

Tang

0.1

Generated by Doxygen 1.9.1

1 Tang: A Template Language	1
1.1 Quick Description	1
1.2 Features	1
1.3 License	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 Tang::AstNode Class Reference	9
5.1.1 Detailed Description	10
5.1.2 Constructor & Destructor Documentation	10
5.1.2.1 AstNode()	10
5.2 Tang::AstNodeInteger Class Reference	10
5.2.1 Detailed Description	11
5.2.2 Constructor & Destructor Documentation	11
5.2.2.1 AstNodeInteger()	11
5.3 Tang::Error Class Reference	12
5.3.1 Detailed Description	13
5.3.2 Constructor & Destructor Documentation	13
5.3.2.1 Error()	13
5.4 Tang::location Class Reference	13
5.4.1 Detailed Description	15
5.5 Tang::position Class Reference	15
5.5.1 Detailed Description	16
5.6 Tang::TangBase Class Reference	16
5.6.1 Detailed Description	17
5.6.2 Constructor & Destructor Documentation	17
5.6.2.1 TangBase()	17
5.7 Tang::TangScanner Class Reference	17
5.7.1 Detailed Description	18
5.7.2 Constructor & Destructor Documentation	18
5.7.2.1 TangScanner()	18
5.7.3 Member Function Documentation	19
5.7.3.1 get_next_token()	19
6 File Documentation	21
6.1 build/generated/location.hh File Reference	21

6.1.1 Detailed Description	22
6.1.2 Function Documentation	22
6.1.2.1 operator<<() [1/2]	23
6.1.2.2 operator<<() [2/2]	23
Index	25

Chapter 1

Tang: A Template Language

1.1 Quick Description

Tang is a C++ Template Language. It takes the form of a library which may be included in other projects. It is under active development, and you can follow its progress here:

- [YouTube playlist](#)
- [GitHub repository](#)

1.2 Features

The following features are planned:

- Native support for Unicode/Utf-8 strings.
- Change from template to script mode using escape tags like PHP.
- Loosely typed, with Python-like indexing and slicing of containers.
- Syntax similar to C/C++/PHP.
- Code compiles to a custom Bytecode and is executed by the Tang VM.
- Fast and thread-safe.

1.3 License

MIT License

Copyright (c) 2022 Corey Pennycuff

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Tang::AstNode	9
Tang::AstNodeInteger	10
Tang::Error	12
Tang::location	13
Tang::position	15
Tang::TangBase	16
TangTangFlexLexer	
Tang::TangScanner	17

Chapter 3

Class Index

3.1 Class List

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

Tang::AstNode	Base class for representing nodes of an Abstract Syntax Tree (AST)	9
Tang::AstNodeInteger	An AstNode that represents an integer literal	10
Tang::Error	Used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error	12
Tang::location	Two points in a source file	13
Tang::position	A point in a source file	15
Tang::TangBase	The base class for the Tang programming language	16
Tang::TangScanner	The Flex lexer class for the main Tang language	17

Chapter 4

File Index

4.1 File List

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

build/generated/ location.hh	
Define the Tang ::location class	21
include/ ast.hpp	??
include/ error.hpp	??
include/ tangBase.hpp	??
include/ tangScanner.hpp	??

Chapter 5

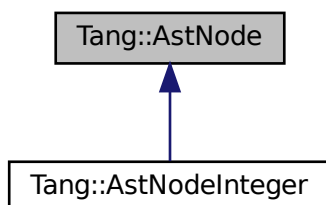
Class Documentation

5.1 Tang::AstNode Class Reference

Base class for representing nodes of an Abstract Syntax Tree (AST).

```
#include <ast.hpp>
```

Inheritance diagram for Tang::AstNode:



Public Member Functions

- virtual std::string [inspect](#) ()
Return a string that describes the contents of the node.

Protected Member Functions

- [AstNode](#) (Tang::location loc)
The generic constructor.

5.1.1 Detailed Description

Base class for representing nodes of an Abstract Syntax Tree (AST).

There will be *many* derived classes, each one conveying the syntactic meaning of the code that it represents.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AstNode()

```
Tang::AstNode::AstNode (
    Tang::location loc ) [inline], [protected]
```

The generic constructor.

It should never be called on its own.

