Symposium: the collaboration place

Generated by Doxygen 1.8.15

Chapter 1

Main Page

Symposium: the collaboration place

Summary

Symposium is an free, real-time, collaborative text editor written in C++ language. Like in the ancient Greece, Symposium allows you to collaborate and exchange ideas with other people in a environment that suits you.

This editor supports the basic functions that a rich text editor usually offers. It's possible create lists, decide the alignment of a paragraph, edit font and style of each font and use buttons and shortcuts for common cut-copy-paste operations. It is also possible to save the document in format pdf.

One of Symposium's strengths is the sophisticated management of sharing options and privileges that each user has over a document. The following modes are supported sharing:

- **Sharing without limits**: a link to the document is activated and remains valid as long as is not explicitly deactivated;
- **Timer sharing**: the link is activated until a certain time, past the which other users, even if in possession of the link, will not be able to add the document between those they have access to.
- **Counter-controlled sharing**: in this mode only predetermined number of users who will take access to the document will be accepted.

Each user can have a different privilege on a document, and the choice of which privilege grant can be decided by the owner when sharing the document. The privileges supported are:

- **Owner**: by default, the creator of the document is the first owner. An owner can edit the document in all its parts, including sharing options and the privileges of others users;
- **Modifier**: can modify every aspect of the document, but not the sharing options neither user privileges associated with the document;
- **Reader**: can only view the document, user cannot modify it. Continue to see real time changes and other users' cursors move around.

It's developed on purpose for an academic project, but it is intended to remain free for your contribution. Symposium is a client-server software that uses **Conflict-Free Replicated Data Types** (CRDT) to make sure all users stay in-sync.

Getting Started

To build and run the project make sure you have the necessary components present in your system.

If you want to proceed with `CMakeList file`, there are 2 main targets: one for the server (SymposiumServer) and one for the client (SymposiumGui).

If you want to proceed with .pro file, there are 2 project in the `directory`: one for the server (`Symposium↩ Gui_Server.pro`) and one for the client (`SymposiumGui.pro`)

Prerequisites

You need the following components to be able to install and run the program:

- boost libraries;
- Qt libraries (version 5.12.3);
- C++ environment (c++17);

In particular we use:

- `Qt Creator`;
- `Clion`;
- `Mingw Distro compiler for boost serialization`

Authors

- **Riccardo Zaccone** - `RickZack`
- **Ksenia Del Conte Akimova** - `KseniaDelConte`
- **Cristian Gianetto** - `addocm`
- **Martina Bellissimo** - `martinaBellissimo`

License

Symposium is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. See the COPYING file for details.

Chapter 2

Namespace Index

2.1 Namespace List

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

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Namespace Documentation

5.1 Symposium Namespace Reference

Contains the forward declaration of classes used in the original Symposium software.

Classes

- class AccessStrategy

  Defines how the permissions on objects of type are handled.
- class askResMessage

  class used to model a message sent by a client asking for a resource
- class clientdispatcherException
- class clientMessage

  class used to model a message sent by a client
- struct Color

  Represent a color in RGB encoding.
- class colorGen

  Color generator for users in Symposium system as functor.
- class cursorMessage

  class used to model a message to update the position of a user's cursor
- class directory

  class used to model a directory, uses Singleton pattern
- class document
- class documentException
- class editLineStyleMessage
- class file

  class used to model a file in the filesystem
- interface filesystem

  class used as interface for a filesystem, made using Composite pattern
- class filesystemException
- struct format
- class loginMessage

  class used to model a message sent by a server as answer to a login message
- class mapMessage

  class used to model an answer message sent by a server for a clientMessage

- class message
- class messageException
- class privMessage

  class used to model a privilege change message sent by a client. It is also used to propagate such a change on other clients
- class RMOAccess

  class used to model a ReadModifyOwn privilege handling on a resource.
- class sendResMessage

  class used to model an answer message sent by a server for a askResMessage
- class serverMessage

  class used to model a message sent by the server
- struct sessionData
- class signUpMessage

  class used to model a sign up message sent by a client
- class symbol
- class symbolMessage

  class used to model a message regarding a symbol
- class SymClient

  class used to model a client of Symposium system
- class SymClientException
- class symlink

  class used to model a pointer to an object of class file
- class SymposiumException

  class used as exception base class for classes in Symposium namespace. It composes a custom error explaining message in the form "function: [functionName], in file: [filename], line: [lineNumber]: [errorMessage]", allowing subclasses to specify an error message. @exceptsafe no-throw
- class SymServer
- class SymServerException

  models an exception occurred in the context of SymServer class @exceptsafe no-throw
- class TrivialAccess

  class used to model the absence of privilege handling on a resource
- class updateActiveMessage

  class used to model the joining of an user to a document
- class updateDocMessage

  class used to model a message sent by a client to close a resource
- class uri

  class used to model resource sharing preferences
- class uriMessage

  class used to model a message for sharing a document
- class user

  class used to model a user of the system
- class userDataMessage

  class used to model a message to change the parameters of a user
- class userException

Enumerations

- enum msgOutcome : char { **failure**, **success** }

  defines the possible outcomes of an operation followed by a message
- enum msgType {
  msgType::registration, msgType::login, msgType::changeUserData, msgType::changeUserPwd, msgType::removeUser, **logout**, msgType::createRes, msgType::createNewDir, msgType::openRes, msgType::openNewRes, msgType::changeResName, msgType::removeRes, msgType::mapChangesToUser, msgType::changePrivileges, msgType::shareRes, msgType::insertSymbol, msgType::removeSymbol, msgType::addActiveUser, msgType::removeActiveUser, msgType::closeRes, msgType::updateCursor, msgType::editLineStyle }

  defines the possible type of messages, that corresponds to an action on server or client
- enum privilege : char { **none**, **readOnly**, **modify**, **owner** }

  defines the possible privileges on a resource
- enum resourceType : char { **directory**, **file**, **symlink** }

  defines the type of a filesystem object
- enum alignType { alignType::left, alignType::right, alignType::center, alignType::justify }
- enum uriPolicy { **inactive**, **activeAlways**, **activeCount**, **activeTimer** }

  defines the policy on an object of class uri

Functions

- std::ostream & operator<< (std::ostream &output, privilege priv)

  output operator for privilege
- privilege & operator-- (privilege &oldPriv)

  pre-decrement operator for privilege
- privilege & operator++ (privilege &oldPriv)

  pre-increment operator for privilege
- privilege operator-- (privilege &oldPriv, int)

  post-decrement operator for privilege
- privilege operator++ (privilege &oldPriv, int)

  post-increment operator for privilege
- bool operator== (const privilege a, const privilege b)

  operator == overload for privilege
- bool operator> (const privilege a, const privilege b)

  operator > overload for privilege
- bool operator< (const privilege a, const privilege b)

  operator < overload for privilege
- bool operator<= (const privilege a, const privilege b)

  operator == overload for privilege
- std::ostream & operator<< (std::ostream &output, resourceType type)

  output operator for resType
- constexpr bool **operator**< (const std::tuple< int, int, Color > &lhs, const std::tuple< int, int, Color > &rhs)

5.1.1 Detailed Description

Contains the forward declaration of classes used in the original Symposium software.

class used to model a server of Symposium system

This namespace is used as a common place where to find the names used in the original project. It's intended to avoid conflicts that can be made if you want to user other classes in the context of Symposium, and to allow you to extend the project without the need to choose names unique in the global context

Handles all the data and actions that are required to a Symposium server. For all of the received client's requestes, the server answer sending an object of class serverMessage (or derivates) with the field *result* set to msgOutcome↩ ::success or msgOutcome::failure, indicating the action's outcome. The client should confirm the action only if it has received a positive outcome.

5.1.2 Enumeration Type Documentation

5.1.2.1 alignType

```
enum Symposium::alignType  [strong]
```

Enumerator

left	to declare that the row has a left alignment
right	to declare that the row has a right alignment
center	to declare that the row has a center alignment
justify	to declare that the row has a justified alignment

5.1.2.2 msgType

```
enum Symposium::msgType  [strong]
```

defines the possible type of messages, that corresponds to an action on server or client

Enumerator

registration	used by client, when asking for a new user registration, see signUpMessage
login	used by server and client, when asking or answering for login of existing user, see clientMessage, loginMessage
changeUserData	used by server and client, when asking for changing the data of an user, see userDataMessage
changeUserPwd	used by server and client, when asking for changing the data of an user, see userDataMessage
removeUser	used by client, when asking to remove the user it is logged with, see clientMessage
createRes	used by server and client, when asking for creating a new document, see sendResMessage, askResMessage
createNewDir	used by server and client, when asking for creating a new directory, see sendResMessage, askResMessage
openRes	user by client, when asking for opening a document that is already in its filesystem,; see askResMessage
openNewRes	used by server and client, when asking for an existing resource via URI, sendResMessage, askResMessage
changeResName	used by server and client, when asking for changing a resource's name, see serverMessage, askResMessage
removeRes	used by client, when asking for a resource removal, serverMessage, askResMessage
mapChangesToUser	used by server and client, when asking for a siteId->username mapping, see mapMessage, updateDocMessage
changePrivileges	used by server and client, when asking for updating or propagating privilege changes, see privMessage
shareRes	used by server and client, when asking for updating or propagating sharing changes, see uriMessage

Enumerator

insertSymbol	used by server and client, when asking for updating or propagating the insertion, see symbolMessage
removeSymbol	used by server and client, when asking for updating or propagating the deletion, see symbolMessage
addActiveUser	used by server, for propagating the document opening of an user, see updateActiveMessage
removeActiveUser	used by server, for propagating the document closing of an user, see updateActiveMessage
closeRes	used by client, when a user wants to close a document, see updateDocMessage
updateCursor	used by client, when a user changes the cursor's position, see cursorMessage
editLineStyle	used by client, when changing alignment and/or index style of a paragraph, see editLineStyleMessage

Chapter 6

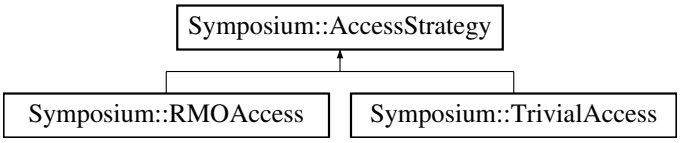
Class Documentation

6.1 Symposium::AccessStrategy Class Reference

Defines how the permissions on objects of type are handled.

`#include <AccessStrategy.h>`

Inheritance diagram for Symposium::AccessStrategy:



Public Member Functions

- virtual bool validateAction (const std::string &targetUser, privilege requested) const =0
  validate an action from user targetUser that requires requested
- virtual privilege setPrivilege (const std::string &targetUser, privilege toGrant)=0
  set the privilege of an user
- virtual privilege **getPrivilege** (const std::string &targetUser) const =0
- virtual std::unordered_map< std::string, privilege > **getPermission** () const =0
- virtual bool **moreOwner** (std::string username) const =0

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &, const unsigned int)

Friends

- class **boost::serialization::access**

6.1.1 Detailed Description

Defines how the permissions on objects of type are handled.

6.1.2 Member Function Documentation

6.1.2.1 setPrivilege()

```
virtual privilege Symposium::AccessStrategy::setPrivilege (
            const std::string & targetUser,
            privilege toGrant )  [pure virtual]
```

set the privilege of an user

Parameters

targetUser	the user the privilege is to be granted
toGrant	the privilege to grant to targetUser

Returns

the privilege previously owned by targetUser, none if no privilege previously owned

Implemented in Symposium::TrivialAccess, and Symposium::RMOAccess.

6.1.2.2 validateAction()

```
virtual bool Symposium::AccessStrategy::validateAction (
            const std::string & targetUser,
            privilege requested ) const  [pure virtual]
```

validate an action from user targetUser that requires requested

Parameters

targetUser	the user who is doing the action
requested	the permission requested by the action

Returns

true if the user is granted the privilege requested

Implemented in Symposium::TrivialAccess, and Symposium::RMOAccess.

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

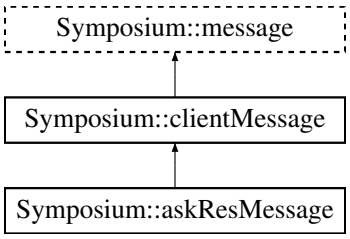
- AccessStrategy.h

6.2 Symposium::askResMessage Class Reference

class used to model a message sent by a client asking for a resource

`#include <message.h>`

Inheritance diagram for Symposium::askResMessage:



Public Member Functions

- askResMessage (msgType action, const std::pair< std::string, std::string > &actionOwner, const std::string &path, const std::string &name, const std::string &resourceId="", privilege accessMode=uri::getDefault↩ Privilege(), uint_positive_cnt::type msgId=0)
- void invokeMethod (SymServer &server) override

  perform an action regarding a resource for the user actionOwner
- void completeAction (SymClient &client, msgOutcome serverResult) override

  completes an action for which the client asked to the server
- bool **operator==** (const askResMessage &rhs) const
- bool **operator!=** (const askResMessage &rhs) const
- const std::string & **getPath** () const
- const std::string & **getName** () const
- const std::string & **getResourceId** () const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Private Attributes

- std::string path
- std::string name
- std::string resourceId
- privilege accessMode

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.2.1 Detailed Description

class used to model a message sent by a client asking for a resource

An instance of this type is use to ask to server to create a resource in *path* with name *name* or to remove a resource named *name* from *path* or to open a resource already known. It is used also to rename an existing resource. This depends on the *action*. If *action* is "openNewRes" then *resourceId* contains the path to the resource (the uri). If *action* is "changeResName" then *resourceId* is the new file name.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 askResMessage()

```
askResMessage::askResMessage (
            msgType action,
            const std::pair< std::string, std::string > & actionOwner,
            const std::string & path,
            const std::string & name,
            const std::string & resourceId = "",
            privilege accessMode = uri::getDefaultPrivilege(),
            uint_positive_cnt::type msgId = 0 )
```

Exceptions

messageException	if *action* is not consistent with the message type

6.2.3 Member Function Documentation

6.2.3.1 completeAction()

```
void askResMessage::completeAction (
            SymClient & client,
            msgOutcome serverResult )  [override], [virtual]
```

completes an action for which the client asked to the server

Parameters

client	the same client that had originated the clientMessage

Some actions on SymClient require to ask to the server to assure that the action is valid and to propagate the change on the server. For such actions, only if the outcome from the server is positive the action can be actually done.

Reimplemented from Symposium::clientMessage.

6.2.3.2 invokeMethod()

```
void askResMessage::invokeMethod (
            SymServer & server )  [override], [virtual]
```

perform an action regarding a resource for the user *actionOwner*

Parameters

server	the server to whom the user is already registered and logged in

Depending on the value of *action*, the *invokeMethod* ask for different actions on the server:

- *action=msgType::createRes* : calls SymServer::createNewSource on *server*. A message of type sendResMessage is sent back to the client, containing the file object created by the server.
- *action=msgType::openRes* : calls SymServer::openSource on *server*. A message of type sendResMessage is sent back to the client, containing the file object stored by the server.
- *action=msgType::openNewRes* : calls SymServer::openNewSource on *server*. A message of type sendResMessage is sent back to the client, containing the symlink object created by the server.
- *action=msgType::changeResName* : calls SymServer::renameResource on *server*. A message of type serveMessage is sent back to the client, to indicate whether the action succeeded or not.
- *action=msgType::createNewDir* : calls SymServer::createNewDir on *server*. A message of type sendResMessage is sent back to the client, containing the directory object created by the server.
- *action=msgType::removeRes* : calls SymServer::removeResource on *server*. A message of type serve↩ Message is sent back to the client, to indicate whether the action succeeded or not.

Reimplemented from Symposium::clientMessage.

6.2.4 Member Data Documentation

6.2.4.1 accessMode

```
privilege Symposium::askResMessage::accessMode  [private]
```

the privilege requested opening the resource

6.2.4.2 name

```
std::string Symposium::askResMessage::name  [private]
```

name to assign to the resource

6.2.4.3 path

```
std::string Symposium::askResMessage::path  [private]
```

path where to put the resource

6.2.4.4 resourceId

```
std::string Symposium::askResMessage::resourceId  [private]
```

when a sharing request is made, contains the path to the resource (the uri).

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

- message.h
- message.cpp

6.3 Symposium::detail::basic_counter< T, N, E, > Class Template Reference

Classes

- struct forward_iterator
- struct iterator
- struct reverse_iterator

Public Types

- typedef T **type**

Public Member Functions

- basic_counter & **operator++** ()
- basic_counter **operator++** (int)
- template<typename T2 , T2 N2>
  basic_counter< typename std::common_type< T, T2 >::type, N > & **operator+=** (const basic_counter< T2, N2 > &rhs)
- template<typename X , std::enable_if_t< type_not_narrow< X, T >::value, int > = 0>
  basic_counter< typename std::common_type< T, X >::type, N > & **operator+=** (const X rhs)
- template<typename T2 , T2 N2>
  basic_counter< typename std::common_type< T, T2 >::type, N > & **operator-=** (const basic_counter< T2, N2 > &rhs)
- template<typename X , std::enable_if_t< type_not_narrow< X, T >::value, int > = 0>
  basic_counter< typename std::common_type< T, X >::type, N > & **operator-=** (const X rhs)
- basic_counter & **operator--** ()
- basic_counter **operator--** (int)
- void **reset** ()
- **operator T** () const
- forward_iterator **begin** ()
- forward_iterator **end** ()
- reverse_iterator **rbegin** ()
- reverse_iterator **rend** ()

Static Public Attributes

- static constexpr T **start_val** =N
- static constexpr T **end_val** =E

Protected Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int)

Private Member Functions

- **basic_counter** (T initializer)

Private Attributes

- T **cnt**

Friends

- class **boost::serialization::access**
- template<typename T1 , T1 N1, typename T2 , T2 N2>
  basic_counter< typename std::common_type< T1, T2 >::type, N1 > **operator+** (const basic_counter< T1, N1 > &lhs, const basic_counter< T2, N2 > &rhs)
- template<typename X , typename U , U M, std::enable_if_t< type_not_narrow< X, U >::value, int > >
  basic_counter< U, M > **operator+** (const basic_counter< U, M > &lhs, const X step)
- template<typename T1 , T1 N1, typename T2 , T2 N2>
  basic_counter< typename std::common_type< T1, T2 >::type, N1 > **operator-** (const basic_counter< T1, N1 > &lhs, const basic_counter< T2, N2 > &rhs)
- template<typename X , typename U , U M, std::enable_if_t< type_not_narrow< X, U >::value, int > >
  basic_counter< U, M > **operator-** (const basic_counter< U, M > &lhs, const X step)

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

- Symposium.h

6.4 basic_counter< T, N, E, > Class Template Reference

Classes

- struct forward_iterator
- struct iterator
- struct reverse_iterator

Public Types

- typedef T **type**

Public Member Functions

- basic_counter & **operator++** ()
- basic_counter **operator++** (int)
- template<typename T2 , T2 N2>
  basic_counter< typename std::common_type< T, T2 >::type, N > & **operator+=** (const basic_counter< T2, N2 > &rhs)
- template<typename X , std::enable_if_t< type_not_narrow< X, T >::value, int > = 0>
  basic_counter< typename std::common_type< T, X >::type, N > & **operator+=** (const X rhs)
- template<typename T2 , T2 N2>
  basic_counter< typename std::common_type< T, T2 >::type, N > & **operator-=** (const basic_counter< T2, N2 > &rhs)
- template<typename X , std::enable_if_t< type_not_narrow< X, T >::value, int > = 0>
  basic_counter< typename std::common_type< T, X >::type, N > & **operator-=** (const X rhs)
- basic_counter & **operator--** ()
- basic_counter **operator--** (int)
- void **reset** ()
- **operator T** () const
- forward_iterator **begin** ()
- forward_iterator **end** ()
- reverse_iterator **rbegin** ()
- reverse_iterator **rend** ()

Static Public Attributes

- static constexpr T **start_val** =N
- static constexpr T **end_val** =E

Protected Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int)

Private Member Functions

- **basic_counter** (T initializer)

Private Attributes

- T **cnt**

Friends

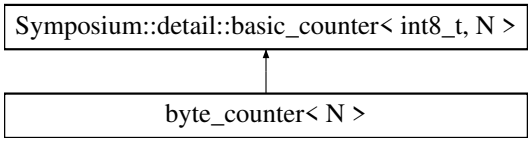
- class **boost::serialization::access**
- template<typename T1 , T1 N1, typename T2 , T2 N2>
  basic_counter< typename std::common_type< T1, T2 >::type, N1 > **operator+** (const basic_counter< T1, N1 > &lhs, const basic_counter< T2, N2 > &rhs)
- template<typename X , typename U , U M, std::enable_if_t< type_not_narrow< X, U >::value, int > >
  basic_counter< U, M > **operator+** (const basic_counter< U, M > &lhs, const X step)
- template<typename T1 , T1 N1, typename T2 , T2 N2>
  basic_counter< typename std::common_type< T1, T2 >::type, N1 > **operator-** (const basic_counter< T1, N1 > &lhs, const basic_counter< T2, N2 > &rhs)
- template<typename X , typename U , U M, std::enable_if_t< type_not_narrow< X, U >::value, int > >
  basic_counter< U, M > **operator-** (const basic_counter< U, M > &lhs, const X step)

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

- counter.h

6.5 byte_counter< N > Struct Template Reference

Inheritance diagram for byte_counter< N >:



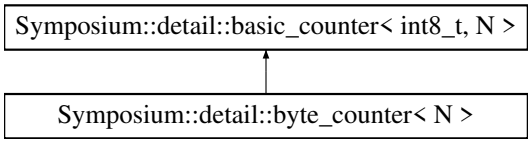
Additional Inherited Members

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

- counter.h

6.6 Symposium::detail::byte_counter< N > Struct Template Reference

Inheritance diagram for Symposium::detail::byte_counter< N >:



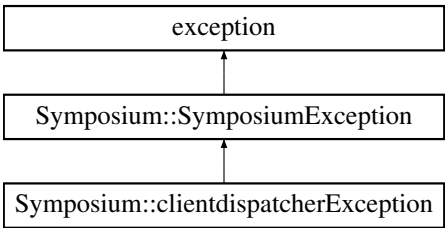
Additional Inherited Members

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

- Symposium.h

6.7 Symposium::clientdispatcherException Class Reference

Inheritance diagram for Symposium::clientdispatcherException:



Public Types

- enum clientdispatcherExceptionCodes { **UnknownMessageAction** =0, **UnknownClassOfMessage**, **Msg↩ ActionNotAllowed** }

  Specific error codes for clientdispatcherException. They are used as indexes to the error table string.

Public Member Functions

- **clientdispatcherException** (clientdispatcherExceptionCodes exceptionCode, const char ∗file, int line, const char ∗func)

Static Private Attributes

- static const char ∗ **clientdispatcherErrors** [ ] ={"The action of this message is not valid", "Unrecognized class of message", "The action of this message is not allowed"}

Additional Inherited Members

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

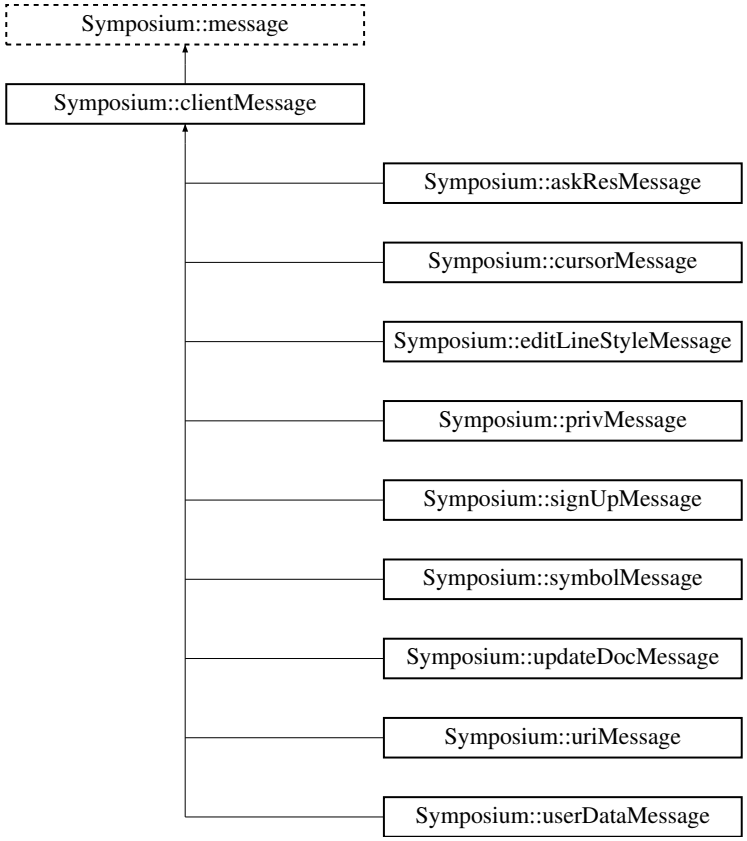
- SymposiumException.h
- SymposiumException.cpp

6.8 Symposium::clientMessage Class Reference

class used to model a message sent by a client

`#include <message.h>`

Inheritance diagram for Symposium::clientMessage:



Public Member Functions

- clientMessage (msgType action, const std::pair< std::string, std::string > &actionOwner, uint_positive_cnt↩ ::type msgId=0)
- const std::pair< std::string, std::string > & **getActionOwner** () const
- clientMessage & clearAuthParam ()

  clear the authentication parameters of the the user who sent this message
- virtual void invokeMethod (SymServer &server)

  make newUser an active user of the server the message is sent to
- virtual void completeAction (SymClient &client, msgOutcome serverResult)

  completes an action for which the client asked to the server
- bool **operator==** (const clientMessage &rhs) const
- bool **operator!=** (const clientMessage &rhs) const

Protected Member Functions

- **clientMessage** (const std::pair< std::string, std::string > &actionOwner, uint_positive_cnt::type msgId=0)

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int)

Private Attributes

- std::pair< std::string, std::string > actionOwner

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.8.1 Detailed Description

class used to model a message sent by a client

A message sent by a client performs an action on the server that has received it. An object of this class is used to log in the user indicated in *actionOwner* and to identify the user who asks for the *action*

6.8.2 Constructor & Destructor Documentation

6.8.2.1 clientMessage()

```
clientMessage::clientMessage (
            msgType action,
            const std::pair< std::string, std::string > & actionOwner,
            uint_positive_cnt::type msgId = 0 )
```

Exceptions

messageException	if *action* is not consistent with the message type

6.8.3 Member Function Documentation

6.8.3.1 clearAuthParam()

```
clientMessage & clientMessage::clearAuthParam ( )
```

clear the authentication parameters of the the user who sent this message

Returns

the message itself

This method is needed when the server has to forward the message to other clients, that mustn't know the authentication parameters of the user who sent this message in the first place

6.8.3.2 completeAction()

```
void clientMessage::completeAction (
            SymClient & client,
            msgOutcome serverResult )  [virtual]
```

completes an action for which the client asked to the server

Parameters

client	the same client that had originated the clientMessage

Some actions on SymClient require to ask to the server to assure that the action is valid and to propagate the change on the server. For such actions, only if the outcome from the server is positive the action can be actually done.

