My Project

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 BC Class Reference

Public Member Functions

- **BC** (const GetPot &dataFile, const std::string &problem, const std::string &section)
- void **setBoundaries** (getfem::mesh &meshRef)
- std::vector< size_type > **getNeumBD** () const
- std::vector< size_type > **getDiriBD** () const
- scalar_type **BCNeum** (const base_node &x, const size_type what, const size_type &flag)
- scalar_type **BCDiri** (const base_node &x, const size_type what, const size_type &flag)

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

- include/BC.h

4.2 Bulk Class Reference

Public Member Functions

- **Bulk** (const GetPot &dataFile, const std::string &section="bulkData/", const std::string &sectionDomain="domain/", const std::string &sectionProblem="nothing", const std::string &domainNumber="nothing")
- void **exportMesh** (std::string const nomefile) const
- getfem::mesh & **getMeshRef** ()
- getfem::mesh **getMesh** () const
- scalar_type **Lx** () const
- scalar_type **Ly** () const
- scalar_type **nSubX** () const
- scalar_type **nSubY** () const

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

- include/Bulk.h

4.3 BulkDatum Class Reference

Public Member Functions

- **BulkDatum** (const GetPot &dataFile, const std::string &section, const std::string &sectionProblem, const std::string &domainNumber, const std::string &datum)
- scalar_type getValue (const base_node &x, const size_type what)

  method to evaluate the string.

4.3.1 Member Function Documentation

4.3.1.1 getValue()

```
scalar_type BulkDatum::getValue (
             const base_node & x,
             const size_type what )
```

method to evaluate the string.

This method takes two input parameters and returns a scalar value.

Parameters

x	coordinates of the point where we want to evaluate the function.
what	index 0 if the datum is a scalar or if we want the first component of the vector; index 1 if we want to evaluate the second component.

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

- include/BulkDatum.h

4.4 FEM Class Reference

Public Member Functions

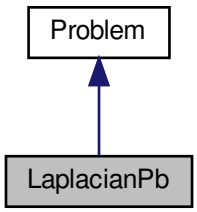
- **FEM** (const getfem::mesh &mesh, const GetPot &dataFile, const std::string &problem, const std::string &variable, const std::string &section="bulkData/", const size_type qdim=1)
- **FEM** (const getfem::mesh &mesh, const std::string femType, const size_type spaceDim)
- size_type **nb_dof** () const
- std::string **type** () const
- getfem::mesh_fem **getFEM** () const
- std::vector< base_node > **getDOFpoints** () const
- base_node **point_of_basic_dof** (size_type i) const

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

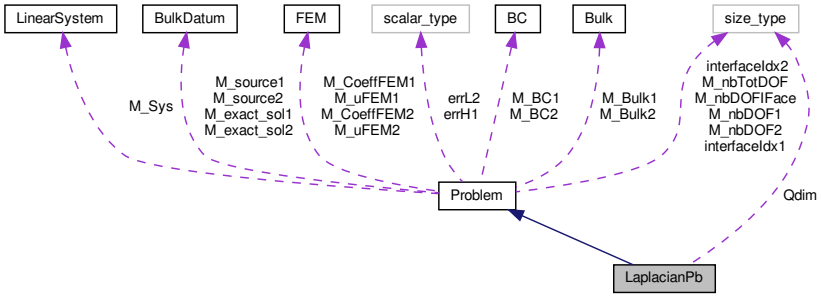
- include/FEM.h

4.5 LaplacianPb Class Reference

Inheritance diagram for LaplacianPb:



Collaboration diagram for LaplacianPb:



Public Member Functions

- **LaplacianPb** (GetPot const &dataFile, Bulk &bulk1, Bulk &bulk2, LinearSystem &extSys)
- void **assembleMatrix** () override

