

myTree  
0.1

Generated by Doxygen 1.8.5

Thu Nov 21 2013 17:10:45



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>Namespace Documentation</b>	<b>5</b>
3.1	node Namespace Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Function Documentation . . . . .	5
3.1.2.1	operator<< . . . . .	5
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
4.1	node::addressError Struct Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.2	tree::addressError Struct Reference . . . . .	7
4.3	node::insertError Struct Reference . . . . .	7
4.3.1	Detailed Description . . . . .	7
4.4	tree::insertError Struct Reference . . . . .	8
4.5	node::Node Class Reference . . . . .	8
4.5.1	Detailed Description . . . . .	8
4.5.2	Constructor & Destructor Documentation . . . . .	9
4.5.2.1	Node . . . . .	9
4.5.2.2	Node . . . . .	9
4.5.2.3	Node . . . . .	9
4.5.2.4	~Node . . . . .	9
4.5.3	Member Function Documentation . . . . .	9
4.5.3.1	addLeft . . . . .	9
4.5.3.2	addLeft . . . . .	9
4.5.3.3	addRight . . . . .	9
4.5.3.4	addRight . . . . .	10
4.5.3.5	deleteLeft . . . . .	10

4.5.3.6	<code>deleteRight</code>	10
4.5.3.7	<code>getLeft</code>	10
4.5.3.8	<code>getRight</code>	10
4.5.3.9	<code>getTop</code>	10
4.5.3.10	<code>normalizeLevel</code>	10
4.5.3.11	<code>operator&lt;</code>	10
4.5.3.12	<code>operator&lt;</code>	10
4.5.3.13	<code>operator=</code>	11
4.5.3.14	<code>operator=</code>	11
4.5.3.15	<code>operator&gt;</code>	11
4.5.3.16	<code>operator&gt;</code>	11
4.6	<code>tree::Tree</code> Class Reference	11

## Index

13

# Chapter 1

## Namespace Index

### 1.1 Namespace List

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

<a href="#">node</a>	– namespace for Node-relation classes . . . . .	<a href="#">5</a>
----------------------	---	-------------------



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">node::addressError</a>	
Thrown if try to access empty place . . . . .	7
<a href="#">tree::addressError</a>	7
<a href="#">node::insertError</a>	
Thrown if try to add node to busy place . . . . .	7
<a href="#">tree::insertError</a>	8
<a href="#">node::Node</a>	
Provides CRUD operations with binary related data . . . . .	8
<a href="#">tree::Tree</a>	11





## Chapter 3

# Namespace Documentation

### 3.1 node Namespace Reference

– namespace for Node-relation classes.

#### Classes

- struct [insertError](#)  
*Thrown if try to add node to busy place.*
- struct [addressError](#)  
*Thrown if try to access empty place.*
- class [Node](#)  
*Provides CRUD operations with binary relationed data.*

#### Functions

- `std::ostream & operator<< (std::ostream &os, const node::Node &cNode)`
- `void \_connect (Node *cNode, const Node *pNode)`

#### 3.1.1 Detailed Description

– namespace for Node-relation classes.

#### 3.1.2 Function Documentation

##### 3.1.2.1 `std::ostream& node::operator<< ( std::ostream & os, const node::Node & cNode )`

Outputs name of current node to output stream



## Chapter 4

# Class Documentation

### 4.1 node::addressError Struct Reference

Thrown if try to access empty place.

```
#include <node.h>
```

#### 4.1.1 Detailed Description

Thrown if try to access empty place.

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

- src/node.h

### 4.2 tree::addressError Struct Reference

#### Public Attributes

- const std::string **msg**

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

- src/tree.h

### 4.3 node::insertError Struct Reference

Thrown if try to add node to busy place.

```
#include <node.h>
```

#### 4.3.1 Detailed Description

Thrown if try to add node to busy place.

