

MindEngine

Generated by Doxygen 1.8.11

Contents

1	Todo List	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	Namespace Documentation	9
5.1	RuleBased Namespace Reference	9
5.1.1	Detailed Description	10
5.1.2	Typedef Documentation	10
5.1.2.1	IdType	10
6	Class Documentation	11
6.1	RuleBased::DataGroup Class Reference	11
6.1.1	Detailed Description	11
6.1.2	Constructor & Destructor Documentation	12
6.1.2.1	DataGroup(const std::string &identifier, DataNode *rightSibling, DataNode *left↔MostChild)	12
6.1.3	Member Function Documentation	12
6.1.3.1	getLeftMostChild() const	12
6.1.3.2	isGroup()	12

6.2	RuleBased::DataGroupMatch Struct Reference	12
6.2.1	Detailed Description	13
6.2.2	Member Function Documentation	13
6.2.2.1	matchesNode(const DataNode *node, void *bindings)	13
6.3	RuleBased::DataNode Class Reference	14
6.3.1	Detailed Description	14
6.3.2	Constructor & Destructor Documentation	14
6.3.2.1	DataNode(const IdType &identifier, DataNode *rightSibling)	14
6.3.3	Member Function Documentation	15
6.3.3.1	getIdentifier() const	15
6.3.3.2	getRightSibling() const	15
6.3.3.3	isDatum() const	15
6.3.3.4	isGroup() const	15
6.4	RuleBased::DataNodeMatch Struct Reference	16
6.4.1	Detailed Description	16
6.4.2	Member Function Documentation	16
6.4.2.1	matches(const DataNode *database, void *bindings)	16
6.4.2.2	matchesChildren(const DataGroup *group, void *bindings)	17
6.4.2.3	matchesNode(const DataNode *node, void *bindings)=0	17
6.5	RuleBased::Datum< T > Class Template Reference	18
6.5.1	Detailed Description	18
6.5.2	Constructor & Destructor Documentation	18
6.5.2.1	Datum(T value)	18
6.5.2.2	Datum(const std::string &identifier, DataNode *rightSibling, T value)	19
6.5.3	Member Function Documentation	19
6.5.3.1	getValue() const	19
6.5.3.2	isDatum()	19
6.5.3.3	setValue(T newValue)	19
6.6	RuleBased::IdCheck Class Reference	20
6.6.1	Detailed Description	20

6.6.2	Member Function Documentation	20
6.6.2.1	isWildcard(IdType identifier)	20
6.7	RuleBased::Match Struct Reference	20
6.7.1	Detailed Description	21
6.7.2	Member Function Documentation	21
6.7.2.1	matches(const DataNode *database, void *bindings)=0	21
6.8	RuleBased::NumberDatumMatch< T > Struct Template Reference	21
6.8.1	Detailed Description	22
6.8.2	Constructor & Destructor Documentation	22
6.8.2.1	NumberDatumMatch(IdType identifier, T min, T max)	22
6.8.3	Member Function Documentation	22
6.8.3.1	matchesNode(const DataNode *node, void *bindings)	22
6.9	RuleBased::Rule Class Reference	23
6.9.1	Detailed Description	23
6.9.2	Member Function Documentation	23
6.9.2.1	action()=0	23
Index		25

Chapter 1

Todo List

Member `RuleBased::DataGroupMatch::matchesNode` (`const DataNode *node, void *bindings`)

Need adding bindings mechanism here

Member `RuleBased::IdType`

Use identifier with strings may decrease performance with large systems due to string-matching operations. Will change this to integer or unsigned integer as soon as I find a way to match identifier with human-readable strings.

