myTree

0.1

Generated by Doxygen 1.8.5

Thu Nov 21 2013 17:10:45

Contents

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Class Index

2.1 Class List

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

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

iName	– 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

iNode	– 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

iName	– 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

iNode	– 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

iName	– 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

iNode	– 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

iNode	– 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

iName	– source name for node to insert.

4.5.3.13 Node & node::Node::operator= (const Node & *iNode*)

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

Parameters

iNode	– source node structure.

4.5.3.14 Node & node::Node::operator= (const std::string & *iName*)

Set current node equal to node with specified name.

Parameters

iName	– source name for node to replace.

4.5.3.15 Node & node::Node::operator> (const Node & *iNode*)

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

Parameters

iNode	– source node structure inserted node.

4.5.3.16 Node & node::Node::operator> (const std::string & *iName*)

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

Parameters

iName	– 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