Util

0.1

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

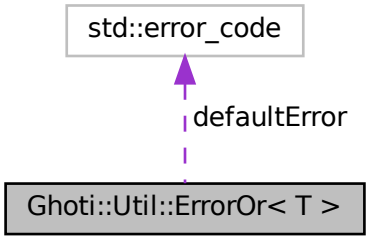
Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Ghoti::Util::ErrorOr< T > Class Template Reference

Collaboration diagram for Ghoti::Util::ErrorOr< T >:



Public Member Functions

- ErrorOr ()

  Default constructor.
- ErrorOr (const T &val)

  Will create an object containing the supplied `T` value.
- ErrorOr (const std::error_code &ec)

  Will create an object containing the supplied `std::error_code`.
- const std::error_code & error_code () const noexcept

  Return the `std::error_code` contained in the object.
- const T & operator* () const

  Return the value as type `T`.
- operator bool () const noexcept

  Indicate whether or not the non-error type is set.

Private Attributes

- std::variant< std::error_code, T > value

  The actual value held by this object.

Static Private Attributes

- static const std::error_code defaultError = {}

  Static helper which holds a default-constructed `std::error_code`, for internal use.

3.1.1 Constructor & Destructor Documentation

3.1.1.1 ErrorOr() [1/3]

```
template<typename T >
Ghoti::Util::ErrorOr< T >::ErrorOr ( )  [inline]
```

Default constructor.

Will create an object containing the default value of type `T`.

3.1.1.2 ErrorOr() [2/3]

```
template<typename T >
Ghoti::Util::ErrorOr< T >::ErrorOr (
            const T & val )  [inline]
```

Will create an object containing the supplied `T` value.

Parameters

val	The `T` type value with which to initialize the object.

3.1.1.3 ErrorOr() [3/3]

```
template<typename T >
Ghoti::Util::ErrorOr< T >::ErrorOr (
            const std::error_code & ec )  [inline]
```

Will create an object containing the supplied `std::error_code`.

Parameters

ec	The `std::error_code` with which to initialize the object.

3.1.2 Member Function Documentation

3.1.2.1 error_code()

```
template<typename T >
const std::error_code& Ghoti::Util::ErrorOr< T >::error_code ( ) const  [inline], [noexcept]
```

Return the `std::error_code` contained in the object.

If no error code is present, then a default-constructed error code will be returned.

Returns

The `std::error_code` held by the object.

3.1.2.2 operator bool()

```
template<typename T >
Ghoti::Util::ErrorOr< T >::operator bool ( ) const  [inline], [noexcept]
```

Indicate whether or not the non-error type is set.

Returns

`false` if value is a `std::error_code`, `true` otherwise.

3.1.2.3 operator*()

```
template<typename T >
const T& Ghoti::Util::ErrorOr< T >::operator* ( ) const  [inline]
```

Return the value as type `T`.

It is up to the programmer to verify that the object does not hold a `std::error_code`.

Returns

The type-`T` value contained in the object.

3.1.3 Member Data Documentation

3.1.3.1 defaultError

```
template<typename T >
const std::error_code Ghoti::Util::ErrorOr< T >::defaultError = {}  [inline], [static], [private]
```

Static helper which holds a default-constructed `std::error_code`, for internal use.

The "internal use" is that this allows the `error_code()` function to be `noexcept` by returning this default error if no error is set.

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

- include/util/errorOr.hpp

3.2 std::hash< Ghoti::shared_string_view > Struct Reference

Hashing function, consistent with `std::string_view`.

`#include <shared_string_view.hpp>`

Public Member Functions

- std::size_t **operator()** (const Ghoti::shared_string_view &ssv) const noexcept

3.2.1 Detailed Description

Hashing function, consistent with `std::string_view`.

Parameters

ssv	The `shared_string_view` to be hashed.

Returns

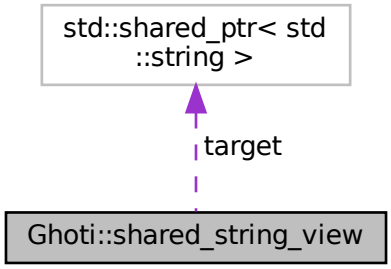
The hashed value.

