Swift Ticket

1.0

Generated by Doxygen 1.8.3.1

Sun Mar 3 2013 16:53:44

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

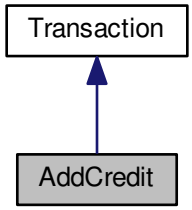
Class Documentation

4.1 AddCredit Class Reference

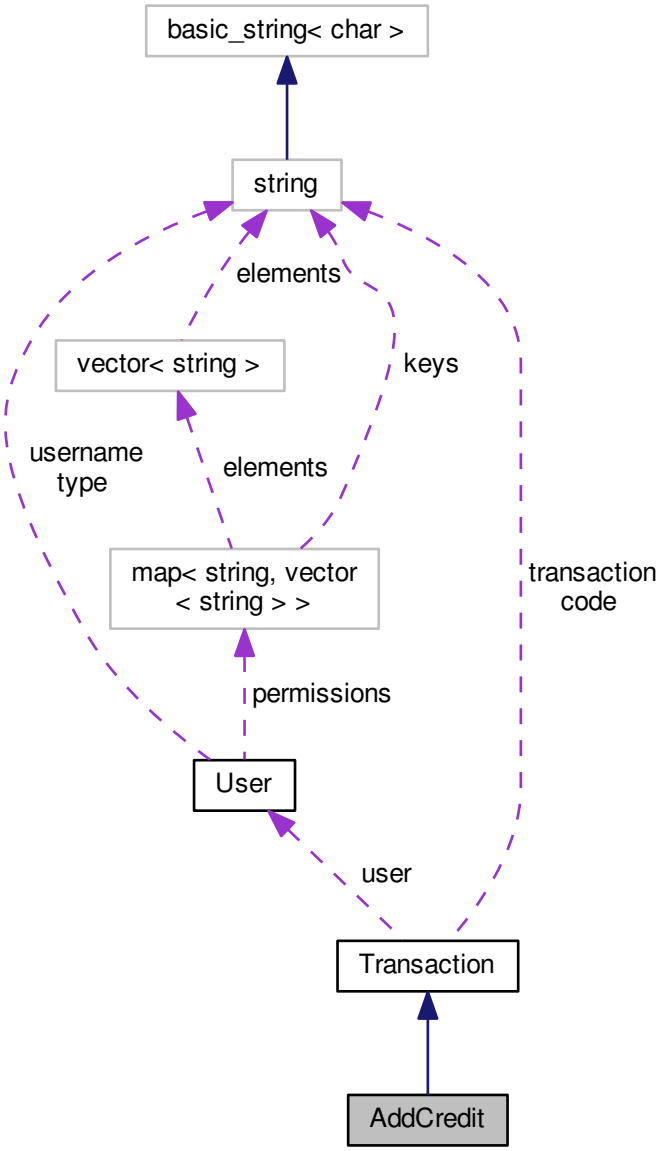
AddCredit Class.

`#include <AddCredit.hpp>`

Inheritance diagram for AddCredit:



Collaboration diagram for AddCredit:



Public Member Functions

- AddCredit (User current_user)

  AddCredit The constructor for the class, requires a user specified to add the credit to.
- void process_credit (string credit)

  process_credit Performs the addition of the credit to the previously specified user account.

- void process_username (CurrentUserAccounts user_accounts)

  process_username Determines the validity of the username specified, and processes that component of the transaction.

Protected Member Functions

- virtual void save_transaction ()

  save_transaction Virtual function signature for class abstraction in C++.

Private Attributes

- double credit

  credit Stores the amount of credit to add.

Additional Inherited Members

4.1.1 Detailed Description

AddCredit Class.

Used for the AddCredit transaction functions and attributes.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AddCredit (User *current_user*)

AddCredit The constructor for the class, requires a user specified to add the credit to.

Parameters

current_user	The user to add the credit specified to.

4.1.3 Member Function Documentation

4.1.3.1 void process_credit (string *credit*)

process_credit Performs the addition of the credit to the previously specified user account.

Parameters

credit	The actual amount of credit to add.

4.1.3.2 void process_username (CurrentUserAccounts *user_accounts*)

process_username Determines the validity of the username specified, and processes that component of the transaction.

Parameters

user_accounts	Provides a handle to the current user accounts object.

4.1.3.3 virtual void save_transaction ( ) [protected], [virtual]

save_transaction Virtual function signature for class abstraction in C++.

Reimplemented from Transaction.

4.1.4 Member Data Documentation

4.1.4.1 double credit [private]

credit Stores the amount of credit to add.

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

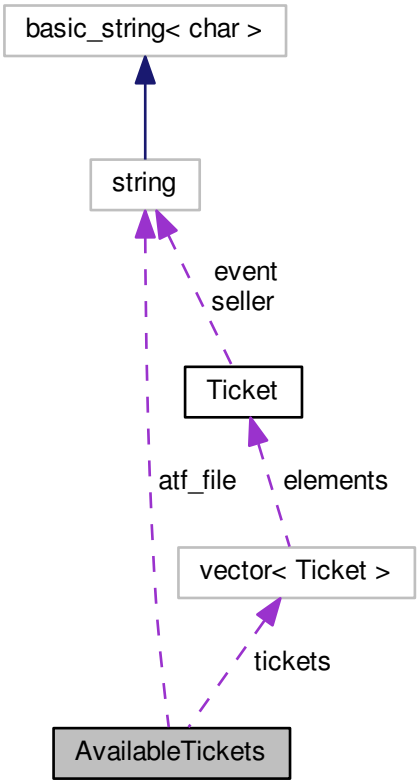
- AddCredit.hpp

4.2 AvailableTickets Class Reference

AvailableTickets Class.

```
#include <AvailableTickets.hpp>
```

Collaboration diagram for AvailableTickets:



Public Member Functions

- AvailableTickets ()
- AvailableTickets (string atf_file)

  AvailableTickets The constructor for the class, requires the location of the available tickets file on disk.
- void display_tickets ()

  display_tickets Displays the tickets which are available.
- Ticket get_ticket (string event, string username)

  get_ticket Gets the tickets for a specific event, from a specified seller.
- bool has_event (string event)

  has_event Determines whether an event exists within the available tickets file.
- bool has_seller (string username)

  has_seller Determines whether a valid seller exists for an event.

Private Member Functions

- void parse ()

parse Takes in the available ticket file, and parses it.

Private Attributes

- string atf_file
- vector< Ticket > tickets

  tickets Stores an array of the actual available tickets.

4.2.1 Detailed Description

AvailableTickets Class.