Static Public Attributes

- static const size_type **Qdim** = 1

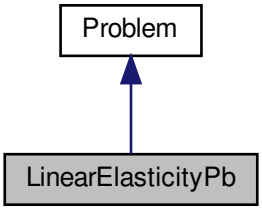
Additional Inherited Members

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

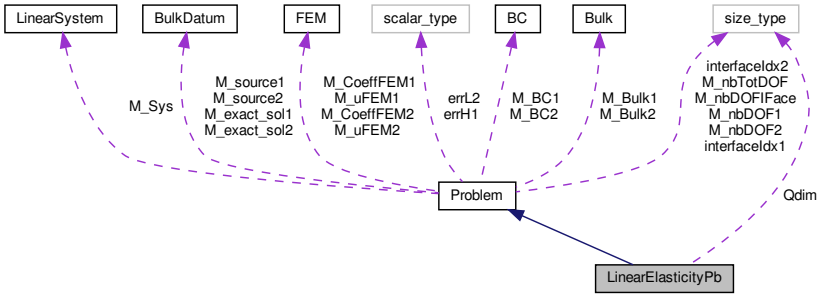
- include/LaplacianPb.h

4.6 LinearElasticityPb Class Reference

Inheritance diagram for LinearElasticityPb:



Collaboration diagram for LinearElasticityPb:



Public Member Functions

- **LinearElasticityPb** (GetPot const &dataFile, Bulk &bulk1, Bulk &bulk2, LinearSystem &extSys)
- void **assembleMatrix** () override

Static Public Attributes

- static const size_type **Qdim** = 2

Additional Inherited Members

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

- include/LinearElasticityPb.h

4.7 LinearSystem Class Reference

Public Member Functions

- void **addToMatrix** (int ndof)
- void **copySubMatrix** (sparseMatrixPtr_Type source, int first_row, int first_column, scalar_type scale=1.0, bool transpose=false)
- void **addSubMatrix** (sparseMatrixPtr_Type source, int first_row, int first_column, scalar_type scale=1.0, bool transpose=false)
- void **extractSubMatrix** (sparseMatrixPtr_Type destination, int first_row, int number_rows, int first_column, int number_cols) const
- void **copySubVector** (scalarVectorPtr_Type source, int first_row, scalar_type scale=1.0)
- void **addSubVector** (scalarVectorPtr_Type source, int first_row, scalar_type scale=1.0)
- void **extractSubVector** (scalarVectorPtr_Type destination, int first_row, std::string where="sol") const
- void **extractSubVector** (scalarVector_Type &destination, int first_row, std::string where="sol") const
- sparseMatrixPtr_Type **getMatrix** () const
- scalarVectorPtr_Type **getRHS** () const
- scalarVectorPtr_Type **getSol** () const
- void **addSubSystem** (LinearSystem *small, size_type shiftRows, size_type shiftColumns)
- void **solve** ()
- void **computeInverse** ()
- void **saveMatrix** (const char *nomefile="Matrix.mm") const
- void **multAddToRHS** (scalarVectorPtr_Type V, int first_row, int first_column, int nrows, int ncols)
- void **multAddToRHS** (sparseMatrixPtr_Type M, scalarVectorPtr_Type V, int first_rowVector, int first_rowRHS, scalar_type scale=1.0, bool transposed=false)
- void **multAddToRHS** (sparseMatrixPtr_Type M, scalarVector_Type &V, int first_rowVector, int first_rowRHS, scalar_type scale=1.0, bool transposed=false)
- void **cleanRHS** ()
- void **cleanMAT** ()
- void **setNullRow** (size_type which)
- void **setMatrixValue** (size_type i, size_type j, scalar_type value)
- void **setRHSValue** (size_type i, scalar_type value)

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

- include/LinearSystem.h

4.8 LifeV::Parser Class Reference

Parser - A string parser for algebraic expressions.

`#include <Parser.h>`

Public Types

Public Types

- typedef std::vector< std::string > **stringsVector_Type**
- typedef std::string::const_iterator **stringIterator_Type**
- typedef ParserSpiritGrammar< stringIterator_Type > **calculator_Type**
- typedef calculator_Type::results_Type **results_Type**

Public Member Functions

Constructors & Destructor

- Parser ()
  Empty constructor (it needs a manual call to setString)
- Parser (const std::string &string)
  Constructor.
- Parser (const Parser &parser)
  Copy constructor.
- virtual ~Parser ()
  Destructor.

Operators

- Parser & operator= (const Parser &parser)
  Operator =.

Methods

- const scalar_type & evaluate (const ID &id=0)
- UInt countSubstring (const std::string &substring)
- void clearVariables ()
  Clear all the variables.

Set Methods

- void setString (const std::string &string, const std::string &stringSeparator=";")
- void setVariable (const std::string &name, const scalar_type &value)

Get Methods

- const scalar_type & variable (const std::string &name)

4.8.1 Detailed Description

Parser - A string parser for algebraic expressions.