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

- include/util/shared_string_view.hpp
- src/shared_string_view.cpp

3.3 Ghoti::shared_string_view Class Reference

Collaboration diagram for Ghoti::shared_string_view:



Public Member Functions

- shared_string_view ()

  Default Constructor.
- shared_string_view (const char *s)

  Constructor.
- shared_string_view (const char *s, size_t length)

  Constructor.
- shared_string_view (const std::string &s)

  Constructor.
- operator std::string_view () const

  Provides a string_view variant of the shared_string_view object.
- size_t length () const

  Return the length of the string represented by the view.
- shared_string_view substr (size_t offset, size_t length)

  Calculate a substring based on the current shared_string_view.
- bool operator== (const Ghoti::shared_string_view &ssv) const

  Compare two shared_string_view objects.
- std::weak_ordering operator<=> (const Ghoti::shared_string_view &ssv) const

  Perform a three-way comparison on two shared_string_view objects.
- shared_string_view & operator+= (const Ghoti::shared_string_view &rhs)

  Perform a concatenation of the supplied string to the existing string view object and apply it to the existing string view object.
- shared_string_view & operator+= (char rhs)

  Perform a concatenation of the supplied character to the existing string view object and apply it to the existing string view object.
- shared_string_view operator+ (const Ghoti::shared_string_view &rhs) const

  Perform a concatenation of the supplied string to the existing string view object.
- std::string_view::const_iterator begin () const

  Provide an iterator from the beginning of the view.
- std::string_view::const_iterator end () const

Provide an iterator pointing to the end of the view.

- std::string_view::const_reverse_iterator rbegin () const

  Provide a reverse iterator from the end of the view.

- std::string_view::const_reverse_iterator rend () const

  Provide a reverse iterator pointing to the beginning of the view.

- char operator[ ] (size_t pos) const

  Perform an index operation into the target string.

Private Member Functions

- shared_string_view (bool)

  Private constructor.

Private Attributes

- std::shared_ptr< std::string > target

  The shared target string pointed to by this object.

- size_t start

  The offset into the shared target string at which this view begins.

- size_t len

  The length of the view.

3.3.1 Constructor & Destructor Documentation

3.3.1.1 shared_string_view() [1/5]

```
shared_string_view::shared_string_view ( )
```

Default Constructor.

Initializes using an empty string.

3.3.1.2 shared_string_view() [2/5]

```
shared_string_view::shared_string_view (
            const char * s )
```

Constructor.

Parameters

s	A C-string used to construct the string.

3.3.1.3 shared_string_view() [3/5]

```
shared_string_view::shared_string_view (
            const char * s,
            size_t length )
```

Constructor.

Parameters

s	A C-string used to construct the string.
length	The length of the C-string.

3.3.1.4 shared_string_view() [4/5]

```
shared_string_view::shared_string_view (
            const std::string & s )
```

Constructor.

Parameters

s	A String object used to construct the string.

3.3.1.5 shared_string_view() [5/5]

```
Ghoti::shared_string_view::shared_string_view (
            bool  )  [explicit], [private]
```

Private constructor.

This constructor is private because it will create an object whose target is not initialized, which should not be done in general.

Parameters

ignore	Only used to denote that this constructor is intentionally being called.

3.3.2 Member Function Documentation

3.3.2.1 begin()

```
string_view::const_iterator shared_string_view::begin ( ) const
```

Provide an iterator from the beginning of the view.

Returns

A forward iterator.

3.3.2.2 end()

```
string_view::const_iterator shared_string_view::end ( ) const
```

Provide an iterator pointing to the end of the view.

Returns

An ending iterator.

3.3.2.3 length()

```
size_t shared_string_view::length ( ) const
```

Return the length of the string represented by the view.

The shared string may be longer, but this is the length of the substring that this view represents.

Returns

The length of the string represented by the view.

3.3.2.4 operator std::string_view()

```
Ghoti::shared_string_view::operator std::string_view ( ) const
```

Provides a string_view variant of the shared_string_view object.

It is up to the programmer to ensure that the shared_string_view object remains in scope while the string_view is in use.

3.3.2.5 operator+()

```
shared_string_view shared_string_view::operator+ (
            const Ghoti::shared_string_view & rhs ) const
```

Perform a concatenation of the supplied string to the existing string view object.

Return a new string view.

Parameters

rhs	A string to be appended to the shared_string_view object.

Returns

The new shared_string_view resulting from the concatenation.

3.3.2.6 operator+=() [1/2]

```
shared_string_view & shared_string_view::operator+= (
            char rhs )
```

Perform a concatenation of the supplied character to the existing string view object and apply it to the existing string view object.

If the target string can be appended to safely, then that will be done. Otherwise, a new internal string will be created.

Because this may modify the target string, all previously-provided std::string_view references will be invalidated. This is similar to the behavior of std::string.cstr(), in which modifying the string will invalidate the c-string pointer.

Parameters

rhs	A character to be appended to the shared_string_view object.

Returns

The amended shared_string_view resulting from the concatenation.

3.3.2.7 operator+=() [2/2]

```
shared_string_view & shared_string_view::operator+= (
            const Ghoti::shared_string_view & rhs )
```

Perform a concatenation of the supplied string to the existing string view object and apply it to the existing string view object.

If the target string can be appended to safely, then that will be done. Otherwise, a new internal string will be created.