Member `RuleBased::NumberDatumMatch< T >::matchesNode` (`const DataNode *node, void *bindings`)

Need adding bindings mechanism here

Member `RuleBased::Rule::action` ()=0

Examine the performance and reusability of using a method. If it is hard to expand, find a way to use a struct/class instead.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

RuleBased

Contains classes to represent Rule-based system's database, which stores knowledge available to the AI agent as well as the implementation of rules and the mechanism to match the rules and the data in the database

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

RuleBased::DataGroupMatch	12
RuleBased::DataNode	14
RuleBased::DataGroup	11
RuleBased::Datum< T >	18
RuleBased::IdCheck	20
RuleBased::Match	20
RuleBased::DataNodeMatch	16
RuleBased::NumberDatumMatch< T >	21
RuleBased::Rule	23

Chapter 4

Class Index

4.1 Class List

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

RuleBased::DataGroup	Represents a non-leaf node, which contains children. Its children can be any DataNode object: either another DataGroup , or a Datum only	11
RuleBased::DataGroupMatch	Matches a group of data in the database. This is done by building a match data structure that mirrors the data structure that is being searched for in the database	12
RuleBased::DataNode	Base class of each node in the database tree. Since every node needs an identifier, but non-leaf nodes contain their children, while leaf nodes store values	14
RuleBased::DataNodeMatch	A struct derived from Match , it is responsible for matching a single DataNode in the database .	16
RuleBased::Datum< T >	A Datum consists of an identifier and and value. In Database's tree structure, a leaf node is a Datum	18
RuleBased::IdCheck	Provides methods to check the IDs	20
RuleBased::Match	Provides the mechanism to match the data item from the rule with any item inside the database	20
RuleBased::NumberDatumMatch< T >	A struct provides mechanism for matching two Datum whose values is number values (int, float,...)	21
RuleBased::Rule	Represent a rule in a Rule-based system. A rule has two components: an if clause is going to be used to match against the database and a function to perform any action required	23

Chapter 5

Namespace Documentation

5.1 RuleBased Namespace Reference

Contains classes to represent Rule-based system's database, which stores knowledge available to the AI agent as well as the implementation of rules and the mechanism to match the rules and the data in the database.

Classes

- class [DataGroup](#)
Represents a non-leaf node, which contains children. Its children can be any [DataNode](#) object: either another [DataGroup](#), or a [Datum](#) only.
- struct [DataGroupMatch](#)
Matches a group of data in the database. This is done by building a match data structure that mirrors the data structure that is being searched for in the database.
- class [DataNode](#)
Base class of each node in the database tree. Since every node needs an identifier, but non-leaf nodes contain their children, while leaf nodes store values.
- struct [DataNodeMatch](#)
A struct derived from [Match](#), it is responsible for matching a single [DataNode](#) in the database.
- class [Datum](#)
A [Datum](#) consists of an identifier and a value. In Database's tree structure, a leaf node is a [Datum](#).
- class [IdCheck](#)
Provides methods to check the IDs.
- struct [Match](#)
Provides the mechanism to match the data item from the rule with any item inside the database.
- struct [NumberDatumMatch](#)
A struct provides mechanism for matching two [Datum](#) whose values are number values (int, float,...).
- class [Rule](#)
Represent a rule in a Rule-based system. A rule has two components: an if clause is going to be used to match against the database and a function to perform any action required.

Typedefs

- typedef std::string [IdType](#)
Currently set the type of the ID of data nodes as string.

5.1.1 Detailed Description

Contains classes to represent Rule-based system's database, which stores knowledge available to the AI agent as well as the implementation of rules and the mechanism to match the rules and the data in the database.

5.1.2 Typedef Documentation

5.1.2.1 `typedef std::string RuleBased::IdType`

Currently set the type of the ID of data nodes as string.

Todo Use identifier with strings may decrease performance with large systems due to string-matching operations. Will change this to integer or unsigned integer as soon as I find a way to match identifier with human-readable strings.

Chapter 6

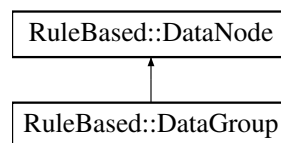
Class Documentation

6.1 RuleBased::DataGroup Class Reference

Represents a non-leaf node, which contains children. Its children can be any [DataNode](#) object: either another [DataGroup](#), or a [Datum](#) only.

```
#include <DataGroup.h>
```

Inheritance diagram for RuleBased::DataGroup:



Public Member Functions

- [DataGroup](#) ()
DataGroup default constructor.
- [DataGroup](#) (const std::string &identifier, [DataNode](#) *rightSibling, [DataNode](#) *leftMostChild)
DataGroup constructor.
- virtual [~DataGroup](#) ()
~DataGroup destructor.
- [DataNode](#) * [getLeftMostChild](#) () const
Data nodes are put into a left-most child, right sibling tree. This function returns the pointer to the left most child of this node.
- bool [isGroup](#) ()
Allows user to check whether this node is a DataGroup or not.

