBOL, LPAREN, RPAREN, AND, OR, NOT, IDENTIFIER,
    ILLEGAL };
00011
00012 class BooleanParser {
00013 public:
00014     BooleanParser();
00015
00016     bool parse(const QString& formula, Expression& expr);
00017     bool check(const QString& formula);
00018
00019 private:
00020     void advance();
00021     bool match(TokenType type);
00022     bool sentence(Expression& expr);
00023     bool disjunction(Expression& expr);
00024     bool conjunction(Expression& expr);
00025     bool negation(Expression& expr);
00026
00027     void tokenize(const QString& formula);
00028
00029     int m_Index;
00030     TokenType m_Sym;
00031     QList<QPair<QString, TokenType>> m_TokenList;
00032
00033     inline static const QMap<TokenType, QString> TokenTypeName = {
00034         { TokenType::START_SYMBOL, QString("<S>") },
00035         { TokenType::END_SYMBOL, QString("<E>") },
00036         { TokenType::LPAREN, QString("lparen") },
00037         { TokenType::RPAREN, QString("rparen") },
00038         { TokenType::NOT, QString("not") },
00039         { TokenType::AND, QString("and") },
00040         { TokenType::OR, QString("or") },
00041         { TokenType::ILLEGAL, QString("illegal") },
00042     };
00043 };

```

## 7.18 src/parser/expression.cpp File Reference

```
#include "expression.hpp"
```

## 7.19 src/parser/expression.hpp File Reference

```
#include <memory>
#include <QList>
#include <QString>
```

### Classes

- class [Expression](#)

### Enumerations

- enum class [Operator](#) : uint8\_t { [NOT](#) , [AND](#) , [OR](#) , [NIL](#) }

### 7.19.1 Enumeration Type Documentation

#### 7.19.1.1 Operator

```
enum class Operator : uint8_t [strong]
```

##### Enumerator

NOT	
AND	
OR	
NIL	

## 7.20 expression.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <memory>
00004
00005 #include <QList>
00006 #include <QString>
00007
00008 enum class Operator : uint8_t { NOT, AND, OR, NIL };
00009
00010 struct Conjunction;
00011 struct Disjunction;
00012 struct Negation;
00013
00014 class Expression
00015 {
00016 public:
00017     Expression() = default;
00018     Expression(Operator op, QString literal = "", const QList<Expression>& = {});
00019     void setOperator(Operator op);
00020     void addOperand(const Expression& expr);
00021     void addOperands(const QList<Expression>& expressions);
00022     const Expression& lhs() const;
00023     const Expression& rhs() const;
00024     const Expression& scope() const;
00025     const QString& literal() const;
00026     Operator op() const;
00027     void clear();
00028
00029 private:
00030     Operator m_Operator = Operator::NIL;
00031     QString m_Literal = "";
00032     QList<std::shared_ptr<Expression>> m_Children = {};
00033 };
```

## 7.21 src/parser/logic.cpp File Reference

```
#include "logic.hpp"
```

### Functions

- bool [isAtomic](#) (const [Expression](#) &expr)
- bool [isConjunction](#) (const [Expression](#) &expr)
- bool [isDisjunction](#) (const [Expression](#) &expr)
- bool [isNegation](#) (const [Expression](#) &expr)
- bool [isDNF](#) (const [Expression](#) &expr)
- bool [isNNF](#) (const [Expression](#) &expr)
- [Expression NNFtoDNF](#) (const [Expression](#) &expr)
- [Expression toDNF](#) (const [Expression](#) &expr)
- [Expression toNNF](#) (const [Expression](#) &expr)