Used for storing the Available Tickets File, and provides functions for interacting with it.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 AvailableTickets ( )

4.2.2.2 AvailableTickets (string *atf_file*)

AvailableTickets The constructor for the class, requires the location of the available tickets file on disk.

Parameters

atf_file	The path to the available tickets file on disk.

4.2.3 Member Function Documentation

4.2.3.1 void display_tickets ( )

display_tickets Displays the tickets which are available.

4.2.3.2 Ticket get_ticket (string *event*, string *username*)

get_ticket Gets the tickets for a specific event, from a specified seller.

Parameters

event	Specifies which event to obtain tickets for.
username	Specifies which username to use as the seller.

Returns

Returns true if the tickets are available for the given seller and event, false if they are not.

4.2.3.3 bool has_event (string *event*)

has_event Determines whether an event exists within the available tickets file.

Parameters

event	Specifies which event to check for.

Returns

Returns true if the event exists, false if it does not.

4.2.3.4 bool has_seller (string *username*)

has_seller Determines whether a valid seller exists for an event.

Parameters

username	Specifies which username to check for.

Returns

Returns true if the seller exists, false if it does not.

4.2.3.5 void parse ( ) [private]

parse Takes in the available ticket file, and parses it.

The output is stored within the AvailableTickets object.

4.2.4 Member Data Documentation

4.2.4.1 string atf_file [private]

4.2.4.2 vector<Ticket> tickets [private]

tickets Stores an array of the actual available tickets.

atf_file Stores the path to the available tickets file on disk.

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

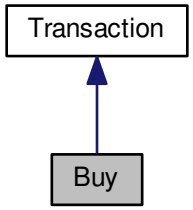
- AvailableTickets.hpp

4.3 Buy Class Reference

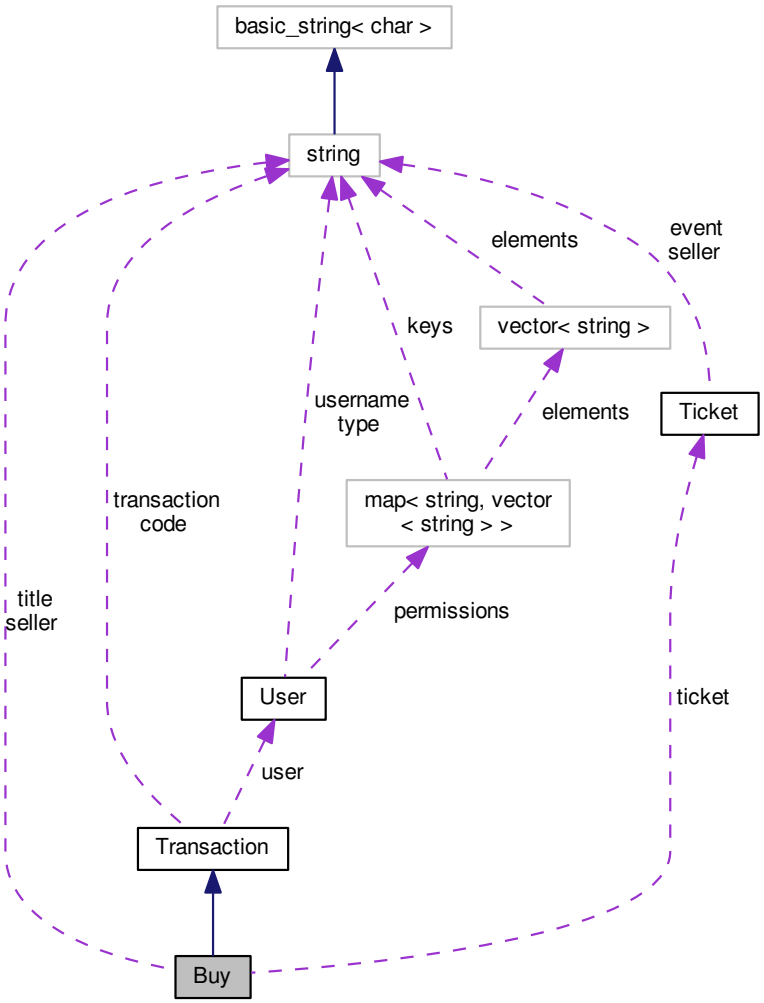
Buy Class.

`#include <Buy.hpp>`

Inheritance diagram for Buy:



Collaboration diagram for Buy:



Public Member Functions

- Buy (User current_user)

  Buy The constructor for the class, requires a reference to the current user who is initiating the transaction.
- void display_cost ()

  display_cost displays the cost per ticket and a total summary of the cost of the tickets.
- void process_confirmation (string confirm)

  process_confirmation Validates and processes the confirmation input for whether the user accepts the transaction.
- void process_seller (string username, AvailableTickets available_tickets)

  process_seller Validates and processes the specified seller for the transaction.
- void process_title (string title, AvailableTickets available_tickets)

process_title Validates and processes the transaction based on the event title specified.

- void process_volume (string volume)

  process_volume Validates and processes the volume of tickets being purchased.

Protected Member Functions

- virtual void save_transaction ()

  save_transaction Virtual function signature for class abstraction in C++.

Private Attributes

- string seller
- Ticket ticket

  ticket Stores a pointer to the ticket being purchased.
- string title
- int volume

Additional Inherited Members

4.3.1 Detailed Description

Buy Class.

Used for the Buy transaction functions and attributes.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Buy (User *current_user*)

Buy The constructor for the class, requires a reference to the current user who is initiating the transaction.

Parameters

current_user	

4.3.3 Member Function Documentation

4.3.3.1 void display_cost ( )

display_cost displays the cost per ticket and a total summary of the cost of the tickets.

4.3.3.2 void process_confirmation (string *confirm*)

process_confirmation Validates and processes the confirmation input for whether the user accepts the transaction.

Parameters

confirm	The confirmation, either 'yes' or 'no'.

4.3.3.3 void process_seller (string *username,* AvailableTickets *available_tickets*)

process_seller Validates and processes the specified seller for the transaction.

Parameters

username	The seller's username.
available_tickets	A handle to the available tickets.

4.3.3.4 void process_title (string *title,* AvailableTickets *available_tickets*)

process_title Validates and processes the transaction based on the event title specified.

Parameters

title	The event title for the tickets being purchased.
available_tickets	A handle to the available tickets.

4.3.3.5 void process_volume (string *volume*)

process_volume Validates and processes the volume of tickets being purchased.

Parameters

volume	The number of tickets being purchased.

4.3.3.6 virtual void save_transaction ( ) [protected],[virtual]

save_transaction Virtual function signature for class abstraction in C++.

Reimplemented from Transaction.