Parameters

<i>loc</i>	The location associated with this node.
------------	---

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

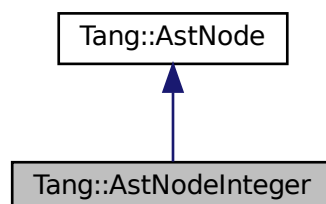
- include/ast.hpp
- src/ast.cpp

5.2 Tang::AstNodeInteger Class Reference

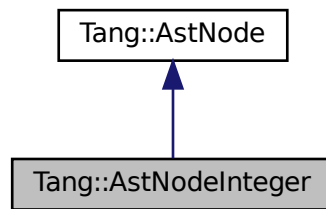
An [AstNode](#) that represents an integer literal.

```
#include <ast.hpp>
```

Inheritance diagram for Tang::AstNodeInteger:



Collaboration diagram for Tang::AstNodeInteger:



Public Member Functions

- [AstNodeInteger](#) (int64_t number, [Tang::location](#) loc)
The constructor.
- virtual std::string [inspect](#) () override
Return a string that describes the contents of the node.

5.2.1 Detailed Description

An [AstNode](#) that represents an integer literal.

Integers are represented by the `int64_t` type, and so are limited in range by that of the underlying type.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AstNodeInteger()

```

Tang::AstNodeInteger::AstNodeInteger (
    int64_t number,
    Tang::location loc ) [inline]
  
```

The constructor.

Parameters

<i>number</i>	The number to represent. @location The location associated with this node.
---------------	--

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

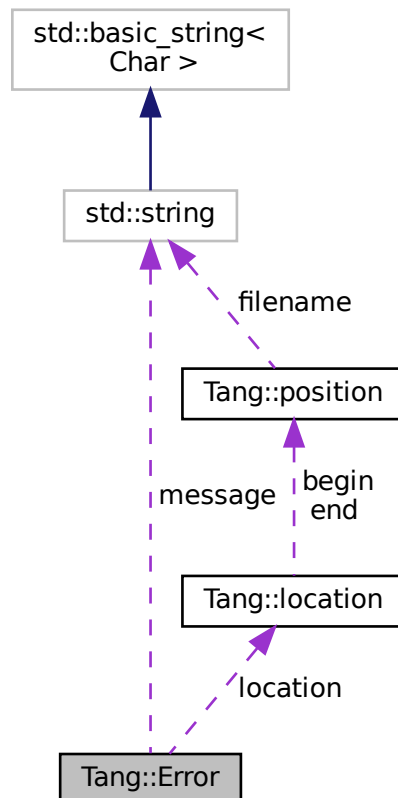
- include/ast.hpp
- src/ast.cpp

5.3 Tang::Error Class Reference

The [Error](#) class is used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error.

```
#include <error.hpp>
```

Collaboration diagram for Tang::Error:



Public Member Functions

- [Error](#) ()
Creates an empty error message.
- [Error](#) (std::string [message](#), Tang::location [location](#))
Creates an error message using the supplied error string and location.

Public Attributes

- std::string [message](#)
The error message as a string.
- Tang::location [location](#)
The location of the error.

5.3.1 Detailed Description

The [Error](#) class is used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Error()

```
Tang::Error::Error (
    std::string message,
    Tang::location location ) [inline]
```

Creates an error message using the supplied error string and location.

Parameters

<i>message</i>	The error message as a string.
<i>location</i>	The location of the error.

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

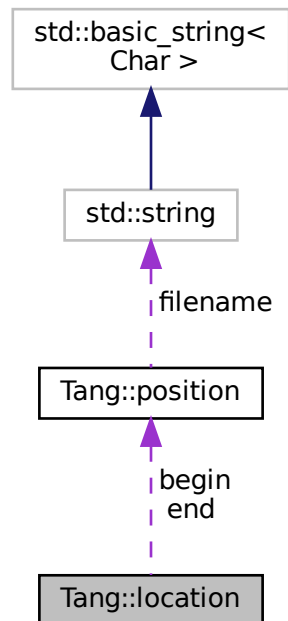
- include/error.hpp
- src/error.cpp

5.4 Tang::location Class Reference

Two points in a source file.

```
#include <location.hh>
```

Collaboration diagram for Tang::location:



Public Types

- typedef `position::filename_type filename_type`
Type for file name.
- typedef `position::counter_type counter_type`
Type for line and column numbers.

Public Member Functions

- `location` (const `position` &b, const `position` &e)
Construct a location from b to e.
- `location` (const `position` &p=`position`())
Construct a 0-width location in p.
- `location` (`filename_type` *f, `counter_type` l=1, `counter_type` c=1)
Construct a 0-width location in f, l, c.
- void `initialize` (`filename_type` *f=((void *) 0), `counter_type` l=1, `counter_type` c=1)
Initialization.

Line and Column related manipulators

- void `step` ()
Reset initial location to final location.
- void `columns` (`counter_type` count=1)
Extend the current location to the COUNT next columns.
- void `lines` (`counter_type` count=1)
Extend the current location to the COUNT next lines.