### 7.21.1 Function Documentation

#### 7.21.1.1 isAtomic()

```
bool isAtomic (  
    const Expression & expr)
```

#### 7.21.1.2 isConjunction()

```
bool isConjunction (  
    const Expression & expr)
```

#### 7.21.1.3 isDisjunction()

```
bool isDisjunction (  
    const Expression & expr)
```

#### 7.21.1.4 isDNF()

```
bool isDNF (  
    const Expression & expr)
```

#### 7.21.1.5 isNegation()

```
bool isNegation (  
    const Expression & expr)
```



### 7.21.1.6 isNNF()

```
bool isNNF (  
    const Expression & expr)
```

### 7.21.1.7 NNFToDNF()

```
Expression NNFToDNF (  
    const Expression & expr)
```

### 7.21.1.8 toDNF()

```
Expression toDNF (  
    const Expression & expr)
```

### 7.21.1.9 toNNF()

```
Expression toNNF (  
    const Expression & expr)
```

## 7.22 src/parser/logic.hpp File Reference

```
#include "expression.hpp"
```

### Functions

- bool [isNegation](#) (const [Expression](#) &expr)
- bool [isDisjunction](#) (const [Expression](#) &expr)
- bool [isConjunction](#) (const [Expression](#) &expr)
- bool [isAtomic](#) (const [Expression](#) &expr)
- bool [isNNF](#) (const [Expression](#) &expr)
- bool [isDNF](#) (const [Expression](#) &expr)
- [Expression toDNF](#) (const [Expression](#) &expr)
- [Expression toNNF](#) (const [Expression](#) &expr)
- [Expression NNFToDNF](#) (const [Expression](#) &expr)

### 7.22.1 Function Documentation

#### 7.22.1.1 isAtomic()

```
bool isAtomic (  
    const Expression & expr)
```

#### 7.22.1.2 isConjunction()

```
bool isConjunction (
    const Expression & expr)
```

#### 7.22.1.3 isDisjunction()

```
bool isDisjunction (
    const Expression & expr)
```

#### 7.22.1.4 isDNF()

```
bool isDNF (
    const Expression & expr)
```

#### 7.22.1.5 isNegation()

```
bool isNegation (
    const Expression & expr)
```

#### 7.22.1.6 isNNF()

```
bool isNNF (
    const Expression & expr)
```

#### 7.22.1.7 NNFtoDNF()

```
Expression NNFtoDNF (
    const Expression & expr)
```

#### 7.22.1.8 toDNF()

```
Expression toDNF (
    const Expression & expr)
```

#### 7.22.1.9 toNNF()

```
Expression toNNF (
    const Expression & expr)
```

## 7.23 logic.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "expression.hpp"
00004
00005 bool isNegation(const Expression& expr);
00006 bool isDisjunction(const Expression& expr);
00007 bool isConjunction(const Expression& expr);
00008 bool isAtomic(const Expression& expr);
00009
00010 bool isNNF(const Expression& expr);
00011 bool isDNF(const Expression& expr);
00012
00013 Expression toDNF(const Expression& expr);
00014 Expression toNNF(const Expression& expr);
00015
00016 Expression NNFtoDNF(const Expression& expr);
```

## 7.24 src/parser/queryparser.cpp File Reference

```
#include "queryparser.hpp"
#include <algorithm>
#include "booleanparser.hpp"
#include "expression.hpp"
#include "logic.hpp"
```

### Functions

- bool [checkQuery](#) (const QString &query)
- QList< QPair< QStringList, QStringList > > [getQueries](#) (const QString &queryStr)

### 7.24.1 Function Documentation

#### 7.24.1.1 checkQuery()

```
bool checkQuery (
    const QString & query)
```

#### 7.24.1.2 getQueries()

```
QList< QPair< QStringList, QStringList > > getQueries (
    const QString & queryStr)
```

## 7.25 src/parser/queryparser.hpp File Reference

```
#include <QList>
#include <QPair>
#include <QString>
#include <QStringList>
```

## Functions

- bool [checkQuery](#) (const QString &queryStr)
- QList< QPair< QStringList, QStringList > > [getQueries](#) (const QString &queryStr)

## 7.25.1 Function Documentation

### 7.25.1.1 checkQuery()

```
bool checkQuery (  
    const QString & queryStr)
```

### 7.25.1.2 getQueries()