Reimplemented in Symposium::editLineStyleMessage, Symposium::userDataMessage, Symposium::uriMessage, Symposium::symbolMessage, Symposium::privMessage, and Symposium::askResMessage.

6.8.3.3 invokeMethod()

```
void clientMessage::invokeMethod (
            SymServer & server )  [virtual]
```

make *newUser* an active user of the server the message is sent to

Parameters

server	the server to login to

Depending on the value of *action*, the *invokeMethod* ask for different actions on the server:

- *action=msgType::login* : calls SymServer::login on *server*. A message of type loginMessage is sent back to the client, containing the user object retrieved by the server.
- *action=msgType::removeUser* : calls SymServer::removeUser on *server*. A message of type serverMessage is sent back to the client, containing the outcome of the action.
- *action=msgType::logout* : calls SymServer::logout on *server*. A message of type serverMessage is sent back to the client, containing the outcome of the action.

Reimplemented in Symposium::editLineStyleMessage, Symposium::cursorMessage, Symposium::userDataMessage, Symposium::uriMessage, Symposium::symbolMessage, Symposium::privMessage, Symposium::updateDocMessage, Symposium::signUpMessage, and Symposium::askResMessage.

6.8.4 Member Data Documentation

6.8.4.1 actionOwner

```
std::pair<std::string, std::string> Symposium::clientMessage::actionOwner  [private]
```

Defines the user (username, password) that has just performed the action

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

- message.h
- message.cpp

6.9 Symposium::Color Struct Reference

Represent a color in RGB encoding.

`#include <Color.h>`

Public Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int version)
- **Color** (uint8_t r, uint8_t g, uint8_t b)
- std::tuple< uint8_t, uint8_t, uint8_t > **getRgb** () const
- std::string **rgb_hex_string** ()
- template<class T >
  operator T () const
  conversion operator from Color to another type
- bool **operator==** (const Color &rhs) const
- bool **operator!=** (const Color &rhs) const

Public Attributes

- uint8_t r
- uint8_t g
- uint8_t b

6.9.1 Detailed Description

Represent a color in RGB encoding.

6.9.2 Member Function Documentation

6.9.2.1 operator T()

```
template<class T >
Symposium::Color::operator T ( ) const  [explicit]
```

conversion operator from Color to another type

Template Parameters

T	the type which the Color has to be converted to

Returns

an instance of type T that represent the converted value for Color

6.9.3 Member Data Documentation

6.9.3.1 b

```
uint8_t Symposium::Color::b
```

blue value for RGB encoding

6.9.3.2 g

```
uint8_t Symposium::Color::g
```

green value for RGB encoding

6.9.3.3 r

```
uint8_t Symposium::Color::r
```

red value for RGB encoding

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

- Color.h
- Color.cpp

6.10 Symposium::colorGen Class Reference

Color generator for users in Symposium system as functor.

```
#include <Color.h>
```

Public Member Functions

- Color **operator()** ()

Static Public Member Functions

- static Color hsv_to_rbg (double h, double s, double v)

  Converts a color encoded in HSV to RGB.

Private Attributes

- double token

Static Private Attributes

- static constexpr double grc = 0.618033988749895

6.10.1 Detailed Description

Color generator for users in Symposium system as functor.

An object of this class is used to generate a color for a user: every time a functor of this class is invoked it returns a Color that is the most perceptually different with respect to the others generated by the same functor.

6.10.2 Member Function Documentation

6.10.2.1 hsv_to_rbg()

```
Color colorGen::hsv_to_rbg (
            double h,
            double s,
            double v )  [static]
```

Converts a color encoded in HSV to RGB.

Parameters

h	hue parameter of HSV encoding
s	saturation parameter of HSV encoding
v	value parameter of HSV encoding

Returns

a corresponding RGB encoding through a Color

6.10.3 Member Data Documentation

6.10.3.1 grc

```
constexpr double Symposium::colorGen::grc = 0.618033988749895  [static], [private]
```

It is the inverse of the golden ratio

6.10.3.2 token

```
double Symposium::colorGen::token  [private]
```

A random semen for the first color generated

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

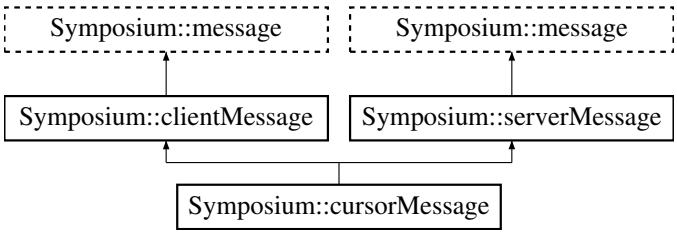
- Color.h
- Color.cpp

6.11 Symposium::cursorMessage Class Reference

class used to model a message to update the position of a user's cursor

```
#include <message.h>
```

Inheritance diagram for Symposium::cursorMessage:



Public Member Functions

- **cursorMessage** (msgType action, const std::pair< std::string, std::string > &actionOwner, msgOutcome result, uint_positive_cnt::type siteId, uint_positive_cnt::type resourceId, unsigned int row, unsigned int col, uint_positive_cnt::type msgId=0)
- void invokeMethod (SymServer &server) override

  notify the server that the position of the user's cursor has changed
- void invokeMethod (SymClient &client) override

  propagate the update of user cursor's position to other clients working on the same resource
- unsigned int **getRow** () const
- unsigned int **getCol** () const
- bool **operator==** (const cursorMessage &rhs) const
- bool **operator!=** (const cursorMessage &rhs) const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Private Attributes

- uint_positive_cnt::type siteId
- uint_positive_cnt::type resourceId
- unsigned row
- unsigned col

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.11.1 Detailed Description

class used to model a message to update the position of a user's cursor

6.11.2 Member Function Documentation

6.11.2.1 invokeMethod() [1/2]

```
void cursorMessage::invokeMethod (
            SymServer & server )  [override], [virtual]
```

notify the server that the position of the user's cursor has changed

Parameters

server	the server the user is active on

Reimplemented from Symposium::clientMessage.

6.11.2.2 invokeMethod() [2/2]

```
void cursorMessage::invokeMethod (
            SymClient & client )  [override], [virtual]
```

propagate the update of user cursor's position to other clients working on the same resource

Parameters

client	the client on which propagate the change

Reimplemented from Symposium::serverMessage.

6.11.3 Member Data Documentation

6.11.3.1 col

```
unsigned Symposium::cursorMessage::col  [private]
```

new column of the cursor

6.11.3.2 resourceId

```
uint_positive_cnt::type Symposium::cursorMessage::resourceId  [private]
```

resourceId of the resource on which the user cursor's position has changed

6.11.3.3 row

```
unsigned Symposium::cursorMessage::row  [private]
```

new row of the cursor

6.11.3.4 siteId

```
uint_positive_cnt::type Symposium::cursorMessage::siteId  [private]
```

siteId of the client that send the message

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

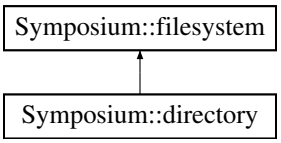
- message.h
- message.cpp

6.12 Symposium::directory Class Reference

class used to model a directory, uses *Singleton* pattern

`#include <filesystem.h>`

Inheritance diagram for Symposium::directory:



Public Member Functions

- virtual std::shared_ptr< filesystem > **get** (const std::string &resPath, const std::string &resId)
- virtual std::shared_ptr< directory > **getDir** (const std::string &resPath, const std::string &resId)
- virtual std::shared_ptr< file > **getFile** (const std::string &resPath, const std::string &resId)
- virtual std::string **setName** (const std::string &resPath, const std::string &resId, const std::string &newName)
- virtual std::shared_ptr< directory > **addDirectory** (const std::string &resName, uint_positive_cnt::type id↩ ToAssign=0)
- virtual std::shared_ptr< file > **addFile** (const std::string &resPath, const std::string &resName, uint_↩ positive_cnt::type idToAssign=0)
- virtual std::shared_ptr< Symposium::symlink > **addLink** (const std::string &symPath, const std::string &symName, const std::string &absPathWithoutId, const std::string &resId, uint_positive_cnt::type idTo↩ Assign=0)
- virtual resourceType resType () const override

  identify the type of current filesystem resource
- virtual document & access (const user &targetUser, const std::string &resPath, const std::string &resId, privilege accessMode)

  traverse the filesystem and invoke access on resName
- virtual std::shared_ptr< filesystem > remove (const user &targetUser, const std::string &resPath, const std↩ ::string &resId)

  traverse the filesystem and invoke remove on resName
- void storeContent () const override

  Call storeContent() on contained filesystem objects.
- virtual std::string print (const std::string &targetUser, bool recursive=true, unsigned int indent=0) const override

  give a textual representation of the content of the current directory
- virtual bool isReadyToRemove (const std::string &username) const override

  check if all resources in the directory is available to remove operation

Static Public Member Functions

- static std::shared_ptr< directory > **emptyDir** ()
- static std::shared_ptr< directory > **getRoot** ()
- static std::tuple< std::string, std::string > separateFirst (std::string path)

  separate the first part of the path which indicate the id of the directory, example: path=./1/2/3 result 1 and 2/3

Protected Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int version)
- **directory** (const std::string &name, const uint_positive_cnt::type &idToAssign=0)

Protected Attributes

- std::vector< std::shared_ptr< filesystem > > contained
- std::weak_ptr< directory > parent
- std::weak_ptr< directory > self

Static Protected Attributes

- static std::shared_ptr< directory > root

Friends

- class **boost::serialization::access**

6.12.1 Detailed Description

class used to model a directory, uses *Singleton* pattern

6.12.2 Member Function Documentation

6.12.2.1 access()

```
document & directory::access (
            const user & targetUser,
            const std::string & resPath,
            const std::string & resId,
            privilege accessMode )  [virtual]
```

traverse the filesystem and invoke *access* on *resName*

Parameters

targetUser	the user who asked for this action
resPath	relative path to the resource from the current directory
resId	the id of the resource to access (file or symlink)
accessMode	the privilege asked by the user for opening the file

Returns

the document contained in the file object

6.12.2.2 isReadyToRemove()

```
bool directory::isReadyToRemove (
            const std::string & username ) const  [override], [virtual]
```

check if all resources in the directory is available to remove operation

Parameters

username	the user who wants to perform the operation

Returns

true is the remove operation is possible, false otherwise

Implements Symposium::filesystem.

6.12.2.3 print()

```
std::string directory::print (
            const std::string & targetUser,
            bool recursive = true,
            unsigned int indent = 0 ) const  [override], [virtual]
```

give a textual representation of the content of the current directory

Parameters

targetUser	the user who asked for this action
recursive	indicates whether the action is to be executed recursively on subdirectories
indent	an optional identation level to distinguish nested objects

Returns

a string containing the representation

Implements Symposium::filesystem.

6.12.2.4 remove()

```
std::shared_ptr< filesystem > directory::remove (
            const user & targetUser,
            const std::string & resPath,
            const std::string & resId )  [virtual]
```

traverse the filesystem and invoke *remove* on *resName*

Parameters

targetUser	the user who asked for this action
resPath	relative path to the resource from the current directory
resId	the name of the resource to remove

Returns

the resource just removed from the filesystem

Removes a file, a symlink or a directory from the current directory. The parameter *targetUser* is used to authenticate the action in case of the target resource *resName* is a file. The directory is remove if all resources are available for deletion

Table 6.17 Removing policy for file and symlink

Resource type	Privilege of user who perform action	Other User Privilege	Active user in the moment of action	Result of the operation
File	owner	owner	Yes/No	File will not be deleted because there are other users with the privilege owner
		readOnly, modify	Yes	File will not be deleted because there are other users working on it
			No	File will be deleted
	readOnly, modify	owner, readOnly, modify	Yes/No	File will not be deleted because there is at least one owner of this document
Symlink	Owner	Owner	Yes/No	Symlink will be deleted and the user will no longer have access to the file pointed
		readOnly, Modify	Yes	Symlink and the file pointed will not be deleted because there are other users working on the file pointed

Resource type	Privilege of user who perform action	Other User Privilege	Active user in the moment of action	Result of the operation
			No	Symlink and the file pointed will be deleted because there are other users working on the file pointed
	readOnly, Modify	Owner, ReadOnly, modify	Yes/No	Symlink will be deleted but the file pointed will not

6.12.2.5 resType()

```
resourceType directory::resType ( ) const  [override], [virtual]
```

identify the type of current filesystem resource

Returns

the type of the current filesystem object (file, directory, symlink)

Implements Symposium::filesystem.

6.12.2.6 separateFirst()

```
std::tuple< std::string, std::string > directory::separateFirst (
            std::string path )  [static]
```

separate the first part of the path which indicate the id of the directory, example: path=./1/2/3 result 1 and 2/3

Parameters

path	the path to divide

Returns

path that has remained and the id of the directory

6.12.3 Member Data Documentation

6.12.3.1 contained

```
std::vector<std::shared_ptr<filesystem> > Symposium::directory::contained  [protected]
```

filesystem objects contained in the directory

6.12.3.2 parent

```
std::weak_ptr<directory> Symposium::directory::parent  [protected]
```

pointer to the parent directory

6.12.3.3 root

```
std::shared_ptr< directory > directory::root  [static], [protected]
```

root directory of the system

6.12.3.4 self

```
std::weak_ptr<directory> Symposium::directory::self  [protected]
```

pointer to itself

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

- filesystem.h
- filesystem.cpp

6.13 Symposium::document Class Reference

Public Member Functions

- **document** (uint_positive_cnt::type id=0)
- uint_positive_cnt::type **getId** () const
- const std::vector< std::vector< symbol > > & **getSymbols** () const
- const std::forward_list< std::pair< const user *, sessionData > > & **getActiveUsers** () const
- unsigned int **getNumchar** () const
- unsigned int countsNumLines () const

  countsNumLines counts the number of line effectively present in the symbols vector
- unsigned int countCharsInLine (unsigned int line) const

  countCharsInLine counts the number of chars that are present in the line
- bool **operator==** (const document &rhs) const
- bool **operator!=** (const document &rhs) const
- virtual document & access (const user &newActive, privilege accessPriv)

  open the current document, loading it from disk if needed
- virtual symbol localInsert (const std::pair< unsigned int, unsigned int > &indexes, symbol &toInsert)

  insert a symbol in the document as consequence of an user's action on the GUI

- virtual symbol localRemove (const std::pair< unsigned int, unsigned int > &indexes, uint_positive_cnt::type siteId)

  remove a symbol in the document as consequence of an user's action on the GUI
- virtual std::pair< unsigned int, unsigned int > remoteInsert (uint_positive_cnt::type siteId, const symbol &toInsert)

  insert a symbol in the document as consequence of a remote user's action
- virtual std::pair< unsigned int, unsigned int > remoteRemove (uint_positive_cnt::type siteId, const symbol &toRemove)

  remove a symbol in the document as consequence of a remote user's action
- virtual std::pair< unsigned int, unsigned int > verifySymbol (const symbol &toVerify)

  set a symbol inside the document as verified
- virtual void updateCursorPos (uint_positive_cnt::type targetSiteId, unsigned int newRow, unsigned int newCol)

  update the position of the cursor for the user target
- virtual void editLineStyle (const std::pair< alignType, unsigned > &newLineStyle, unsigned row)

  update the alignment and/or the indexStyle of the document's paragraph row
- std::wstring toText () const

  give a representation of the document ad sequence of wide characters
- virtual void close (const user &noLongerActive)

  removes the user noLongerActive from activeUsers
- void store () const

  Store permanently the content of the document onto the disk.
- bool load ()

  Load the content of the document from disk.
- virtual std::set< uint_positive_cnt::type > retrieveSiteIds () const

  retrieves the set of siteId in the current document
- const std::vector< std::pair< alignType, unsigned int > > & **getAlignmentStyle** () const

Static Public Member Functions

- static void doLightSerializing (const std::function< void(void)> &op)

  executes op assuring that, if serialize method wil be invoked, the serialization will involve all object's attributes

Static Public Attributes

- static constexpr wchar_t **emptyChar** =0x1F
- static const symbol **emptySymbol**
- static bool **doLoadAndStore** =true

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int)
- symbol generatePosition (const std::pair< unsigned int, unsigned int > indexes, const symbol &toInsert)

  it determines the globally unique fractional index position of the new character.
- std::pair< unsigned int, unsigned int > findInsertIndex (const symbol &symbol) const

  it searches for the position of the inserted symbol
- std::pair< unsigned int, unsigned int > findEndPosition (unsigned int lines, const symbol &lastSymbol) const

  it finds the position of the last Symbol to insert the symbol at the end of vector

- unsigned int findInsertInLine (const symbol &ch, const std::vector< symbol > &vector, unsigned int line) const

  it searches the position in a line for the symbol that has to be inserted throught the RemoteInsert operation
- std::pair< unsigned int, unsigned int > findPosition (const symbol &symbol) const

  it searches for the position of a symbol in order to find it and eliminate it
- unsigned int findIndexInLine (const symbol &sym, const std::vector< symbol > &vector, unsigned int charsInLine) const

  findIndexInLine searches the position in a line for the symbol that has to be deleted throught the RemoteDelete operation
- symbol findPosBefore (const std::pair< unsigned int, unsigned int > &pair) const

  searches the position before the one of the considered value
- symbol findPosAfter (const std::pair< unsigned int, unsigned int > &pair) const

  searches the position after the one of the considered value
- std::vector< int > generatePosBetween (const std::vector< int > &posBefore, const std::vector< int > &posAfter, std::vector< int > newPos, int level, const symbol &b, const symbol &a)

  recoursive algorithm to dynamically generates the relative position of a symbol inserted in between two other ones
- char retrieveStrategy (unsigned int level)

  it modifies the strategy parameter
- void updateOtherCursorPos (uint_positive_cnt::type targetSiteId, unsigned int newRow, unsigned int newCol, const symbol &symb, bool ins)

  updateOtherCursorPos to update the position of the cursors different from mine in the document
- void checkIndexes (const std::pair< unsigned int, unsigned int > &toAccess) const

  check that the indexes passed as parameter are valid

Static Private Member Functions

- static unsigned int generateIdBetween (uint_positive_cnt::type id1, uint_positive_cnt::type id2, char boundaryStrategy)

  finds the a correct value for the position of a symbol inserted between other two symbols

Private Attributes

- uint_positive_cnt::type id
- std::vector< std::vector< symbol > > symbols
- std::forward_list< std::pair< const user *, sessionData > > activeUsers
- std::vector< std::pair< alignType, unsigned > > alignmentStyle
- unsigned numchar
- std::vector< char > **strategyCache**
- wchar_t **strategy** ='r'
- bool **loaded**

Static Private Attributes

- static uint_positive_cnt idCounter
- static const std::string **basePath** ="./docs/"
- static bool **serializeFull** =true

Friends

- class **boost::serialization::access**

6.13.1 Member Function Documentation

6.13.1.1 access()

```
document & document::access (
            const user & newActive,
            privilege accessPriv )  [virtual]
```

open the current document, loading it from disk if needed

Parameters

newActiveUser	the user who attempts to open the document
accessPriv	the privilege granted to the user for this access

Returns

the document itself

This method is called by `file::access` and perform the actions needed to open a document, such as loading or storing it, and add *newActiveUser* to *activeUsers*.

6.13.1.2 checkIndexes()

```
void document::checkIndexes (
            const std::pair< unsigned int, unsigned int > & toAccess ) const  [private]
```

check that the indexes passed as parameter are valid

Parameters

toAccess	the indexes of the symbol that it's going to be accessed

6.13.1.3 close()

```
void document::close (
            const user & noLongerActive )  [virtual]
```

removes the user *noLongerActive* from *activeUsers*

Parameters

noLongerActive	the user that is no longer active on the current document

This method is called when a client closes the document it is working on.

6.13.1.4 countCharsInLine()

```
unsigned int document::countCharsInLine (
            unsigned int line ) const
```

countCharsInLine counts the number of chars that are present in the *line*

Parameters

line	

Returns

the number of chars that are present in the *line*

6.13.1.5 countsNumLines()

```
unsigned int document::countsNumLines ( ) const
```

countsNumLines counts the number of line effectively present in the symbols vector

Returns

number of lines

6.13.1.6 doLightSerializing()

```
void document::doLightSerializing (
            const std::function< void(void)> & op )  [static]
```

executes *op* assuring that, if serialize method wil be invoked, the serialization will involve all object's attributes

Parameters

op	the operation to be executed with full serialization

6.13.1.7 editLineStyle()

```
void document::editLineStyle (
```

```
            const std::pair< alignType, unsigned > & newLineStyle,
            unsigned row )  [virtual]
```

update the alignment and/or the indexStyle of the document's paragraph *row*

Parameters

newLineStyle	new alignment and indexStyle to apply
row	the row to apply *newLineStyle* to

6.13.1.8 findEndPosition()

```
std::pair< unsigned int, unsigned int > document::findEndPosition (
            unsigned int lines,
            const symbol & lastSymbol ) const  [private]
```

it finds the position of the last Symbol to insert the symbol at the end of *vector*

Parameters

aChar	the lastSymbol in the vector
vector	is the LastLine in the vector
lines	is the number of lines in the vector

Returns

the position

6.13.1.9 findIndexInLine()

```
unsigned int document::findIndexInLine (
            const symbol & sym,
            const std::vector< symbol > & vector,
            unsigned int charsInLine ) const  [private]
```

findIndexInLine searches the position in a line for the symbol that has to be deleted throught the RemoteDelete operation

Parameters

sym	symbol that has to be deleted
vector	corresponds to the line in which the symbol is
charsInLine	the number of the chars in Line

Returns

6.13.1.10 findInsertIndex()

```
std::pair< unsigned int, unsigned int > document::findInsertIndex (
            const symbol & symbol ) const  [private]
```

it searches for the position of the inserted symbol

Parameters

symbol	inserted symbol

Returns

position

6.13.1.11 findInsertInLine()

```
unsigned int document::findInsertInLine (
            const symbol & ch,
            const std::vector< symbol > & vector,
            unsigned int line ) const  [private]
```

it searches the position in a line for the symbol that has to be inserted throught the RemoteInsert operation

Parameters

ch	symbol
vector	line in which the symbol is searched

Returns

the index in the line at which the symbol is

6.13.1.12 findPosAfter()

```
symbol document::findPosAfter (
            const std::pair< unsigned int, unsigned int > & pair ) const  [private]
```

searches the position after the one of the considered value

Parameters

pair	the indexes

Returns

the symbol. If there no exists a following symbol, it returns an empty Symbol

< it could be zero

< there are no chars in line or no chars after the current pos, there is no a pos-after

< there is a pos-after

6.13.1.13 findPosBefore()

```
symbol document::findPosBefore (
            const std::pair< unsigned int, unsigned int > & pair ) const  [private]
```

searches the position before the one of the considered value

Parameters

pair	the indexes

Returns

the symbol. If there no exists a previous symbol, it returns an emptySymbol

< line is !=0 -> line-1 can't be a negative number

< in a previous line w.r.t the one in which I am, I have at least '\r' character

6.13.1.14 findPosition()

```
std::pair< unsigned int, unsigned int > document::findPosition (
            const symbol & symbol ) const  [private]
```

it searches for the position of a symbol in order to find it and eliminate it

Parameters

symbol	the symbol to search

Returns

the position of *symbol*

6.13.1.15 generateIdBetween()

```
unsigned int document::generateIdBetween (
            uint_positive_cnt::type id1,
            uint_positive_cnt::type id2,
            char boundaryStrategy )  [static], [private]
```

finds the a correct value for the position of a symbol inserted between other two symbols

Parameters

id1	
id2	
boundaryStrategy	

Returns

6.13.1.16 generatePosBetween()

```
std::vector< int > document::generatePosBetween (
            const std::vector< int > & posBefore,
            const std::vector< int > & posAfter,
            std::vector< int > newPos,
            int level,
            const symbol & b,
            const symbol & a )  [private]
```

recoursive algorithm to dynamically generates the relative position of a symbol inserted in between two other ones

Parameters

posBefore	the posBefore
posAfter	the posAfter

Returns

the searched position

6.13.1.17 generatePosition()

```
symbol document::generatePosition (
            const std::pair< unsigned int, unsigned int > indexes,
            const symbol & toInsert )  [private]
```

it determines the globally unique fractional index position of the new character.

Parameters

indexes	position of the adjacent characters used to generate the position of the new one
toInsert	the value to insert

6.13.1.18 load()

```
bool document::load ( )
```

Load the content of the document from disk.