Because this may modify the target string, all previously-provided std::string_view references will be invalidated. This is similar to the behavior of std::string.cstr(), in which modifying the string will invalidate the c-string pointer.

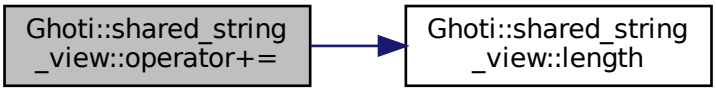
Parameters

rhs	A string to be appended to the shared_string_view object.

Returns

The amended shared_string_view resulting from the concatenation.

Here is the call graph for this function:



3.3.2.8 operator<=>()

```
std::weak_ordering shared_string_view::operator<=> (
            const Ghoti::shared_string_view & ssv ) const
```

Perform a three-way comparison on two shared_string_view objects.

Parameters

ssv	The right hand side operator.

Returns

A weak ordering indicator of the two objects.

3.3.2.9 operator==()

```
bool shared_string_view::operator== (
            const Ghoti::shared_string_view & ssv ) const
```

Compare two shared_string_view objects.

Parameters

ssv	The right hand side operator.

Returns

True if the objects have equivalent values, false otherwise.

3.3.2.10 operator[]()

```
char shared_string_view::operator[] (
            size_t pos ) const
```

Perform an index operation into the target string.

Parameters

pos	The 0-based index position.

Returns

The character at the position requested.

3.3.2.11 rbegin()

```
string_view::const_reverse_iterator shared_string_view::rbegin ( ) const
```

Provide a reverse iterator from the end of the view.

Returns

A reverse iterator.

3.3.2.12 rend()

```
string_view::const_reverse_iterator shared_string_view::rend ( ) const
```

Provide a reverse iterator pointing to the beginning of the view.

Returns

An reverse ending iterator.

3.3.2.13 substr()

```
shared_string_view shared_string_view::substr (
            size_t offset,
            size_t length )
```

Calculate a substring based on the current shared_string_view.

If the substring is out of range, then an empty view will be provided. If the substring length requested is greater than what is available, then the returned substring will contain as many characters as possible, within the limits of the parent string view range.

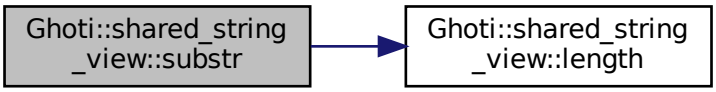
Parameters

offset	The 0-based offset from which the substring should start.
length	The length of the substring desired.

Returns

A new shared_string_view of the requested substring.

Here is the call graph for this function:



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

- include/util/shared_string_view.hpp
- src/shared_string_view.cpp

Chapter 4

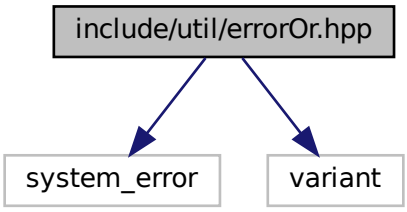
File Documentation

4.1 include/util/errorOr.hpp File Reference

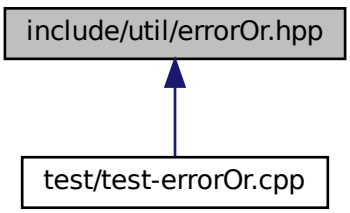
Header file containing the definitions of the ErrorOr class.

```
#include <system_error>
#include <variant>
```

Include dependency graph for errorOr.hpp:



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



Classes

- class Ghoti::Util::ErrorOr< T >

4.1.1 Detailed Description

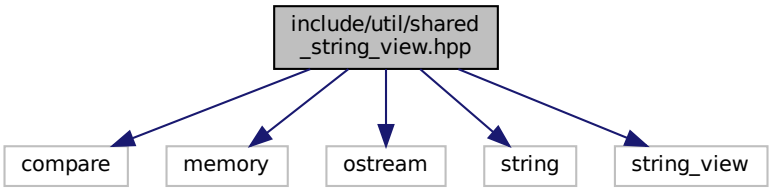
Header file containing the definitions of the ErrorOr class.

4.2 include/util/shared_string_view.hpp File Reference

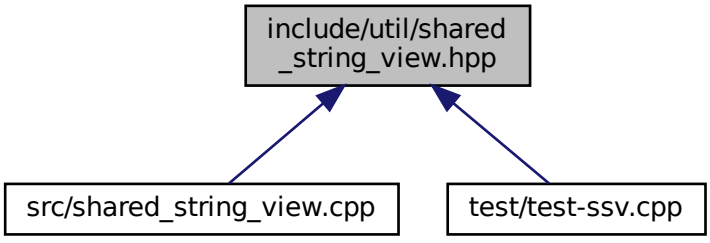
Header file containing the definitions of the shared_string_view class.

```
#include <compare>
#include <memory>
#include <ostream>
#include <string>
#include <string_view>
```

Include dependency graph for shared_string_view.hpp:



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



Classes

- class Ghoti::shared_string_view
- struct std::hash< Ghoti::shared_string_view >

  Hashing function, consistent with `std::string_view`.