```
QList< QPair< QStringList, QStringList > > getQueries (  
    const QString & queryStr)
```

## 7.26 queryparser.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once  
00002  
00003 #include <QList>  
00004 #include <QPair>  
00005 #include <QString>  
00006 #include <QStringList>  
00007  
00008 bool checkQuery(const QString& queryStr);  
00009 QList<QPair<QStringList, QStringList> getQueries(const QString& queryStr);
```

# Index

- [\\_range](#)
    - [helperfunctions.hpp, 40](#)
  - [~ExportOptionsDialog](#)
    - [ExportOptionsDialog, 16](#)
  - [~VendorDialog](#)
    - [VendorDialog, 19](#)
- [addOperand](#)
  - [Expression, 17](#)
- [addOperands](#)
  - [Expression, 17](#)
- [addPromptsToTree](#)
  - [helperfunctions.cpp, 24](#)
  - [helperfunctions.hpp, 33](#)
- [ai21](#)
  - [HPB, 10](#)
- [AlephAlpha](#)
  - [HPB, 10](#)
- [AND](#)
  - [booleanparser.hpp, 46](#)
  - [expression.hpp, 47](#)
- [anthropic](#)
  - [HPB, 10](#)
- [Ask](#)
  - [helperfunctions.cpp, 25](#)
  - [helperfunctions.hpp, 34](#)
- [BooleanParser, 15](#)
  - [BooleanParser, 15](#)
  - [check, 15](#)
  - [parse, 15](#)
- [booleanparser.hpp](#)
  - [AND, 46](#)
  - [END\\_SYMBOL, 45](#)
  - [IDENTIFIER, 46](#)
  - [ILLEGAL, 46](#)
  - [LPAREN, 46](#)
  - [NOT, 46](#)
  - [OR, 46](#)
  - [RPAREN, 46](#)
  - [START\\_SYMBOL, 45](#)
  - [TokenType, 45](#)
- [check](#)
  - [BooleanParser, 15](#)
- [checkQuery](#)
  - [queryparser.cpp, 51](#)
  - [queryparser.hpp, 52](#)
- [clear](#)
  - [Expression, 17](#)
- [cohere](#)
  - [HPB, 10](#)
- [deleteDatasetFromTree](#)
  - [helperfunctions.cpp, 25](#)
  - [helperfunctions.hpp, 34](#)
- [DTColumnCount](#)
  - [HPB, 11](#)
- [DTDatasetNameColumn](#)
  - [HPB, 11](#)
- [DTLMListColumn](#)
  - [HPB, 11](#)
- [DTNumberOfModels](#)
  - [HPB, 11](#)
- [eleutherai](#)
  - [HPB, 10](#)
- [END\\_SYMBOL](#)
  - [booleanparser.hpp, 45](#)
- [ExportOptionsDialog, 16](#)
  - [~ExportOptionsDialog, 16](#)
  - [ExportOptionsDialog, 16](#)
- [Expression, 16](#)
  - [addOperand, 17](#)
  - [addOperands, 17](#)
  - [clear, 17](#)
  - [Expression, 17](#)
  - [lhs, 17](#)
  - [literal, 17](#)
  - [op, 18](#)
  - [rhs, 18](#)
  - [scope, 18](#)
  - [setOperator, 18](#)
- [expression.hpp](#)
  - [AND, 47](#)
  - [NIL, 47](#)
  - [NOT, 47](#)
  - [Operator, 47](#)
  - [OR, 47](#)
- [generateCustomDataset](#)
  - [helperfunctions.cpp, 25](#)
  - [helperfunctions.hpp, 34](#)
- [getCID](#)
  - [helperfunctions.cpp, 26](#)
  - [helperfunctions.hpp, 35](#)
- [getDatasetBase](#)
  - [helperfunctions.cpp, 26](#)
  - [helperfunctions.hpp, 35](#)
- [getDatasetSpec](#)