Returns

a bool indicating success of failure on loading

6.13.1.19 localInsert()

```
symbol document::localInsert (
            const std::pair< unsigned int, unsigned int > & indexes,
            symbol & toInsert )  [virtual]
```

insert a symbol in the document as consequence of an user's action on the GUI

Parameters

toInsert	symbol to insert

6.13.1.20 localRemove()

```
symbol document::localRemove (
            const std::pair< unsigned int, unsigned int > & indexes,
            uint_positive_cnt::type siteId )  [virtual]
```

remove a symbol in the document as consequence of an user's action on the GUI

Parameters

indexes	symbol to remove

6.13.1.21 remoteInsert()

```
std::pair< unsigned int, unsigned int > document::remoteInsert (
            uint_positive_cnt::type siteId,
            const symbol & toInsert )  [virtual]
```

insert a symbol in the document as consequence of a remote user's action

Parameters

siteId	the site id of the user performing the insertion
toInsert	symbol to insert

Returns

the position of the inserted symbol

6.13.1.22 remoteRemove()

```
std::pair< unsigned int, unsigned int > document::remoteRemove (
            uint_positive_cnt::type siteId,
            const symbol & toRemove )  [virtual]
```

remove a symbol in the document as consequence of a remote user's action

Parameters

siteId	the site id of the user performing the removal
toRemove	symbol to remove

Returns

the position of the removed symbol

6.13.1.23 retrieveSiteIds()

```
std::set< uint_positive_cnt::type > document::retrieveSiteIds ( ) const  [virtual]
```

retrieves the set of siteId in the current document

Returns

a set of siteIds

6.13.1.24 retrieveStrategy()

```
char document::retrieveStrategy (
            unsigned int level )  [private]
```

it modifies the *strategy* parameter

Parameters

level	

Returns

6.13.1.25 toText()

```
std::wstring document::toText ( ) const
```

give a representation of the document ad sequence of wide characters

Returns

a string of wide characters with the document's content

6.13.1.26 updateCursorPos()

```
void document::updateCursorPos (
            uint_positive_cnt::type targetSiteId,
            unsigned int newRow,
            unsigned int newCol )  [virtual]
```

update the position of the cursor for the user *target*

Parameters

target↩ SiteId	the user whose cursor's position has been changed
newRow	the new row number
newCol	the new column number
toInsert	to verify if a '\r' symbol has beed introduced
ins	true if a symbol has been inserted, false if a symbol has been removed

6.13.1.27 updateOtherCursorPos()

```
void document::updateOtherCursorPos (
            uint_positive_cnt::type targetSiteId,
            unsigned int newRow,
            unsigned int newCol,
            const symbol & symb,
            bool ins )  [private]
```

updateOtherCursorPos to update the position of the cursors different from mine in the document

Parameters

target↩ SiteId	my siteId to find all the users different from me
newRow	i index in the document
newCol	j index in the document
symb	symbol that has been inserted. Useful to verify if I'm in a particular case \r
ins	to understand if the action is an insertion or a remove

6.13.1.28 verifySymbol()

```
std::pair< unsigned int, unsigned int > document::verifySymbol (
            const symbol & toVerify )  [virtual]
```

set a symbol inside the document as verified

Parameters

toVerify	the symbol to be marked as verified

6.13.2 Member Data Documentation

6.13.2.1 activeUsers

```
std::forward_list<std::pair<const user *, sessionData> > Symposium::document::activeUsers
[private]
```

list of users currently active on the document, with the current privilege

6.13.2.2 alignmentStyle

```
std::vector<std::pair<alignType,unsigned> > Symposium::document::alignmentStyle  [private]
```

vector that contains for each row the alignment left/right/center/justify and the style index

6.13.2.3 id

```
uint_positive_cnt::type Symposium::document::id  [private]
```

unique identifier for the document

6.13.2.4 idCounter

```
uint_positive_cnt document::idCounter  [static], [private]
```

id to be assigned to the next created document

6.13.2.5 numchar

```
unsigned Symposium::document::numchar  [private]
```

number of printable characters

6.13.2.6 symbols

```
std::vector<std::vector<symbol> > Symposium::document::symbols  [private]
```

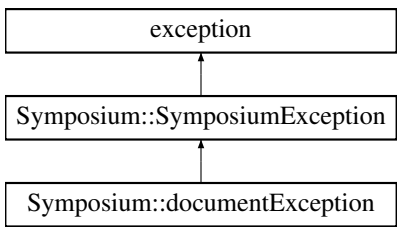
container of characters and metadata for CRDT

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

- document.h
- document.cpp

6.14 Symposium::documentException Class Reference

Inheritance diagram for Symposium::documentException:



Public Types

- enum **docExceptionCodes** {
  positionNotFound =0, **fixPositionSorting**, **outOfBounds**, **deletingEmptyChar**,
  insertingAfterNewLine }

Public Member Functions

- **documentException** (docExceptionCodes exceptionCode, const char *file, int line, const char *func)

Static Private Attributes

- static const char * **documentErrors** [ ]

Additional Inherited Members

6.14.1 Member Data Documentation

6.14.1.1 documentErrors

```
const char * documentException::documentErrors  [static], [private]
```

Initial value:

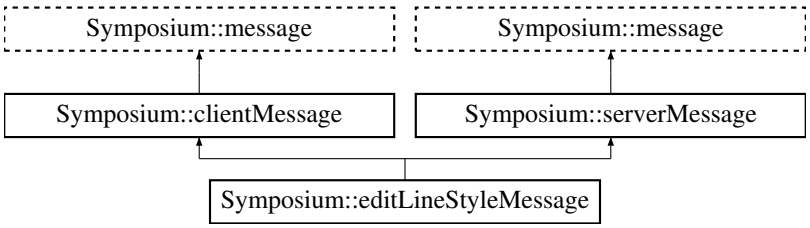
```
={"Error in finding the position of the symbol searched", "Fix Position Sorting", "trying to access index of
      of bounds",
                                                "Logic error, trying to delete line delimiter emptyChar",
      "Logic error, trying to insert in the same line after a newline character"}
```

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

- SymposiumException.h
- SymposiumException.cpp

6.15 Symposium::editLineStyleMessage Class Reference

Inheritance diagram for Symposium::editLineStyleMessage:



Public Member Functions

- **editLineStyleMessage** (msgType action, const std::pair< std::string, std::string > &actionOwner, msgOutcome result, const std::pair< alignType, unsigned > &oldLineStyle, const std::pair< alignType, unsigned > &newLineStyle, uint_positive_cnt::type docId, unsigned row, uint_positive_cnt::type msgId=0)
- void invokeMethod (SymServer &server) override

  Notify the server that the alignment and/or the index style of a paragraph has been changed.
- void invokeMethod (SymClient &client) override

  propagate the update alignment and/or the index style to other clients working on the same resource
- void completeAction (SymClient &client, msgOutcome serverResult) override

  confirm the changing of alignment and/or the index style made by a client or abort it
- bool **operator==** (const editLineStyleMessage &rhs) const
- bool **operator!=** (const editLineStyleMessage &rhs) const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Private Attributes

- std::pair< alignType, unsigned > **oldLineStyle**
- std::pair< alignType, unsigned > **newLineStyle**
- uint_positive_cnt::type **docId**
- unsigned **row**

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.15.1 Member Function Documentation

6.15.1.1 completeAction()

```
void editLineStyleMessage::completeAction (
            SymClient & client,
            msgOutcome serverResult )  [override], [virtual]
```

confirm the changing of alignment and/or the index style made by a client or abort it

Parameters

client	the client which sent the message

Depending on the value of *result*, the *invokeMethod* ask for different actions on the client:

- *result=msgOutcome::success* : calls nothing
- *result=msgOutcome::failure* : calls SymClient::changeLineStyle with *oldLineStyle* as parameter

Reimplemented from Symposium::clientMessage.

6.15.1.2 invokeMethod() [1/2]

```
void editLineStyleMessage::invokeMethod (
            SymServer & server )  [override], [virtual]
```

Notify the server that the alignment and/or the index style of a paragraph has been changed.

Parameters

server	the server the user is active on

Reimplemented from Symposium::clientMessage.

6.15.1.3 invokeMethod() [2/2]

```
void editLineStyleMessage::invokeMethod (
            SymClient & client )  [override], [virtual]
```

propagate the update alignment and/or the index style to other clients working on the same resource

Parameters

client	the client on which propagate the change

Reimplemented from Symposium::serverMessage.

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

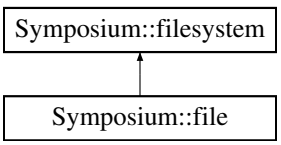
- message.h
- message.cpp

6.16 Symposium::file Class Reference

class used to model a file in the filesystem

`#include <filesystem.h>`

Inheritance diagram for Symposium::file:



Public Member Functions

- **file** (const std::string &name, uint_positive_cnt::type idToAssign=0)
- const document & **getDoc** () const
- const std::unordered_map< std::string, privilege > getUsers ()

  retrieve all users who can access the file invoke AccessStrategy::getPermission
- virtual privilege getUserPrivilege (const std::string &targetUser) const override

retrieve the privilege of a user on the current file

- virtual privilege setUserPrivilege (const std::string &targetUser, privilege newPrivilege) override
  set the privilege of targetUser to newPrivilege for the current file
- virtual uri setSharingPolicy (const std::string &actionUser, const uri &newSharingPrefs) override
  set new sharingPolicy for a file
- resourceType resType () const override
  identify the type of current filesystem resource
- virtual document & access (const user &targetUser, privilege accessMode)
  open the document associated with the current file
- void storeContent () const override
  Store the contained document in a proper file.
- std::string print (const std::string &targetUser, bool recursive=false, unsigned int indent=0) const override
  give a textual representation of the file
- void deleteFromStrategy (const std::string &userName)
  invoke AccessStrategy::deleteUser
- void replacement (std::shared_ptr< file > replace)
  substitute this file with the one pass as the parameter
- virtual bool validateAction (const std::string &userName, privilege priv)
  invoke AccessStrategy::validateAction
- virtual bool isReadyToRemove (const std::string &username) const override
  check if the resource is available to remove operation, so no users working on document
- bool **operator==** (const file &rhs) const
- bool **operator!=** (const file &rhs) const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int version)

Private Attributes

- document doc

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.16.1 Detailed Description

class used to model a file in the filesystem

6.16.2 Member Function Documentation

6.16.2.1 access()

```
document & file::access (
            const user & targetUser,
            privilege accessMode )  [virtual]
```

open the document associated with the current file

Parameters

targetUser	the user who asked for this action
accessMode	the privilege asked for the resource

Returns

the document contained in the file object

On server side, send the document to the client that has requested it checking for the privilege granted to it. Calls document::access on the document. On client side, request the server to send a document object and, after having received it, return it to the GUI

6.16.2.2 deleteFromStrategy()

```
void file::deleteFromStrategy (
            const std::string & userName )
```

invoke AccessStrategy::deleteUser

Parameters

userName	to delete from strategy object

Returns

true if the operation was successful, false instead

6.16.2.3 getUserPrivilege()

```
privilege file::getUserPrivilege (
            const std::string & targetUser ) const  [override], [virtual]
```

retrieve the privilege of a user on the current file

Parameters

targetUser	the user whose privilege is to be retrieved

Returns

the privilege of *targetUser*

Implements Symposium::filesystem.

6.16.2.4 getUsers()

```
const std::unordered_map< std::string, privilege > file::getUsers ( )
```

retrieve all users who can access the file invoke AccessStrategy::getPermission

Returns

unorder_map<username, privilege>

6.16.2.5 isReadyToRemove()

```
bool file::isReadyToRemove (
            const std::string & username ) const  [override], [virtual]
```

check if the resource is available to remove operation, so no users working on document

Parameters

username	the user who wants to perform the operation

Returns

true is the remove operation is possible, false otherwise

Implements Symposium::filesystem.

6.16.2.6 print()

```
std::string file::print (
            const std::string & targetUser,
            bool recursive = false,
            unsigned int indent = 0 ) const  [override], [virtual]
```

give a textual representation of the file

Parameters

targetUser	the user who asked for this action
recursive	for a file is meaningless
indent	an optional indentation level to distinguish nested objects

Returns

a string containing the representation

For a file, *print(targetUser)* shows the name of the file and the privilege that *targetUser* has on it

Implements Symposium::filesystem.

6.16.2.7 replacement()

```
void file::replacement (
            std::shared_ptr< file > replace )
```

substitute this file with the one pass as the parameter

Parameters

replace	the file to replace with

Warning

the content of *replaced* is moved into this

6.16.2.8 resType()

```
resourceType file::resType ( ) const  [override], [virtual]
```

identify the type of current filesystem resource

Returns

the type of the current filesystem object (file, directory, symlink)

Implements Symposium::filesystem.

6.16.2.9 setSharingPolicy()

```
uri file::setSharingPolicy (
            const std::string & actionUser,
            const uri & newSharingPrefs )  [override], [virtual]
```

set new *sharingPolicy* for a file

Parameters

actionUser	the user who is performing the action
newSharingPrefs	new sharing preferences for the resource

Returns

the old *sharingPolicy*

Verifies that *actionUser* is enabled to make such an action and replace the old *sharingPolicy*

Implements Symposium::filesystem.

6.16.2.10 setUserPrivilege()

```
privilege file::setUserPrivilege (
            const std::string & targetUser,
            privilege newPrivilege )  [override], [virtual]
```

set the privilege of *targetUser* to *newPrivilege* for the current file

Parameters

targetUser	the user whose privilege is to be modified
newPrivilege	the privilege to grant to *targetUser*

On client side this method asks the server for privilege changing sending a message of type privMessage: if the message outcome from the server is positive, then confirm the action, otherwise revert it. On server side validate the action, perform the action and send a serverMessage with the outcome.

Implements Symposium::filesystem.

6.16.2.11 validateAction()

```
bool file::validateAction (
            const std::string & userName,
            privilege priv )  [virtual]
```

invoke AccessStrategy::validateAction

Parameters

userName	of who wants to perform the action
priv	the privilege with which user wants to perform the action

Returns

true if the operation is allowed, false instead

6.16.3 Member Data Documentation

6.16.3.1 doc

`document Symposium::file::doc [private]`

document to handle

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

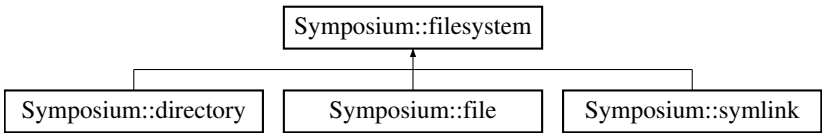
- filesystem.h
- filesystem.cpp

6.17 Symposium::filesystem Interface Reference

class used as interface for a filesystem, made using *Composite* pattern

`#include <filesystem>`

Inheritance diagram for Symposium::filesystem:



Public Member Functions

- void **setName** (const std::string &name)
- **filesystem** (const std::string &name, const uint_positive_cnt::type idToAssign=0)
- uint_positive_cnt::type **getId** () const
- const std::string & **getName** () const
- virtual void storeContent () const

  Store the content handled by the filesystem object (if any)
- virtual privilege getUserPrivilege (const std::string &targetUser) const

  retrieve the privilege of a user on the current filesystem object
- virtual uri & **getSharingPolicy** ()
- virtual privilege setUserPrivilege (const std::string &targetUser, privilege newPrivilege)

  set the privilege of targetUser to newPrivilege for the current filesystem object
- virtual resourceType resType () const =0

  identify the type of current filesystem resource
- virtual uri setSharingPolicy (const std::string &actionUser, const uri &newSharingPrefs)

  set new sharingPolicy for a filesystem object
- virtual std::string **print** (const std::string &targetUser, bool recursive=false, unsigned int indent=0) const =0
- bool moreOwner (const std::string &username)

  check if the object has other owners besides the name of the username pass as parameter
- virtual bool isReadyToRemove (const std::string &username) const =0

  check if the resource is available to remove operation
- bool **operator==** (const filesystem &rhs) const
- bool **operator!=** (const filesystem &rhs) const

Static Public Member Functions

- static std::tuple< std::string, std::string > separate (const std::string &path)

  separate the last part of path which indicate the id of the resource, example: path=./1/2/3 result 1/2 and 3
- static bool pathIsValid2 (const std::string &toCheck)

  check if the path has a valid format

Protected Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int version)

Protected Attributes

- std::string name
- uri sharingPolicy
- std::unique_ptr< AccessStrategy > strategy

Private Attributes

- uint_positive_cnt::type id

Static Private Attributes

- static uint_positive_cnt idCounter

Friends

- class **boost::serialization::access**

6.17.1 Detailed Description

class used as interface for a filesystem, made using *Composite* pattern

In the application design only objects of subclass file can be shared, but *sharingPolicy* could refer to a directory. This is done on purpose to allow future extensions of this module. As long as the mentioned behaviour is desired, for objects of subclasses directory and symlink *sharingPolicy* must indicate that the resource is not sharable

6.17.2 Member Function Documentation

6.17.2.1 getUserPrivilege()

```
privilege filesystem::getUserPrivilege (
            const std::string & targetUser ) const  [virtual]
```

retrieve the privilege of a user on the current filesystem object

Parameters

targetUser	the user whose privilege is to be retrieved

Returns

the privilege of *targetUser*

Exceptions

Implemented in Symposium::file.

6.17.2.2 isReadyToRemove()

```
virtual bool Symposium::filesystem::isReadyToRemove (
            const std::string & username ) const  [pure virtual]
```

check if the resource is available to remove operation

Parameters

username	the user who wants to perform the operation

Returns

true is the remove operation is possible, false otherwise

Implemented in Symposium::directory, Symposium::symlink, and Symposium::file.

6.17.2.3 moreOwner()

```
bool filesystem::moreOwner (
            const std::string & username )
```

check if the object has other owners besides the name of the username pass as parameter

Parameters

username	

Returns

true if there is only one user and this user is *username*, false instead

6.17.2.4 pathIsValid2()

```
bool filesystem::pathIsValid2 (
            const std::string & toCheck )  [static]
```

check if the path has a valid format

Parameters

toCheck	path to check

Returns

true if the format is ok, false instead

6.17.2.5 resType()

```
virtual resourceType Symposium::filesystem::resType ( ) const  [pure virtual]
```

identify the type of current filesystem resource

Returns

the type of the current filesystem object (file, directory, symlink)

Implemented in Symposium::directory, Symposium::symlink, and Symposium::file.

6.17.2.6 separate()

```
std::tuple< std::string, std::string > filesystem::separate (
            const std::string & path )  [static]
```

separate the last part of path which indicate the id of the resource, example: path=./1/2/3 result 1/2 and 3

Parameters

path	the path to divide

Returns

path to resource and id of the resource

6.17.2.7 setSharingPolicy()

```
uri filesystem::setSharingPolicy (
            const std::string & actionUser,
            const uri & newSharingPrefs ) [virtual]
```

set new *sharingPolicy* for a filesystem object

Parameters

actionUser	the user who is performing the action
newSharingPrefs	new sharing preferences for the resource

Returns

the old *sharingPolicy*

Exceptions

Implemented in Symposium::file.

6.17.2.8 setUserPrivilege()

```
privilege filesystem::setUserPrivilege (
            const std::string & targetUser,
            privilege newPrivilege ) [virtual]
```

set the privilege of *targetUser* to *newPrivilege* for the current filesystem object

Parameters

targetUser	the user whose privilege is to be modified
newPrivilege	the privilege to grant to *targetUser*

Exceptions

Implemented in Symposium::file.

6.17.3 Member Data Documentation

6.17.3.1 id

```
uint_positive_cnt::type Symposium::filesystem::id  [private]
```

unique identifier for the filesystem object, used also for identifying objects along a path

6.17.3.2 idCounter

```
uint_positive_cnt filesystem::idCounter  [static], [private]
```

id to be assigned to the next created filesystem object

6.17.3.3 name

```
std::string Symposium::filesystem::name  [protected]
```

resource name

6.17.3.4 sharingPolicy

```
uri Symposium::filesystem::sharingPolicy  [protected]
```

sharing policy applied to the resource

6.17.3.5 strategy

```
std::unique_ptr<AccessStrategy> Symposium::filesystem::strategy  [protected]
```

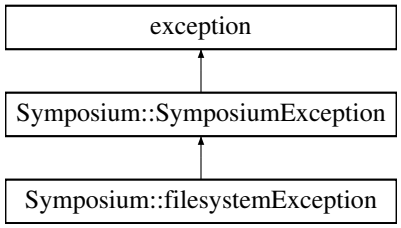
embodies the privileges associated with users

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

- filesystem.h
- filesystem.cpp

6.18 Symposium::filesystemException Class Reference

Inheritance diagram for Symposium::filesystemException:



Public Types

- enum filesystemExceptionCodes {
  objSha =0, **pathEmpty**, **pathNvalid**, **changePriv**,
  notOwn, **noPermission**, **noGet**, **noGetDir**,
  noGetFile, **sameName**, **notOnlyOwn**, **someoneWork**,
  notOwnDelete }

  Specific error codes for filesystemException. They are used as indexes to the error table string.

Public Member Functions

- **filesystemException** (filesystemExceptionCodes exceptionCode, const char ∗file, int line, const char ∗func)

Static Private Attributes

- static const char ∗ **filesystemErrors** [ ]

Additional Inherited Members

6.18.1 Member Data Documentation

6.18.1.1 filesystemErrors

```
const char * filesystemException::filesystemErrors  [static], [private]
```

Initial value:

```
={"Object is not shareable", "The path is empty", "The format of path is not valid",
                                                    "You are an owner, so you cannot change your
       privilege", "You are not an owner to share this file",
                                                    "You have not permission to access this file in this
       mode", "The element has not been found with get",
                                                    "Directory are you searching for, is not present",
       "File are you searching for, is not present",
                                                    "You already have an element with the same name", "You
       are not the only owner of this file, so you cannot delete it",
                                                    "It is not possible to delete:someone is working on
       document", "Ask an owner to delete this file, you cannot do it"}
```

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

- SymposiumException.h
- SymposiumException.cpp

6.19 Symposium::format Struct Reference

Public Member Functions

- **format** (const std::string &ft, bool bold, bool underline, bool italic, unsigned size, Color col, unsigned indexStyle, alignType type)
- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int)

Public Attributes

- std::string **familyType**
- bool **isBold**
- bool **isUnderlined**
- bool **isItalic**
- unsigned **size**
- Color **col**
- unsigned indexStyle
- alignType type

6.19.1 Member Data Documentation

6.19.1.1 indexStyle

```
unsigned Symposium::format::indexStyle
```

to set the style of the text

6.19.1.2 type

```
alignType Symposium::format::type
```

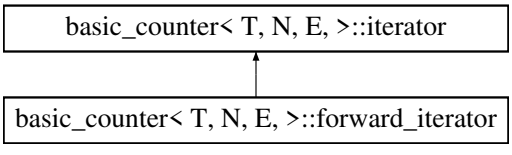
to set the kind of alignment of the text

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

- symbol.h
- symbol.cpp

6.20 basic_counter< T, N, E, >::forward_iterator Struct Reference

Inheritance diagram for basic_counter< T, N, E, >::forward_iterator:



Public Member Functions

- **forward_iterator** (basic_counter &c)
- forward_iterator **begin** ()
- forward_iterator **end** ()
- forward_iterator & **operator++** ()
- forward_iterator **operator++** (int)

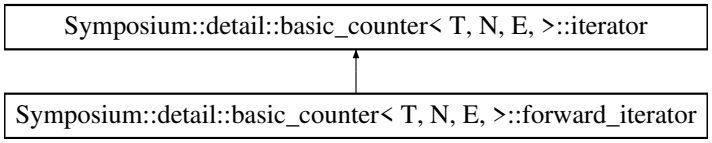
Additional Inherited Members

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

- counter.h

6.21 Symposium::detail::basic_counter< T, N, E, >::forward_iterator Struct Reference

Inheritance diagram for Symposium::detail::basic_counter< T, N, E, >::forward_iterator:



Public Member Functions

- **forward_iterator** (basic_counter &c)
- forward_iterator **begin** ()
- forward_iterator **end** ()
- forward_iterator & **operator++** ()
- forward_iterator **operator++** (int)

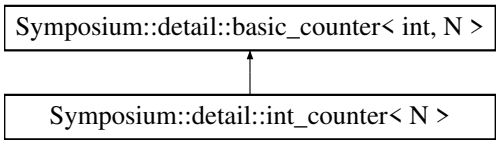
Additional Inherited Members

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

- Symposium.h

6.22 Symposium::detail::int_counter< N > Struct Template Reference

Inheritance diagram for Symposium::detail::int_counter< N >:



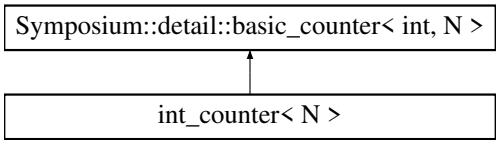
Additional Inherited Members

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

- Symposium.h

6.23 int_counter< N > Struct Template Reference

Inheritance diagram for int_counter< N >:



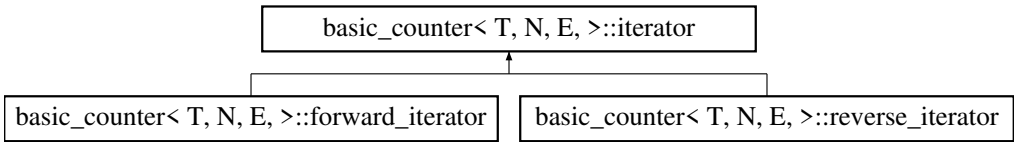
Additional Inherited Members

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

- counter.h

6.24 basic_counter< T, N, E, >::iterator Struct Reference

Inheritance diagram for basic_counter< T, N, E, >::iterator:



Public Member Functions

- **iterator** (basic_counter &c)
- const basic_counter & **operator** * ()
- bool **operator==** (const iterator &rhs) const
- bool **operator!=** (const iterator &rhs) const

Public Attributes

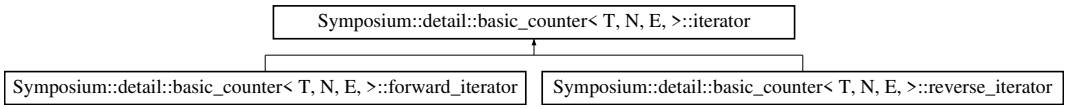
- basic_counter **cnt**

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

- counter.h

6.25 Symposium::detail::basic_counter< T, N, E, >::iterator Struct Reference

Inheritance diagram for Symposium::detail::basic_counter< T, N, E, >::iterator:



Public Member Functions

- **iterator** (basic_counter &c)
- const basic_counter & **operator** * ()
- bool **operator==** (const iterator &rhs) const
- bool **operator!=** (const iterator &rhs) const

Public Attributes

- basic_counter **cnt**

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

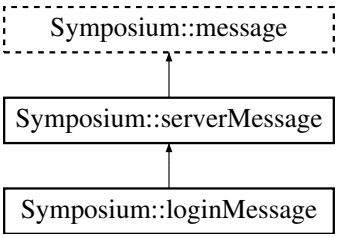
- Symposium.h

6.26 Symposium::loginMessage Class Reference

class used to model a message sent by a server as answer to a login message

`#include <message.h>`

Inheritance diagram for Symposium::loginMessage:



Public Member Functions

- loginMessage (msgType action, msgOutcome result, const user &loggedUser, uint_positive_cnt::type msgId=0)
- void invokeMethod (SymClient &client) override

  enable the client have the same representation for an user after a login
- bool **operator==** (const loginMessage &rhs) const
- bool **operator!=** (const loginMessage &rhs) const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Private Attributes

- user **loggedUser**

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.26.1 Detailed Description

class used to model a message sent by a server as answer to a login message

The client sends a clientMessage object with the authentication parameters and receives a loginMessage object with all the data of the logged user. In this message *action=msgType::registration* or *action=msgType::login*

6.26.2 Constructor & Destructor Documentation

6.26.2.1 loginMessage()

```
loginMessage::loginMessage (
            msgType action,
            msgOutcome result,
            const user & loggedUser,
            uint_positive_cnt::type msgId = 0 )
```

Exceptions

messageException	if *action* is not consistent with the message type

6.26.3 Member Function Documentation

6.26.3.1 invokeMethod()

```
void loginMessage::invokeMethod (
            SymClient & client )  [override], [virtual]
```

enable the client have the same representation for an user after a login

Parameters

client	the client instance to fill with user's data

When this message is received by the client, the *invokeMethod* calls SymClient::setLoggedUser on the *client* passed as parameter.