Author

(s) Cristiano Malossi, Gilles Fourestey

See `ParserSpiritGrammar` class for more details.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Parser() [1/2]

```
LifeV::Parser::Parser (
            const std::string & string )  [explicit]
```

Constructor.

Parameters

string	expression to parse

4.8.2.2 Parser() [2/2]

```
LifeV::Parser::Parser (
            const Parser & parser )  [explicit]
```

Copy constructor.

Parameters

parser	Parser

4.8.3 Member Function Documentation

4.8.3.1 countSubstring()

```
UInt LifeV::Parser::countSubstring (
            const std::string & substring )
```

Count how many times a substring is present in the string (utility for BCInterfaceFunction)

Parameters

substring	string to find

Returns

number of substring

4.8.3.2 evaluate()

```
const scalar_type& LifeV::Parser::evaluate (
            const ID & id = 0 )
```

Evaluate the expression

Parameters

id	expression index (starting from 0)

Returns

computed value

4.8.3.3 operator=()

```
Parser& LifeV::Parser::operator= (
            const Parser & parser )
```

Operator =.

Parameters

parser	Parser

Returns

reference to a copy of the class

4.8.3.4 setString()

```
void LifeV::Parser::setString (
            const std::string & string,
            const std::string & stringSeparator = ";" )
```

Set string function

Parameters

string	Expression to evaluate
stringSeparator	Separator identifier (default -> ";")

4.8.3.5 setVariable()

```
void LifeV::Parser::setVariable (
            const std::string & name,
            const scalar_type & value )
```

Set/replace a variable

Parameters

name	name of the parameter
value	value of the parameter

4.8.3.6 variable()

```
const scalar_type& LifeV::Parser::variable (
            const std::string & name )
```

Get variable

Parameters

name	name of the parameter

Returns

value of the variable

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

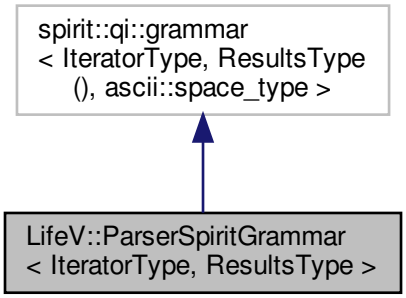
- include/Parser.h

4.9 LifeV::ParserSpiritGrammar< IteratorType, ResultsType > Class Template Reference

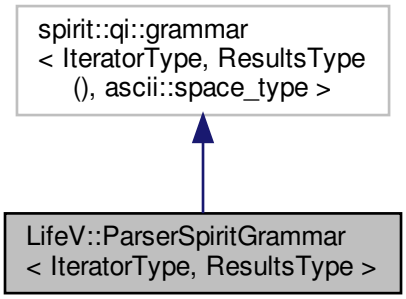
ParserSpiritGrammar - A string parser grammar based on `boost::spirit::qi`.

`#include <ParserSpiritGrammar.h>`

Inheritance diagram for LifeV::ParserSpiritGrammar< IteratorType, ResultsType >:



Collaboration diagram for LifeV::ParserSpiritGrammar< IteratorType, ResultsType >:



Public Types

Public Types

- typedef IteratorType **iterator_Type**
- typedef boost::iterator_range< iterator_Type > **iteratorRange_Type**
- typedef ResultsType **results_Type**

Public Member Functions

Constructors & Destructor

- ParserSpiritGrammar ()
  Constructor.
- ParserSpiritGrammar (const ParserSpiritGrammar &spiritGrammar)
  Copy constructor.
- virtual ~ParserSpiritGrammar ()
  Destructor.

Operators

- ParserSpiritGrammar & operator= (const ParserSpiritGrammar &spiritGrammar)
  Operator =.

Methods

- void assignVariable (const iteratorRange_Type &stringIteratorRange, const scalar_type &value)
- void clearVariables ()
  Clear all the variables.

Set Methods

- void setDefaultVariables ()
  Set default variables.
- void setVariable (const std::string &name, const scalar_type &value)

Get Methods

- scalar_type & variable (const std::string &name)

4.9.1 Detailed Description

template<typename IteratorType = std::string::const_iterator, typename ResultsType = std::vector < scalar_type >>
class LifeV::ParserSpiritGrammar< IteratorType, ResultsType >

ParserSpiritGrammar - A string parser grammar based on `boost::spirit::qi`.