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

- src/node.h

## 4.4 tree::insertError Struct Reference

### Public Attributes

- const std::string **msg**

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

- src/tree.h

## 4.5 node::Node Class Reference

Provides CRUD operations with binary relationed data.

```
#include <node.h>
```

### Public Member Functions

- [Node](#) ()
- [Node](#) (const std::string &iName)
- [Node](#) (const [Node](#) &iNode)
- [~Node](#) ()
- [Node](#) & [getLeft](#) () const
- [Node](#) & [getRight](#) () const
- [Node](#) & [getTop](#) () const
- [Node](#) & [addLeft](#) (const std::string &iName)
- [Node](#) & [addLeft](#) (const [Node](#) &iNode)
- [Node](#) & [addRight](#) (const std::string &iName)
- [Node](#) & [addRight](#) (const [Node](#) &iNode)
- [Node](#) & [deleteLeft](#) ()
- [Node](#) & [deleteRight](#) ()
- std::string [getName](#) () const
- [Node](#) & [setName](#) (const std::string &iName)
- [Node](#) & [operator=](#) (const [Node](#) &iNode)
- [Node](#) & [operator=](#) (const std::string &iName)
- [Node](#) & [operator<](#) (const [Node](#) &iNode)
- [Node](#) & [operator<](#) (const std::string &iName)
- [Node](#) & [operator>](#) (const [Node](#) &iNode)
- [Node](#) & [operator>](#) (const std::string &iName)
- unsigned int [getLevel](#) () const
- unsigned int [normalizeLevel](#) ()

### Friends

- void [\\_connect](#) ([Node](#) \*cNode, const [Node](#) \*pNode)

### 4.5.1 Detailed Description

Provides CRUD operations with binary relationed data.

## 4.5.2 Constructor & Destructor Documentation

### 4.5.2.1 node::Node::Node ( )

Creates node with empty name and zero depth.

### 4.5.2.2 node::Node::Node ( const std::string & iName )

Creates node with specified name and zero depth.

Parameters

<i>iName</i>	– name for node.
--------------	------------------

### 4.5.2.3 node::Node::Node ( const Node & iNode )

Creates node from other node structure (copy constructor). Level may be not normalized, but grown.

Parameters

<i>iNode</i>	– source node to copy.
--------------	------------------------

### 4.5.2.4 node::Node::~~Node ( )

Recursive destructor.

## 4.5.3 Member Function Documentation

### 4.5.3.1 Node & node::Node::addLeft ( const std::string & iName )

Inserts node with specified name as left child, if not exists. Normalizes inserted node. Throws [node::insertError](#) exception in other way.

Parameters

<i>iName</i>	– name of node to insert.
--------------	---------------------------

### 4.5.3.2 Node & node::Node::addLeft ( const Node & iNode )

Inserts node structure as left child, if not exists. Normalizes inserted subtree. Throws [node::insertError](#) exception in other way.

Parameters

<i>iNode</i>	– source of node structure to insert.
--------------	---------------------------------------

### 4.5.3.3 Node & node::Node::addRight ( const std::string & iName )

Inserts node with specified name as right child, if not exists. Normalizes inserted node. Throws [node::insertError](#) exception in other way.

## Parameters

<i>iName</i>	– name of node to insert.
--------------	---------------------------

**4.5.3.4 Node & node::Node::addRight ( const Node & iNode )**

Inserts node structure as right child, if not exists. Normalizes inserted subtree. Throws [node::insertError](#) exception in other way.

## Parameters

<i>iNode</i>	– source of node structure to insert.
--------------	---------------------------------------

**4.5.3.5 Node & node::Node::deleteLeft ( )**

Deletes left subtree.

**4.5.3.6 Node & node::Node::deleteRight ( )**

Deletes right subtree.

**4.5.3.7 Node & node::Node::getLeft ( ) const**

Returns reference to left child, if exists. Throws [node::addressError](#) exception in other way.

**4.5.3.8 Node & node::Node::getRight ( ) const**

Returns reference to right child, if exists. Throws [node::addressError](#) exception in other way.

**4.5.3.9 Node & node::Node::getTop ( ) const**

Returns reference to parent node, if exists. Throws [node::addressError](#) exception in other way.

**4.5.3.10 unsigned int node::Node::normalizeLevel ( )**

Makes node levels growing in order, counted form root. Root depth == 0. Returns counted depth of tree.

**4.5.3.11 Node & node::Node::operator< ( const Node & iNode )**

Add specified node structure as left child. Normalizes inserted subtree. Throws [node::insertError](#) exception if try to insert iNode to busy place.

## Parameters

<i>iNode</i>	– source node structure inserted node.
--------------	--

**4.5.3.12 Node & node::Node::operator< ( const std::string & iName )**

Add node with specified name as left child. Throws [node::insertError](#) exception if try to insert iNode to busy place.