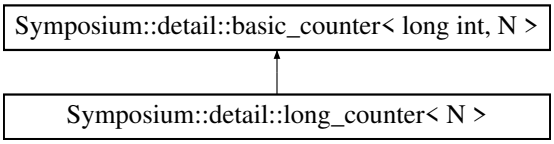
Reimplemented from Symposium::serverMessage.

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

- message.h
- message.cpp

6.27 Symposium::detail::long_counter< N > Struct Template Reference

Inheritance diagram for Symposium::detail::long_counter< N >:



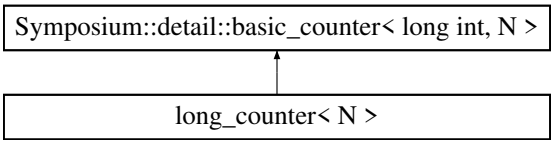
Additional Inherited Members

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

- Symposium.h

6.28 long_counter< N > Struct Template Reference

Inheritance diagram for long_counter< N >:



Additional Inherited Members

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

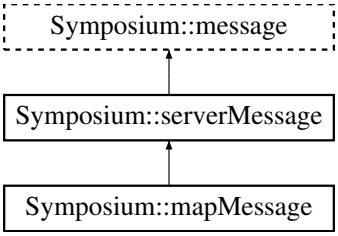
- counter.h

6.29 Symposium::mapMessage Class Reference

class used to model an answer message sent by a server for a clientMessage

`#include <message.h>`

Inheritance diagram for Symposium::mapMessage:



Public Member Functions

- mapMessage (msgType action, msgOutcome result, const std::map< uint_positive_cnt::type, user > &siteIdToUser, uint_positive_cnt::type msgId=0)
- void invokeMethod (SymClient &client) override

  update the mapping siteId->pair<user, color> with the information sent by the server
- bool **operator==** (const mapMessage &rhs) const
- bool **operator!=** (const mapMessage &rhs) const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Private Attributes

- std::map< uint_positive_cnt::type, user > **siteIdToUser**

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.29.1 Detailed Description

class used to model an answer message sent by a server for a clientMessage

The server answers with the map siteId->user for the document the current user is working on. The client ask for this answer with a updateDocMessage with *action=msgType::mapChangesToUser* In this message *action=msgType::mapChangesToUser* : calls

6.29.2 Constructor & Destructor Documentation

6.29.2.1 mapMessage()

```
mapMessage::mapMessage (
            msgType action,
            msgOutcome result,
            const std::map< uint_positive_cnt::type, user > & siteIdToUser,
            uint_positive_cnt::type msgId = 0 )
```

Exceptions

messageException	if *action* is not consistent with the message type

6.29.3 Member Function Documentation

6.29.3.1 invokeMethod()

```
void mapMessage::invokeMethod (
            SymClient & client )  [override], [virtual]
```

update the mapping siteId->pair<user, color> with the information sent by the server

Parameters

client	the client instance to update

When this message is received by the client, the *invokeMethod* calls SymClient::setUserColors on the *client* passed as parameter.

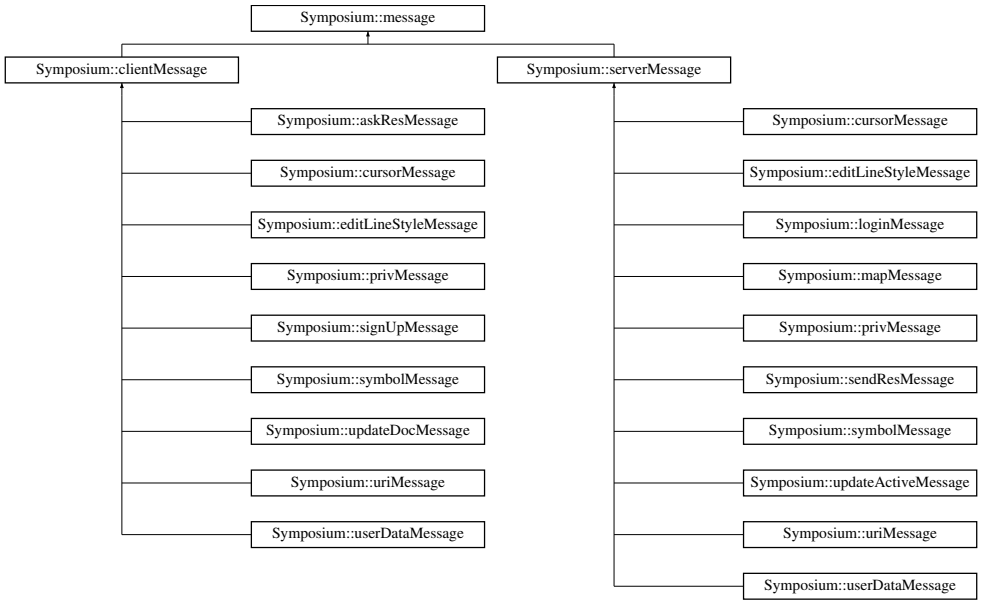
Reimplemented from Symposium::serverMessage.

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

- message.h
- message.cpp

6.30 Symposium::message Class Reference

Inheritance diagram for Symposium::message:



Public Member Functions

- uint_positive_cnt::type **getMsgId** () const
- msgType **getAction** () const
- bool **isFinalMex** () const
- bool **operator==** (const message &rhs) const
- bool **operator!=** (const message &rhs) const

Protected Member Functions

- **message** (uint_positive_cnt::type msgId=0)

Protected Attributes

- uint_positive_cnt::type msgId
- msgType action

Static Protected Attributes

- static uint_positive_cnt **msgCounter**

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Friends

- class **boost::serialization::access**

6.30.1 Member Data Documentation

6.30.1.1 action

```
msgType Symposium::message::action  [protected]
```

Defines the action for the current message

6.30.1.2 msgId

```
uint_positive_cnt::type Symposium::message::msgId  [protected]
```

random identifier for the message, used when a message is followed by an answer

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

- message.h
- message.cpp

6.31 message Interface Reference

class used as interface for a message

```
#include <message>
```

6.31.1 Detailed Description

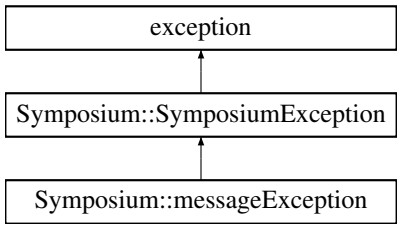
class used as interface for a message

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

- message.h

6.32 Symposium::messageException Class Reference

Inheritance diagram for Symposium::messageException:



Public Types

- enum messageExceptionCodes {
  action =0, **notClient**, **notSucc**, **askResMes**,
  sendResMes, **symb**, **upAct**, **upDoc**,
  userData, **cursor** }

  Specific error codes for messageException. They are used as indexes to the error table string.

Public Member Functions

- **messageException** (messageExceptionCodes exceptionCode, const char ∗file, int line, const char ∗func)

Static Private Attributes

- static const char ∗ **messageErrors** [ ]

Additional Inherited Members

6.32.1 Member Data Documentation

6.32.1.1 messageErrors

```
const char * messageException::messageErrors  [static], [private]
```

Initial value:

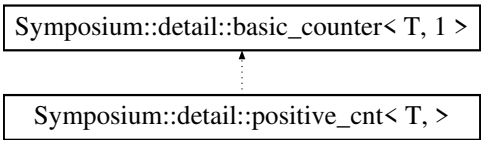
```
={"Action is not consistent with the message type", "This is not a client message",
                                                    "The action had not succeded", "This is not an askRes
       message",
                                                    "This is not a sendRes  message", "This is not a symbol
       message",
                                                    "This is not an updateActive message", "This is not an
       updateDoc message",
                                                    "This is not an userData message", "This is not a cursor
       message"}
```

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

- SymposiumException.h
- SymposiumException.cpp

6.33 Symposium::detail::positive_cnt< T, > Struct Template Reference

Inheritance diagram for Symposium::detail::positive_cnt< T, >:



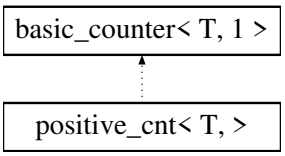
Additional Inherited Members

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

- Symposium.h

6.34 positive_cnt< T, > Struct Template Reference

Inheritance diagram for positive_cnt< T, >:



Additional Inherited Members

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

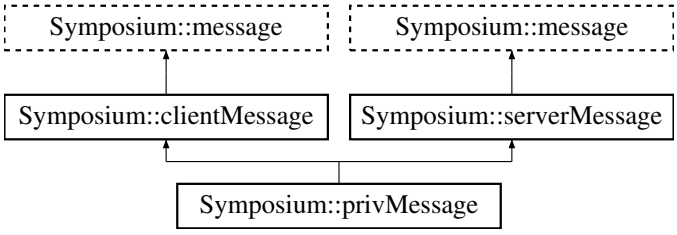
- counter.h

6.35 Symposium::privMessage Class Reference

class used to model a privilege change message sent by a client. It is also used to propagate such a change on other clients

`#include <message.h>`

Inheritance diagram for Symposium::privMessage:



Public Member Functions

- privMessage (msgType action, const std::pair< std::string, std::string > &actionOwner, msgOutcome result, const std::string &resourceId, const std::string &targetUser, privilege newPrivilege, uint_positive_cnt::type msgId=0)
- void invokeMethod (SymServer &server) override

  asks the server to modify a file privilege
- void invokeMethod (SymClient &client) override

  propagate the changes made by a client over a file privilege to other clients
- void completeAction (SymClient &client, msgOutcome serverResult) override

  completes an action for which the client asked to the server
- bool **operator==** (const privMessage &rhs) const
- bool **operator!=** (const privMessage &rhs) const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Private Attributes

- std::string resourceId
- std::string targetUser
- privilege newPrivilege

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.35.1 Detailed Description

class used to model a privilege change message sent by a client. It is also used to propagate such a change on other clients

The client sends the resource identifier for the target resource, the user whose privilege has to be changed and the new privilege. The server forward this message to other clients that share a privilege on that resource to inform them about what changed. Object of this class have *action=msgType::changePrivileges*

6.35.2 Constructor & Destructor Documentation

6.35.2.1 privMessage()

```
privMessage::privMessage (
            msgType action,
            const std::pair< std::string, std::string > & actionOwner,
            msgOutcome result,
            const std::string & resourceId,
            const std::string & targetUser,
            privilege newPrivilege,
            uint_positive_cnt::type msgId = 0 )
```

Exceptions

messageException	if *action* is not consistent with the message type

6.35.3 Member Function Documentation

6.35.3.1 completeAction()

```
void privMessage::completeAction (
            SymClient & client,
            msgOutcome serverResult )  [override], [virtual]
```

completes an action for which the client asked to the server

Parameters

client	the same client that had originated the clientMessage

Some actions on SymClient require to ask to the server to assure that the action is valid and to propagate the change on the server. For such actions, only if the outcome from the server is positive the action can be actually done.

Reimplemented from Symposium::clientMessage.

6.35.3.2 invokeMethod() [1/2]

```
void privMessage::invokeMethod (
            SymServer & server )  [override], [virtual]
```

asks the server to modify a file privilege

Parameters

server	the server the user is active on

When this message is received by the server, the *invokeMethod* calls SymServer::editPrivilege on the *server* passed as parameter.

Reimplemented from Symposium::clientMessage.

6.35.3.3 invokeMethod() [2/2]

```
void privMessage::invokeMethod (
            SymClient & client )  [override], [virtual]
```

propagate the changes made by a client over a file privilege to other clients

Parameters

client	the client on which propagate the change

When this message is received by the client, the *invokeMethod* calls SymClient::editPrivilege on the *client* passed as parameter.

Reimplemented from Symposium::serverMessage.

6.35.4 Member Data Documentation

6.35.4.1 newPrivilege

```
privilege Symposium::privMessage::newPrivilege  [private]
```

new privilege to assign to *targetUser* for *resourceId*

6.35.4.2 resourceId

```
std::string Symposium::privMessage::resourceId  [private]
```

path to the resource (the uri)

6.35.4.3 targetUser

```
std::string Symposium::privMessage::targetUser  [private]
```

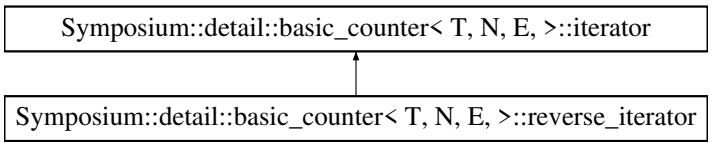
username of the user whose privilege on *resourceId* has to be changed

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

- message.h
- message.cpp

6.36 Symposium::detail::basic_counter< T, N, E, >::reverse_iterator Struct Reference

Inheritance diagram for Symposium::detail::basic_counter< T, N, E, >::reverse_iterator:



Public Member Functions

- **reverse_iterator** (basic_counter &c)
- reverse_iterator **begin** ()
- reverse_iterator **end** ()
- reverse_iterator & **operator++** ()
- reverse_iterator **operator++** (int)

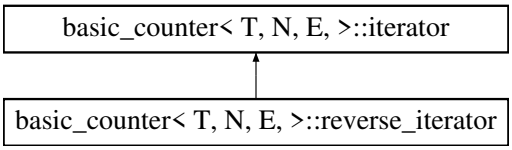
Additional Inherited Members

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

- Symposium.h

6.37 basic_counter< T, N, E, >::reverse_iterator Struct Reference

Inheritance diagram for basic_counter< T, N, E, >::reverse_iterator:



Public Member Functions

- **reverse_iterator** (basic_counter &c)
- reverse_iterator **begin** ()
- reverse_iterator **end** ()
- reverse_iterator & **operator++** ()
- reverse_iterator **operator++** (int)

Additional Inherited Members

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

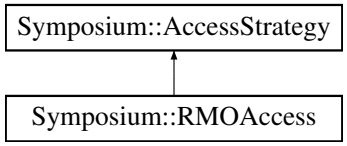
- counter.h

6.38 Symposium::RMOAccess Class Reference

class used to model a ReadModifyOwn privilege handling on a resource.

`#include <AccessStrategy.h>`

Inheritance diagram for Symposium::RMOAccess:



Public Member Functions

- bool validateAction (const std::string &targetUser, privilege requested) const override
  validate an action from user targetUser that requires requested
- privilege setPrivilege (const std::string &targetUser, privilege toGrant) override
  set the privilege of an user
- privilege **getPrivilege** (const std::string &targetUser) const override
- std::unordered_map< std::string, privilege > **getPermission** () const override
- bool **moreOwner** (std::string username) const override
- bool **operator==** (const RMOAccess &rhs) const
- bool **operator!=** (const RMOAccess &rhs) const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int)

Private Attributes

- std::unordered_map< std::string, privilege > permission

Friends

- class **boost::serialization::access**

6.38.1 Detailed Description

class used to model a ReadModifyOwn privilege handling on a resource.

6.38.2 Member Function Documentation

6.38.2.1 setPrivilege()

```
privilege RMOAccess::setPrivilege (
            const std::string & targetUser,
            privilege toGrant )  [override], [virtual]
```

set the privilege of an user

Parameters

targetUser	the user the privilege is to be granted
toGrant	the privilege to grant to targetUser

Returns

the privilege previously owned by targetUser, none if no privilege previously owned

Implements Symposium::AccessStrategy.

6.38.2.2 validateAction()

```
bool RMOAccess::validateAction (
            const std::string & targetUser,
            privilege requested ) const  [override], [virtual]
```

validate an action from user targetUser that requires requested

Parameters

targetUser	the user who is doing the action
requested	the permission requested by the action

Returns

true if the user is granted the privilege requested

Implements Symposium::AccessStrategy.

6.38.3 Member Data Documentation

6.38.3.1 permission

```
std::unordered_map<std::string, privilege> Symposium::RMOAccess::permission  [private]
```

username and related privilege for the resource

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

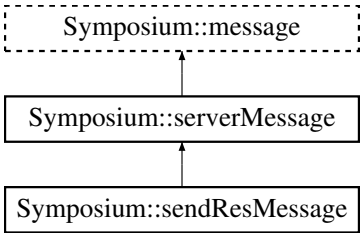
- AccessStrategy.h
- AccessStrategy.cpp

6.39 Symposium::sendResMessage Class Reference

class used to model an answer message sent by a server for a askResMessage

`#include <message.h>`

Inheritance diagram for Symposium::sendResMessage:



Public Member Functions

- sendResMessage (msgType action, msgOutcome result, std::shared_ptr< filesystem > resource, uint_positive_cnt::type symId=0, uint_positive_cnt::type msgId=0)
- void invokeMethod (SymClient &client) override

  make a client receive a new resource after a request

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Private Attributes

- uint_positive_cnt::type symId {}
- std::shared_ptr< filesystem > **resource**

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.39.1 Detailed Description

class used to model an answer message sent by a server for a askResMessage

6.39.2 Constructor & Destructor Documentation

6.39.2.1 sendResMessage()

```
sendResMessage::sendResMessage (
            msgType action,
            msgOutcome result,
            std::shared_ptr< filesystem > resource,
            uint_positive_cnt::type symId = 0,
            uint_positive_cnt::type msgId = 0 )
```

Exceptions

messageException	if *action* is not consistent with the message type

6.39.3 Member Function Documentation

6.39.3.1 invokeMethod()

```
void sendResMessage::invokeMethod (
            SymClient & client )  [override], [virtual]
```

make a client receive a new resource after a request

Parameters

client	the client instance to update

Depending on the value of *action*, the *invokeMethod* ask for different actions on the client:

- *action=msgType::createRes* : calls SymClient::createNewSource
- *action=msgType::createNewDir* : calls SymClient::createNewDir
- *action=msgType::openNewRes* : calls SymClient::openNewSource
- *action=msgType::changeResName* : calls SymClient::renameResource
- *action=msgType::removeRes* : calls SymClient::removeResource

Reimplemented from Symposium::serverMessage.

6.39.4 Member Data Documentation

6.39.4.1 symId

```
uint_positive_cnt::type Symposium::sendResMessage::symId {}  [private]
```

in case of *action=msgType::openNewRes* , the id assigned to the symlink

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

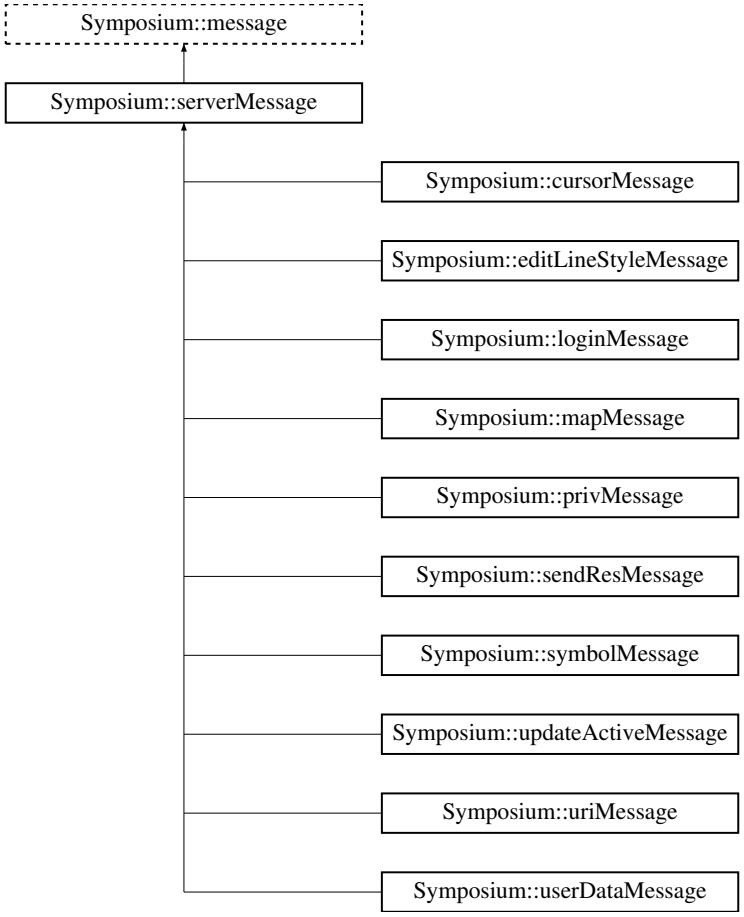
- message.h
- message.cpp

6.40 Symposium::serverMessage Class Reference

class used to model a message sent by the server

`#include <message.h>`

Inheritance diagram for Symposium::serverMessage:



Public Member Functions

- serverMessage (msgType action, msgOutcome result, uint_positive_cnt::type msgId=0)
- const std::string & **getErrDescr** () const
- void **setErrDescr** (const std::string &errDescr)
- msgOutcome **getResult** () const
- virtual void invokeMethod (SymClient &client)

  invoke an action on the SymClient given as parameter
- bool isRelatedTo (const clientMessage &other) const

  indicates of a message from a SymServer is related to a SymClient 's one
- bool **operator==** (const serverMessage &rhs) const
- bool **operator!=** (const serverMessage &rhs) const

Protected Member Functions

- **serverMessage** (msgOutcome result, uint_positive_cnt::type msgId=0)

Protected Attributes

- msgOutcome result
- std::string **errDescr**

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.40.1 Detailed Description

class used to model a message sent by the server

6.40.2 Constructor & Destructor Documentation

6.40.2.1 serverMessage()

```
serverMessage::serverMessage (
            msgType action,
            msgOutcome result,
            uint_positive_cnt::type msgId = 0 )
```

Exceptions

messageException	if *action* is not consistent with the message type

6.40.3 Member Function Documentation

6.40.3.1 invokeMethod()

```
void serverMessage::invokeMethod (
            SymClient & client )  [virtual]
```

invoke an action on the SymClient given as parameter

Parameters

client	the client instance on which perform the action

Reimplemented in Symposium::editLineStyleMessage, Symposium::cursorMessage, Symposium::userDataMessage, Symposium::uriMessage, Symposium::symbolMessage, Symposium::privMessage, Symposium::updateActiveMessage, Symposium::sendResMessage, Symposium::mapMessage, and Symposium::loginMessage.

6.40.3.2 isRelatedTo()

```
bool serverMessage::isRelatedTo (
            const clientMessage & other ) const
```

indicates of a message from a SymServer is related to a SymClient 's one

Parameters

other	the message this message could be in response to

Returns

true if the current message is in response to the one passed as argument; false otherwise

6.40.4 Member Data Documentation

6.40.4.1 result

```
msgOutcome Symposium::serverMessage::result  [protected]
```

result of an operation asked to the server

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

- message.h
- message.cpp

6.41 Symposium::sessionData Struct Reference

Public Member Functions

- **sessionData** (privilege p=privilege::none, unsigned int row=0, unsigned int col=0)
- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int)
- bool **operator==** (const sessionData &rhs) const
- bool **operator!=** (const sessionData &rhs) const

Public Attributes

- unsigned int **row**
- unsigned int **col**
- std::chrono::system_clock::time_point **tmstmp**
- privilege **p**

Friends

- class **boost::serialization::access**

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

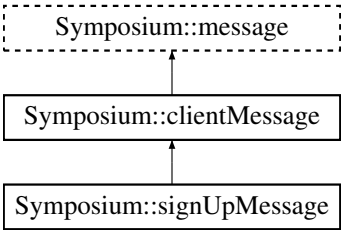
- document.h

6.42 Symposium::signUpMessage Class Reference

class used to model a sign up message sent by a client

`#include <message.h>`

Inheritance diagram for Symposium::signUpMessage:



Public Member Functions

- signUpMessage (msgType action, const std::pair< std::string, std::string > &actionOwner, const user &newUser, uint_positive_cnt::type msgId=0)
- const user & **getNewUser** () const
- void invokeMethod (SymServer &server) override

  make newUser a registered user of the server the message is sent to
- bool **operator==** (const signUpMessage &rhs) const
- bool **operator!=** (const signUpMessage &rhs) const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Private Attributes

- user newUser

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.42.1 Detailed Description

class used to model a sign up message sent by a client

6.42.2 Constructor & Destructor Documentation

6.42.2.1 signUpMessage()

```
signUpMessage::signUpMessage (
            msgType action,
            const std::pair< std::string, std::string > & actionOwner,
            const user & newUser,
            uint_positive_cnt::type msgId = 0 )
```

Exceptions

messageException	if *action* is not consistent with the message type

6.42.3 Member Function Documentation

6.42.3.1 invokeMethod()

```
void signUpMessage::invokeMethod (
            SymServer & server )  [override], [virtual]
```

make *newUser* a registered user of the server the message is sent to

Parameters

server	the server to sign up to

When this message is received by the server, the *invokeMethod* calls SymServer::addUser on the *server* passed as parameter.

Reimplemented from Symposium::clientMessage.

