

Semiautodoc

Aleksandra Prohorova, Andrey Barekov

24 April, 2021

Problem

- ▶ necessity of documentation
- ▶ desire to edit the documentation separately from code
- ▶ opportunity of manual fix if something went wrong

Requirements

- ▶ automatic parsing of classes and function's names from source code
- ▶ edit names that were produced as a result of parsing
- ▶ add names to the current list if some of them were missed and delete names that seem unimportant, add description for each element to get most relevant documentation
- ▶ get readable markdown file from tree structure of classes and functions

Project structure

- ▶ static library for parsing
- ▶ unit and functional tests for static library
- ▶ GUI

Example

Results of parsing

The screenshot shows the Semiautodoc application interface. On the left is a file explorer showing a project structure with folders like 'build', 'parser', 'include', 'src', and 'test'. The 'include' folder is expanded, showing files like 'class.hpp', 'element.hpp', 'function.hpp', and 'parser.hpp'. The 'element.hpp' file is selected and its contents are displayed in the main window. The main window has a table with two columns: 'Name' and 'Description'. The table contains one entry for the 'class Element'.

Name	Description
class Element	<pre>Element(); Element(std::string name); ~Element(); bool operator==(const Element& element) const; bool operator!=(const Element& element) const; std::string getName() const; std::string getDescription() const; void addElement(pointer newElement); void show(std::ostream& out); bool isComposite() const; listOfElements& getListOfElements(); vectorOfElements& getVectorOfElements(); pointer getParent(); int getRow(); void setName(std::string name); void setDescription(std::string description); void removeElement(int row); void addElement(int row, Element::pointer newElement);</pre>

At the bottom of the main window, there are three buttons: 'Remove row', 'Insert row', and 'Insert node'.

Example

Edit results of parsing

The screenshot shows a Qt IDE window titled 'element.hpp *'. The left sidebar displays a project structure with folders 'build', 'parser', 'include', 'src', and 'test', and files 'CMakeLists.txt', 'build_script.sh', and 'CMakeLists.txt'. The main editor area shows the 'element.hpp' file with a table of code and descriptions.

Name	Description
<code>class Element</code>	
<code>Element();</code>	
<code>Element(std::string name);</code>	
<code>~Element();</code>	
<code>bool operator==(const Element& element) const;</code>	Ввиду некоторых причин невозможно воспользоваться средствами автоматического управления ресурсами, так что ... Необходимо для сравнения ожидаемого и полученного дерева в тестах
<code>std::string getName() const;</code>	
<code>std::string getDescription() const;</code>	
<code>void addElement(pointer newElement);</code>	
<code>void show(std::ostream& out);</code>	
<code>bool isComposite() const;</code>	
<code>listOfElements& getListOfElements();</code>	Информация, необходимая для решения того, выводить ли таблицу для вложенных элементов
<code>vectorOfElements& getVectorOfElements();</code>	
<code>pointer getParent();</code>	Необходимо для создания QTreeView, информация о родителе сохраняется при добавлении элемента
<code>int getRow();</code>	
<code>void setName(std::string name);</code>	
<code>void setDescription(std::string description);</code>	
<code>void removeElement(int row);</code>	
<code>void addElement(int row, Element::pointer newElement);</code>	

At the bottom of the editor, there are three buttons: 'Remove row', 'Insert row', and 'Insert node'.

Example

Markdown file

22 lines (21 sloc) | 1.36 KB

Raw Blame

class Element

Name	Description
Element();	
Element(std::string name);	
~Element();	Ввиду некоторых причин невозможно воспользоваться средствами автоматического управления ресурсами, так что освободить память приходится вручную
bool operator==(const Element& element) const;	Необходимо для сравнения ожидаемого и полученного дерева в тестах
bool operator!=(const Element& element) const;	
std::string getName() const;	
std::string getDescription() const;	
void addElement(pointer newElement);	
void show(std::ostream& out);	
bool isComposite() const;	Информация, необходимая для решения того, выводить ли таблицу для вложенных элементов
listOfElements& getListOfElements();	
vectorOfElements getVectorOfElements();	
pointer getParent();	Необходимо для создания QTreeView, информация о родителе сохраняется при добавлении элемента
int getRow();	
void setName(std::string name);	
void setDescription(std::string description);	
void removeElement(int row);	
void addElement(int row, Element* pointer newElement);	

Source code

[**https://github.com/aleksandraprohorova/semiautodoc**](https://github.com/aleksandraprohorova/semiautodoc)