4.3.4 Member Data Documentation

4.3.4.1 string seller [private]

4.3.4.2 Ticket ticket [private]

ticket Stores a pointer to the ticket being purchased.

title Stores the title of the event for purchase. volume Stores the volume of tickets to purchase. seller Stores the name of the seller for the transaction.

4.3.4.3 string title [private]

4.3.4.4 int volume [private]

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

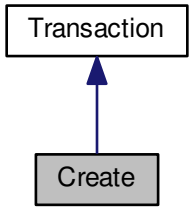
- Buy.hpp

4.4 Create Class Reference

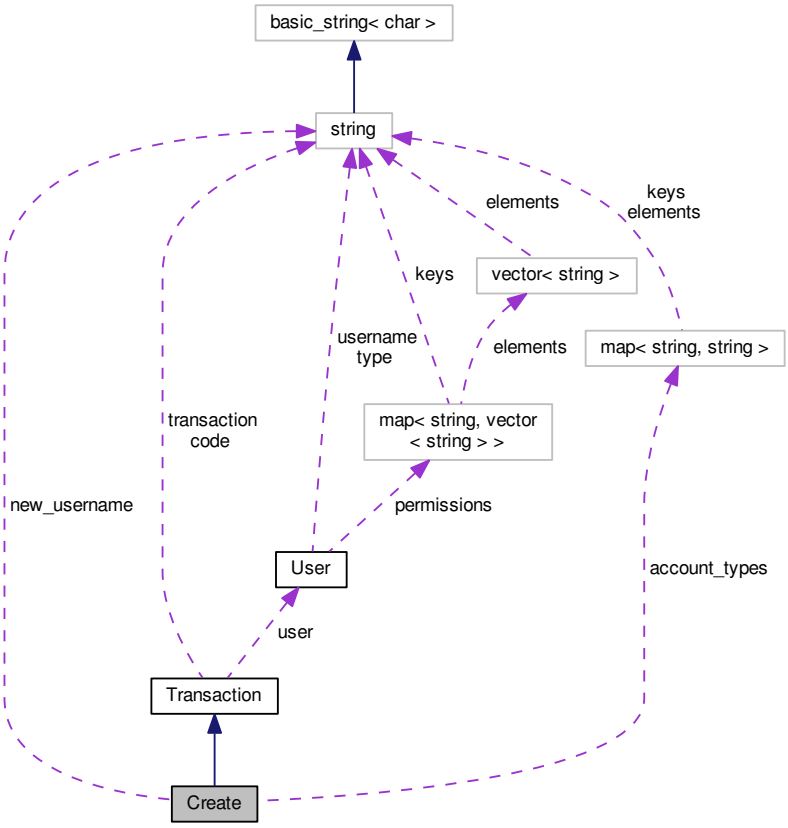
Create Class.

`#include <Create.hpp>`

Inheritance diagram for Create:



Collaboration diagram for Create:



Public Member Functions

- Create (User current_user)

  Create The constructor for the class, requires a reference to the current user initiating the transaction.
- void process_type (string type)

  process_type Validates and processes the type of user account specified for creation.
- void process_username (string username, CurrentUserAccounts user_accounts)

  process_username Validates and processes the username entered for creation.

Protected Member Functions

- virtual void save_transaction ()

  save_transaction Virtual function signature for class abstraction in C++.

Private Attributes

- map< string, string > account_types

types Stores the types of usernames that can be used.

- string new_username

Additional Inherited Members

4.4.1 Detailed Description

Create Class.

Used for the create transaction functions and attributes.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Create (User *current_user*)

Create The constructor for the class, requires a reference to the current user initiating the transaction.

Parameters

current_user	

4.4.3 Member Function Documentation

4.4.3.1 void process_type (string *type*)

process_type Validates and processes the type of user account specified for creation.

Parameters

type	The type of user account to create.

4.4.3.2 void process_username (string *username,* CurrentUserAccounts *user_accounts*)

process_username Validates and processes the username entered for creation.

Parameters

username	The username to create.
user_accounts	A handle to the current user accounts.

4.4.3.3 virtual void save_transaction ( ) [protected], [virtual]

save_transaction Virtual function signature for class abstraction in C++.

Reimplemented from Transaction.

4.4.4 Member Data Documentation

4.4.4.1 map<string, string> account_types [private]

Initial value:

```
= {
            {"admin", "AA"},
            {"full-standard", "FS"},
            {"buy-standard", "BS"},
            {"sell-standard", "SS"}
    }
```

types Stores the types of usernames that can be used.

new_username Stores the new username to create.

4.4.4.2 string new_username [private]

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

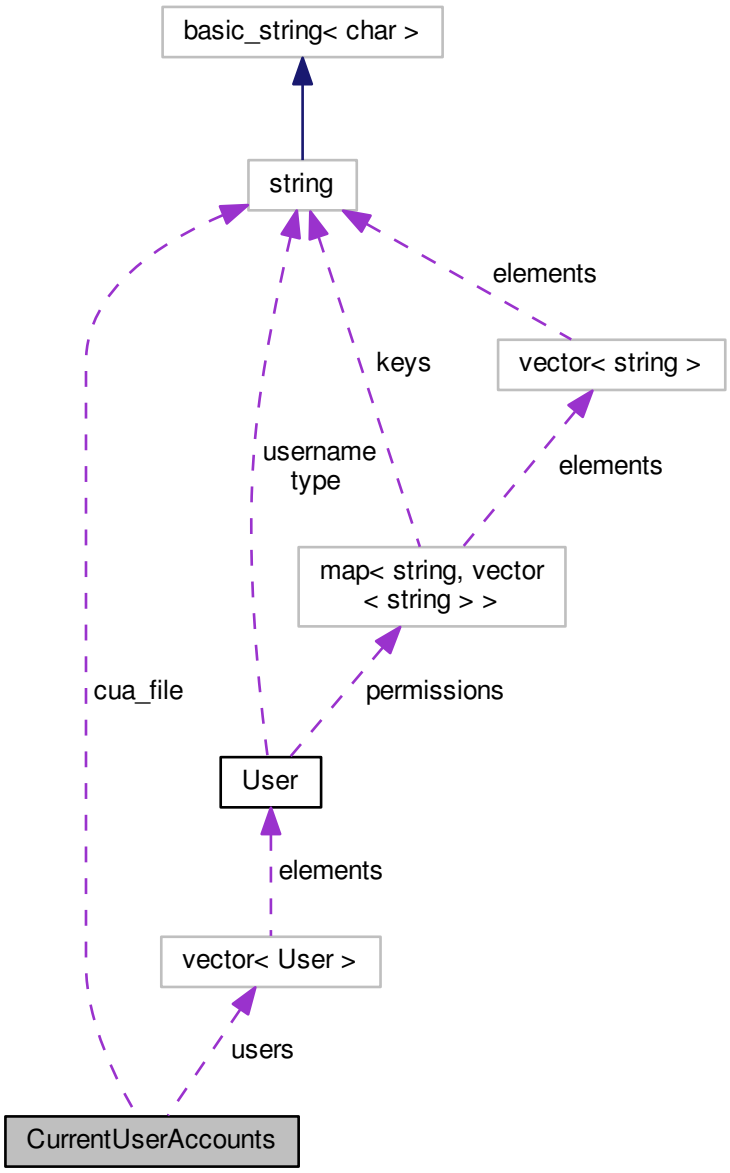
- Create.hpp

4.5 CurrentUserAccounts Class Reference

CurrentUserAccounts Class.