Author

(s) Cristiano Malossi

`ParserSpiritGrammar` is a `boost::spirit::qi` based class to perform evaluation of `std::string` expressions.

EXAMPLE - HOW TO USE Let's consider the following example: suppose that we have this function:

[u,v,w] = f(x,y,z,t)

where

u(x) = a∗b∗x v(x,y) = a/b∗sqrt(x^2 + y^2) w(t) = b∗t;

with "a" and "b" constants such that a=5.12345, b=9.999999.

To evaluate function f(x,y,z,t), we use this syntax:

string = "a=5.12345 ; b=9.999999 ; (a∗b∗x, a/b∗sqrt(x^2 + y^2), b∗t)"

where semicolons (";") separate constants and commas (",") separate output functions.

NOTE: Currently ParserSpiritGrammar works with the following operators:

```
*  +, -, *, /, ^, sqrt(), sin(), cos(), tan(), exp(), log(), log10(), >, <.
*
```

4.9.2 Constructor & Destructor Documentation

4.9.2.1 ParserSpiritGrammar()

```
template<typename IteratorType , typename ResultsType >
LifeV::ParserSpiritGrammar< IteratorType, ResultsType >::ParserSpiritGrammar (
            const ParserSpiritGrammar< IteratorType, ResultsType > & spiritGrammar )  [explicit]
```

Copy constructor.

Parameters

ParserSpiritGrammar	ParserSpiritGrammar

4.9.3 Member Function Documentation

4.9.3.1 assignVariable()

```
template<typename IteratorType = std::string::const_iterator, typename ResultsType = std↩
::vector < scalar_type >>
void LifeV::ParserSpiritGrammar< IteratorType, ResultsType >::assignVariable (
            const iteratorRange_Type & stringIteratorRange,
            const scalar_type & value )  [inline]
```

Assign a variable using a `boost::iterator_range`

Parameters

stringIteratorRange	name of the parameter
value	value of the parameter

4.9.3.2 operator=()

```
template<typename IteratorType , typename ResultsType >
ParserSpiritGrammar< IteratorType, ResultsType > & LifeV::ParserSpiritGrammar< IteratorType,
ResultsType >::operator= (
            const ParserSpiritGrammar< IteratorType, ResultsType > & spiritGrammar )
```

Operator =.

Parameters

SpiritGrammar	ParserSpiritGrammar

Returns

reference to a copy of the class

4.9.3.3 setVariable()

```
template<typename IteratorType , typename ResultsType >
void LifeV::ParserSpiritGrammar< IteratorType, ResultsType >::setVariable (
            const std::string & name,
            const scalar_type & value )  [inline]
```

Set/replace a variable

Parameters

name	name of the parameter
value	value of the parameter

4.9.3.4 variable()

```
template<typename IteratorType = std::string::const_iterator, typename ResultsType = std↩
::vector < scalar_type >>
scalar_type& LifeV::ParserSpiritGrammar< IteratorType, ResultsType >::variable (
            const std::string & name )  [inline]
```

Get variable

Parameters

name	name of the parameter

Returns

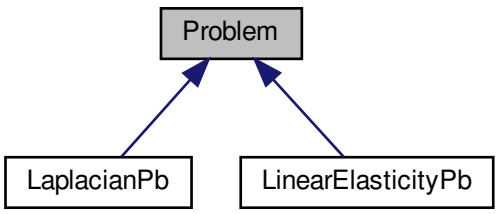
value of the variable

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

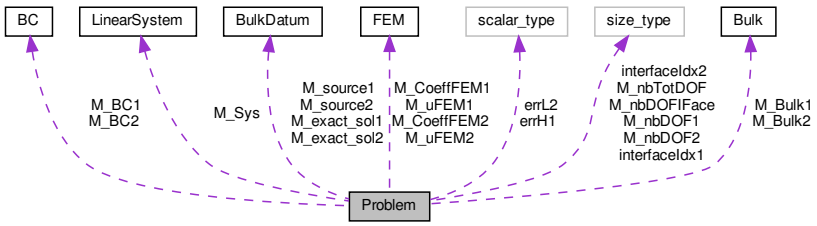
- include/ParserSpiritGrammar.h

4.10 Problem Class Reference

Inheritance diagram for Problem:



Collaboration diagram for Problem:



Public Member Functions

- **Problem** (GetPot const &dataFile, std::string const problem, Bulk &bulk1, Bulk &bulk2, const size_type dim, LinearSystem &extSys)
- FEM **getFEM** (size_type const idx) const
- LinearSystem & **getSYS** () const
- size_type **getNDOF** (std::string const variable="all") const
- scalar_type **getL2ERR** () const
- scalar_type **getH1ERR** () const
- virtual void **assembleMatrix** ()=0
- void **assembleRHS** ()
- void **enforceStrongBC** (size_type const domainIdx)
- void **enforceInterfaceJump** ()
- void **solve** ()
- void **extractSol** (scalarVector_Type &destSol, std::string const variable="all")
- void **exportVtk** (std::string const folder="./vtk", std::string const what="all")
- void **computeErrors** ()

Protected Attributes

- Bulk & **M_Bulk1**
- Bulk & **M_Bulk2**
- BC **M_BC1**
- BC **M_BC2**
- size_type **interfaceIdx1**
- size_type **interfaceIdx2**
- FEM **M_uFEM1**
- FEM **M_uFEM2**
- FEM **M_CoeffFEM1**
- FEM **M_CoeffFEM2**
- getfem::mesh_im **M_intMethod1**
- getfem::mesh_im **M_intMethod2**
- LinearSystem & **M_Sys**
- BulkDatum **M_exact_sol1**
- BulkDatum **M_exact_sol2**
- BulkDatum **M_source1**
- BulkDatum **M_source2**
- scalarVector_Type **M_uSol**
- scalarVector_Type **M_uSol1**

- scalarVector_Type **M_uSol2**
- size_type **M_nbDOF1**
- size_type **M_nbDOF2**
- size_type **M_nbTotDOF**
- size_type **M_nbDOFIFace**
- sizeVector_Type **dof_IFace1**
- sizeVector_Type **dof_IFace2**
- sizeVector_Type **M_rowsStrongBC1**
- sizeVector_Type **M_rowsStrongBC2**
- sizeVector_Type **M_rowsIFace1**
- sizeVector_Type **M_rowsIFace2**
- sizeVector_Type **M_rowsStrongBCFlags1**
- sizeVector_Type **M_rowsStrongBCFlags2**
- sizeVector_Type **M_rowsIFace**
- scalar_type **errL2**
- scalar_type **errH1**

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

- include/Problem.h

Chapter 5

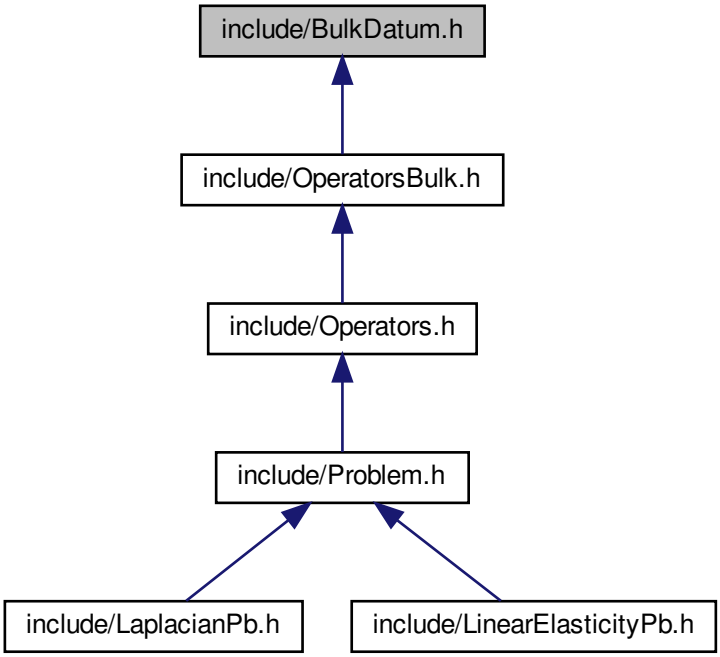
File Documentation

5.1 include/BulkDatum.h File Reference

```
#include "Core.h"
#include "Parser.h"
```

Include dependency graph for BulkDatum.h:

This graph shows which files directly or indirectly include this file:



Classes

- class BulkDatum

5.1.1 Detailed Description

is a class for any kind of data related to the problem.