6.42.4 Member Data Documentation

6.42.4.1 newUser

```
user Symposium::signUpMessage::newUser  [private]
```

new SympUser data inserted by the user

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

- message.h
- message.cpp

6.43 Symposium::symbol Class Reference

Public Member Functions

- **symbol** (wchar_t ch, int siteId, int counter, const std::vector< int > &pos, bool verified=false)
- wchar_t **getCh** () const
- int **getSiteId** () const
- int **getCounter** () const
- const std::vector< int > & **getPos** () const
- bool **isVerified** () const
- symbol & **setVerified** ()
- bool **operator<** (const symbol &rhs) const
- bool **operator>** (const symbol &rhs) const
- bool **operator<=** (const symbol &rhs) const
- bool **operator>=** (const symbol &rhs) const
- bool **operator==** (const symbol &rhs) const
- bool **operator!=** (const symbol &rhs) const
- void **setCharFormat** (const format &value)
- format **getCharFormat** () const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int version)

Private Attributes

- wchar_t ch
- int siteId
- int **counter**
- std::vector< int > pos
- bool verified
- format charFormat

Friends

- class **boost::serialization::access**

6.43.1 Member Data Documentation

6.43.1.1 ch

```
wchar_t Symposium::symbol::ch  [private]
```

the character the symbols carries

6.43.1.2 charFormat

```
format Symposium::symbol::charFormat  [private]
```

attributes of the symbol for rich text editor

6.43.1.3 pos

```
std::vector<int> Symposium::symbol::pos  [private]
```

vector that defines the fractional indices of the symbol inside a document

6.43.1.4 siteId

```
int Symposium::symbol::siteId  [private]
```

identifies the user that generated the symbol

6.43.1.5 verified

```
bool Symposium::symbol::verified  [private]
```

true if the symbol has been verified by the server or has been inserted by remoteInsert

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

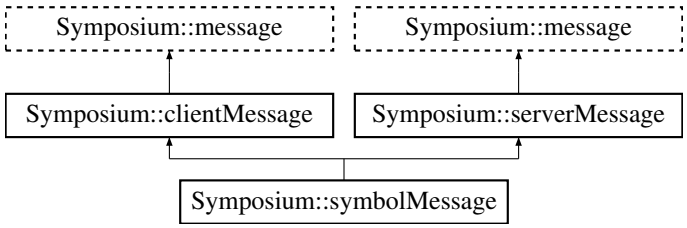
- symbol.h
- symbol.cpp

6.44 Symposium::symbolMessage Class Reference

class used to model a message regarding a symbol

```
#include <message.h>
```

Inheritance diagram for Symposium::symbolMessage:



Public Member Functions

- symbolMessage (msgType action, const std::pair< std::string, std::string > &actionOwner, msgOutcome result, uint_positive_cnt::type siteId, uint_positive_cnt::type resourceId, const symbol &sym, uint_positive_↩ cnt::type msgId=0)
- uint_positive_cnt::type **getSiteId** () const
- symbolMessage & verifySym ()

  set the contained symbol as "verified" calling symbol::setVerified on sym
- const symbol & getSym () const

  normal getter for member sym, normally used to inspect the content
- void invokeMethod (SymServer &server) override

  propagate the local change on a document content on the server
- void invokeMethod (SymClient &client) override

  propagate the changes made by a client over a document content to other clients
- void completeAction (SymClient &client, msgOutcome serverResult) override

  confirm the insertion or deletion of a symbol made by a client or abort it
- bool **operator==** (const symbolMessage &rhs) const
- bool **operator!=** (const symbolMessage &rhs) const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Private Attributes

- uint_positive_cnt::type siteId
- uint_positive_cnt::type resourceId
- symbol sym

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.44.1 Detailed Description

class used to model a message regarding a symbol

The client sends the symbol to be inserted or removed. The server forwards this message to other clients that are working on the same document. Object of this class have *action=msgType::insertSymbol* or *action=msgType::removeSymbol*

6.44.2 Constructor & Destructor Documentation

6.44.2.1 symbolMessage()

```
symbolMessage::symbolMessage (
            msgType action,
            const std::pair< std::string, std::string > & actionOwner,
            msgOutcome result,
            uint_positive_cnt::type siteId,
            uint_positive_cnt::type resourceId,
            const symbol & sym,
            uint_positive_cnt::type msgId = 0 )
```

Exceptions

messageException	if *action* is not consistent with the message type

6.44.3 Member Function Documentation

6.44.3.1 completeAction()

```
void symbolMessage::completeAction (
            SymClient & client,
            msgOutcome serverResult )  [override], [virtual]
```

confirm the insertion or deletion of a symbol made by a client or abort it

Parameters

client	the client which sent the message

Depending on the value of *action* and *result*, the *invokeMethod* ask for different actions on the client:

- *result=msgOutcome::success* :
  - *action=msgType::insertSymbol* : calls SymClient::verifySymbol
  - *action=msgType::removeSymbol* : calls nothing
- *result=msgOutcome::failure* :
  - *action=msgType::insertSymbol* : calls SymClient::remoteRemove
  - *action=msgType::removeSymbol* : calls SymClient::remoteInsert

Reimplemented from Symposium::clientMessage.

6.44.3.2 getSym()

```
const symbol & symbolMessage::getSym ( ) const
```

normal getter for member *sym*, normally used to inspect the content

Returns

a const reference to the symbol hold by the message

6.44.3.3 invokeMethod() [1/2]

```
void symbolMessage::invokeMethod (
            SymServer & server )  [override], [virtual]
```

propagate the local change on a document content on the server

Parameters

server	the server the user is active on

Depending on the value of *action*, the *invokeMethod* ask for different actions on the server:

- *action=msgType::insertSymbol* : calls SymServer::remoteInsert
- *action=msgType::removeSymbol* : calls SymServer::remoteRemove

Reimplemented from Symposium::clientMessage.

6.44.3.4 invokeMethod() [2/2]

```
void symbolMessage::invokeMethod (
            SymClient & client )  [override], [virtual]
```

propagate the changes made by a client over a document content to other clients

Parameters

client	the client on which propagate the change

Depending on the value of *action*, the *invokeMethod* ask for different actions on the client:

- *action=msgType::insertSymbol* : calls SymClient::remoteInsert
- *action=msgType::removeSymbol* : calls SymClient::remoteRemove

Reimplemented from Symposium::serverMessage.

6.44.3.5 verifySym()

```
symbolMessage & symbolMessage::verifySym ( )
```

set the contained symbol as "verified" calling symbol::setVerified on *sym*

Returns

a reference to the symbolMessage itself

6.44.4 Member Data Documentation

6.44.4.1 resourceId

```
uint_positive_cnt::type Symposium::symbolMessage::resourceId  [private]
```

id of the resource the client is working on

6.44.4.2 siteId

```
uint_positive_cnt::type Symposium::symbolMessage::siteId  [private]
```

siteId of the client that send the message

6.44.4.3 sym

```
symbol Symposium::symbolMessage::sym  [private]
```

symbol to be inserted or deleted

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

- message.h
- message.cpp

6.45 Symposium::SymClient Class Reference

class used to model a client of Symposium system

```
#include <SymClient.h>
```

Public Member Functions

- signUpMessage signUp (const std::string &username, const std::string &pwd, const std::string &nickname, const std::string &iconPath)

  constructs a signUpMessage to send to the server to ask for registration
- virtual void signUp (const user &logged)

  assign to loggedUser the user returned by the server after having sent a signUpMessage
- clientMessage logIn (const std::string &username, const std::string &pwd)

  constructs a clientMessage to send to the server to aks for authentication
- virtual void logIn (const user &logged)

  assign to loggedUser the user returned by the server after having sent a clientMessage
- askResMessage openSource (const std::string &resPath, const std::string &resId, privilege reqPriv)

  constructs a askResMessage to send to the server to ask to open a document
- virtual void openSource (const std::string &resPath, const std::string &resId, const std::shared_ptr< file > fileAsked, privilege reqPriv)

  add to activeFile the fileAsked and opens the document adding it to activeDoc
- virtual askResMessage openNewSource (const std::string &absolutePath, privilege reqPriv, const std::string &destPath, const std::string &destName="")

  constructs a askResMessage to send to the server to ask to open a document
- virtual void openNewSource (const std::string &absolutePath, privilege reqPriv, const std::string &destPath, const std::string &destName, uint_positive_cnt::type idToAssign, const std::shared_ptr< file > fileAsked)

  add to activeFile the fileAsked and opens the document adding it to activeDoc
- virtual askResMessage createNewSource (const std::string &resPath, const std::string &resName)

  constructs a askResMessage to send to the server to ask to create a file
- virtual void createNewSource (const std::string &resPath, const std::string &resName, uint_positive_cnt::type idToAssign, const std::shared_ptr< file > fileCreated)

add to activeFile the fileAsked and opens the document adding it to activeDoc

- virtual askResMessage createNewDir (const std::string &resPath, const std::string &resName)

  constructs a askResMessage to send to the server to ask to create a directory

- virtual void createNewDir (const std::string &resPath, const std::string &resName, uint_positive_cnt::type id↩ ToAssign)

  add the directory to user's filesystem

- virtual symbolMessage localInsert (uint_positive_cnt::type docId, const symbol &newSym, const std::pair< unsigned int, unsigned int > &index)

  insert a symbol on an opened document and constructs a message to sent to the server

- virtual symbolMessage localRemove (uint_positive_cnt::type docId, const std::pair< unsigned int, unsigned int > &indexes)

  remove a symbol on an opened document and constructs a message to sent to the server

- virtual void remoteInsert (uint_positive_cnt::type siteId, uint_positive_cnt::type docId, const symbol &new↩ Sym)

  propagate a symbol insertion on document content made by another user

- virtual void remoteRemove (uint_positive_cnt::type siteId, uint_positive_cnt::type docId, const symbol &rm↩ Sym)

  propagate a symbol deletion on document content made by another user

- virtual void verifySymbol (uint_positive_cnt::type docId, const symbol &sym)

  set the symbol of the opened document that has resourceId to "verified"

- virtual cursorMessage updateCursorPos (uint_positive_cnt::type docId, unsigned int row, unsigned int col)

  constructs a message to sent to the server to inform other users that local user has moved his cursor

- virtual void updateCursorPos (uint_positive_cnt::type userSiteId, uint_positive_cnt::type docId, unsigned int row, unsigned int col)

  update the position of user's cursor that has that userSiteId on the document

- virtual privMessage editPrivilege (const std::string &targetUser, const std::string &resPath, const std::string &resId, privilege newPrivilege)

  constructs a privMessage to send to the server to ask to change privileges for a resource

- virtual privilege editPrivilege (const std::string &targetUser, const std::string &resPath, const std::string &res↩ Id, privilege newPrivilege, bool msgRcv)

  edit the privilege of targetUser user for the resource resName in resPath to newPrivilege

- virtual uriMessage shareResource (const std::string &resPath, const std::string &resId, const uri &newPrefs)

  constructs a uriMessage to send to the server to ask to change sharing preferences for a resource

- virtual std::shared_ptr< filesystem > shareResource (const std::string &actionUser, const std::string &res↩ Path, const std::string &resId, const uri &newPrefs, bool msgRcv)

  set new sharing preferences for a resource

- virtual askResMessage renameResource (const std::string &resPath, const std::string &resId, const std↩ ::string &newName)

  constructs a uriMessage to send to the server to ask to change the name of a resource

- virtual std::shared_ptr< filesystem > renameResource (const std::string &resPath, const std::string &resId, const std::string &newName, bool msgRcv)

  removes a resource from remover 's home directory

- virtual askResMessage removeResource (const std::string &resPath, const std::string &resId)

  constructs a askResMessage to send to the server to ask to remove a resource

- virtual std::shared_ptr< filesystem > removeResource (const std::string &resPath, const std::string &resId, bool msgRcv)

  removes a resource from remover 's home directory

- std::string showDir (bool recursive=false) const

  show the content of user's root directory

- updateDocMessage closeSource (uint_positive_cnt::type docId)

  close a document for a user

- virtual userDataMessage editUser (user &newUserData)

changes user's data

- virtual const user editUser (user &newUserData, bool msgRcv)

  changes user's data
- clientMessage removeUser (const std::string &pwd)

  delete a user from the system
- virtual void removeUser (bool msgRcv)

  confirm deletion of user
- clientMessage logout ()

  ask to disconnect a user from the system
- virtual void logout (bool msgRcv)

  disconnect a user from the system
- updateDocMessage mapSiteIdToUser (const document &currentDoc)

  create a updateDocMessage to get from the server the mapping between username ad siteID of currentDoc 's activeUsers
- editLineStyleMessage localEditLineStyle (uint_positive_cnt::type docId, const std::pair< alignType, unsigned > &oldLineStyle, const std::pair< alignType, unsigned > &newLineStyle, unsigned row)

  edit the alignment and/or the indexStyle of a document's paragraph row
- virtual void remoteEditLineStyle (uint_positive_cnt::type docId, const std::pair< alignType, unsigned > &newLineStyle, unsigned row)

  propagate an update of line style on document made by another user
- virtual void setUserColors (uint_positive_cnt::type docId, const std::map< uint_positive_cnt::type, user > &siteIdToUser)

  assign to each user working on the same document on which loggedUser is working a unique color
- virtual void addActiveUser (uint_positive_cnt::type docId, user &targetUser, privilege Priv)

  add targetUser to the list of active users of the document
- virtual void removeActiveUser (uint_positive_cnt::type docId, user &targetUser)

  remove targetUser to the list of active users of the document
- virtual std::shared_ptr< clientMessage > retrieveRelatedMessage (const serverMessage &smex)

  retrieve a clientMessage previously sent by the client related to smex, removing it from unanswered
- virtual ~SymClient ()=default

  the default destroyer
- const std::forward_list< std::pair< const user *, sessionData > > onlineUsersonDocument (uint_positive_cnt::type documentID)

  it provides the online users list on the specified document
- const std::unordered_map< std::string, privilege > allUsersonDocument (uint_positive_cnt::type documentID)

  it provides the list of all users who modified the specified document
- const user & userData ()

  it provides the logged user with all data
- std::string directoryContent (std::string &ID_Cartella, std::string &path)

  show the content of directory with ID ID_Cartella
- Color colorOfUser (uint_positive_cnt::type resId, uint_positive_cnt::type siteId)

  retrieve the color assigned to an user for a document
- Color **colorOfUserbyUsername** (uint_positive_cnt::type resId, const std::string &username)
- const document & **getActiveDocumenttoOpenbyID** (uint_positive_cnt::type id)
- const file & **getActiveFiletoOpenbyID** (uint_positive_cnt::type id)
- bool **controlFileIsActive** (uint_positive_cnt::type id)
- bool **controlDocumentIsActive** (uint_positive_cnt::type id)

Protected Member Functions

- virtual const user & **getLoggedUser** () const

Protected Attributes

- user loggedUser
- std::map< std::string, std::pair< const user, int > > usersOnDocuments
- std::forward_list< std::shared_ptr< file > > activeFile
- std::forward_list< std::pair< document *, colorGen > > activeDoc
- std::map< std::pair< uint_positive_cnt::type, uint_positive_cnt::type >, std::pair< user, Color > > userColors
- std::forward_list< std::shared_ptr< clientMessage > > unanswered

Private Member Functions

- document * getActiveDocumentbyID (uint_positive_cnt::type id)

  get the document with specified ID
- colorGen & getColorGeneratorbyDocumentiID (uint_positive_cnt::type id)

  get the colorGen associated with document that it has the specified id
- const std::shared_ptr< file > getFilebyDocumentID (uint_positive_cnt::type id)

  get the file that content the document with specified ID
- virtual void setLoggedUser (const user &loggedUser)

  set all the details of the user just logged
- const user & **addUsersOnDocument** (const user &toInsert)
- const user & **getUsersOnDocument** (const std::string &username)
- void **removeUsersOnDocument** (const std::string &username)
- void **assignUsersColor** (colorGen &c, document &d)
- bool **handleDocException** (const std::function< void(void)> &op)

6.45.1 Detailed Description

class used to model a client of Symposium system

An instance of this class is intended to represent a logged user working on some documents. If a user wants to open multiple documents at the same time, then it can have multiple windows all sharing the same *loggedUser* object. The server is enabled to accept request for different documents by the same user

6.45.2 Member Function Documentation

6.45.2.1 addActiveUser()

```
void SymClient::addActiveUser (
            uint_positive_cnt::type docId,
            user & targetUser,
            privilege Priv )  [virtual]
```

add *targetUser* to the list of active users of the document

Parameters

docId	the id of the document the update refers to
targetUser	the user to add
Priv	the privilege of the user to add

6.45.2.2 allUsersonDocument()

```
const std::unordered_map< std::string, privilege > SymClient::allUsersonDocument (
            uint_positive_cnt::type documentID )
```

it provides the list of all users who modified the specified document

Parameters

documentID	the id of the document

Returns

the list of all users

6.45.2.3 closeSource()

```
updateDocMessage SymClient::closeSource (
            uint_positive_cnt::type docId )
```

close a document for a user

Parameters

docId	document to be closed

When a user client side wants to remove a resource, it sends a updateDocMessage and removes the document with id *resourceId* from *activeDoc* and *activeFile*.

6.45.2.4 colorOfUser()

```
Color SymClient::colorOfUser (
            uint_positive_cnt::type resId,
            uint_positive_cnt::type siteId )
```

retrieve the color assigned to an user for a document

Parameters

resId	the document the mapping is required for
siteId	the siteId of the user for which the color is asked

Returns

the Color corresponding to the user having *siteId* for the document having *resId*

6.45.2.5 createNewDir() [1/2]

```
askResMessage SymClient::createNewDir (
            const std::string & resPath,
            const std::string & resName )  [virtual]
```

constructs a askResMessage to send to the server to ask to create a directory

Parameters

resPath	the path of the directory to create, relative to user's home directory
resName	the name of the directory to create

Returns

the name of the file to create

6.45.2.6 createNewDir() [2/2]

```
void SymClient::createNewDir (
            const std::string & resPath,
            const std::string & resName,
            uint_positive_cnt::type idToAssign )  [virtual]
```

add the directory to user's filesystem

Parameters

resPath	is the path where the user want to put the directory into
resName	is the name to assign to the new directory
idToAssign	is the id assigned to the directory from the server

The new directory is created by calling user::newDirectory with the (*path*, *name*) taken from the previously sent askResMessage and *idToAssign* taken from the directory object contained in the sendResMessage just received

6.45.2.7 createNewSource() [1/2]

```
askResMessage SymClient::createNewSource (
            const std::string & resPath,
            const std::string & resName )  [virtual]
```

constructs a askResMessage to send to the server to ask to create a file

Parameters

resPath	the path of the file to create, relative to user's home directory
resName	the name of the file to create

Returns

a properly constructed askResMessage to send to the server

6.45.2.8 createNewSource() [2/2]

```
void SymClient::createNewSource (
            const std::string & resPath,
            const std::string & resName,
            uint_positive_cnt::type idToAssign,
            const std::shared_ptr< file > fileCreated )  [virtual]
```

add to *activeFile* the *fileAsked* and opens the document adding it to *activeDoc*

Parameters

resPath	is the path where the user want to put the file into
resName	is the name to assign to the new file
idToAssign	is the id assigned to the file from the server
fileCreated	the file sent back by the server in a sendResMessage

The new file is created by calling user::newFile with the (*path*, *name*) taken from the previously sent askResMessage and *idToAssign* taken from the file object contained in the sendResMessage just received

6.45.2.9 directoryContent()

```
std::string SymClient::directoryContent (
            std::string & ID_Cartella,
            std::string & path )
```

show the content of directory with ID ID_Cartella

Parameters

ID_Cartella	the ID of directory
path	path where the directory is located

Returns

a string with all content of request directory

6.45.2.10 editPrivilege() [1/2]

```
privMessage SymClient::editPrivilege (
            const std::string & targetUser,
            const std::string & resPath,
            const std::string & resId,
            privilege newPrivilege )  [virtual]
```

constructs a privMessage to send to the server to ask to change privileges for a resource

Parameters

targetUser	the user whose privilege has to be modified
resPath	the relative path of the resource
resId	the id of the resource
newPrivilege	the new privilege to be granted to *targetUser*

Returns

a properly constructed privMessage to send to the server

When a user client side want to edit the privilege of another user on a resource, it sends a PrivMessage. The server will answer with a serveMessage indicating if the action has been done successfully. The message is put on *unanswered*, so when the client will receive an answer for a message, it will invoke privilege SymClient::editPrivilege that will actually perform the privilege change.

6.45.2.11 editPrivilege() [2/2]

```
privilege SymClient::editPrivilege (
            const std::string & targetUser,
            const std::string & resPath,
            const std::string & resId,
            privilege newPrivilege,
            bool msgRcv )  [virtual]
```

edit the privilege of *targetUser* user for the resource *resName* in *resPath* to *newPrivilege*

Parameters

targetUser	the user whose privilege has to be modified
resPath	the relative path of the resource
resId	the id of the resource
newPrivilege	the new privilege to be granted to *targetUser*
msgRcv	indicate whether the method is called after having sent a privMessage

Returns

the old privilege of *targetUser* had on the resource

When a server answers to a privMessage, it sends a serverMessage indicating if the action has been done successfully. When a client receives this message, it searches for a related message in *unanswered*, and calls *completeAction* on it passing itself. The method *completeAction* of privMessage calls this function with msgRcv=true, while the method *invokeMethod* calls with msgRcv=false. This parameter msgRcv is used only to distinguish method signatures of this method and the one that returns a privMessage.

6.45.2.12 editUser() [1/2]

```
userDataMessage SymClient::editUser (
            user & newUserData )  [virtual]
```

changes user's data

Parameters

newUserData	a user object filled with new data

Returns

the old user data

When a user client side wants to change its data, it sends a userDataMessage. The server will answer with a serveMessage indicating if the action has been done successfully. The message is put on *unanswered*, so when the client will receive an answer for a message, it will invoke the analogous function *editUser(user&, bool)* to actually perform the changing if the outcome is positive.

6.45.2.13 editUser() [2/2]

```
const user SymClient::editUser (
            user & newUserData,
            bool msgRcv )  [virtual]
```

changes user's data

Parameters

newUserData	a user object filled with new data
msgRcv	indicates if the function is called because a userDataMessage was sent or not

Returns

the old user data

When the response for a userDataMessage is received, this function is called with msgRcv=true, elseif the function is called because a changing in user data has to be propagated to this client msgRcv=false

6.45.2.14 getActiveDocumentbyID()

```
document * SymClient::getActiveDocumentbyID (
            uint_positive_cnt::type id )  [private]
```

get the document with specified ID

Parameters

id	the id of the request document

Returns

the request document

6.45.2.15 getColorGeneratorbyDocumentiID()

```
colorGen & SymClient::getColorGeneratorbyDocumentiID (
            uint_positive_cnt::type id )  [private]
```

get the colorGen associated with document that it has the specified id

Parameters

id	the document the colorGen is required for

Returns

the colorGen associated with specified document

6.45.2.16 getFilebyDocumentID()

```
const std::shared_ptr< file > SymClient::getFilebyDocumentID (
            uint_positive_cnt::type id )  [private]
```

get the file that content the document with specified ID

Parameters

id	the id of the content document into the file

Returns

the file that content the document with specified ID

6.45.2.17 localEditLineStyle()

```
editLineStyleMessage SymClient::localEditLineStyle (
            uint_positive_cnt::type docId,
            const std::pair< alignType, unsigned > & oldLineStyle,
            const std::pair< alignType, unsigned > & newLineStyle,
            unsigned row )
```

edit the alignment and/or the indexStyle of a document's paragraph *row*

Parameters

docId	the id of the document that has to be modified
oldLineStyle	the actual alignment and indexStyle of the document at row *row*
newLineStyle	the new alignment and indexStyle to apply at row *row*
row	the document's row that has to be modified

Returns

a properly constructed editLineStyleMessage to send to the server

6.45.2.18 localInsert()

```
symbolMessage SymClient::localInsert (
            uint_positive_cnt::type docId,
            const symbol & newSym,
            const std::pair< unsigned int, unsigned int > & index )  [virtual]
```

insert a symbol on an opened document and constructs a message to sent to the server

Parameters

docId	the document the insertion refers to
newSym	the symbol to insert

Returns

a properly constructed symbolMessage to send to the server

6.45.2.19 localRemove()

```
symbolMessage SymClient::localRemove (
            uint_positive_cnt::type docId,
            const std::pair< unsigned int, unsigned int > & indexes )  [virtual]
```

remove a symbol on an opened document and constructs a message to sent to the server

Parameters

docId	the document the deletion refers to
indexes	the position of the symbol to remove

Returns

a properly constructed symbolMessage to send to the server

6.45.2.20 logIn() [1/2]

```
clientMessage SymClient::logIn (
            const std::string & username,
            const std::string & pwd )
```

constructs a clientMessage to send to the server to aks for authentication

Parameters

username	the username of the user to login
pwd	the password the user chase

Returns

a properly constructed clientMessage to send to the server

6.45.2.21 logIn() [2/2]

```
void SymClient::logIn (
            const user & logged )  [virtual]
```

assign to *loggedUser* the user returned by the server after having sent a clientMessage

Parameters

logged	the user sent back by the server in a signUpMessage

6.45.2.22 logout()

```
clientMessage SymClient::logout ( )
```

ask to disconnect a user from the system

Returns

a properly constructed clientMessage to send to the server

6.45.2.23 mapSiteIdToUser()

```
updateDocMessage SymClient::mapSiteIdToUser (
            const document & currentDoc )
```

create a updateDocMessage to get from the server the mapping between username ad siteID of *currentDoc* 's activeUsers

Parameters

currentDoc	document you are working on

Returns

a properly constructed updateDocMessage to send to the server

6.45.2.24 onlineUsersonDocument()

```
const std::forward_list< std::pair< const user *, sessionData > > SymClient::onlineUserson↩
Document (
            uint_positive_cnt::type documentID )
```

it provides the online users list on the specified document

Parameters

documentID	the id of the document

Returns

the list of online users

6.45.2.25 openNewSource() [1/2]

```
askResMessage SymClient::openNewSource (
            const std::string & absolutePath,
            privilege reqPriv,
            const std::string & destPath,
            const std::string & destName = "" )  [virtual]
```

constructs a askResMessage to send to the server to ask to open a document

Parameters

absolutePath	the absolute path to the resource to open
reqPriv	the privilege requested opening the file
destPath	the path inside user's home directory where to put a symlink to the file

Returns

a properly constructed askResMessage to send to the server

6.45.2.26 openNewSource() [2/2]

```
void SymClient::openNewSource (
            const std::string & absolutePath,
            privilege reqPriv,
            const std::string & destPath,
            const std::string & destName,
            uint_positive_cnt::type idToAssign,
            const std::shared_ptr< file > fileAsked )  [virtual]
```

add to *activeFile* the *fileAsked* and opens the document adding it to *activeDoc*

Parameters

absolutePath	the absolute path to the resource to open
reqPriv	the privilege requested opening the file
destPath	the path where to put the symlink to *name*, inside *opener* 's home directory
destName	the name to assign to the symlink
idToAssign	is the id assigned to the file from the server
fileAsked	the file sent back by the server in a sendResMessage

6.45.2.27 openSource() [1/2]

```
askResMessage SymClient::openSource (
            const std::string & resPath,
            const std::string & resId,
            privilege reqPriv )
```

constructs a askResMessage to send to the server to ask to open a document

Parameters

resPath	the path of the file to open, relative to user's home directory
resId	the id of the file to open
reqPriv	the privilege requested opening the file

Returns

a properly constructed askResMessage to send to the server

6.45.2.28 openSource() [2/2]

```
void SymClient::openSource (
            const std::string & resPath,
            const std::string & resId,
            const std::shared_ptr< file > fileAsked,
            privilege reqPriv )  [virtual]
```

add to *activeFile* the *fileAsked* and opens the document adding it to *activeDoc*

Parameters

resPath	the path of the file to open, relative to user's home directory
resId	the id of the file to open
fileAsked	the file sent back by the server in a sendResMessage

6.45.2.29 remoteEditLineStyle()

```
void SymClient::remoteEditLineStyle (
            uint_positive_cnt::type docId,
            const std::pair< alignType, unsigned > & newLineStyle,
            unsigned row )  [virtual]
```

propagate an update of line style on document made by another user

Parameters

docId	the id of the document that has been modified
newLineStyle	the new alignment and indexStyle applied at row *row*
row	the document's row that has been modified

6.45.2.30 remoteInsert()

```
void SymClient::remoteInsert (
            uint_positive_cnt::type siteId,
            uint_positive_cnt::type docId,
            const symbol & newSym )  [virtual]
```

propagate a symbol insertion on document content made by another user

Parameters

siteId	the site id of the user performing the insertion
docId	the document the insertion refers to
newSym	the symbol to insert

This method is called after having received a symbolMessage

6.45.2.31 remoteRemove()

```
void SymClient::remoteRemove (
            uint_positive_cnt::type siteId,
            uint_positive_cnt::type docId,
            const symbol & rmSym )  [virtual]
```

propagate a symbol deletion on document content made by another user

Parameters

siteId	the site id of the user performing the removal
docId	the document the deletion refers to
rmSym	the symbol to remove

This method is called after having received a symbolMessage

6.45.2.32 removeActiveUser()

```
void SymClient::removeActiveUser (
            uint_positive_cnt::type docId,
            user & targetUser )  [virtual]
```

remove *targetUser* to the list of active users of the document

Parameters

docId	the id of the document the update refers to
targetUser	the user to remove

6.45.2.33 removeResource() [1/2]

```
askResMessage SymClient::removeResource (
            const std::string & resPath,
            const std::string & resId )  [virtual]
```

constructs a askResMessage to send to the server to ask to remove a resource

Parameters

resPath	the relative path to the user's *home* directory where to delete the file
resId	the resource's id

Returns

a properly constructed askResMessage to send to the server

When a user client side wants remove a resource, it sends a askResMessage. The server will answer with a serve↩ Message indicating if the action has been done successfully. The message is put on *unanswered*, so when the client will receive an answer for a message, it will invoke std::shared_ptr<filesystem> SymClient::removeResource that will actually perform the removing.