`#include <CurrentUserAccounts.hpp>`

Collaboration diagram for CurrentUserAccounts:



Public Member Functions

- CurrentUserAccounts ()
- CurrentUserAccounts (string cua_file)

  CurrentUserAccounts The constructor for the class, requires a reference to the current user accounts file path.
- void display_users ()

display_users Displays the users within the current user accounts file.

- User get_user (string username)

  get_user Returns a user object based on a user account stored within the current user accounts file.

- bool has_user (string username)

  has_user Validates and verifies whether a user exists within the current user accounts file.

Private Member Functions

- void parse ()

  parse Takes in the current user accounts file, and parses it.

Private Attributes

- string cua_file
- vector< User > users

  users The list of current users.

4.5.1 Detailed Description

CurrentUserAccounts Class.

Used for storing the current user accounts file, and provides functions for interacting with it.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 CurrentUserAccounts ( )

4.5.2.2 CurrentUserAccounts (string *cua_file*)

CurrentUserAccounts The constructor for the class, requires a reference to the current user accounts file path.

Parameters

cua_file	The path to the current user accounts file on disk.

4.5.3 Member Function Documentation

4.5.3.1 void display_users ( )

display_users Displays the users within the current user accounts file.

4.5.3.2 User get_user (string *username*)

get_user Returns a user object based on a user account stored within the current user accounts file.

Parameters

username	The name for the user specified.

Returns

Returns a User object, containing the data for the user account specified.

4.5.3.3 bool has_user (string *username*)

has_user Validates and verifies whether a user exists within the current user accounts file.

Parameters

username	The username to check.

Returns

Returns true if the user exists, false if it does not.

4.5.3.4 void parse ( ) [private]

parse Takes in the current user accounts file, and parses it.

The output is stored within the CurrentUserAccounts object.

4.5.4 Member Data Documentation

4.5.4.1 string cua_file [private]

4.5.4.2 vector<User> users [private]

users The list of current users.

cua_file The path to the current user accounts file.

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

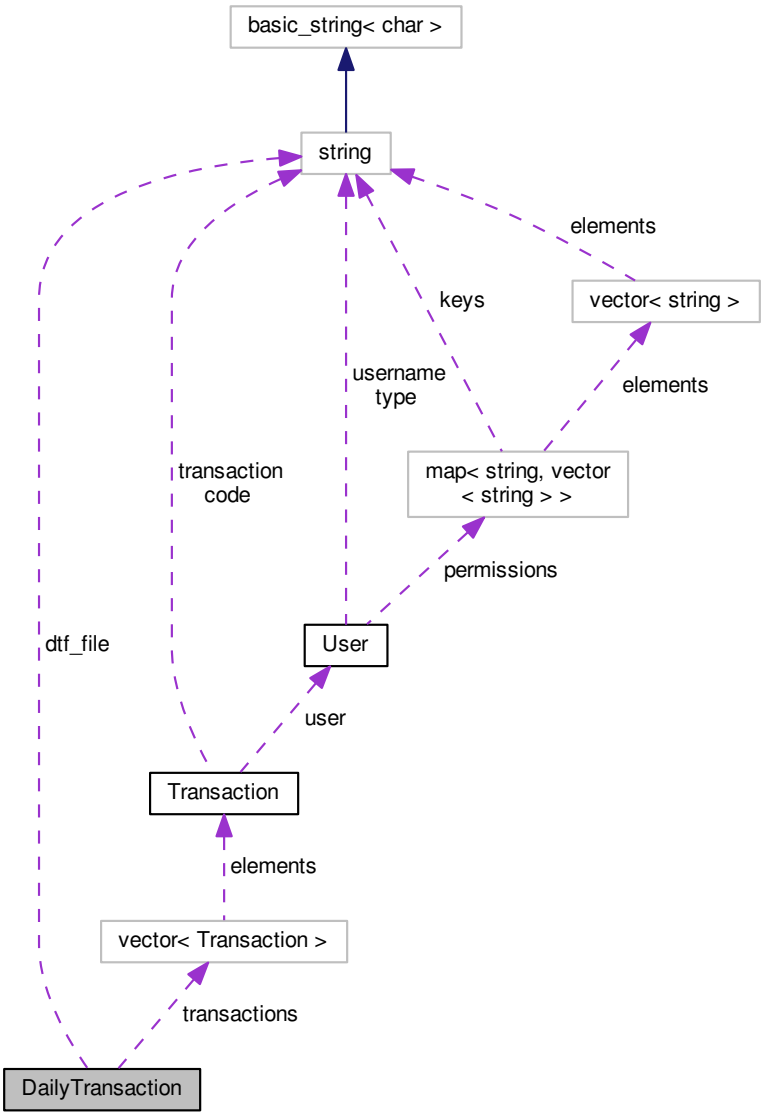
- CurrentUserAccounts.hpp

4.6 DailyTransaction Class Reference

DailyTransaction Class.

`#include <DailyTransaction.hpp>`

Collaboration diagram for DailyTransaction:



Public Member Functions

- DailyTransaction ()
- DailyTransaction (string dtf_file)

  DailyTransaction The constructor for the class, requires a reference to the daily transaction file path.
- void save (Transaction transaction)

  save Saves a transaction to the daily transaction file.
- void write ()

  write Writes the daily transaction file out to disk.

Private Attributes

- string dtf_file
- vector< Transaction > transactions

  transactions The list of transactions currently in the daily transaction file.

4.6.1 Detailed Description

DailyTransaction Class.

Used for storing the daily transaction file and provides functions for interacting with it.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 DailyTransaction ( )

4.6.2.2 DailyTransaction (string *dtf_file*)

DailyTransaction The constructor for the class, requires a reference to the daily transaction file path.

Parameters

dtf_file	The path to the daily transaction file on disk.

4.6.3 Member Function Documentation

4.6.3.1 void save (Transaction *transaction*)

save Saves a transaction to the daily transaction file.