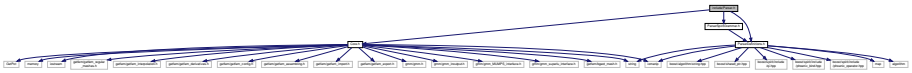
This class is used to store the string describing the parameters (diffusion, Lamè...) and the functions (forcing, exact solution) related to the problem we are interested in. They can be both scalars and vectors.

5.2 include/Parser.h File Reference

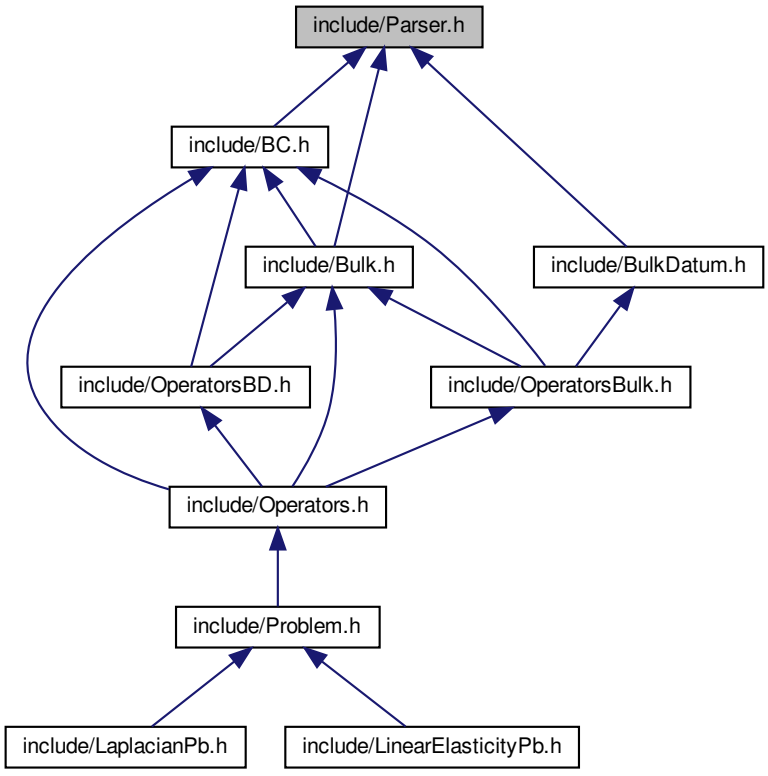
File containing the Parser interface.

```
#include "Core.h"
#include "ParserDefinitions.h"
#include "ParserSpiritGrammar.h"
```

Include dependency graph for Parser.h:



This graph shows which files directly or indirectly include this file:



Classes

- class LifeV::Parser

  Parser - A string parser for algebraic expressions.

5.2.1 Detailed Description

File containing the Parser interface.

Date

07-04-2009

Author

Cristiano Malossi cristiano.malossi@epfl.ch

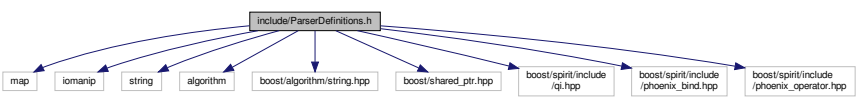
Gilles Fourestey gilles.fourestey@epfl.ch Cristiano Malossi cristiano.malossi@epfl.ch