- helperfunctions.cpp, 26
  - helperfunctions.hpp, 35
- getFiltersFromDatasetList
  - helperfunctions.cpp, 26
  - helperfunctions.hpp, 35
- getHelmTaskDirs
  - helperfunctions.cpp, 26
  - helperfunctions.hpp, 35
- getModelList
  - helperfunctions.hpp, 36
- getName
  - helperfunctions.cpp, 27
  - helperfunctions.hpp, 36
- getPID
  - helperfunctions.cpp, 27
  - helperfunctions.hpp, 36
- getPrompt
  - helperfunctions.cpp, 27
  - helperfunctions.hpp, 36
- getPromptText
  - helperfunctions.cpp, 27
- getQueries
  - queryparser.cpp, 51
  - queryparser.hpp, 52
- getReferences
  - helperfunctions.cpp, 27
  - helperfunctions.hpp, 36
- getReferencesText
  - helperfunctions.cpp, 27
- getSamples
  - helperfunctions.cpp, 28
  - helperfunctions.hpp, 36
- getSelectedDatasetNames
  - helperfunctions.cpp, 28
  - helperfunctions.hpp, 36
- getTaskInstances
  - helperfunctions.cpp, 28
  - helperfunctions.hpp, 37
- hasSelectedPrompts
  - helperfunctions.cpp, 29
  - helperfunctions.hpp, 37
- hasSpecifications
  - helperfunctions.cpp, 29
  - helperfunctions.hpp, 37
- helperfunctions.cpp
  - addPromptsToTree, 24
  - Ask, 25
  - deleteDatasetFromTree, 25
  - generateCustomDataset, 25
  - getCID, 26
  - getDatasetBase, 26
  - getDatasetSpec, 26
  - getFiltersFromDatasetList, 26
  - getHelmTaskDirs, 26
  - getName, 27
  - getPID, 27
  - getPrompt, 27
  - getPromptText, 27
  - getReferences, 27
  - getReferencesText, 27
  - getSamples, 28
  - getSelectedDatasetNames, 28
  - getTaskInstances, 28
  - hasSelectedPrompts, 29
  - hasSpecifications, 29
  - isPrompt, 29
  - isSelected, 29
  - loadHelmDataConfig, 29
  - matches, 30
  - PopUp, 30
  - setCID, 31
  - setSelectedStatus, 31
  - splitDatasetName, 31
  - transformDatasetTree, 31
  - transformPromptTree, 31
  - Warn, 32
- helperfunctions.hpp
  - \_range, 40
  - addPromptsToTree, 33
  - Ask, 34
  - deleteDatasetFromTree, 34
  - generateCustomDataset, 34
  - getCID, 35
  - getDatasetBase, 35
  - getDatasetSpec, 35
  - getFiltersFromDatasetList, 35
  - getHelmTaskDirs, 35
  - getModelList, 36
  - getName, 36
  - getPID, 36
  - getPrompt, 36
  - getReferences, 36
  - getSamples, 36
  - getSelectedDatasetNames, 36
  - getTaskInstances, 37
  - hasSelectedPrompts, 37
  - hasSpecifications, 37
  - isPrompt, 37
  - isSelected, 38
  - loadHelmDataConfig, 38
  - matches, 38
  - PopUp, 38
  - prettyPrint, 39
  - setCID, 39
  - setSelectedStatus, 39
  - splitDatasetName, 39
  - transformDatasetTree, 39
  - transformPromptTree, 40
  - Warn, 40
- HPB, 9
  - ai21, 10
  - AlephAlpha, 10
  - anthropic, 10
  - cohere, 10
  - DTColumnCount, 11
  - DTDatasetNameColumn, 11