Parameters

transaction	The transaction to be saved.

4.6.3.2 void write ( )

write Writes the daily transaction file out to disk.

4.6.4 Member Data Documentation

4.6.4.1 string dtf_file [private]

4.6.4.2 vector<Transaction> transactions [private]

transactions The list of transactions currently in the daily transaction file.

dtf_file The path to the daily transaction file on disk.

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

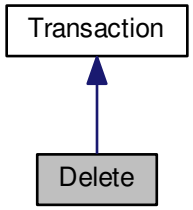
- DailyTransaction.hpp

4.7 Delete Class Reference

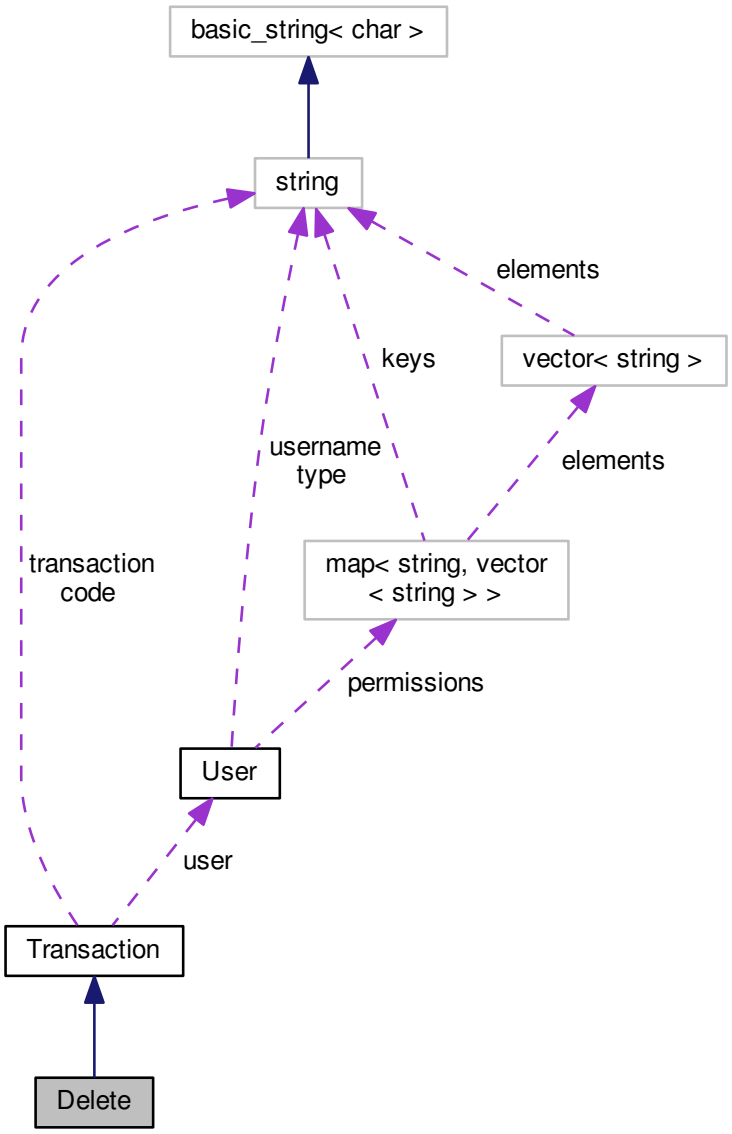
Delete Class.

`#include <Delete.hpp>`

Inheritance diagram for Delete:



Collaboration diagram for Delete:



Public Member Functions

- Delete (User current_user)

  Delete The constructor for the class, requires a reference to the current user session.
- void process_username (string username, CurrentUserAccounts user_accounts)

  process_username Validates and processes a username for deletion.

Protected Member Functions

- virtual void save_transaction ()

  save_transaction Virtual function signature used for interface creation in C++.

Additional Inherited Members

4.7.1 Detailed Description

Delete Class.

Used for the delete transaction functions and attributes.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Delete (User *current_user*)

Delete The constructor for the class, requires a reference to the current user session.

Parameters

current_user	

4.7.3 Member Function Documentation

4.7.3.1 void process_username (string *username,* CurrentUserAccounts *user_accounts*)

process_username Validates and processes a username for deletion.

Parameters

username	The username to delete.
user_accounts	A handle to the current user accounts.

4.7.3.2 virtual void save_transaction ( ) [protected], [virtual]

save_transaction Virtual function signature used for interface creation in C++.

Reimplemented from Transaction.

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

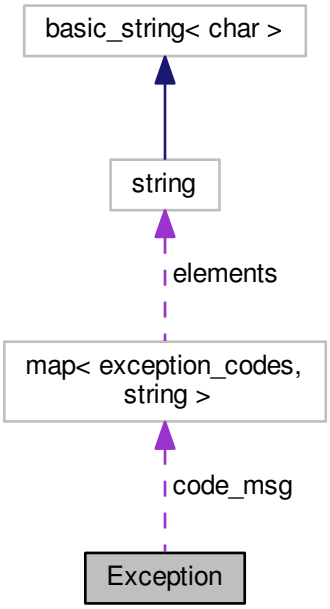
- Delete.hpp

4.8 Exception Class Reference

Exception Class.

```
#include <Exception.hpp>
```

Collaboration diagram for Exception:



Public Member Functions

- Exception (exception_codes code)

  Exception The constructor for the class, requires an exception code.
- string mesg ()

  mesg Outputs an exception message to the console.

Private Attributes

- exception_codes code
- map< exception_codes, string > code_msg

  code_msg Maps the transaction error codes to error messages.

4.8.1 Detailed Description

Exception Class.

Used for all exceptions within every transaction, and provides functions for providing diagnostic output.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Exception (exception_codes *code*)

Exception The constructor for the class, requires an exception code.

Parameters

code	The exception code to use.

4.8.3 Member Function Documentation

4.8.3.1 string mesg ( )

mesg Outputs an exception message to the console.

Returns

Returns an exception string.

4.8.4 Member Data Documentation

4.8.4.1 exception_codes code [private]

4.8.4.2 map<exception_codes, string> code_msg [private]

code_msg Maps the transaction error codes to error messages.