5.3 include/ParserDefinitions.h File Reference

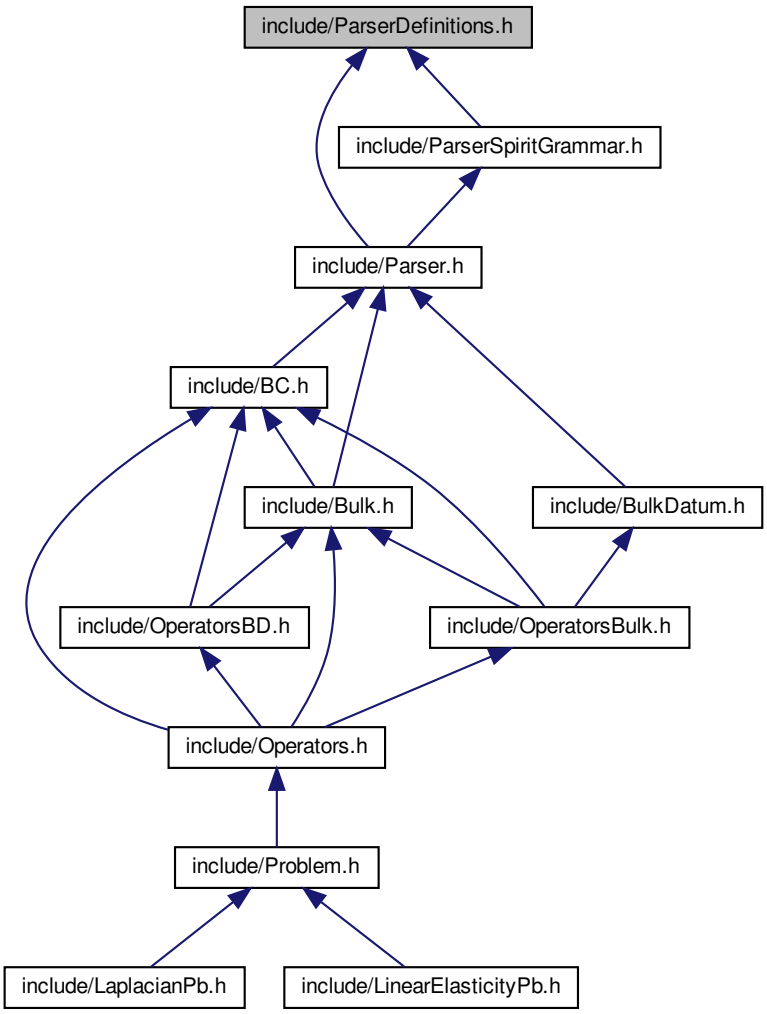
File containing the Parser definitions.

```
#include <map>
#include <iomanip>
#include <string>
#include <algorithm>
#include <boost/algorithm/string.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/spirit/include/qi.hpp>
#include <boost/spirit/include/phoenix_bind.hpp>
#include <boost/spirit/include/phoenix_operator.hpp>
```

Include dependency graph for ParserDefinitions.h:



This graph shows which files directly or indirectly include this file:



5.3.1 Detailed Description

File containing the Parser definitions.

Date

29-01-2010

Author

Cristiano Malossi cristiano.malossi@epfl.ch

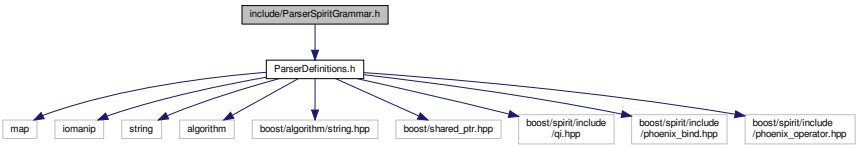
Cristiano Malossi cristiano.malossi@epfl.ch

5.4 include/ParserSpiritGrammar.h File Reference

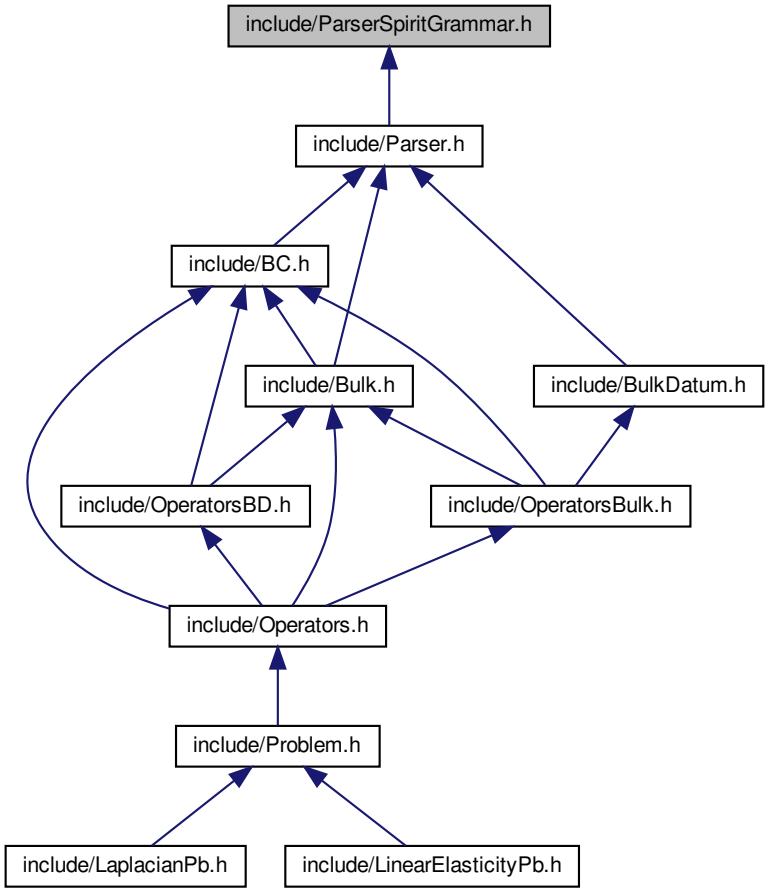
File containing the Parser grammar.

```
#include "ParserDefinitions.h"
```