Additional Inherited Members

6.1.1 Detailed Description

Represents a non-leaf node, which contains children. Its children can be any [DataNode](#) object: either another [DataGroup](#), or a [Datum](#) only.

See also

[DataNode](#)
[Datum](#)

6.1.2 Constructor & Destructor Documentation

6.1.2.1 **RuleBased::DataGroup::DataGroup** (const std::string & *identifier*, *DataNode* * *rightSibling*, *DataNode* * *leftMostChild*)

[DataGroup](#) constructor.

Parameters

<i>identifier</i>	a string parameter.
<i>rightSibling</i>	a DataNode pointer representing this node's right sibling in tree.
<i>leftMostChild</i>	a DataNode pointer representing this node's left most child in tree.

6.1.3 Member Function Documentation

6.1.3.1 **DataNode** * **RuleBased::DataGroup::getLeftMostChild** () const

Data nodes are put into a left-most child, right sibling tree. This function returns the pointer to the left most child of this node.

Returns

The left most child of this data group node.

6.1.3.2 **bool** **RuleBased::DataGroup::isGroup** ()

Allows user to check whether this node is a [DataGroup](#) or not.

Returns

true if this node is a [DataGroup](#), otherwise returns false.

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

- Engine/Database/DataGroup.h
- Engine/Database/DataGroup.cpp

6.2 RuleBased::DataGroupMatch Struct Reference

Matches a group of data in the database. This is done by building a match data structure that mirrors the data structure that is being searched for in the database.

```
#include <DataGroupMatch.h>
```

Public Member Functions

- virtual bool [matchesNode](#) (const [DataNode](#) *node, void *bindings)

Tries to match the given data node from the database against the criteria. Method used: a recursive matching algorithm that travels down the given node, trying to match it against the structure of this match and its children.

Public Attributes

- [IdType](#) identifier
The identifier to match.
- [DataNodeMatch](#) * [leftMostChild](#)
The first sub-match in this group.

6.2.1 Detailed Description

Matches a group of data in the database. This is done by building a match data structure that mirrors the data structure that is being searched for in the database.

6.2.2 Member Function Documentation

6.2.2.1 bool RuleBased::DataGroupMatch::matchesNode (const [DataNode](#) * node, void * bindings) [virtual]

Tries to match the given data node from the database against the criteria. Method used: a recursive matching algorithm that travels down the given node, trying to match it against the structure of this match and its children.

Parameters

<i>node</i>	The database node to match on.
<i>bindings</i>	When part of the if clause matches a wild card, it is added to the bindings. This parameter is both input and output parameter.

Returns

true if matches, otherwise return false.

Todo Need adding bindings mechanism here

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

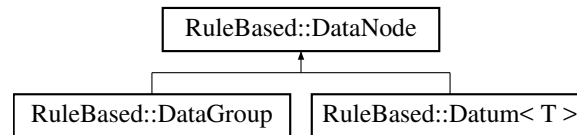
- Engine/Rules/DataGroupMatch.h
- Engine/Rules/DataGroupMatch.cpp

6.3 RuleBased::DataNode Class Reference

Base class of each node in the database tree. Since every node needs an identifier, but non-leaf nodes contain their children, while leaf nodes store values.

```
#include <DataNode.h>
```

Inheritance diagram for RuleBased::DataNode:



Public Member Functions

- [DataNode](#) ()
DataNode default constructor.
- [DataNode](#) (const [IdType](#) &identifier, [DataNode](#) *rightSibling)
DataNode constructor with parameters.
- virtual [~DataNode](#) ()
~DataNode destructor.
- [IdType](#) getIdentifier () const
The data nodes have unique identifiers.
- [DataNode](#) * getRightSibling () const
Data nodes are put into a left-most child, right sibling tree. This function is used to get the sibling next to current node.
- virtual bool isGroup () const
Allows user to check whether this node is a DataGroup or not.
- virtual bool isDatum () const
Allows user to check whether this node is a Datum or not.