## Parameters

<i>iName</i>	– source name for node to insert.
--------------	-----------------------------------

## 4.5.3.13 Node &amp; node::Node::operator= ( const Node &amp; iNode )

Set current node equal to *copy* of specified node structure.

## Parameters

<i>iNode</i>	– source node structure.
--------------	--------------------------

## 4.5.3.14 Node &amp; node::Node::operator= ( const std::string &amp; iName )

Set current node equal to node with specified name.

## Parameters

<i>iName</i>	– source name for node to replace.
--------------	------------------------------------

## 4.5.3.15 Node &amp; node::Node::operator&gt; ( const Node &amp; iNode )

Add specified node structure as right child. Normalizes inserted subtree. Throws [node::insertError](#) exception if try to insert iNode to busy place.

## Parameters

<i>iNode</i>	– source node structure inserted node.
--------------	--

## 4.5.3.16 Node &amp; node::Node::operator&gt; ( const std::string &amp; iName )

Add node with specified name as right child. Throws [node::insertError](#) exception if try to insert iNode to busy place.

## Parameters

<i>iName</i>	– source name for node to insert.
--------------	-----------------------------------

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

- src/node.h

## 4.6 tree::Tree Class Reference

### Public Member Functions

- **Tree** (const std::string &iName)
- **Tree** (const [node::Node](#) &iNode)
- **Tree** (const [tree::Tree](#) &iTree)
- **Tree** & **addLeft** (const std::string &iName)
- **Tree** & **addLeft** (const [node::Node](#) &iNode)
- **Tree** & **addLeft** (const [Tree](#) &iTree)
- **Tree** & **addRight** (const std::string &iName)
- **Tree** & **addRight** (const [node::Node](#) &iNode)
- **Tree** & **addRight** (const [Tree](#) &iTree)

- [Tree](#) & **moveLeft** ()
- [Tree](#) & **moveRight** ()
- [Tree](#) & **moveTop** ()
- [Tree](#) & **moveRoot** ()
- [Tree](#) & **deleteLeft** ()
- [Tree](#) & **deleteRight** ()
- std::string **getName** () const
- [Tree](#) & **setName** (const std::string &iName)
- [Tree](#) & **operator=** (const [tree::Tree](#) &iTree)
- [Tree](#) & **operator=** (const [node::Node](#) &iNode)
- [Tree](#) & **operator=** (const std::string &iName)
- [Tree](#) & **operator<** (const [tree::Tree](#) &iTree)
- [Tree](#) & **operator<** (const [node::Node](#) &iNode)
- [Tree](#) & **operator<** (const std::string &iName)
- [Tree](#) & **operator>** (const [tree::Tree](#) &iTree)
- [Tree](#) & **operator>** (const [node::Node](#) &iNode)
- [Tree](#) & **operator>** (const std::string &iName)
- unsigned int **getLevel** () const
- unsigned int **getDepth** () const
- [node::Node](#) & **getNodes** () const
- TreeNodes **search** (const std::string &sName) const
- TreeContent **getContent** () const
- TreeNodes **operator[]** (const std::string &sName) const

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

- src/tree.h



# Index

~Node  
  node::Node, 9

addLeft  
  node::Node, 9

addRight  
  node::Node, 9, 10

deleteLeft  
  node::Node, 10

deleteRight  
  node::Node, 10

getLeft  
  node::Node, 10

getRight  
  node::Node, 10

getTop  
  node::Node, 10

Node  
  node::Node, 9

node, 5  
  operator<<, 5

node::Node, 8  
  ~Node, 9  
  addLeft, 9  
  addRight, 9, 10  
  deleteLeft, 10  
  deleteRight, 10  
  getLeft, 10  
  getRight, 10  
  getTop, 10  
  Node, 9  
  normalizeLevel, 10  
  operator<, 10  
  operator>, 11  
  operator=, 11

node::addressError, 7

node::insertError, 7

normalizeLevel  
  node::Node, 10

operator<  
  node::Node, 10

operator<<  
  node, 5

operator>  
  node::Node, 11

operator=  
  node::Node, 11

tree::Tree, 11

tree::addressError, 7

tree::insertError, 8