6.45.2.34 removeResource() [2/2]

```
std::shared_ptr< filesystem > SymClient::removeResource (
            const std::string & resPath,
            const std::string & resId,
            bool msgRcv )  [virtual]
```

removes a resource from *remover* 's *home* directory

Parameters

resPath	the relative path to the user's *home* directory where to create the file
resId	the resource's id
msgRcv	indicates whether this method is called after having received a serverMessage

Returns

the resource just removed

When a server answers to a askResMessage, it sends a serverMessage indicating if the action has been done successfully. When a client receives this message, it searches for a related message in *unanswered*, and calls *completeAction* on it passing itself. The method *completeAction* of askResMessage calls this function with msg↩ Rcv=true, while the method *invokeMethod* calls with msgRcv=false. This parameter msgRcv is used only to distinguish method signatures of this method and the one that returns a askResMessage.

6.45.2.35 removeUser() [1/2]

```
clientMessage SymClient::removeUser (
            const std::string & pwd )
```

delete a user from the system

Parameters

pwd	the password the user chase

Returns

a properly constructed clientMessage to send to the server

6.45.2.36 removeUser() [2/2]

```
void SymClient::removeUser (
            bool msgRcv )  [virtual]
```

confirm deletion of user

Parameters

msgRcv	indicates if the function is called because a userDataMessage was sent or not

6.45.2.37 renameResource() [1/2]

```
askResMessage SymClient::renameResource (
            const std::string & resPath,
            const std::string & resId,
            const std::string & newName )  [virtual]
```

constructs a uriMessage to send to the server to ask to change the name of a resource

Parameters

resPath	the relative path to the user's *home* directory where to create the file
resId	the resource's id
newName	the new resource's name

Returns

a properly constructed askResMessage to send to the server

When a user client side wants to set a new name for a resource, it sends a askResMessage. The server will answer with a serveMessage indicating if the action has been done successfully. The message is put on *unanswered*, so when the client will receive an answer for a message, it will invoke std::shared_ptr<filesystem> SymClient::renameResource that will actually perform the renaming.

6.45.2.38 renameResource() [2/2]

```
std::shared_ptr< filesystem > SymClient::renameResource (
            const std::string & resPath,
            const std::string & resId,
            const std::string & newName,
            bool msgRcv )  [virtual]
```

removes a resource from *remover*'s *home* directory

Parameters

resPath	the relative path to the user's *home* directory where to create the file
resId	the resource's id
newName	the new resource's name
msgRcv	indicates whether this method is called after having received a serverMessage

Returns

the resource just renamed

When a server answers to a askResMessage, it sends a serverMessage indicating if the action has been done successfully. When a client receives this message, it searches for a related message in *unanswered*, and calls *completeAction* on it passing itself. The method *completeAction* of askResMessage calls this function with msg↩ Rcv=true, while the method *invokeMethod* calls with msgRcv=false. This parameter msgRcv is used only to distinguish method signatures of this method and the one that returns a askResMessage.

6.45.2.39 retrieveRelatedMessage()

```
std::shared_ptr< clientMessage > SymClient::retrieveRelatedMessage (
            const serverMessage & smex )  [virtual]
```

retrieve a clientMessage previously sent by the client related to *smex*, removing it from *unanswered*

Parameters

smex	the serverMessage sent by the server that is supposed to be a response for a previous clientMessage

Returns

the clientMessage that is related to the received *smex*

6.45.2.40 setLoggedUser()

```
void SymClient::setLoggedUser (
            const user & loggedUser )  [private], [virtual]
```

set all the details of the user just logged

Parameters

loggedUser	the user object containing all the information of the logged user

When the client wants to perform login it calls the logIn method. When the server answers, the loginMessage calls setLoggedUser, passing the user object transmitted by the user

6.45.2.41 setUserColors()

```
void SymClient::setUserColors (
            uint_positive_cnt::type docId,
            const std::map< uint_positive_cnt::type, user > & siteIdToUser )  [virtual]
```

assign to each user working on the same document on which *loggedUser* is working a unique color

Parameters

docId	the identified of the document the mapping was requested for
siteIdToUser	the mapping siteId->user asked to the server

Receive the mapping siteId->user from the server and assigns to each user a unique color among the colors assigned to the users working on the same document.

6.45.2.42 shareResource() [1/2]

```
uriMessage SymClient::shareResource (
            const std::string & resPath,
            const std::string & resId,
            const uri & newPrefs )  [virtual]
```

constructs a uriMessage to send to the server to ask to change sharing preferences for a resource

Parameters

resPath	the relative path of the resource
resId	the id of the resource
newPrefs	new sharing preferences to set the resource to

Returns

a properly constructed uriMessage to send to the server

When a user client side want to share a resource with another user, it sends a uriMessage. The server will answer with a serveMessage indicating if the action has been done successfully. The message is put on *unanswered*, so when the client will receive an answer for a message, it will invoke uri SymClient::shareResource that will actually perform the sharing preference change.

6.45.2.43 shareResource() [2/2]

```
std::shared_ptr< filesystem > SymClient::shareResource (
            const std::string & actionUser,
            const std::string & resPath,
            const std::string & resId,
            const uri & newPrefs,
            bool msgRcv )  [virtual]
```

set new sharing preferences for a resource

Parameters

resPath	the relative path of the resource
resId	the id of the resource
newPrefs	new sharing preferences for the resource
msgRcv	indicate whether the method is called after having sent a uriMessage

Returns

the old *sharingPolicy*

When a server answers to a uriMessage, it sends a serverMessage indicating if the action has been done successfully. When a client receives this message, it searches for a related message in *unanswered*, and calls *completeAction* on it passing itself. The method *completeAction* of uriMessage calls this function with msgRcv=true, while the method *invokeMethod* calls with msgRcv=false. This parameter msgRcv is used only to distinguish method signatures of this method and the one that returns a uriMessage.

6.45.2.44 showDir()

```
std::string SymClient::showDir (
            bool recursive = false ) const
```

show the content of user's root directory

Parameters

recursive	true if the method should be recursive, false otherwise

Returns

a string with all content of user's root directory

6.45.2.45 signUp() [1/2]

```
signUpMessage SymClient::signUp (
            const std::string & username,
            const std::string & pwd,
            const std::string & nickname,
            const std::string & iconPath )
```

constructs a signUpMessage to send to the server to ask for registration

Parameters

username	the username of the user to register
pwd	the password the user has chosen
nickname	the nickName chosen by the user
iconPath	the path (relative to the software directory) of the chosen icon

Returns

a properly constructed signUpMessage to send to the server

6.45.2.46 signUp() [2/2]

```
void SymClient::signUp (
            const user & logged )  [virtual]
```

assign to *loggedUser* the user returned by the server after having sent a signUpMessage

Parameters

logged	the user sent back by the server in a signUpMessage

6.45.2.47 updateCursorPos() [1/2]

```
cursorMessage SymClient::updateCursorPos (
            uint_positive_cnt::type docId,
            unsigned int row,
            unsigned int col )  [virtual]
```

constructs a message to sent to the server to inform other users that local user has moved his cursor

Parameters

doc↩ Id	the id of the document
row	X coordinate of the new position of cursor
col	Y coordinate of the new position of cursor

Returns

a properly constructed cursorMessage to send to the server

6.45.2.48 updateCursorPos() [2/2]

```
void SymClient::updateCursorPos (
            uint_positive_cnt::type userSiteId,
            uint_positive_cnt::type docId,
            unsigned int row,
            unsigned int col )  [virtual]
```

update the position of user's cursor that has that userSiteId on the document

Parameters

user↩ SiteId	the site id of the user moving his cursor
docId	the id of the document
row	X coordinate of the new position of user's cursor
col	Y coordinate of the new position of user's cursor

This method is called after having received a cursorMessage

6.45.2.49 userData()

```
const user & SymClient::userData ( )
```

it provides the logged user with all data

Returns

the logged user

6.45.2.50 verifySymbol()

```
void SymClient::verifySymbol (
            uint_positive_cnt::type docId,
            const symbol & sym )  [virtual]
```

set the symbol of the opened document that has *resourceId* to "verified"

Parameters

doc↩ Id	the id of the document the symbol refers to
sym	the symbol to verify

When the server answers to a client's symbolMessage, it sends a serverMessage containing the outcome of the action. When this message is received by the client, the related previously sent message is retrieved and symbolMessage::completeAction is called: it calls this method if the outcome is positive.

6.45.3 Member Data Documentation

6.45.3.1 activeDoc

```
std::forward_list<std::pair<document *, colorGen> > Symposium::SymClient::activeDoc  [protected]
```

list of files the active documents are related to

6.45.3.2 activeFile

```
std::forward_list<std::shared_ptr<file> > Symposium::SymClient::activeFile [protected]
```

list of active documents

6.45.3.3 loggedUser

```
user Symposium::SymClient::loggedUser [protected]
```

users map with whom we have open documents in common, with counter for each user

6.45.3.4 unanswered

```
std::forward_list<std::shared_ptr<clientMessage> > Symposium::SymClient::unanswered [protected]
```

messages sent by client that have not been received an answer

6.45.3.5 userColors

```
std::map<std::pair<uint_positive_cnt::type, uint_positive_cnt::type>, std::pair<user, Color>
> Symposium::SymClient::userColors [protected]
```

map {siteId, documentId}->{user, color}

6.45.3.6 usersOnDocuments

```
std::map<std::string, std::pair<const user, int> > Symposium::SymClient::usersOnDocuments
[protected]
```

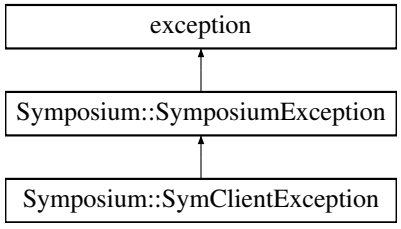
active users, indexed by username

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

- SymClient.h
- SymClient.cpp

6.46 Symposium::SymClientException Class Reference

Inheritance diagram for Symposium::SymClientException:



Public Types

- enum SymClientExceptionCodes { **noActiveDocument** =0, **noRelatedMessage**, **nocolorOfUser**, **noActiveFile** }

  Specific error codes for SymClientException. They are used as indexes to the error table string.

Public Member Functions

- **SymClientException** (SymClientExceptionCodes exceptionCode, const char *file, int line, const char *func)

Static Private Attributes

- static const char * **SymClientErrors** [ ] ={"No active document with that ID", "No relative message found", "User not found on that document", "No active file with that ID"}

Additional Inherited Members

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

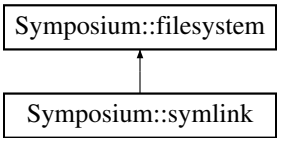
- SymposiumException.h
- SymposiumException.cpp

6.47 Symposium::symlink Class Reference

class used to model a pointer to an object of class file

`#include <filesystem.h>`

Inheritance diagram for Symposium::symlink:



Public Member Functions

- **symlink** (const std::string &symName, const std::string &absPathWithoutId, const std::string &resId, uint_positive_cnt::type idToAssign=0)
- std::string **getPath** ()
- const std::string & **getResId** () const
- resourceType resType () const override

  identify the type of current filesystem resource
- document & access (const user &targetUser, privilege accessMode)

  access the file named resId located in absPathWithoutId
- virtual std::string print (const std::string &targetUser, bool recursive=false, unsigned int indent=0) const override

  give a textual representation of the symlink
- virtual bool isReadyToRemove (const std::string &username) const override

  the remove operation is possible if: the user is not only owner of the file pointed, or the user has lower privilege then owner or if the user is the only owner of the file pointed this method has to check that the file pointed doesn't have any active user

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int version)

Private Attributes

- std::string absPathWithoutId
- std::string resId

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.47.1 Detailed Description

class used to model a pointer to an object of class file

In the application design only pointers to objects of class file are allowed, but *absPathWithoutId* could point to a directory. As long as th

6.47.2 Member Function Documentation

6.47.2.1 access()

```
document & Symposium::symlink::access (
            const user & targetUser,
            privilege accessMode )
```

access the file named *resId* located in *absPathWithoutId*

Retrieves the resource indicated in parameters and call the method @ access on it. The resource should be a file, because pointers to directories are not allowed in this system design and pointers to symlink are meaningless.

6.47.2.2 isReadyToRemove()

```
bool Symposium::symlink::isReadyToRemove (
            const std::string & username ) const  [override], [virtual]
```

the remove operation is possible if: the user is not only owner of the file pointed, or the user has lower privilege then owner or if the user is the only owner of the file pointed this method has to check that the file pointed doesn't have any active user

Parameters

username	the user who wants to perform the operation

Returns

true if the operation is possible, false otherwise

Implements Symposium::filesystem.

6.47.2.3 print()

```
std::string Symposium::symlink::print (
            const std::string & targetUser,
            bool recursive = false,
            unsigned int indent = 0 ) const  [override], [virtual]
```

give a textual representation of the symlink

Parameters

targetUser	the user who asked for this action
recursive	for a symlink is meaningless
indent	an optional identation level to distinguish nested objects

Returns

a string containing the representation

For a symlink, *print(targetUser)* shows the name of the symlink and the privileges granted to *targetUser* for the file pointed by the symlink

Implements Symposium::filesystem.

6.47.2.4 resType()

```
resourceType Symposium::symlink::resType ( ) const  [override], [virtual]
```

identify the type of current filesystem resource

Returns

the type of the current filesystem object (file, directory, symlink)

Implements Symposium::filesystem.

6.47.3 Member Data Documentation

6.47.3.1 absPathWithoutId

```
std::string Symposium::symlink::absPathWithoutId  [private]
```

absolute path to a *file*, obtained as concatenation of *id*

6.47.3.2 resId

```
std::string Symposium::symlink::resId  [private]
```

id of the file pointed

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

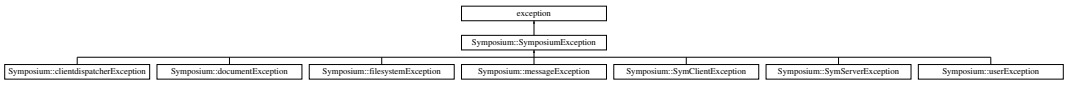
- filesystem.h
- filesystem.cpp

6.48 Symposium::SymposiumException Class Reference

class used as exception base class for classes in Symposium namespace. It composes a custom error explaining message in the form "function: [functionName], in file: [filename], line: [lineNumber]: [errorMessage]", allowing subclasses to specify an error message. @exceptsafe no-throw

```
#include <SymposiumException.h>
```

Inheritance diagram for Symposium::SymposiumException:



Public Member Functions

- const char * **what** () const noexcept override
- const char * **getErrorCodeMsg** () const noexcept

Protected Member Functions

- **SymposiumException** (const char *file, const int line, const char *func, const char *errDescr)

Private Attributes

- char **errorMsg** [msgMaxLen]
- const char * **errorCodeMsg**

Static Private Attributes

- static constexpr int **msgMaxLen** = 400

6.48.1 Detailed Description

class used as exception base class for classes in Symposium namespace. It composes a custom error explaining message in the form "function: [functionName], in file: [filename], line: [lineNumber]: [errorMessage]", allowing subclasses to specify an error message. @exceptsafe no-throw

Every specific Symposium exception class must derive from this class and use the SymposiumException constructor. The exception safety is no-throw because the class is designed to encode the error explanation string starting from C-style const literals composed in a static-allocated C-style string of predefined constant length. Function *snprintf* is used in constructor, so buffer overflow is avoided. No memory is allocated nor de-allocated in methods.

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

- SymposiumException.h
- SymposiumException.cpp

6.49 Symposium::SymServer Class Reference

Public Member Functions

- **SymServer** (bool loading=true, bool storing=true)
- SymServer & **operator=** (SymServer &&other)=default
- bool **operator==** (const SymServer &rhs) const
- bool **operator!=** (const SymServer &rhs) const
- virtual const user & addUser (user &newUser, uint_positive_cnt::type respMsgId)

  adds a new user to the set of users registered to the system
- virtual const user & login (const std::string &username, const std::string &pwd, uint_positive_cnt::type respMsgId)

  log in an already registered user, adding it to active
- virtual std::shared_ptr< file > openSource (const std::string &opener, const std::string &resPath, const std::string &resId, privilege reqPriv, uint_positive_cnt::type respMsgId)

  access a document that is already in the user's filesystem
- virtual std::shared_ptr< file > openNewSource (const std::string &opener, const std::string &absolutePath, const std::string &destPath, const std::string &destName, privilege reqPriv, uint_positive_cnt::type respMsgId)

  access a user's document via uri to the filesystem of the another user
- virtual const document & createNewSource (const std::string &opener, const std::string &resPath, const std::string &resName, uint_positive_cnt::type respMsgId)

  creates a new file with an empty document inside
- virtual std::shared_ptr< directory > createNewDir (const std::string &opener, const std::string &resPath, const std::string &resName, uint_positive_cnt::type respMsgId)

  creates a new directory in the user's filesystem
- virtual void remoteInsert (const std::string &inserter, uint_positive_cnt::type docId, const symbolMessage &symMsg)

  update a document with a new symbol from a client
- virtual void remoteRemove (const std::string &remover, uint_positive_cnt::type docId, const symbolMessage &rmMsg)

update a document removing a symbol

- virtual void updateCursorPos (const std::string &targetUser, uint_positive_cnt::type docId, const cursorMessage &crMsg)

  update the current position of user's cursor inside one of the documents he's working on
- virtual privilege editPrivilege (const std::string &actionUser, const std::string &targetUser, const std::string &resPath, const std::string &resId, privilege newPrivilege, uint_positive_cnt::type respMsgId)

  edit the privilege of targetUser user for the resource resName in resPath to newPrivilege
- virtual std::shared_ptr< filesystem > shareResource (const std::string &actionUser, const std::string &resPath, const std::string &resId, const uri &newPrefs, uint_positive_cnt::type respMsgId)

  set new sharing preferences for a resource
- virtual std::shared_ptr< filesystem > renameResource (const std::string &renamer, const std::string &resPath, const std::string &resId, const std::string &newName, uint_positive_cnt::type respMsgId)

  renames a resource from remover 's home directory
- virtual std::shared_ptr< filesystem > removeResource (const std::string &remover, const std::string &resPath, const std::string &resId, uint_positive_cnt::type respMsgId)

  removes a resource from remover 's home directory
- virtual void closeSource (const std::string &actionUser, uint_positive_cnt::type docId, uint_positive_cnt::type respMsgId)

  close a document for a user
- virtual const user & editUser (const std::string &username, user &newUserData, uint_positive_cnt::type respMsgId)

  changes user's data
- virtual void removeUser (const std::string &username, const std::string &pwd, uint_positive_cnt::type respMsgId)

  removes an user to the set of users registered to the system
- virtual void logout (const std::string &username, uint_positive_cnt::type respMsgId)

  performs a log out, removing the user from active
- virtual void hardLogout (uint_positive_cnt::type siteId)

  performs a logout as a result of unexpected detach of a client
- virtual std::map< uint_positive_cnt::type, user > mapSiteIdToUser (const std::string &actionUser, uint_positive_cnt::type docId, uint_positive_cnt::type respMsgId)

  maps siteIds to users to allow a client to identify the owner of each change in a document
- virtual void editLineStyle (const std::string &actionUser, uint_positive_cnt::type docId, const std::pair< alignType, unsigned > &newLineStyle, unsigned row, const editLineStyleMessage &editMsg)

  edit the alignment and/or indexStyle of the document's paragraph row inside one of the documents he's working on
- std::pair< const uint_positive_cnt::type, std::shared_ptr< serverMessage > > extractNextMessage ()

  extract a message from the message queue associated with an user
- uint_positive_cnt::type getSiteIdOfUser (const std::string &username) const

  retrieve the site id of an user
- void store () const

  store the server data (users registered and filesystem structure) onto the disk
- bool load ()

  load data (users registered and filesystem structure) from disk
- virtual ~SymServer ()

  performs storing operations of the data of the whole system

Public Attributes

- bool **loadData**
- bool storeData

Static Public Attributes

- static const user **unknownUser**
- static constexpr char storeFile [ ] ="./serverData.dat"

Protected Member Functions

- virtual user & registerUser (const user &toInsert)

  insert a new user in the table of registered users
- virtual user & getRegistered (const std::string &username)

  return the registered user associated with username
- virtual void **removeRegistered** (const std::string &username)
- virtual bool **userIsRegistered** (const std::string &toCheck) const noexcept
- virtual const user & **findUserBySiteId** (int id) const noexcept
- virtual bool **userIsActive** (const std::string &username) const

Protected Attributes

- std::unordered_map< std::string, user > registered
- std::unordered_map< std::string, const user * > active
- std::unordered_map< std::string, std::forward_list< document * > > workingDoc
- std::unordered_map< uint_positive_cnt::type, std::queue< std::shared_ptr< serverMessage > > > siteIdToMex
- std::unordered_map< uint_positive_cnt::type, std::forward_list< uint_positive_cnt::type > > resIdToSiteId
- std::shared_ptr< directory > rootDir

Static Protected Attributes

- static uint_positive_cnt idCounter

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)
- std::pair< bool, document * > userIsWorkingOnDocument (const std::string &username, uint_positive_cnt↩ ::type resourceId) const

  searches in the workingDoc map for the document that has the given resourceId
- uint_positive_cnt::type handleAccessToDoc (const std::string &actionUser, const std::string &resName, const std::string &pathFromUserHome, const user &actionU)

  Handles the access of actionUser to the document to change the privileges of a target user.
- void handleUserState (const std::string &targetUser, uint_positive_cnt::type docId, bool working=true)

  Handles control on the state of targetUser, to impose that it should or should not work on the resource.
- std::forward_list< uint_positive_cnt::type > siteIdsFor (uint_positive_cnt::type resId, uint_positive_cnt::type siteIdToExclude=0) const

  extract the siteIds associated to the given resId, excluding siteIdToExclude from the result
- std::forward_list< uint_positive_cnt::type > resIdOfDocOfUser (const std::string &username) const

  extract the resIds of the documents associated with the user names username
- std::forward_list< uint_positive_cnt::type > siteIdOfUserOfDoc (const std::forward_list< uint_positive_cnt↩ ::type > &resIds, unsigned int siteIdToExclude=-1) const

  extract the siteIds of the users that are associated with at least one of resIds in resIds

- void insertMessageForSiteIds (const std::forward_list< uint_positive_cnt::type > &siteIds, const std::shared_ptr< serverMessage > &toSend)

  Insert a copy of the message toSend in the message queue associated with every siteIds in siteIds.
- void closeAllDocsAndPropagateMex (const user &loggedOut, const std::forward_list< document * > &listOfDocs, uint_positive_cnt::type respMsgId)

  Call document::close on all the documents left opened by the user that just logged out and propagate the message about the disjoin of the user on that documents.
- void handleLeavingUser (const user &loggedOut)

  Removes the siteId of the user that just logged out from the list of siteIds associated with every resource that the user left opened.
- void generateSimpleResponse (unsigned int recvSiteId, msgType action, uint_positive_cnt::type respMsgId)

  generate a simple response with msgOutcome::success for a client request

Static Private Member Functions

- static bool userIsValid (const user &toCheck) noexcept

  checks that part of user's data that matters for the server
- static std::pair< std::string, std::string > fromLocalPathToGlobal (const user &actionU, const std::string &resPath, const std::string &resId)

  retrieves the absolute path of a resource

Friends

- class **boost::serialization::access**

6.49.1 Member Function Documentation

6.49.1.1 addUser()

```
const user & SymServer::addUser (
            user & newUser,
            uint_positive_cnt::type respMsgId )  [virtual]
```

adds a new user to the set of users registered to the system

Parameters

newUser	the user to be added to the system

Returns

the user just added

Exceptions

SymServerException	thrown if newUser is already registered
SymServerException	thrown if newUser data is not correct

When a client asks for the registration of a new user via a signUpMessage, the server validates the user data received (a signUpMessage contains a whole new user), assignes to it a subdirectory inside *rootDir*, adds the user to *registered* and send back to the client the user structure just filled inside a loginMessage. Note that the *siteId* of the user is assigned by the server, as the user's *home* directory structure and the *hashSalt* (that mustn't be sent to the client inside the loginMessage), so the user filled client side is always incomplete.

6.49.1.2 closeAllDocsAndPropagateMex()

```
void SymServer::closeAllDocsAndPropagateMex (
            const user & loggedOut,
            const std::forward_list< document * > & listOfDocs,
            uint_positive_cnt::type respMsgId )  [private]
```

Call document::close on all the documents left opened by the user that just logged out and propagate the message about the disjoin of the user on that documents.