Protected Attributes

- [IdType](#) identifier
Each item of data should be identified to know what the value means.
- [DataNode](#) * rightSibling
The right sibling node of this node, or NULL if this node is the right most child.

6.3.1 Detailed Description

Base class of each node in the database tree. Since every node needs an identifier, but non-leaf nodes contain their children, while leaf nodes store values.

See also

[DataGroup](#)
[Datum](#)

6.3.2 Constructor & Destructor Documentation

6.3.2.1 RuleBased::DataNode::DataNode (const [IdType](#) & identifier, [DataNode](#) * rightSibling)

[DataNode](#) constructor with parameters.

Parameters

<i>identifier</i>	an ID parameter.
<i>rightSibling</i>	a DataNode pointer representing this node's right sibling in tree. structure

6.3.3 Member Function Documentation

6.3.3.1 IdType RuleBased::DataNode::getIdentifier () const

The data nodes have unique identifiers.

Returns

The identifier of this node.

6.3.3.2 DataNode * RuleBased::DataNode::getRightSibling () const

Data nodes are put into a left-most child, right sibling tree. This function is used to get the sibling next to current node.

Returns

The right sibling node of this node, or NULL if this node is the right most child.

6.3.3.3 bool RuleBased::DataNode::isDatum () const [virtual]

Allows user to check whether this node is a [Datum](#) or not.

Returns

true if this node is a [Datum](#), otherwise returns false.

6.3.3.4 bool RuleBased::DataNode::isGroup () const [virtual]

Allows user to check whether this node is a [DataGroup](#) or not.

Returns

true if this node is a [DataGroup](#), otherwise returns false.

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

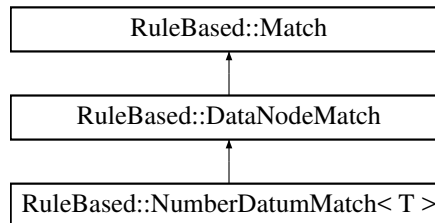
- Engine/Database/DataNode.h
- Engine/Database/DataNode.cpp

6.4 RuleBased::DataNodeMatch Struct Reference

A struct derived from [Match](#), it is responsible for matching a single [DataNode](#) in the database.

```
#include <DataNodeMatch.h>
```

Inheritance diagram for RuleBased::DataNodeMatch:



Public Member Functions

- virtual bool [matches](#) (const [DataNode](#) *database, void *bindings)
Matches the given database by checking each element in the database against the matchesNode method.
- bool [matchesChildren](#) (const [DataGroup](#) *group, void *bindings)
Matches all the children of the given group to see if any of them pass the matchesNode test. This is used in the implementation of the matches method to iterate through items in the database.
- virtual bool [matchesNode](#) (const [DataNode](#) *node, void *bindings)=0
Matches the data node from the database against the criteria in this match.

Public Attributes

- [DataNodeMatch](#) * [rightSibling](#)
The right sibling of this node in match tree.

6.4.1 Detailed Description

A struct derived from [Match](#), it is responsible for matching a single [DataNode](#) in the database.

Conceptually, the match class could match the whole database in a single operation: it is only ever fed the whole database, so it can do with that what it likes. However in practical, the vast majority of matching requirements involve trying to find a single item in the database. This struct's match method iterates through the items in the database, and calls matchesNode on each one. Matches items can be implemented in sub-classes to check if the item fulfils the match criteria.

Data node matches are arranged into tree, just like how data nodes are.

See also

[Match](#)

6.4.2 Member Function Documentation

6.4.2.1 bool RuleBased::DataNodeMatch::matches (const [DataNode](#) * database, void * bindings) [virtual]

Matches the given database by checking each element in the database against the matchesNode method.

Parameters

<i>database</i>	The database node to match on.
<i>bindings</i>	When part of the if clause matches a wild card, it is added to the bindings. This parameter is both input and output parameter.