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

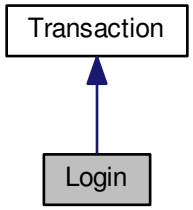
- Exception.hpp

4.9 Login Class Reference

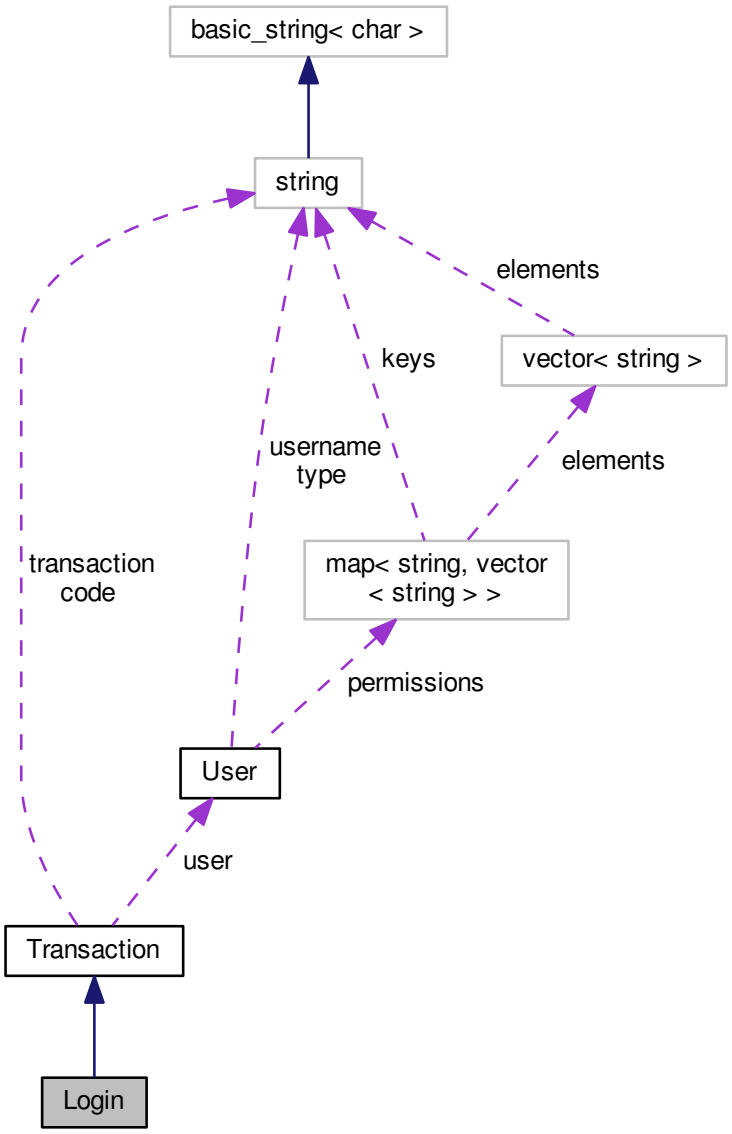
Login Class.

`#include <Login.hpp>`

Inheritance diagram for Login:



Collaboration diagram for Login:



Public Member Functions

- Login (User current_user)

  Login The constructor for the class, requires a handle to the current user logged in (if applicable).

- User process_username (string username, CurrentUserAccounts user_accounts)

  process_username Validates and processes the username specified to log in with.

Protected Member Functions

- virtual void save_transaction ()

  save_transaction Virtual function signature for class abstraction in C++.

Additional Inherited Members

4.9.1 Detailed Description

Login Class.

Used for the login transaction functions and attributes.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Login (User *current_user*)

Login The constructor for the class, requires a handle to the current user logged in (if applicable).

Parameters

current_user	The current user logged in.

4.9.3 Member Function Documentation

4.9.3.1 User process_username (string *username,* CurrentUserAccounts *user_accounts*)

process_username Validates and processes the username specified to log in with.

Parameters

username	The username for the account to log in.
user_accounts	A handle to the current user accounts.

Returns

Returns a User object for the user which was logged in, or null if an exception occurs.

4.9.3.2 virtual void save_transaction ( ) [protected], [virtual]

save_transaction Virtual function signature for class abstraction in C++.

Reimplemented from Transaction.

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

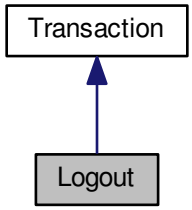
- Login.hpp

4.10 Logout Class Reference

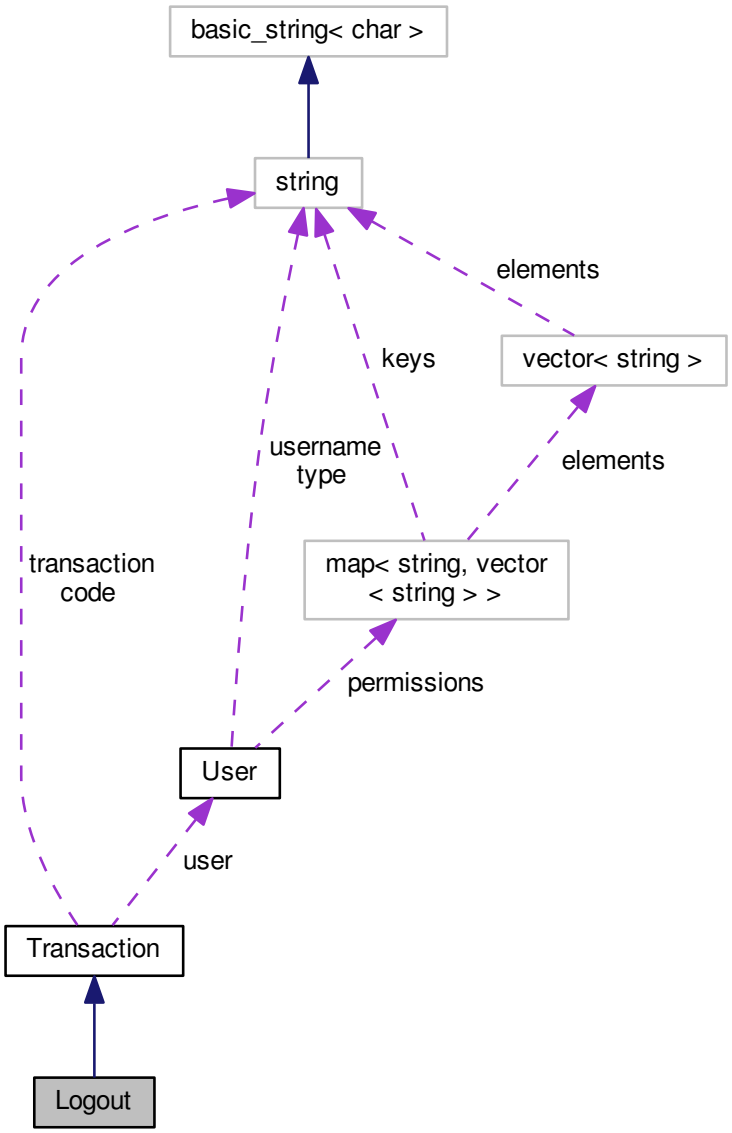
Logout Class.