Public Attributes

- [position begin](#)
Beginning of the located region.
- [position end](#)
End of the located region.

5.4.1 Detailed Description

Two points in a source file.

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

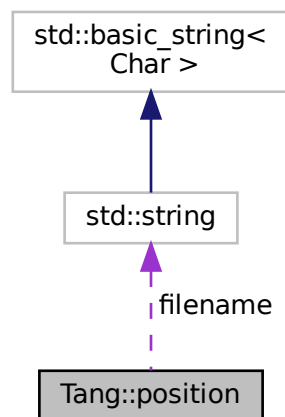
- build/generated/[location.hh](#)

5.5 Tang::position Class Reference

A point in a source file.

```
#include <location.hh>
```

Collaboration diagram for Tang::position:



Public Types

- typedef const std::string [filename_type](#)
Type for file name.
- typedef int [counter_type](#)
Type for line and column numbers.

Public Member Functions

- [position](#) ([filename_type](#) *f=((void *) 0), [counter_type](#) l=1, [counter_type](#) c=1)
Construct a position.
- void [initialize](#) ([filename_type](#) *fn=((void *) 0), [counter_type](#) l=1, [counter_type](#) c=1)
Initialization.

Line and Column related manipulators

- void [lines](#) ([counter_type](#) count=1)
(line related) Advance to the COUNT next lines.
- void [columns](#) ([counter_type](#) count=1)
(column related) Advance to the COUNT next columns.

Public Attributes

- [filename_type](#) * [filename](#)
File name to which this position refers.
- [counter_type](#) [line](#)
Current line number.
- [counter_type](#) [column](#)
Current column number.

5.5.1 Detailed Description

A point in a source file.

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

- build/generated/[location.hh](#)

5.6 Tang::TangBase Class Reference

The base class for the Tang programming language.

```
#include <tangBase.hpp>
```

Public Member Functions

- [TangBase](#) ()
The constructor.

5.6.1 Detailed Description

The base class for the Tang programming language.

This class is the fundamental starting point to compile and execute a Tang program. It may be considered in three parts:

1. It acts as an extendable interface through which additional "library" functions can be added to the language. It is intentionally designed that each instance of [TangBase](#) will have its own library functions.
2. It provides methods to compile scripts and templates, resulting in a Program object.
3. The Program object may then be executed, providing instance-specific context information (*i.e.*, state).

5.6.2 Constructor & Destructor Documentation

5.6.2.1 TangBase()

```
Tang::TangBase::TangBase ( )
```

The constructor.

Isn't it glorious.

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

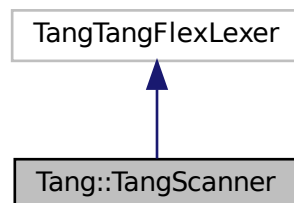
- include/tangBase.hpp
- src/tangBase.cpp

5.7 Tang::TangScanner Class Reference

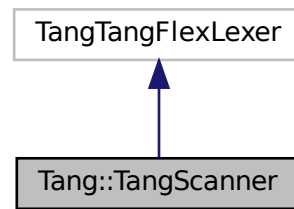
The Flex lexer class for the main Tang language.

```
#include <tangScanner.hpp>
```

Inheritance diagram for Tang::TangScanner:



Collaboration diagram for Tang::TangScanner:



Public Member Functions

- [TangScanner](#) (std::istream &arg_yyin, std::ostream &arg_yyout)
The constructor for the Scanner.
- virtual Tang::TangParser::symbol_type [get_next_token](#) ()
A pass-through function that we supply so that we can provide a Bison 3 token return type instead of the `int` that is returned by the default class configuration.

5.7.1 Detailed Description

The Flex lexer class for the main Tang language.

Flex requires that our lexer class inherit from yyFlexLexer, an "intermediate" class whose real name is "TangTangFlexLexer". We are subclassing it so that we can override the return type of [get_next_token\(\)](#), for compatibility with Bison 3 tokens.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 TangScanner()

```
Tang::TangScanner::TangScanner (
    std::istream & arg_yyin,
    std::ostream & arg_yyout ) [inline]
```

The constructor for the Scanner.

The design of the Flex lexer is to tokenize the contents of an input stream, and to write any error messages to an output stream. In our implementation, however, errors are returned differently, so the output stream is never used. It's presence is retained, however, in case it is needed in the future.

For now, the general approach should be to supply the input as a string stream, and to use std::cout as the output.