Functions

- std::ostream & Ghoti::operator<< (std::ostream &out, const Ghoti::shared_string_view &ssv)

  Insertion operator.

4.2.1 Detailed Description

Header file containing the definitions of the shared_string_view class.

4.2.2 Function Documentation

4.2.2.1 operator<<()

```
ostream & Ghoti::operator<< (
            std::ostream & out,
            const Ghoti::shared_string_view & ssv )
```

Insertion operator.

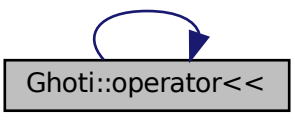
Parameters

out	The output stream to be written to.
ssv	The `shared_string_view` to be inserted into the stream.

Returns

The output stream.

Here is the call graph for this function:

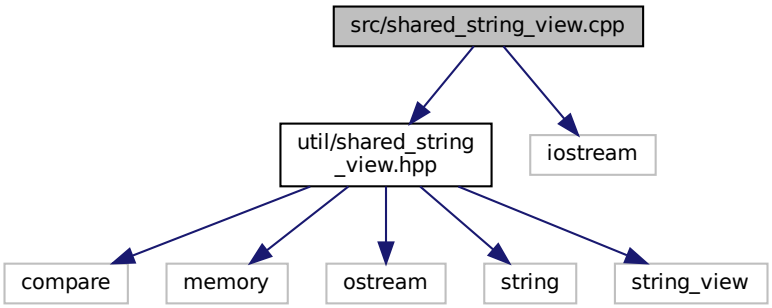


4.3 src/shared_string_view.cpp File Reference

Define the shared_string_view class.

```
#include "util/shared_string_view.hpp"
#include <iostream>
```

Include dependency graph for shared_string_view.cpp:



4.3.1 Detailed Description

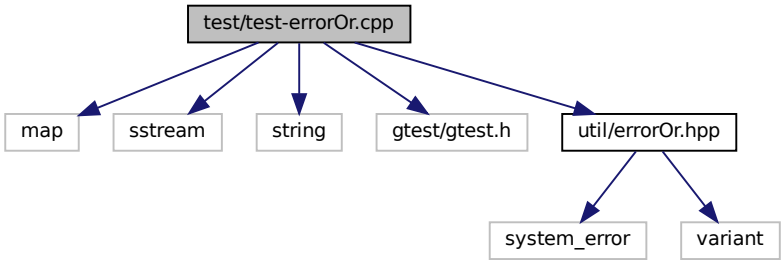
Define the shared_string_view class.

4.4 test/test-errorOr.cpp File Reference

Test the shared_string_view behavior.

```
#include <map>
#include <sstream>
#include <string>
#include <gtest/gtest.h>
#include "util/errorOr.hpp"
```

Include dependency graph for test-errorOr.cpp:



Functions

- **TEST** (ErrorOr, Constructor)
- int **main** (int argc, char **argv)

4.4.1 Detailed Description

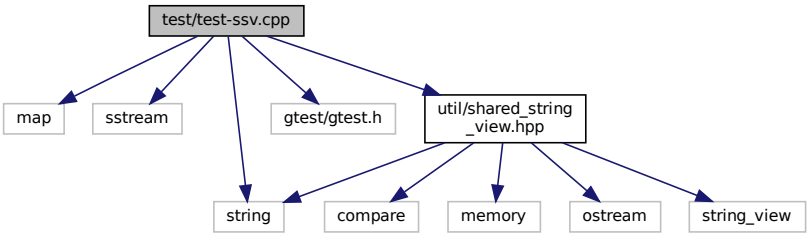
Test the shared_string_view behavior.

4.5 test/test-ssv.cpp File Reference

Test the shared_string_view behavior.

```
#include <map>
#include <sstream>
#include <string>
#include <gtest/gtest.h>
#include "util/shared_string_view.hpp"
```

Include dependency graph for test-ssv.cpp:



Functions

- **TEST** (Constructor, Length)
- **TEST** (Operator, string_stream)
- **TEST** (Operator, Extraction)
- **TEST** (Operator, ThreeWayComparison)
- **TEST** (Operator, PlusEqual)
- **TEST** (Operator, Plus)
- **TEST** (Method, substr)
- **TEST** (Method, ForwardIterator)
- **TEST** (Method, ReverseIterator)
- **TEST** (Method, Index)
- **TEST** (Aux, Hash)
- int **main** (int argc, char **argv)

4.5.1 Detailed Description

Test the shared_string_view behavior.

Index