Parameters

loggedOut	the user that just logged out
listOfDocs	the user of docs the user was working on

6.49.1.3 closeSource()

```
void SymServer::closeSource (
            const std::string & actionUser,
            uint_positive_cnt::type docId,
            uint_positive_cnt::type respMsgId )  [virtual]
```

close a document for a user

Parameters

actionUser	the user who wants to close the document
docId	document's id to be closed

This method is invoked by receiving a updateDocMessage and has the effect of calling document::close and the removal of *actionUser* from *workingDoc* for *toClose*

6.49.1.4 createNewDir()

```
std::shared_ptr< directory > SymServer::createNewDir (
            const std::string & opener,
            const std::string & resPath,
            const std::string & resName,
            uint_positive_cnt::type respMsgId )  [virtual]
```

creates a new directory in the user's filesystem

Parameters

opener	the user who made the request
resPath	the relative path to the *opener* 's *home* directory where to create the directory
resName	the name of the new directory

Returns

the directory just created

Exceptions

SymServerException	thrown if the user opener is not logged in
filesystemException	rethrown if there are problems creating the resource

When a client asks for a new directory, the server tries to do it and send back to the client a sendResMessage containing the directory just added to *path*.

6.49.1.5 createNewSource()

```
const document & SymServer::createNewSource (
            const std::string & opener,
            const std::string & resPath,
            const std::string & resName,
            uint_positive_cnt::type respMsgId )  [virtual]
```

creates a new file with an empty document inside

Parameters

opener	the user who made the request
resPath	the relative path to the *opener* 's *home* directory where to create the file
resName	the name of the new file

Returns

the document just created

Exceptions

SymServerException	thrown if the user opener is not logged in
filesystemException	rethrown if there are problems creating the resource

When a client asks for a new directory, the server tries to do it and send back to the client a sendResMessage containing the resource just added to *path*.

6.49.1.6 editLineStyle()

```
void SymServer::editLineStyle (
            const std::string & actionUser,
            uint_positive_cnt::type docId,
            const std::pair< alignType, unsigned > & newLineStyle,
            unsigned row,
            const editLineStyleMessage & editMsg )  [virtual]
```

edit the alignment and/or indexStyle of the document's paragraph *row* inside one of the documents he's working on

Parameters

docId	the identifier of the document whose paragraph has to be edited
newLineStyle	new alignment and indexStyle to apply
row	the row to apply *newLineStyle* to

6.49.1.7 editPrivilege()

```
privilege SymServer::editPrivilege (
            const std::string & actionUser,
            const std::string & targetUser,
            const std::string & resPath,
            const std::string & resId,
            privilege newPrivilege,
            uint_positive_cnt::type respMsgId )  [virtual]
```

edit the privilege of *targetUser* user for the resource *resName* in *resPath* to *newPrivilege*

Parameters

actionUser	the user who made the request
targetUser	the user whose privilege has to be modified
resPath	the relative path to the *actionUser* 's *home* directory
resId	the id of the resource
newPrivilege	the new privilege to be granted to *targetUser*

Returns

the old privilege of *targetUser* had on the resource

When a user client side want to edit the privilege of another user on a resource, it sends a PrivMessage and waits for an answer from the server. The server checks that *actionUser* is in *registered* and in *active*, and that *targetUser* is in *registered* and is not working on the resource in *resPath* named *resName*. Calls user::openFile to get the document and then calls user::editPrivilege on *actionUser* At the end send a serverMessage with the action outcome

6.49.1.8 editUser()

```
const user & SymServer::editUser (
            const std::string & username,
```

```
            user & newUserData,
            uint_positive_cnt::type respMsgId )  [virtual]
```

changes user's data

Parameters

username	the username of the user whose data are to be changed
newUserData	a user object filled with new data

Returns

the new user data

The server validates the new data and changes the user in *registered* and in *active* and sends back to the client a serverMessage with the outcome. The server must send a userDataMessage to the users in *active* that share some files with the user identified by *username* if the user changed the nickname or the icon

6.49.1.9 extractNextMessage()

```
std::pair< const uint_positive_cnt::type, std::shared_ptr< serverMessage > > SymServer↩
::extractNextMessage ( )
```

extract a message from the message queue associated with an user

Returns

a pair containing the siteId of the user the message is to send to and the message itself

6.49.1.10 fromLocalPathToGlobal()

```
std::pair< std::string, std::string > SymServer::fromLocalPathToGlobal (
            const user & actionU,
            const std::string & resPath,
            const std::string & resId )  [static], [private]
```

retrieves the absolute path of a resource

Parameters

actionU	the user who has the resource pointed by *resPath* and *resId*
resPath	the path of the resource relative to *actionU's* filesystem
resId	the id of the resource

Returns

a pair containing the absolute path and the id of the resource

6.49.1.11 generateSimpleResponse()

```
void SymServer::generateSimpleResponse (
            unsigned int recvSiteId,
            msgType action,
            uint_positive_cnt::type respMsgId )  [private]
```

generate a simple response with msgOutcome::success for a client request

Parameters

recv↩ SiteId	the siteId of the client to send the confirm to
action	the msgType of the received clientMessage

6.49.1.12 getRegistered()

```
user & SymServer::getRegistered (
            const std::string & username )  [protected], [virtual]
```

return the registered user associated with *username*

Parameters

username	the username of the registered user

Returns

a reference to the user

Warning

before a call to this method, the user with *username* must exist in server memory, otherwise an exception is thrown

Exceptions

std::out_of_range	can be thrown if the method is called without previously verifying that the user is registered

6.49.1.13 getSiteIdOfUser()

```
uint_positive_cnt::type SymServer::getSiteIdOfUser (
```

```
            const std::string & username ) const
```

retrieve the site id of an user

Parameters

username	the username of the user whose id is to be retrieved

Returns

the site id of that user

6.49.1.14 handleAccessToDoc()

```
uint_positive_cnt::type SymServer::handleAccessToDoc (
            const std::string & actionUser,
            const std::string & resName,
            const std::string & pathFromUserHome,
            const user & actionU )  [private]
```

Handles the access of *actionUser* to the document to change the privileges of a target user.

Parameters

actionUser	the name of the user who is changing the privileges of the target user
resName	the resource name
pathFromUserHome	the resource path, stating form *actionUser's* home directory
actionU	the user who is changing the privileges of the target user

Returns

the resId of the document involved in the privilege change

6.49.1.15 handleLeavingUser()

```
void SymServer::handleLeavingUser (
            const user & loggedOut )  [private]
```

Removes the siteId of the user that just logged out from the list of siteIds associated with every resource that the user left opened.

Parameters

loggedOut	the user that just logged out

6.49.1.16 handleUserState()

```
void SymServer::handleUserState (
            const std::string & targetUser,
            uint_positive_cnt::type docId,
            bool working = true )  [private]
```

Handles control on the state of *targetUser*, to impose that it should or should not work on the resource.

Parameters

targetUser	the user whose state is to be controlled
docId	the resource for which the working state of the user should be controlled
working	indicates whether an exception is to be raised if the user work or do not work on the resource

Exceptions

SymServerException	thrown if *targetUser* has state *working* on resource *docId*

6.49.1.17 hardLogout()

```
void SymServer::hardLogout (
            uint_positive_cnt::type siteId )  [virtual]
```

performs a logout as a result of unexpected detach of a client

Parameters

siteId	the siteId of the detached user

6.49.1.18 insertMessageForSiteIds()

```
void SymServer::insertMessageForSiteIds (
            const std::forward_list< uint_positive_cnt::type > & siteIds,
            const std::shared_ptr< serverMessage > & toSend )  [private]
```

Insert a copy of the message *toSend* in the message queue associated with every siteIds in *siteIds*.

Parameters

siteIds	the list of user (by means of their siteId) the message *toSend* should be forwarded to
toSend	the message to send to every user that has siteId in *siteIds*

6.49.1.19 load()

```
bool SymServer::load ( )
```

load data (users registered and filesystem structure) from disk

Returns

true if data has been found and correctly loaded, false otherwise

6.49.1.20 login()

```
const user & SymServer::login (
            const std::string & username,
            const std::string & pwd,
            uint_positive_cnt::type respMsgId )  [virtual]
```

log in an already registered user, adding it to *active*

Parameters

username	the username of the user who is performing the log in
pwd	the user's password

Returns

the logged in user

Exceptions

SymServerException	thrown if the user with username is not registered
SymServerException	thrown if the pwd is wrong
SymServerException	thrown if the user with username is already logged in

When a client asks for the login via a clientMessage, the server checks that the *username* and *password* corresponds to a user contained in *registered*, then sends the retrieved user object to the client, after having darken the *password* and the *hashSalt*, via a loginMessage

6.49.1.21 logout()

```
void SymServer::logout (
            const std::string & username,
            uint_positive_cnt::type respMsgId )  [virtual]
```

performs a log out, removing the user from *active*

Parameters

username	the username of the user who is performing the log out

Returns

the logged out user

6.49.1.22 mapSiteIdToUser()

```
std::map< uint_positive_cnt::type, user > SymServer::mapSiteIdToUser (
            const std::string & actionUser,
            uint_positive_cnt::type docId,
            uint_positive_cnt::type respMsgId )  [virtual]
```

maps siteIds to users to allow a client to identify the owner of each change in a document

Parameters

actionUser	the user who is asking for the mapping
docId	the id of the document for which the client asked for the mapping

Returns

the mapping siteId->user

This method id invoked by receiving a updateDocMessage with @action *action=msgType::mapChangesToUser* . It verifies that *actionUser* is registered and active and that it is currently working on the document that has the same *resourceId*. Calls document::retrieveSiteIds to know which users has modified the document, then produce the map with users in *registered*. Sends to the client a mapMessage

6.49.1.23 openNewSource()

```
std::shared_ptr< file > SymServer::openNewSource (
            const std::string & opener,
            const std::string & absolutePath,
            const std::string & destPath,
            const std::string & destName,
            privilege reqPriv,
            uint_positive_cnt::type respMsgId )  [virtual]
```

access a user's document via uri to the filesystem of the another user

Parameters

opener	the user who made the request
absolutePath	the absolute path of the requested document
destPath	the path where to put the symlink to *name*, inside *opener* 's home directory
destName	the name to assign to the symlink
reqPriv	the privilege requested opening the document

Returns

the document just retrieved

Exceptions

SymServerException	thrown if the user opener is not logged in
filesystemException	rethrown if there are problems regarding the asked resource

When a client asks for a document for which the user has no privilege with askResMessage, the server checks that the file in *resourceId* is available and then that the file is shareable. Under these conditions, the server adds the *opener* to the subset of users who have some privilege, adds a symlink to the file in *destPath* named *destName* and send the document inside a sendResMessage If the operation succeed, the server sends a updateActiveMessage to the clients working on the document and send back to the client a sendResMessage with the symlink just created

6.49.1.24 openSource()

```
std::shared_ptr< file > SymServer::openSource (
            const std::string & opener,
            const std::string & resPath,
            const std::string & resId,
            privilege reqPriv,
            uint_positive_cnt::type respMsgId )  [virtual]
```

access a document that is already in the user's filesystem

Parameters

opener	the user who wants to access the document
resPath	the path of the file, relative to user's home directory
resId	the file's id
reqPriv	the privilege requested opening the document

Returns

the document just retrieved

Exceptions

SymServerException	thrown if the user opener is not logged in
filesystemException	rethrown if there are problems regarding the asked resource

When a client asks for a document, the server checks that the file named *name* in *path* is available and then that *opener* has a privilege less or equal to *reqPriv* on it. If these requirements are met, then the server sends the document inside a sendResMessage If the operation succeed, the server sends a updateActiveMessage to the clients working on the document

6.49.1.25 registerUser()

```
user & SymServer::registerUser (
            const user & toInsert )  [protected], [virtual]
```

insert a new user in the table of registered users

Parameters

toInsert	the user to be inserted

Returns

a reference to the inserted user

6.49.1.26 remoteInsert()

```
void SymServer::remoteInsert (
            const std::string & inserter,
            uint_positive_cnt::type docId,
            const symbolMessage & symMsg )  [virtual]
```

update a document with a new symbol from a client

Parameters

inserter	the user who is working on the document
docId	the id of the document inside *workingDoc*
symMsg	message received containing the symbol to insert

When a user client side inserts a new symbol, the client sends to the server a symbolMessage containing the symbol, the *resourceId* and the *siteId*. The server validates the message checking that *inserter* is a valid user and that is a user currently working on the document indicated by *resourceId*, then update its own copy of the document and propagate the update to the other clients putting the message in *workingQueue*

6.49.1.27 remoteRemove()

```
void SymServer::remoteRemove (
            const std::string & remover,
            uint_positive_cnt::type docId,
            const symbolMessage & rmMsg )  [virtual]
```

update a document removing a symbol

Parameters

remover	the user who is working on the document
docId	the id of the document inside *workingDoc*
rmMsg	message received containing the symbol to remove

When a user client side removes a symbol, the client sends to the server a symbolMessage containing the symbol, the *resourceId* and the *siteId*. The server validates the message checking that *remover* is a valid user and that is a user currently working on the document indicated by *resourceId*, then update its own copy of the document and propagate the update to the other clients putting the message in *workingQueue*

6.49.1.28 removeResource()

```
std::shared_ptr< filesystem > SymServer::removeResource (
            const std::string & remover,
            const std::string & resPath,
            const std::string & resId,
            uint_positive_cnt::type respMsgId )  [virtual]
```

removes a resource from *remover* 's *home* directory

Parameters

remover	the user who is asking to remove its resource
resPath	the relative path to the *remover* 's *home* directory where to create the file
resId	the resource's id

Returns

the resource just removed

When a user client side wants remove a resource, it sends a askResMessage and waits for an answer from the server. The server checks that *actionUser* is in *registered* and in *active*, then calls user::renameResource on *remover*.

6.49.1.29 removeUser()

```
void SymServer::removeUser (
            const std::string & username,
            const std::string & pwd,
            uint_positive_cnt::type respMsgId )  [virtual]
```

removes an user to the set of users registered to the system

Parameters

username	the name of the user to remove
pwd	the user's password

Returns

the user just removed

6.49.1.30 renameResource()

```
std::shared_ptr< filesystem > SymServer::renameResource (
            const std::string & renamer,
            const std::string & resPath,
            const std::string & resId,
            const std::string & newName,
            uint_positive_cnt::type respMsgId )  [virtual]
```

renames a resource from *remover* 's *home* directory

Parameters

renamer	the user who is asking to rename its resource
resPath	the relative path to the *renamer* 's *home* directory where to find the resource
resId	the resource's id
newName	the new resource's name

Returns

the resource just renamed

When a user client side wants to set a new name for a resource, it sends a askResMessage and waits for an answer from the server. The server checks that *actionUser* is in *registered* and in *active*, then calls user::renameResource on *renamer*. Please note that, in case the filesystem object is a symlink, this method renames the symlink, not the resource pointed. Sends back a serveMessage, to indicate whether the action succeeded or not.

6.49.1.31 resIdOfDocOfUser()

```
std::forward_list< uint_positive_cnt::type > SymServer::resIdOfDocOfUser (
            const std::string & username ) const  [private]
```

extract the resIds of the documents associated with the user names *username*

Parameters

username	the name of the user for which the mapping is needed

Returns

a list of resIds

6.49.1.32 shareResource()

```
std::shared_ptr< filesystem > SymServer::shareResource (
            const std::string & actionUser,
            const std::string & resPath,
```

```
            const std::string & resId,
            const uri & newPrefs,
            uint_positive_cnt::type respMsgId )  [virtual]
```

set new sharing preferences for a resource

Parameters

actionUser	the user who made the request
resPath	the relative path to the *actionUser* 's *home* directory
resId	the id of the resource
newPrefs	new sharing preferences for the resource

Returns

the old *sharingPolicy*

When a user client side wants to set a new sharing policy for a resource, it sends a uriMessage and waits for an answer from the server. The server checks that *actionUser* is in *registered* and in *active*, then calls user::shareResource on *actionUser* At the end send a serverMessage with the action outcome

6.49.1.33 siteIdOfUserOfDoc()

```
std::forward_list< uint_positive_cnt::type > SymServer::siteIdOfUserOfDoc (
            const std::forward_list< uint_positive_cnt::type > & resIds,
            unsigned int siteIdToExclude = -1 ) const  [private]
```

extract the siteIds of the users that are associated with at least one of resIds in *resIds*

Parameters

resIds	a list of resource ids for which a mapping with the working user's siteId is needed
siteIdToExclude	a siteId to exclude from the result, tipically the one of the user who asked for this

Returns

a list of siteIds

6.49.1.34 siteIdsFor()

```
std::forward_list< uint_positive_cnt::type > SymServer::siteIdsFor (
            uint_positive_cnt::type resId,
            uint_positive_cnt::type siteIdToExclude = 0 ) const  [private]
```

extract the siteIds associated to the given *resId*, excluding *siteIdToExclude* from the result

Parameters

resId	the resource id of the resource we want the siteIds associated with
siteIdToExclude	a siteId to exclude from the result, tipically the one of the user who asked for this

Returns

a list of siteIds

6.49.1.35 updateCursorPos()

```
void SymServer::updateCursorPos (
            const std::string & targetUser,
            uint_positive_cnt::type docId,
            const cursorMessage & crMsg )  [virtual]
```

update the current position of user's cursor inside one of the documents he's working on

Parameters

targetUser	the user whose cursor position has changed
docId	the id of the document in which the user's cursor has been moved
crMsg	the message received by the client

6.49.1.36 userIsValid()

```
bool SymServer::userIsValid (
            const user & toCheck )  [static], [private], [noexcept]
```

checks that part of user's data that matters for the server

Parameters

toCheck	the user object whose parameters has to be checked

Returns

true if username and nickname are not empty and if icon path is a valid path

6.49.1.37 userIsWorkingOnDocument()

```
std::pair< bool, document * > SymServer::userIsWorkingOnDocument (
            const std::string & username,
            uint_positive_cnt::type resourceId ) const  [private]
```

searches in the *workingDoc* map for the document that has the given *resourceId*

Parameters

username	the username of the user whose working state on a document is to be checked
resourceId	the id of the resource we want to know if the user is working on

Returns

a pair that containt {false, nulltpr} if the user is not working on the resource that has the given *resourceId*, or {true, pointer to document} if the user is working on the resource

6.49.2 Member Data Documentation

6.49.2.1 active

```
std::unordered_map<std::string, const user *> Symposium::SymServer::active  [protected]
```

active users, indexed by username

6.49.2.2 idCounter

```
uint_positive_cnt SymServer::idCounter  [static], [protected]
```

siteId to be assigned to the next registered user

6.49.2.3 registered

```
std::unordered_map<std::string, user> Symposium::SymServer::registered  [protected]
```

registered users, indexed by username

6.49.2.4 resIdToSiteId

```
std::unordered_map<uint_positive_cnt::type, std::forward_list<uint_positive_cnt::type> >
Symposium::SymServer::resIdToSiteId  [protected]
```

list of users involved in a document, by means of *siteIds* and *resIds*

6.49.2.5 rootDir

```
std::shared_ptr<directory> Symposium::SymServer::rootDir  [protected]
```

virtual filesystem of the Symposium server

6.49.2.6 siteIdToMex

```
std::unordered_map<uint_positive_cnt::type, std::queue<std::shared_ptr<serverMessage> > >
Symposium::SymServer::siteIdToMex  [protected]
```

messages queues associated with every user by means of *siteId*

6.49.2.7 storeData

```
bool Symposium::SymServer::storeData
```

flags to indicate whether the server data is to be stored and loaded

6.49.2.8 storeFile

```
constexpr char Symposium::SymServer::storeFile[] ="./serverData.dat"  [static]
```

path of the file the server data is stored onto and loaded from

6.49.2.9 workingDoc

```
std::unordered_map<std::string, std::forward_list<document *> > Symposium::SymServer::working↩
Doc  [protected]
```

list of document each user is working on

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

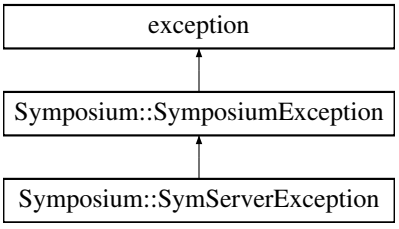
- SymServer.h
- SymServer.cpp

6.50 Symposium::SymServerException Class Reference

models an exception occurred in the context of SymServer class @exceptsafe no-throw

```
#include <SymposiumException.h>
```

Inheritance diagram for Symposium::SymServerException:



Public Types

- enum SymServerExceptionCodes {
  userAlreadyExist =0, **userWrongParam**, **userNotRegistered**, **userWrongPwd**,
  userAlreadyActive, **userNotLogged**, **userNotWorkingOnDoc**, **actionUserNotLoggedOrTargetUser↩ NotRegistered**,
  userWorkingOnDoc, **userNotFound**, **noUserWorkingOnRes** }

  Specific error codes for SymServerException. They are used as indexes to the error table string.

Public Member Functions

- **SymServerException** (SymServerExceptionCodes exceptionCode, const char *file, int line, const char *func)

Static Private Attributes

- static const char * **SymServerErrors** [ ]

Additional Inherited Members

6.50.1 Detailed Description

models an exception occurred in the context of SymServer class @exceptsafe no-throw

Constructor simply performs lookup on *SymServerErrors* table and pass the error message to SymposiumException constructor. The exception safety is no-throw because the *SymServerErrors* table is statically allocated and conversions from the underlying type to SymServerExceptionCodes are not allowed, so there is no way to access an illegal element of *SymServerErrors*.

6.50.2 Member Data Documentation

6.50.2.1 SymServerErrors

```
const char * SymServerException::SymServerErrors  [static], [private]
```

Initial value:

```
={"the user already exists",
                          "the user has wrong parameters", "the user is not registered", "wrong
      password",
                          "user already active","the user is not logged in", "user is not working on the
      document",
                          "actionUser is not logged in or targetUser is not registered",
                          "user is working on the document", "user not found",
                          "There are no users working on this resource"}
```

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

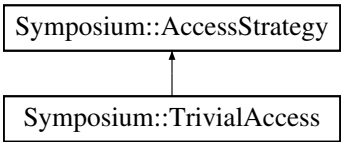
- SymposiumException.h
- SymposiumException.cpp

6.51 Symposium::TrivialAccess Class Reference

class used to model the absence of privilege handling on a resource

`#include <AccessStrategy.h>`

Inheritance diagram for Symposium::TrivialAccess:



Public Member Functions

- bool validateAction (const std::string &targetUser, privilege requested) const override
  validate an action from user targetUser that requires requested
- privilege setPrivilege (const std::string &targetUser, privilege toGrant) override
  set the privilege of an user
- privilege **getPrivilege** (const std::string &targetUser) const override
- std::unordered_map< std::string, privilege > **getPermission** () const override
- bool **moreOwner** (std::string username) const override

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int)

Friends

- class **boost::serialization::access**

6.51.1 Detailed Description

class used to model the absence of privilege handling on a resource

6.51.2 Member Function Documentation

6.51.2.1 setPrivilege()

```
privilege TrivialAccess::setPrivilege (
            const std::string & targetUser,
            privilege toGrant )  [override], [virtual]
```

set the privilege of an user

Parameters

targetUser	the user the privilege is to be granted
toGrant	the privilege to grant to targetUser

Returns

the privilege previously owned by targetUser, none if no privilege previously owned

Implements Symposium::AccessStrategy.

6.51.2.2 validateAction()

```
bool TrivialAccess::validateAction (
            const std::string & targetUser,
            privilege requested ) const  [override], [virtual]
```

validate an action from user targetUser that requires requested

Parameters

targetUser	the user who is doing the action
requested	the permission requested by the action

Returns

true if the user is granted the privilege requested

Implements Symposium::AccessStrategy.

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

- AccessStrategy.h
- AccessStrategy.cpp

6.52 type_not_narrow< Source, Dest, > Struct Template Reference

Static Public Attributes

- static constexpr bool **value** = std::numeric_limits<Source>::max() <= std::numeric_limits<Dest>::max()

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

- counter.h

6.53 Symposium::detail::type_not_narrow< Source, Dest, > Struct Template Reference

Static Public Attributes

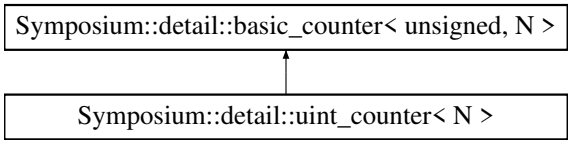
- static constexpr bool **value** = std::numeric_limits<Source>::max() <= std::numeric_limits<Dest>::max()

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

- Symposium.h

6.54 Symposium::detail::uint_counter< N > Struct Template Reference

Inheritance diagram for Symposium::detail::uint_counter< N >:



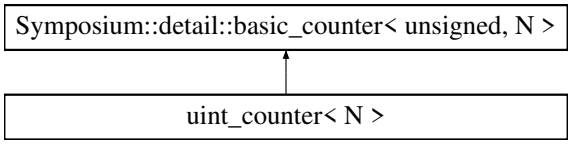
Additional Inherited Members

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

- Symposium.h

6.55 uint_counter< N > Struct Template Reference

Inheritance diagram for uint_counter< N >:



Additional Inherited Members

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

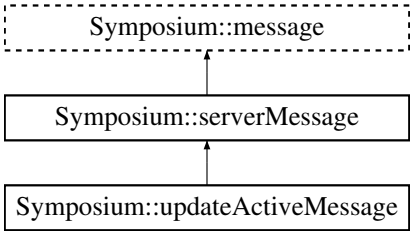
- counter.h

6.56 Symposium::updateActiveMessage Class Reference

class used to model the joining of an user to a document

`#include <message.h>`

Inheritance diagram for Symposium::updateActiveMessage:



Public Member Functions

- updateActiveMessage (msgType action, msgOutcome result, const user &newUser, uint_positive_cnt::type resourceId, privilege priv=privilege::readOnly, uint_positive_cnt::type msgId=0)
- int **getResourceId** () const
- privilege **getUserPrivilege** () const
- void invokeMethod (SymClient &client) override

  enable all the clients active on document know what other users are active on the document
- bool **operator==** (const updateActiveMessage &rhs) const
- bool **operator!=** (const updateActiveMessage &rhs) const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Private Attributes

- user newUser
- uint_positive_cnt::type resourceId
- privilege userPrivilege

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.56.1 Detailed Description

class used to model the joining of an user to a document

The server sends this message when an user access or closes a document. If a user opens a document, then becomes *active* on it and the other clients must be able to see who is active on the document: in this case *action=msgType::addActiveUser* . Similarly, when a user closes a document, since it is not longer active on it, the server sends a *updateActiveMessage* with *action=msgType::removeActiveUser* . Client side, if *newUser* is not in any set (readers, writers, owners), then add the user to the proper set basing on *userPrivilege*

6.56.2 Constructor & Destructor Documentation

6.56.2.1 updateActiveMessage()

```
updateActiveMessage::updateActiveMessage (
            msgType action,
            msgOutcome result,
            const user & newUser,
            uint_positive_cnt::type resourceId,
            privilege priv = privilege::readOnly,
            uint_positive_cnt::type msgId = 0 )
```

Exceptions

messageException	if *action* is not consistent with the message type

6.56.3 Member Function Documentation

6.56.3.1 invokeMethod()

```
void updateActiveMessage::invokeMethod (
            SymClient & client )  [override], [virtual]
```

enable all the clients active on document know what other users are active on the document

Parameters

client	the client instance to update

Depending on the value of *action*, the *invokeMethod* ask for different actions on the client:

- *action=msgType::addActiveUser* : calls SymClient::addActiveUser
- *action=msgType::removeActiveUser* : calls SymClient::removeActiveUser

Reimplemented from Symposium::serverMessage.