Returns

true if matches, else returns false.

Implements [RuleBased::Match](#).

6.4.2.2 bool RuleBased::DataNodeMatch::matchesChildren (const DataGroup * *group*, void * *bindings*)

Matches all the children of the given group to see if any of them pass the matchesNode test. This is used in the implementation of the matches method to iterate through items in the database.

Parameters

<i>group</i>	The group to match on.
<i>bindings</i>	When part of the if clause matches a wild card, it is added to the bindings. This parameter is both input and output parameter.

Returns

true if matches, else return false.

6.4.2.3 virtual bool RuleBased::DataNodeMatch::matchesNode (const DataNode * *node*, void * *bindings*) [pure virtual]

Matches the data node from the database against the criteria in this match.

Parameters

<i>node</i>	The database node to match on.
<i>bindings</i>	When part of the if clause matches a wild card, it is added to the bindings. This parameter is both input and output parameter.

Returns

true if matches, else return false.

Implemented in [RuleBased::NumberDatumMatch< T >](#).

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

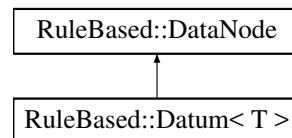
- Engine/Rules/DataNodeMatch.h
- Engine/Rules/DataNodeMatch.cpp

6.5 RuleBased::Datum< T > Class Template Reference

A [Datum](#) consists of an identifier and and value. In Database's tree structure, a leaf node is a [Datum](#).

```
#include <Datum.h>
```

Inheritance diagram for RuleBased::Datum< T >:



Public Member Functions

- [Datum](#) (T value)
Datum constructor.
- [Datum](#) (const std::string &identifier, [DataNode](#) *rightSibling, T value)
Datum constructor.
- virtual [~Datum](#) ()
~Datum destructor
- void [setValue](#) (T newValue)
To change the value of the Datum.
- T [getValue](#) () const
To get the value of the Datum.
- bool [isDatum](#) ()
Allows user to check whether this node is a Datum or not.

Additional Inherited Members

6.5.1 Detailed Description

```
template<class T>
class RuleBased::Datum< T >
```

A [Datum](#) consists of an identifier and and value. In Database's tree structure, a leaf node is a [Datum](#).

See also

[DataNode](#)
[DataGroup](#)

6.5.2 Constructor & Destructor Documentation

6.5.2.1 template<class T > RuleBased::Datum< T >::Datum (T value)

[Datum](#) constructor.

Parameters

<i>value</i>	the value that the Datum holds.
--------------	---

6.5.2.2 `template<class T> RuleBased::Datum< T >::Datum (const std::string & identifier, DataNode * rightSibling, T value)`

[Datum](#) constructor.

Parameters

<i>identifier</i>	a string parameter.
<i>rightSibling</i>	a DataNode pointer representing this node's right sibling in tree.
<i>value</i>	the value that the Datum holds.

6.5.3 Member Function Documentation

6.5.3.1 `template<class T> T RuleBased::Datum< T >::getValue () const`

To get the value of the [Datum](#).

Returns

The value that the datum is currently holding.

6.5.3.2 `template<class T> bool RuleBased::Datum< T >::isDatum ()`

Allows user to check whether this node is a [Datum](#) or not.

Returns

true if this node is a [Datum](#), otherwise returns false.

6.5.3.3 `template<class T> void RuleBased::Datum< T >::setValue (T newValue)`

To change the value of the [Datum](#).

Parameters

<i>newValue</i>	The new value that is going to be assigned to the Datum .
-----------------	---

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

- Engine/Database/Datum.h
- Engine/Database/Datum.cpp

6.6 RuleBased::IdCheck Class Reference

Provides methods to check the IDs.

```
#include <IdCheck.h>
```

Static Public Member Functions

- static bool [isWildcard](#) ([IdType](#) identifier)
Check if the given database item's identifier is a wildcard or not.

6.6.1 Detailed Description

Provides methods to check the IDs.