Include dependency graph for ParserSpiritGrammar.h:



This graph shows which files directly or indirectly include this file:



Classes

- class LifeV::ParserSpiritGrammar< IteratorType, ResultsType >

  ParserSpiritGrammar - A string parser grammar based on `boost::spirit::qi`.

5.4.1 Detailed Description

File containing the Parser grammar.

Date

05-02-2010

Author

Cristiano Malossi cristiano.malossi@epfl.ch

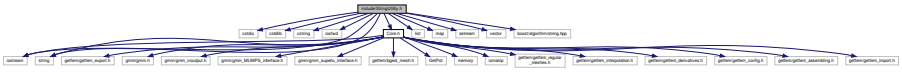
Cristiano Malossi cristiano.malossi@epfl.ch

5.5 include/StringUtility.h File Reference

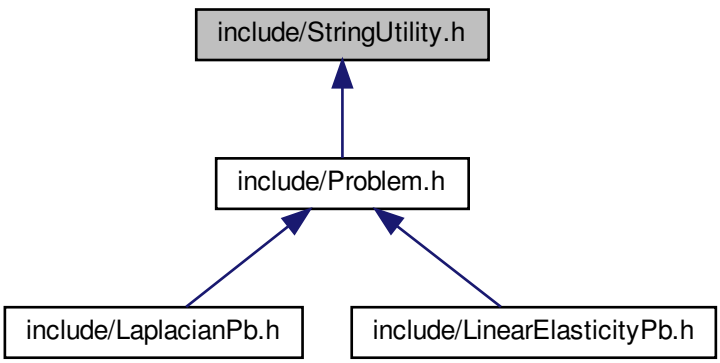
std::string utilities

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iosfwd>
#include <iostream>
#include <list>
#include <map>
#include <sstream>
#include <string>
#include <vector>
#include <boost/algorithm/string.hpp>
#include "Core.h"
```

Include dependency graph for StringUtility.h:



This graph shows which files directly or indirectly include this file:



Functions

- std::istream & LifeV::eatLine (std::istream &s)
- std::istream & LifeV::eatComments (std::istream &s)

  skip lines starting with '!%#;$'
- std::istream & LifeV::nextGoodLine (std::istream &s, std::string &line)

  gets next uncommented line
- std::string & LifeV::setStringLength (std::string &s, unsigned int len, char c)
- int LifeV::atoi (const std::string &s)

  extends atoi to STL std::strings (from Stroustrup)
- std::string **LifeV::operator+** (const std::string &str, const int i)
- std::string **LifeV::operator+** (const std::string &str, const long int i)
- std::string **LifeV::operator+** (const std::string &str, const unsigned int i)
- template<typename EntryType >
  void **LifeV::parseList** (const std::string &slist, std::list< EntryType > &list)
- double **LifeV::string2number** (const std::string &s)
- template<typename NumberType >
  std::string **LifeV::number2string** (const NumberType &n)
- template<typename EnumeratorType >
  std::string **LifeV::enum2String** (const EnumeratorType &Enum, const std::map< std::string, Enumerator↩
  Type > &Map)
- template<typename NumberType >
  void **LifeV::string2numbersVector** (const std::string &string, std::vector< NumberType > &numberVector)

5.5.1 Detailed Description

std::string utilities

Date

13-12-2010

Author

Radu Popescu radu.popescu@epfl.ch

5.5.2 Function Documentation

5.5.2.1 eatLine()

```
std::istream& LifeV::eatLine (
            std::istream & s )
```

It gets a the next line from std::istream

5.5.2.2 setStringLength()

```
std::string& LifeV::setStringLength (
             std::string & s,
             unsigned int len,
             char c )
```

always return a std::string with len characters

- if the s has more than len characters : keep only the first len
- if the s has less than len characters : complete with c until len

Index