`#include <Logout.hpp>`

Inheritance diagram for Logout:



Collaboration diagram for Logout:



Public Member Functions

- Logout (User current_user)

  Logout The constructor for the class, requires a handle to the current user logged in.

Protected Member Functions

- virtual void save_transaction ()

  save_transaction Virtual function signature for class abstraction in C++.

Additional Inherited Members

4.10.1 Detailed Description

Logout Class.

Used for the logout transaction functions and attributes.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Logout (User *current_user*)

Logout The constructor for the class, requires a handle to the current user logged in.

Parameters

current_user	The current user logged in, to be logged out.

4.10.3 Member Function Documentation

4.10.3.1 virtual void save_transaction ( ) [protected], [virtual]

save_transaction Virtual function signature for class abstraction in C++.

Reimplemented from Transaction.

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

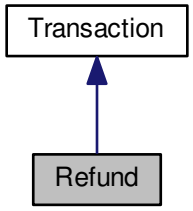
- Logout.hpp

4.11 Refund Class Reference

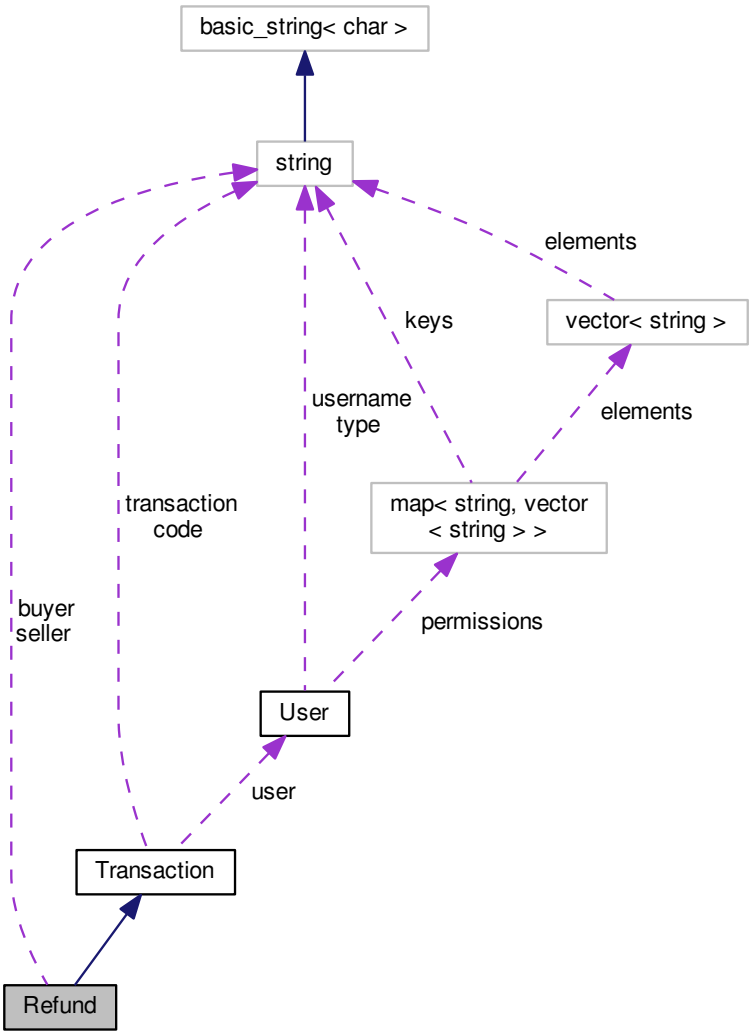
Refund Class.

```
#include <Refund.hpp>
```

Inheritance diagram for Refund:



Collaboration diagram for Refund:



Public Member Functions

- Refund (User current_user)

  Refund The constructor for the class, requires a handle to the current user logged in.
- void process_buyer (string buyer, CurrentUserAccounts user_accounts)

  process_buyer Validates and processes the specified buyer for the refund.
- void process_credit (string credit)

  process_credit Validates and processes the specified amount for the refund.
- void process_seller (string seller, CurrentUserAccounts user_accounts)

  process_seller Validates and processes the specified seller for the refund.

Protected Member Functions

- virtual void save_transaction ()

  save_transaction Virtual function signature for class abstraction in C++.

Private Attributes

- string buyer

  buyer Contains the buyer for the original sale to be refunded.
- double credit
- string seller

Additional Inherited Members

4.11.1 Detailed Description

Refund Class.

Used for the refund transaction functions and attributes.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Refund (User *current_user*)

Refund The constructor for the class, requires a handle to the current user logged in.

Parameters

current_user	A handle to the current user logged in.

4.11.3 Member Function Documentation

4.11.3.1 void process_buyer (string *buyer,* CurrentUserAccounts *user_accounts*)

process_buyer Validates and processes the specified buyer for the refund.

Parameters

buyer	The buyer for the sale to be refunded.
user_accounts	A handle to the current user accounts.

4.11.3.2 void process_credit (string *credit*)

process_credit Validates and processes the specified amount for the refund.

Parameters

credit	The specified refund amount, in dollars.

4.11.3.3 void process_seller (string *seller,* CurrentUserAccounts *user_accounts*)

process_seller Validates and processes the specified seller for the refund.

Parameters

seller	The seller for the sale to be refunded.
user_accounts	A handle to the current user accounts.

4.11.3.4 virtual void save_transaction ( ) [protected], [virtual]

save_transaction Virtual function signature for class abstraction in C++.

Reimplemented from Transaction.

4.11.4 Member Data Documentation

4.11.4.1 string buyer [private]

buyer Contains the buyer for the original sale to be refunded.

seller Contains the seller for the refund. credit Contains the amount of credit to refund to the buyer, in dollars.