6.6.2 Member Function Documentation

6.6.2.1 bool RuleBased::IdCheck::isWildcard ([IdType](#) identifier) [static]

Check if the given database item's identifier is a wildcard or not.

See also

[IdType](#)

Returns

True if it is a wildcard.

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

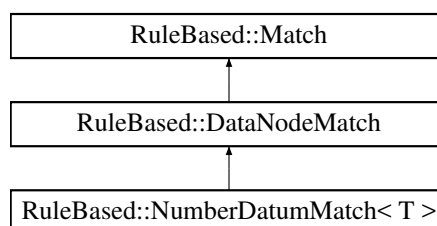
- Engine/Rules/IdCheck.h
- Engine/Rules/IdCheck.cpp

6.7 RuleBased::Match Struct Reference

Provides the mechanism to match the data item from the rule with any item inside the database.

```
#include <Match.h>
```

Inheritance diagram for RuleBased::Match:



Public Member Functions

- virtual bool [matches](#) (const [DataNode](#) *database, void *bindings)=0
Check the match on the database.

6.7.1 Detailed Description

Provides the mechanism to match the data item from the rule with any item inside the database.

6.7.2 Member Function Documentation

6.7.2.1 virtual bool RuleBased::Match::matches (const [DataNode](#) * *database*, void * *bindings*) [pure virtual]

Check the match on the database.

Parameters

<i>database</i>	The database to match on.
<i>bindings</i>	When part of the if clause matches a wild card, it is added to the bindings. This parameter is both input and output parameter.

Returns

true if matches, else returns false.

Implemented in [RuleBased::DataNodeMatch](#).

The documentation for this struct was generated from the following file:

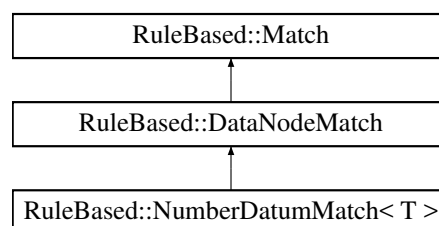
- Engine/Rules/Match.h

6.8 RuleBased::NumberDatumMatch< T > Struct Template Reference

A struct provides mechanism for matching two [Datum](#) whose values is number values (int, float,...).

```
#include <NumberDatumMatch.h>
```

Inheritance diagram for RuleBased::NumberDatumMatch< T >:



Public Member Functions

- virtual bool [matchesNode](#) (const [DataNode](#) *node, void *bindings)
Matches the given database node.
- [NumberDatumMatch](#) ([IdType identifier](#), T min, T max)
Create a match object with the given identifier and range. This range-based approach allows you to match a range with more flexibility. For example, if you want to check the HP of the agent is lower than 80, you just create a matching range of [0, 79] (since both limits are inclusive).

Public Attributes

- T [min](#)
The minimum value of the matching range (inclusive).
- T [max](#)
The maximum value of the matching range (inclusive).
- [IdType identifier](#)
The identifier to match.

6.8.1 Detailed Description

```
template<typename T>
struct RuleBased::NumberDatumMatch< T >
```

A struct provides mechanism for matching two [Datum](#) whose values is number values (int, float,...).

It uses the range-based approach, since every number values in game is limited in a specific range, for example: HP is from 0 to 100, ammo is from 0 to 50, and so on. That means, the comparison operators greater than, greater than or equal, equal, less than, less than or equal can be combined into a single match struct.

Note

This struct acts not only as the mechanism of matching two number [Datum](#) nodes, it also is the example of how to implement your own type of matching with other data structures.

See also

[Match](#)
[DataNodeMatch](#)

6.8.2 Constructor & Destructor Documentation

6.8.2.1 `template<typename T > RuleBased::NumberDatumMatch< T >::NumberDatumMatch (IdType identifier, T min, T max)`

Create a match object with the given identifier and range. This range-based approach allows you to match a range with more flexibility. For example, if you want to check the HP of the agent is lower than 80, you just create a matching range of [0, 79] (since both limits are inclusive).

Note

Remember to have max value greater than or equal to the min value.

6.8.3 Member Function Documentation

6.8.3.1 `template<typename T > bool RuleBased::NumberDatumMatch< T >::matchesNode (const DataNode * node, void * bindings) [virtual]`

Matches the given database node.