- DTLMListColumn, [11](#)
- DTNumberOfModels, [11](#)
- eleutherai, [10](#)
- list\_10, [11](#)
- list\_2, [11](#)
- list\_24, [11](#)
- list\_32, [11](#)
- list\_39, [11](#)
- list\_40, [12](#)
- list\_42, [12](#)
- list\_6, [12](#)
- list\_66a, [12](#)
- list\_66b, [12](#)
- list\_67, [12](#)
- list\_69, [12](#)
- list\_70, [13](#)
- lmsys, [10](#)
- meta, [10](#)
- microsoft, [10](#)
- mistralai, [10](#)
- mosaicml, [10](#)
- openai, [10](#)
- PTCIDColumn, [13](#)
- PTColumnCount, [13](#)
- PTDatasetBaseColumn, [13](#)
- PTDatasetSpecColumn, [13](#)
- PTHasSpecificationsColumn, [13](#)
- PTIsPromptColumn, [13](#)
- PTIsSelectedColumn, [13](#)
- PTNameIDColumn, [13](#)
- PTPromptContentsColumn, [14](#)
- PTReferencesColumn, [14](#)
- stanford, [10](#)
- tiuae, [10](#)
- together, [11](#)
- Vendor, [10](#)
- writer\_palmyra, [11](#)
- id
  - LanguageModel, [19](#)
- IDENTIFIER
  - booleanparser.hpp, [46](#)
- ILLEGAL
  - booleanparser.hpp, [46](#)
- isAtomic
  - logic.cpp, [48](#)
  - logic.hpp, [49](#)
- isConjunction
  - logic.cpp, [48](#)
  - logic.hpp, [49](#)
- isDisjunction
  - logic.cpp, [48](#)
  - logic.hpp, [50](#)
- isDNF
  - logic.cpp, [48](#)
  - logic.hpp, [50](#)
- isNegation
  - logic.cpp, [48](#)
  - logic.hpp, [50](#)
- isNNF
  - logic.cpp, [48](#)
  - logic.hpp, [50](#)
- isPrompt
  - helperfunctions.cpp, [29](#)
  - helperfunctions.hpp, [37](#)
- isSelected
  - helperfunctions.cpp, [29](#)
  - helperfunctions.hpp, [38](#)
- LanguageModel, [18](#)
  - id, [19](#)
  - LanguageModel, [18](#)
  - name, [19](#)
  - parameters, [19](#)
  - vendor, [19](#)
- lhs
  - Expression, [17](#)
- list\_10
  - HPB, [11](#)
- list\_2
  - HPB, [11](#)
- list\_24
  - HPB, [11](#)
- list\_32
  - HPB, [11](#)
- list\_39
  - HPB, [11](#)
- list\_40
  - HPB, [12](#)
- list\_42
  - HPB, [12](#)
- list\_6
  - HPB, [12](#)
- list\_66a
  - HPB, [12](#)
- list\_66b
  - HPB, [12](#)
- list\_67
  - HPB, [12](#)
- list\_69
  - HPB, [12](#)
- list\_70
  - HPB, [13](#)
- literal
  - Expression, [17](#)
- lmsys
  - HPB, [10](#)
- loadHelmDataConfig
  - helperfunctions.cpp, [29](#)
  - helperfunctions.hpp, [38](#)
- logic.cpp
  - isAtomic, [48](#)
  - isConjunction, [48](#)
  - isDisjunction, [48](#)
  - isDNF, [48](#)
  - isNegation, [48](#)
  - isNNF, [48](#)
  - NNFtoDNF, [49](#)