6.56.4 Member Data Documentation

6.56.4.1 newUser

```
user Symposium::updateActiveMessage::newUser [private]
```

user who joined the document

6.56.4.2 resourceId

```
uint_positive_cnt::type Symposium::updateActiveMessage::resourceId [private]
```

identifier of the opened resource

6.56.4.3 userPrivilege

```
privilege Symposium::updateActiveMessage::userPrivilege [private]
```

the privilege *newUser* has on *resourceId*

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

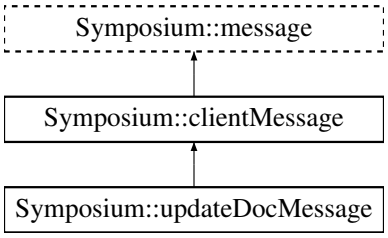
- message.h
- message.cpp

6.57 Symposium::updateDocMessage Class Reference

class used to model a message sent by a client to close a resource

```
#include <message.h>
```

Inheritance diagram for Symposium::updateDocMessage:



Public Member Functions

- updateDocMessage (msgType action, const std::pair< std::string, std::string > &actionOwner, uint_positive_cnt::type resourceId, uint_positive_cnt::type msgId=0)
- uint_positive_cnt::type **getResourceId** () const
- void invokeMethod (SymServer &server) override

  asks the server for an action on the document indicated by resourceId the actionUser is working on
- bool **operator==** (const updateDocMessage &rhs) const
- bool **operator!=** (const updateDocMessage &rhs) const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int)

Private Attributes

- uint_positive_cnt::type resourceId

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.57.1 Detailed Description

class used to model a message sent by a client to close a resource

6.57.2 Constructor & Destructor Documentation

6.57.2.1 updateDocMessage()

```
updateDocMessage::updateDocMessage (
            msgType action,
            const std::pair< std::string, std::string > & actionOwner,
            uint_positive_cnt::type resourceId,
            uint_positive_cnt::type msgId = 0 )
```

Exceptions

messageException	if *action* is not consistent with the message type

6.57.3 Member Function Documentation

6.57.3.1 invokeMethod()

```
void updateDocMessage::invokeMethod (
            SymServer & server )  [override], [virtual]
```

asks the server for an action on the document indicated by *resourceId* the *actionUser* is working on

Parameters

server	the server the user is active on

Depending on the value of *action*, the *invokeMethod* ask for different actions on the server:

- *action=msgType::closeRes* : calls SymServer::closeSource on *server*.
- *action=msgType::mapChangesToUser* : calls SymServer::mapSiteIdToUser on *server*. A message of type mapMessage is sent back to the client

Reimplemented from Symposium::clientMessage.

6.57.4 Member Data Documentation

6.57.4.1 resourceId

```
uint_positive_cnt::type Symposium::updateDocMessage::resourceId  [private]
```

identifier of the resource on which perform the action

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

- message.h
- message.cpp

6.58 Symposium::uri Class Reference

class used to model resource sharing preferences

```
#include <uri.h>
```

Public Member Functions

- **uri** (uriPolicy activePolicy=uriPolicy::inactive)
- uriPolicy **getActivePolicy** () const
- unsigned int **getSharesLeft** () const
- privilege **getGranted** () const
- void activateAlways (privilege newPrivilege=defaultPrivilege)
  set activePolicy as activateAlways with newPrivilege
- void activateCount (unsigned int shares, privilege newPrivilege=defaultPrivilege)
  set activePolicy as activeCount with newPrivilege
- void activateTimer (std::chrono::system_clock::time_point endTime, privilege newPrivilege=defaultPrivilege)
  set activePolicy as activeTimer with newPrivilege
- void deactivate ()
  set activePolicy as inactive
- virtual privilege getShare (privilege requested)
  check uri validity with requested privilege requested by user

Static Public Member Functions

- static privilege **getDefaultPrivilege** ()

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int)

Private Attributes

- uriPolicy activePolicy
- unsigned int sharesLeft
- std::chrono::system_clock::time_point stopTime
- privilege granted

Static Private Attributes

- static constexpr privilege **defaultPrivilege** = privilege::modify

Friends

- class **boost::serialization::access**

Related Functions

(Note that these are not member functions.)

- bool operator== (const uri &rhs) const
  operator == overload for uri
- bool operator!= (const uri &rhs) const
  operator != overload for uri

6.58.1 Detailed Description

class used to model resource sharing preferences

The intend of this class is to enable users to share their own resources and obtain a privilege on other's ones. The settings used in an object of this class are not retroactive, meaning that a changing in a uri object does not affect users who already have a privilege on the resource the uri object refers to

6.58.2 Member Function Documentation

6.58.2.1 activateAlways()

```
void uri::activateAlways (
            privilege newPrivilege = defaultPrivilege )
```

set activePolicy as activateAlways with newPrivilege

Parameters

newPrivilege	the privilege to set

6.58.2.2 activateCount()

```
void uri::activateCount (
            unsigned int shares,
            privilege newPrivilege = defaultPrivilege )
```

set activePolicy as activeCount with newPrivilege

Parameters

shares	the number of possible shares
newPrivilege	the privilege to set

6.58.2.3 activateTimer()

```
void uri::activateTimer (
            std::chrono::system_clock::time_point endTime,
            privilege newPrivilege = defaultPrivilege )
```

set activePolicy as activeTimer with newPrivilege

Parameters

endTime	the end of the uri validity
newPrivilege	the privilege to set

6.58.2.4 getShare()

```
privilege uri::getShare (
            privilege requested )  [virtual]
```

check uri validity with requested privilege requested by user

Parameters

requested	the privilege which user want

Returns

requested privilege if it is <= than granted privilege or granted privilege if requested privilege is > than granted

6.58.3 Member Data Documentation

6.58.3.1 activePolicy

```
uriPolicy Symposium::uri::activePolicy  [private]
```

the policy currently active

6.58.3.2 granted

```
privilege Symposium::uri::granted  [private]
```

privilege that the resource owner decided to grant via uri

6.58.3.3 sharesLeft

```
unsigned int Symposium::uri::sharesLeft  [private]
```

number of shares that can still be accepted

6.58.3.4 stopTime

```
std::chrono::system_clock::time_point Symposium::uri::stopTime  [private]
```

end time of sharing

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

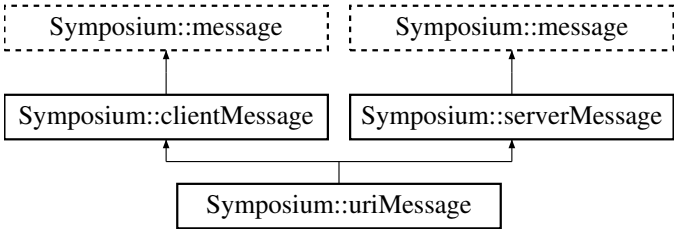
- uri.h
- uri.cpp

6.59 Symposium::uriMessage Class Reference

class used to model a message for sharing a document

```
#include <message.h>
```

Inheritance diagram for Symposium::uriMessage:



Public Member Functions

- uriMessage (msgType action, const std::pair< std::string, std::string > &actionOwner, msgOutcome result, const std::string &path, const std::string &name, const uri &sharingPrefs, uint_positive_cnt::type msgId=0)
- const uri & **getSharingPrefs** () const
- void invokeMethod (SymServer &server) override

  asks the server to modify a file sharing preferences'
- void invokeMethod (SymClient &client) override

  propagate the changes made by a client over a file sharing preferences to other clients
- void completeAction (SymClient &client, msgOutcome serverResult) override

  completes an action for which the client asked to the server
- bool **operator==** (const uriMessage &rhs) const
- bool **operator!=** (const uriMessage &rhs) const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Private Attributes

- std::string **path**
- std::string **name**
- uri **sharingPrefs**

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.59.1 Detailed Description

class used to model a message for sharing a document

The client sends an object of type uri with the sharing preferences changed by the user. The server forwards this message to other clients that are enabled to see the sharing preferences of a document.

6.59.2 Constructor & Destructor Documentation

6.59.2.1 uriMessage()

```
uriMessage::uriMessage (
            msgType action,
            const std::pair< std::string, std::string > & actionOwner,
            msgOutcome result,
            const std::string & path,
            const std::string & name,
            const uri & sharingPrefs,
            uint_positive_cnt::type msgId = 0 )
```

Exceptions

messageException	if *action* is not consistent with the message type

6.59.3 Member Function Documentation

6.59.3.1 completeAction()

```
void uriMessage::completeAction (
            SymClient & client,
            msgOutcome serverResult )  [override], [virtual]
```

completes an action for which the client asked to the server

Parameters

client	the same client that had originated the clientMessage

Some actions on SymClient require to ask to the server to assure that the action is valid and to propagate the change on the server. For such actions, only if the outcome from the server is positive the action can be actually done.

Reimplemented from Symposium::clientMessage.

6.59.3.2 invokeMethod() [1/2]

```
void uriMessage::invokeMethod (
            SymServer & server )  [override], [virtual]
```

asks the server to modify a file sharing preferences'

Parameters

server	the server the user is active on

When this message is received by the server, the *invokeMethod* calls SymServer::shareResource on the *server* passed as parameter.

Reimplemented from Symposium::clientMessage.

6.59.3.3 invokeMethod() [2/2]

```
void uriMessage::invokeMethod (
            SymClient & client )  [override], [virtual]
```

propagate the changes made by a client over a file sharing preferences to other clients

Parameters

client	the client on which propagate the change

When this message is received by the client, the *invokeMethod* calls SymClient::shareResource on the *client* passed as parameter.

Reimplemented from Symposium::serverMessage.

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

- message.h
- message.cpp

6.60 Symposium::user Class Reference

class used to model a user of the system

```
#include <user.h>
```

Public Member Functions

- user (const std::string &username, const std::string &pwd, const std::string &nickname, const std::string &iconPath, uint_positive_cnt::type siteId, std::shared_ptr< directory > home)

  creates a new user
- const std::string & **getUsername** () const
- uint_positive_cnt::type **getSiteId** () const
- const std::string & **getNickname** () const
- const std::string & **getIconPath** () const
- const std::shared_ptr< directory > & **getHome** () const
- const std::string & **getPwdHash** () const
- void **setNickname** (const std::string &nickname)
- void **setSiteId** (uint_positive_cnt::type siteId)
- void **setIconPath** (const std::string &iconPath)
- void **setHome** (std::shared_ptr< directory > home)
- virtual void **setNewData** (const user &newData)
- virtual bool **hasPwd** (const std::string &pwd) const
- bool **operator==** (const user &rhs) const
- bool **operator!=** (const user &rhs) const
- virtual std::string showDir (bool recursive=false) const

  shows the entire user's home directory
- virtual std::shared_ptr< file > newFile (const std::string &resName, const std::string &resPath="./", uint_positive_cnt::type idToAssign=0) const

creates a new file named resId in pathFromHome

- virtual std::shared_ptr< directory > newDirectory (const std::string &resName, const std::string &resPath="./", uint_positive_cnt::type idToAssign=0) const

  creates a new directory named dirName in pathFromHome

- virtual std::pair< int, std::shared_ptr< file > > accessFile (const std::string &absolutePath, const std::string &destPath, const std::string &destName, privilege reqPriv) const

  adds a link to the resource for which the user has been granted a privilege

- virtual std::shared_ptr< file > openFile (const std::string &resPath, const std::string &resId, privilege accessMode) const

  open a file already present in the user's filesystem

- virtual privilege editPrivilege (const std::string &otherUser, const std::string &resPath, const std::string &resId, privilege newPrivilege) const

  edit the privilege of otherUser user for the resource resName in resPath to newPrivilege

- virtual std::shared_ptr< filesystem > shareResource (const std::string &resPath, const std::string &resId, const uri &newPrefs) const

  set new sharing preferences for a resource

- virtual std::shared_ptr< filesystem > renameResource (const std::string &resPath, const std::string &resId, const std::string &newName) const

  renames a resource

- virtual std::shared_ptr< filesystem > removeResource (const std::string &resPath, const std::string &resId) const

  removes a filesystem object

- user makeCopyNoPwd () const

  constructs a copy of the current user object clearing the pwdHash and the hashSalt

Static Public Member Functions

- static void **hideAuthParams** (const std::function< void(void)> &op)
- static bool noCharPwd (const std::string &pass)
- static bool noNumPwd (const std::string &pass)
- static bool noSpecialCharPwd (const std::string &pass)
- static bool noSpaceUsername (const std::string &username)

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int version)

Static Private Member Functions

- static std::string saltGenerate ()

  generate random salt for password

- static bool correctFormatResPath (const std::string &path)

  check if the relative path passed as argument is correct

- static bool correctFormatAbsolutePathWithId (const std::string &path)

  check if the absolute path passed as argument is correct

Private Attributes

- std::string username
- std::string pwdHash
- std::string hashSalt
- uint_positive_cnt::type siteId
- std::string nickname
- std::string iconPath
- std::shared_ptr< directory > home

Static Private Attributes

- static constexpr char **noChar** [ ] ="1234567890?!$+-/.,@^_ "
- static constexpr char **noNum** [ ] ="abcdefghijklmnopqrstuvwxyz?!$+-/.,@^_ "
- static constexpr char **noSpecialChar** [ ] ="abcdefghijklmnopqrstuvwxyz1234567890 "
- static bool **HideParamOnSer** =false

Friends

- class **boost::serialization::access**

6.60.1 Detailed Description

class used to model a user of the system

A user is identified by its *username*, that is used in login step, and by a *siteId*, that is a number given by server the user is hosted by, used in the CRDT algorithm to distinguish symbols from different users. For each user the system store a cryptographic hash value of its password, to ensure more privacy and security. Each user has a reference to a part of the server's virtual filesystem,a directory under "/" called as the username. The server and the clients share the same set of icons, so *iconPath* is the common path to icons for the server and the client. When an *incomplete* user is built client side to be sent inside a signUpMessage, *pwdHash* contains the plaintext password, *hashSalt*, *siteId* and *home* are empty.

6.60.2 Constructor & Destructor Documentation

6.60.2.1 user()

```
user::user (
            const std::string & username,
            const std::string & pwd,
            const std::string & nickname,
            const std::string & iconPath,
            uint_positive_cnt::type siteId,
            std::shared_ptr< directory > home )
```

creates a new user

Exceptions

6.60.3 Member Function Documentation

6.60.3.1 accessFile()

```
std::pair< int, std::shared_ptr< file > > user::accessFile (
            const std::string & absolutePath,
            const std::string & destPath,
            const std::string & destName,
            privilege reqPriv ) const  [virtual]
```

adds a link to the resource for which the user has been granted a privilege

Parameters

absolutePath	the uri of the resource to be linked by the symlink
destPath	the path to put the resource into
destName	the name of the new link

Exceptions

6.60.3.2 correctFormatAbsolutePathWithId()

```
bool user::correctFormatAbsolutePathWithId (
            const std::string & path )  [static], [private]
```

check if the absolute path passed as argument is correct

Parameters

path	the path to check

Returns

true if the path is correct otherwise return false

6.60.3.3 correctFormatResPath()

```
bool user::correctFormatResPath (
            const std::string & path )  [static], [private]
```

check if the relative path passed as argument is correct

Parameters

path	the path to check

Returns

true if the path is correct otherwise return false

6.60.3.4 editPrivilege()

```
privilege user::editPrivilege (
            const std::string & otherUser,
            const std::string & resPath,
            const std::string & resId,
            privilege newPrivilege ) const  [virtual]
```

edit the privilege of *otherUser* user for the resource *resName* in *resPath* to *newPrivilege*

Parameters

otherUser	the user whose privilege has to be modified
resPath	relative path to the resource from the current user's *home*
resId	the id of the target resource
newPrivilege	the new privilege to be granted to *targetUser*

Exceptions

6.60.3.5 makeCopyNoPwd()

```
user user::makeCopyNoPwd ( ) const
```

constructs a copy of the current user object clearing the *pwdHash* and the *hashSalt*

Returns

the constructed copy

6.60.3.6 newDirectory()

```
std::shared_ptr< directory > user::newDirectory (
            const std::string & resName,
            const std::string & resPath = "./",
            uint_positive_cnt::type idToAssign = 0 ) const  [virtual]
```

creates a new directory named *dirName* in *pathFromHome*

Parameters

resName	name to be assigned to the new directory
resPath	path inside the home to put the directory. By default is the home itself

Returns

the pointer to the directory just created

6.60.3.7 newFile()

```
std::shared_ptr< file > user::newFile (
            const std::string & resName,
            const std::string & resPath = "./",
            uint_positive_cnt::type idToAssign = 0 ) const  [virtual]
```

creates a new file named *resId* in *pathFromHome*

Parameters

resName	name to be assigned to the new file
resPath	path inside the home to put the file. By default is the home itself

Returns

the pointer to the file just created

6.60.3.8 noCharPwd()

```
bool user::noCharPwd (
            const std::string & pass )  [static]
```

Parameters

pass	the password to control

Returns

true if the @pass don't have any alphabetic character and false if it does

6.60.3.9 noNumPwd()

```
bool user::noNumPwd (
            const std::string & pass )  [static]
```

Parameters

pass	the password to control

Returns

true if the @pass don't have any number and false if it does

6.60.3.10 noSpaceUsername()

```
bool user::noSpaceUsername (
            const std::string & username )  [static]
```

Parameters

username	the password to control

Returns

true if the @username don't have any space and false if it does

6.60.3.11 noSpecialCharPwd()

```
bool user::noSpecialCharPwd (
            const std::string & pass )  [static]
```

Parameters

pass	the password to control

Returns

true if the @pass don't have any special character and false if it does

6.60.3.12 openFile()

```
std::shared_ptr< file > user::openFile (
            const std::string & resPath,
            const std::string & resId,
            privilege accessMode ) const  [virtual]
```

open a file already present in the user's filesystem

Parameters

resPath	location of the file relative to the *home* directory
resId	id of the file to be opened
accessMode	the privilege asked by the user for opening the file

Returns

the document contained in the file

This method calls the directory::access method on *home* passing the current user as the one asking the action

6.60.3.13 removeResource()

```
std::shared_ptr< filesystem > user::removeResource (
            const std::string & resPath,
            const std::string & resId ) const  [virtual]
```

removes a filesystem object

Parameters

resPath	the path inside the user's home directory where the target file is located
resId	the file id

Returns

the filesystem object just removed from the user's filesystem

6.60.3.14 renameResource()

```
std::shared_ptr< filesystem > user::renameResource (
            const std::string & resPath,
```

```
            const std::string & resId,
            const std::string & newName ) const  [virtual]
```

renames a resource

Parameters

resPath	the relative path to the *home* directory where to find the resource
resId	the resource's id
newName	the new resource's name

Returns

the resource just renamed

Changes the attribute *name* of the filesystem object. Please note that, in case the filesystem object is a symlink, this method renames the symlink, not the resource pointed.

6.60.3.15 saltGenerate()

```
std::string user::saltGenerate ( )  [static], [private]
```

generate random salt for password

Returns

salt

6.60.3.16 shareResource()

```
std::shared_ptr< filesystem > user::shareResource (
            const std::string & resPath,
            const std::string & resId,
            const uri & newPrefs ) const  [virtual]
```

set new sharing preferences for a resource

Parameters

resPath	the relative path of the resource
resId	the id of the resource
newPrefs	new sharing preferences for the resource

Returns

the old *sharingPolicy*

Calls directory::getFile on *home* and then `file::setSharingPolicy` on the retrieved file

6.60.3.17 showDir()

```
std::string user::showDir (
            bool recursive = false ) const  [virtual]
```

shows the entire user's home directory

Parameters

recursive	indicates whether the visit must be recursive or not

Returns

a concatenation string of the content of the home directory to be showed by the GUI

6.60.4 Member Data Documentation

6.60.4.1 hashSalt

```
std::string Symposium::user::hashSalt  [private]
```

random generated string, used as a salt to *pwdHash*

6.60.4.2 home

```
std::shared_ptr<directory> Symposium::user::home  [private]
```

user's virtual filesystem

6.60.4.3 iconPath

```
std::string Symposium::user::iconPath  [private]
```

path to the user's icon in program installation folder

6.60.4.4 nickname

```
std::string Symposium::user::nickname  [private]
```

name chosen by the user to be showed to other users

6.60.4.5 pwdHash

```
std::string Symposium::user::pwdHash  [private]
```

user password's hash value

6.60.4.6 siteId

```
uint_positive_cnt::type Symposium::user::siteId  [private]
```

unique identifier for the user, used in CRDT logic

6.60.4.7 username

```
std::string Symposium::user::username  [private]
```

unique identifier for the user, used also for login

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

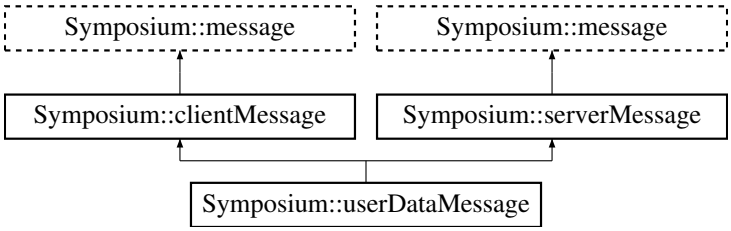
- user.h
- user.cpp

6.61 Symposium::userDataMessage Class Reference

class used to model a message to change the parameters of a user

```
#include <message.h>
```

Inheritance diagram for Symposium::userDataMessage:



Public Member Functions

- userDataMessage (msgType action, const std::pair< std::string, std::string > &actionOwner, msgOutcome result, const user &newUserData, uint_positive_cnt::type msgId=0)
- void invokeMethod (SymServer &server) override

  asks the server to modify a user's data
- void invokeMethod (SymClient &client) override

  propagate the changes made by a client over a user to other clients
- void completeAction (SymClient &client, msgOutcome serverResult) override

  completes an action for which the client asked to the server
- bool **operator==** (const userDataMessage &rhs) const
- bool **operator!=** (const userDataMessage &rhs) const

Private Member Functions

- template<class Archive >
  void **serialize** (Archive &ar, unsigned int)

Private Attributes

- user **newUserData**

Friends

- class **boost::serialization::access**

Additional Inherited Members

6.61.1 Detailed Description

class used to model a message to change the parameters of a user

6.61.2 Constructor & Destructor Documentation

6.61.2.1 userDataMessage()

```
userDataMessage::userDataMessage (
            msgType action,
            const std::pair< std::string, std::string > & actionOwner,
            msgOutcome result,
            const user & newUserData,
            uint_positive_cnt::type msgId = 0 )
```

Exceptions

messageException	if *action* is not consistent with the message type

6.61.3 Member Function Documentation

6.61.3.1 completeAction()

```
void userDataMessage::completeAction (
            SymClient & client,
            msgOutcome serverResult )  [override], [virtual]
```

completes an action for which the client asked to the server

Parameters

client	the same client that had originated the clientMessage

Some actions on SymClient require to ask to the server to assure that the action is valid and to propagate the change on the server. For such actions, only if the outcome from the server is positive the action can be actually done.

Reimplemented from Symposium::clientMessage.

6.61.3.2 invokeMethod() [1/2]

```
void userDataMessage::invokeMethod (
            SymServer & server )  [override], [virtual]
```

asks the server to modify a user's data

Parameters

server	the server the user is active on

When this message is received by the server, the *invokeMethod* calls SymServer::editUser on the *server* passed as parameter.

Reimplemented from Symposium::clientMessage.

6.61.3.3 invokeMethod() [2/2]

```
void userDataMessage::invokeMethod (
            SymClient & client )  [override], [virtual]
```

propagate the changes made by a client over a user to other clients

Parameters

client	the client on which propagate the change

When this message is received by the client, the *invokeMethod* calls SymClient::editUser on the *client* passed as parameter. A message of this type is sent from the server to clients only if:

- *action==msgType::changeUserData* ;

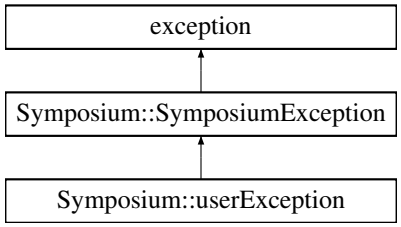
Reimplemented from Symposium::serverMessage.

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

- message.h
- message.cpp

6.62 Symposium::userException Class Reference

Inheritance diagram for Symposium::userException:



Public Types

- enum userExceptionCodes {
  shortPwd =0, **longPwd**, **noCharPwd**, **noNumPwd**,
  noSpecialCharPwd, **nickname**, **noPriv**, **noPermissionToChange**,
  LinkNoCorrect, **haveLink**, **path**, **pathForLink**,
  minPriv, **userSpace** }

  Specific error codes for userException. They are used as indexes to the error table string.

Public Member Functions

- **userException** (userExceptionCodes exceptionCode, const char *file, int line, const char *func)

Static Private Attributes

- static const char * **userErrors** [ ]

Additional Inherited Members

6.62.1 Member Data Documentation

6.62.1.1 userErrors

```
const char * userException::userErrors  [static], [private]
```

Initial value:

```
={"Too short password", "Too long password",
                                    "You don’t use any alphabetic character for password", "You don’t
       use any number for password",
                                    "You don’t use any special character for password", "You can choose
       a nickname",
                                    "You don’t have any privilege to this file anymore", "You cannot
       upgrade this privilege",
                                    "Inserted link is not correct", "You already have the access to
       this file",
                                    "The path for new resource is not correct", "The link isn’t
       correct, check it and try again",
                                    "You have to have reader privilege as a minimum privilege", "Your
       username has not contain spaces"}
```

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

- SymposiumException.h
- SymposiumException.cpp

Index