Parameters

<i>node</i>	The database node to match on.
<i>bindings</i>	When part of the if clause matches a wild card, it is added to the bindings. This parameter is both input and output parameter.

Returns

true if matches, else return false.

Todo Need adding bindings mechanism here

Implements [RuleBased::DataNodeMatch](#).

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

- Engine/Rules/NumberDatumMatch.h
- Engine/Rules/NumberDatumMatch.cpp

6.9 RuleBased::Rule Class Reference

Represent a rule in a Rule-based system. A rule has two components: an if clause is going to be used to match against the database and a function to perform any action required.

```
#include <Rule.h>
```

Public Member Functions

- virtual void [action](#) ()=0
The action is going to be carried out when the rule matches.

Public Attributes

- [Match](#) * [ifClause](#)
Consist of a set of data items, in a similar format to those in the database.

6.9.1 Detailed Description

Represent a rule in a Rule-based system. A rule has two components: an if clause is going to be used to match against the database and a function to perform any action required.

6.9.2 Member Function Documentation

6.9.2.1 virtual void RuleBased::Rule::action () [pure virtual]

The action is going to be carried out when the rule matches.

Todo Examine the performance and reusability of using a method. If it is hard to expand, find a way to use a struct/class instead.

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

- Engine/Rules/Rule.h

Index

- action
 - RuleBased::Rule, [23](#)
- DataGroup
 - RuleBased::DataGroup, [12](#)
- DataNode
 - RuleBased::DataNode, [14](#)
- Datum
 - RuleBased::Datum, [18](#), [19](#)
- getIdentifier
 - RuleBased::DataNode, [15](#)
- getLeftMostChild
 - RuleBased::DataGroup, [12](#)
- getRightSibling
 - RuleBased::DataNode, [15](#)
- getValue
 - RuleBased::Datum, [19](#)
- IdType
 - RuleBased, [10](#)
- isDatum
 - RuleBased::DataNode, [15](#)
 - RuleBased::Datum, [19](#)
- isGroup
 - RuleBased::DataGroup, [12](#)
 - RuleBased::DataNode, [15](#)
- isWildcard
 - RuleBased::IdCheck, [20](#)
- matches
 - RuleBased::DataNodeMatch, [16](#)
 - RuleBased::Match, [21](#)
- matchesChildren
 - RuleBased::DataNodeMatch, [17](#)
- matchesNode
 - RuleBased::DataGroupMatch, [13](#)
 - RuleBased::DataNodeMatch, [17](#)
 - RuleBased::NumberDatumMatch, [22](#)
- NumberDatumMatch
 - RuleBased::NumberDatumMatch, [22](#)
- RuleBased, [9](#)
 - IdType, [10](#)
- RuleBased::DataGroup, [11](#)
 - DataGroup, [12](#)
 - getLeftMostChild, [12](#)
 - isGroup, [12](#)
- RuleBased::DataGroupMatch, [12](#)
 - matchesNode, [13](#)
- RuleBased::DataNode, [14](#)
 - DataNode, [14](#)
 - getIdentifier, [15](#)
 - getRightSibling, [15](#)
 - isDatum, [15](#)
 - isGroup, [15](#)
- RuleBased::DataNodeMatch, [16](#)
 - matches, [16](#)
 - matchesChildren, [17](#)
 - matchesNode, [17](#)
- RuleBased::Datum
 - Datum, [18](#), [19](#)
 - getValue, [19](#)
 - isDatum, [19](#)
 - setValue, [19](#)
- RuleBased::Datum < T >, [18](#)
- RuleBased::IdCheck, [20](#)
 - isWildcard, [20](#)
- RuleBased::Match, [20](#)
 - matches, [21](#)
- RuleBased::NumberDatumMatch
 - matchesNode, [22](#)
 - NumberDatumMatch, [22](#)
- RuleBased::NumberDatumMatch < T >, [21](#)
- RuleBased::Rule, [23](#)
 - action, [23](#)
- setValue
 - RuleBased::Datum, [19](#)