Parameters

<i>arg_yyin</i>	The input stream to be tokenized
<i>arg_yyout</i>	The output stream (not currently used)

5.7.3 Member Function Documentation

5.7.3.1 get_next_token()

```
virtual Tang::TangParser::symbol_type Tang::TangScanner::get_next_token ( ) [virtual]
```

A pass-through function that we supply so that we can provide a Bison 3 token return type instead of the `int` that is returned by the default class configuration.

Returns

A Bison 3 token representing the lexeme that was recognized.

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

- include/tangScanner.hpp

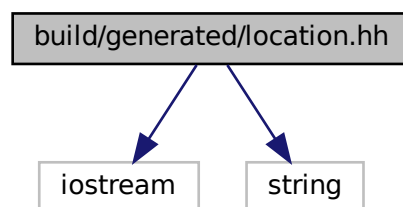
Chapter 6

File Documentation

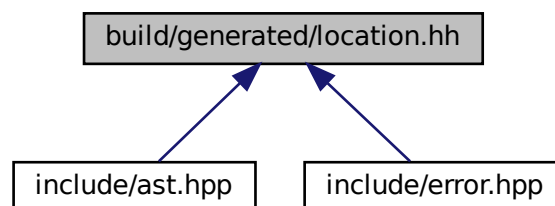
6.1 build/generated/location.hh File Reference

Define the Tang ::location class.

```
#include <iostream>
#include <string>
Include dependency graph for location.hh:
```



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



Classes

- class [Tang::position](#)
A point in a source file.
- class [Tang::location](#)
Two points in a source file.

Macros

- `#define YY_NULLPTR ((void*)0)`

Functions

- `position & Tang::operator+= (position &res, position::counter_type width)`
Add width columns, in place.
- `position Tang::operator+ (position res, position::counter_type width)`
Add width columns.
- `position & Tang::operator-= (position &res, position::counter_type width)`
Subtract width columns, in place.
- `position Tang::operator- (position res, position::counter_type width)`
Subtract width columns.
- `template<typename YYChar > std::basic_ostream< YYChar > & Tang::operator<< (std::basic_ostream< YYChar > &ostr, const position &pos)`
Intercept output stream redirection.
- `location & Tang::operator+= (location &res, const location &end)`
Join two locations, in place.
- `location Tang::operator+ (location res, const location &end)`
Join two locations.
- `location & Tang::operator+= (location &res, location::counter_type width)`
Add width columns to the end position, in place.
- `location Tang::operator+ (location res, location::counter_type width)`
Add width columns to the end position.
- `location & Tang::operator-= (location &res, location::counter_type width)`
Subtract width columns to the end position, in place.
- `location Tang::operator- (location res, location::counter_type width)`
Subtract width columns to the end position.
- `template<typename YYChar > std::basic_ostream< YYChar > & Tang::operator<< (std::basic_ostream< YYChar > &ostr, const location &loc)`
Intercept output stream redirection.

6.1.1 Detailed Description

Define the Tang ::location class.

6.1.2 Function Documentation

6.1.2.1 operator<<() [1/2]

```
template<typename YYChar >
std::basic_ostream<YYChar>& Tang::operator<< (
    std::basic_ostream< YYChar > & ostr,
    const location & loc )
```

Intercept output stream redirection.

Parameters

<i>ostr</i>	the destination output stream
<i>loc</i>	a reference to the location to redirect

Avoid duplicate information.

6.1.2.2 operator<<() [2/2]

```
template<typename YYChar >
std::basic_ostream<YYChar>& Tang::operator<< (
    std::basic_ostream< YYChar > & ostr,
    const position & pos )
```

Intercept output stream redirection.

Parameters

<i>ostr</i>	the destination output stream
<i>pos</i>	a reference to the position to redirect

Index

- AstNode
 - Tang::AstNode, [10](#)
- AstNodeInteger
 - Tang::AstNodeInteger, [11](#)
- build/generated/location.hh, [21](#)
- Error
 - Tang::Error, [13](#)
- get_next_token
 - Tang::TangScanner, [19](#)
- location.hh
 - operator<<, [22](#), [23](#)
- operator<<
 - location.hh, [22](#), [23](#)
- Tang::AstNode, [9](#)
 - AstNode, [10](#)
- Tang::AstNodeInteger, [10](#)
 - AstNodeInteger, [11](#)
- Tang::Error, [12](#)
 - Error, [13](#)
- Tang::location, [13](#)
- Tang::position, [15](#)
- Tang::TangBase, [16](#)
 - TangBase, [17](#)
- Tang::TangScanner, [17](#)
 - get_next_token, [19](#)
 - TangScanner, [18](#)
- TangBase
 - Tang::TangBase, [17](#)
- TangScanner
 - Tang::TangScanner, [18](#)