- toDNF, [49](#)
  - toNNF, [49](#)
- logic.hpp
  - isAtomic, [49](#)
  - isConjunction, [49](#)
  - isDisjunction, [50](#)
  - isDNF, [50](#)
  - isNegation, [50](#)
  - isNNF, [50](#)
  - NNFtoDNF, [50](#)
  - toDNF, [50](#)
  - toNNF, [50](#)
- LPAREN
  - booleanparser.hpp, [46](#)
- matches
  - helperfunctions.cpp, [30](#)
  - helperfunctions.hpp, [38](#)
- meta
  - HPB, [10](#)
- microsoft
  - HPB, [10](#)
- mistralai
  - HPB, [10](#)
- mosaicml
  - HPB, [10](#)
- name
  - LanguageModel, [19](#)
- NIL
  - expression.hpp, [47](#)
- NNFtoDNF
  - logic.cpp, [49](#)
  - logic.hpp, [50](#)
- NOT
  - booleanparser.hpp, [46](#)
  - expression.hpp, [47](#)
- op
  - Expression, [18](#)
- openai
  - HPB, [10](#)
- Operator
  - expression.hpp, [47](#)
- OR
  - booleanparser.hpp, [46](#)
  - expression.hpp, [47](#)
- parameters
  - LanguageModel, [19](#)
- parse
  - BooleanParser, [15](#)
- PopUp
  - helperfunctions.cpp, [30](#)
  - helperfunctions.hpp, [38](#)
- prettyPrint
  - helperfunctions.hpp, [39](#)
- PTCIDColumn
  - HPB, [13](#)
- PTColumnCount
  - HPB, [13](#)
- PTDatasetBaseColumn
  - HPB, [13](#)
- PTDatasetSpecColumn
  - HPB, [13](#)
- PTHasSpecificationsColumn
  - HPB, [13](#)
- PTIsPromptColumn
  - HPB, [13](#)
- PTIsSelectedColumn
  - HPB, [13](#)
- PTNameIDColumn
  - HPB, [13](#)
- PTPromptContentsColumn
  - HPB, [14](#)
- PTReferencesColumn
  - HPB, [14](#)
- queryparser.cpp
  - checkQuery, [51](#)
  - getQueries, [51](#)
- queryparser.hpp
  - checkQuery, [52](#)
  - getQueries, [52](#)
- rhs
  - Expression, [18](#)
- RPAREN
  - booleanparser.hpp, [46](#)
- scope
  - Expression, [18](#)
- setCID
  - helperfunctions.cpp, [31](#)
  - helperfunctions.hpp, [39](#)
- setOperator
  - Expression, [18](#)
- setSelectedStatus
  - helperfunctions.cpp, [31](#)
  - helperfunctions.hpp, [39](#)
- splitDatasetName
  - helperfunctions.cpp, [31](#)
  - helperfunctions.hpp, [39](#)
- src/dialogs/exportoptionsdialog.cpp, [21](#)
- src/dialogs/exportoptionsdialog.hpp, [21](#)
- src/dialogs/vendordialog.cpp, [22](#)
- src/dialogs/vendordialog.hpp, [22](#)
- src/helperfunctions.cpp, [23](#)
- src/helperfunctions.hpp, [32, 41](#)
- src/hpb\_globals.hpp, [42, 43](#)
- src/languagemodel.cpp, [44](#)
- src/languagemodel.hpp, [44](#)
- src/parser/booleanparser.cpp, [45](#)
- src/parser/booleanparser.hpp, [45, 46](#)
- src/parser/expression.cpp, [46](#)
- src/parser/expression.hpp, [47](#)
- src/parser/logic.cpp, [48](#)
- src/parser/logic.hpp, [49, 51](#)



- src/parser/queryparser.cpp, [51](#)
- src/parser/queryparser.hpp, [51](#), [52](#)
- stanford
  - HPB, [10](#)
- START\_SYMBOL
  - booleanparser.hpp, [45](#)
- tiiuae
  - HPB, [10](#)
- toDNF
  - logic.cpp, [49](#)
  - logic.hpp, [50](#)
- together
  - HPB, [11](#)
- TokenType
  - booleanparser.hpp, [45](#)
- toNNF
  - logic.cpp, [49](#)
  - logic.hpp, [50](#)
- transformDatasetTree
  - helperfunctions.cpp, [31](#)
  - helperfunctions.hpp, [39](#)
- transformPromptTree
  - helperfunctions.cpp, [31](#)
  - helperfunctions.hpp, [40](#)
- Ui, [14](#)
- Vendor
  - HPB, [10](#)
- vendor
  - LanguageModel, [19](#)
- VendorDialog, [19](#)
  - ~VendorDialog, [19](#)
  - VendorDialog, [19](#)
- Warn
  - helperfunctions.cpp, [32](#)
  - helperfunctions.hpp, [40](#)
- writer\_palmyra
  - HPB, [11](#)