4.11.4.2 double credit [private]

4.11.4.3 string seller [private]

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

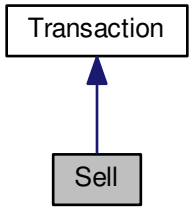
- Refund.hpp

4.12 Sell Class Reference

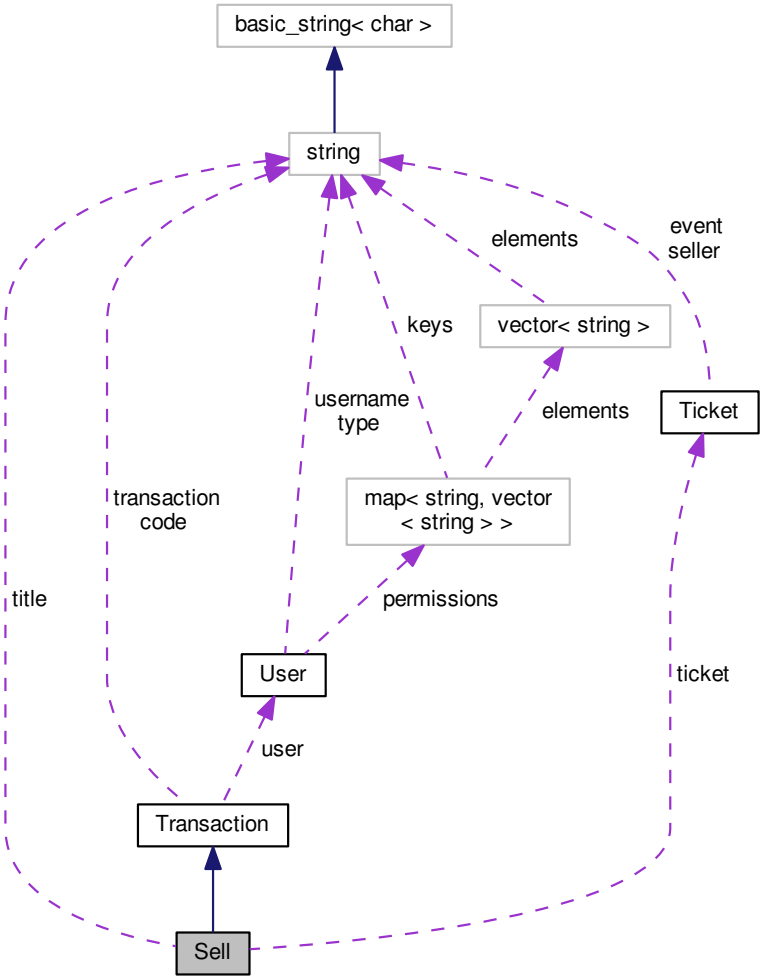
Sell Class.

`#include <Sell.hpp>`

Inheritance diagram for Sell:



Collaboration diagram for Sell:



Public Member Functions

- Sell (User current_user)

  Sell The constructor of the function, requires a handle to the current user logged in.

- void process_price (string price)

  process_price Validates and processes the specified price for the tickets.

- void process_title (string title)

  process_title Validates and processes the title for the ticket sale.

- void process_volume (string volume)

  process_volume Validates and processes the specified number of tickets to be sold.

Protected Member Functions

- virtual void save_transaction ()

  save_transaction Virtual function signature for class abstraction in C++.

Private Attributes

- double price
- Ticket ticket

  ticket Contains the ticket to be sold.
- string title
- int volume

Additional Inherited Members

4.12.1 Detailed Description

Sell Class.

Used for the sell transaction functions and attributes.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 Sell (User *current_user*)

Sell The constructor of the function, requires a handle to the current user logged in.

Parameters

current_user	A handle to the current user.

4.12.3 Member Function Documentation

4.12.3.1 void process_price (string *price*)

process_price Validates and processes the specified price for the tickets.

Parameters

price	The price, in dollars, for the tickets being sold.

4.12.3.2 void process_title (string *title*)

process_title Validates and processes the title for the ticket sale.

Parameters

title	The title for the event tickets being sold.

4.12.3.3 void process_volume (string *volume*)

process_volume Validates and processes the specified number of tickets to be sold.

Parameters

volume	The number of tickets to be sold.

4.12.3.4 virtual void save_transaction ( ) [protected], [virtual]

save_transaction Virtual function signature for class abstraction in C++.

Reimplemented from Transaction.

4.12.4 Member Data Documentation

4.12.4.1 double price [private]

4.12.4.2 Ticket ticket [private]

ticket Contains the ticket to be sold.

title Contains the title for the ticket sale. price Contains the price for the ticket sales. volume Contains the volume of tickets to sell.

4.12.4.3 string title [private]

4.12.4.4 int volume [private]

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

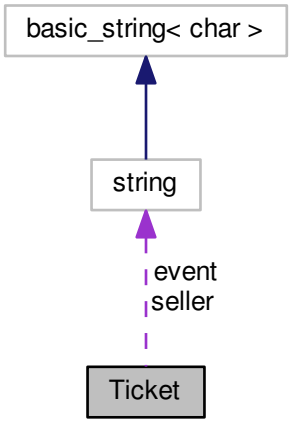
- Sell.hpp

4.13 Ticket Class Reference

Ticket Class.

`#include <Ticket.hpp>`

Collaboration diagram for Ticket:



Public Member Functions

- Ticket ()
- Ticket (string event, string seller, int volume, double price)

  Ticket The constructor for the class, requires all class attributes to be defined.
- string get_event ()

  get_event Access function for the event title.
- double get_price ()

  get_price Access function for the ticket price.
- string get_seller ()

  get_seller Access function for the seller.
- int get_volume ()

  get_volume Access function for the ticket volume.

Private Attributes

- string event

  event The event title for the tickets.
- double price
- string seller
- int volume

4.13.1 Detailed Description

Ticket Class.

Used for storing ticket data, and provides functions for interacting with them.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Ticket ( )

4.13.2.2 Ticket (string *event,* string *seller,* int *volume,* double *price*)

Ticket The constructor for the class, requires all class attributes to be defined.

Parameters

event	The event title for the tickets.
seller	The seller of the tickets.
volume	The amount of tickets available.
price	The cost of the tickets in dollars.

4.13.3 Member Function Documentation

4.13.3.1 string get_event ( )

get_event Access function for the event title.

Returns

Returns the event title as a string.

4.13.3.2 double get_price ( )

get_price Access function for the ticket price.

Returns

Returns the cost of the tickets, in dollars.

4.13.3.3 string get_seller ( )

get_seller Access function for the seller.

Returns

Returns the seller as a string.

4.13.3.4 int get_volume ( )

get_volume Access function for the ticket volume.

Returns

Returns the amount of tickets available as an integer.

4.13.4 Member Data Documentation

4.13.4.1 string event [private]

event The event title for the tickets.

seller The seller of the tickets. volume The amount of tickets available. price The cost of the tickets in dollars.

4.13.4.2 double price [private]

4.13.4.3 string seller [private]

4.13.4.4 int volume [private]

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

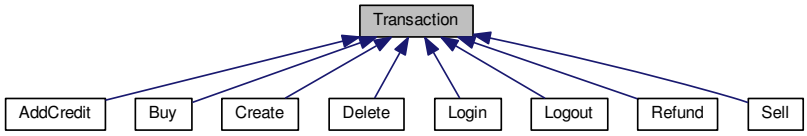
- Ticket.hpp

4.14 Transaction Class Reference

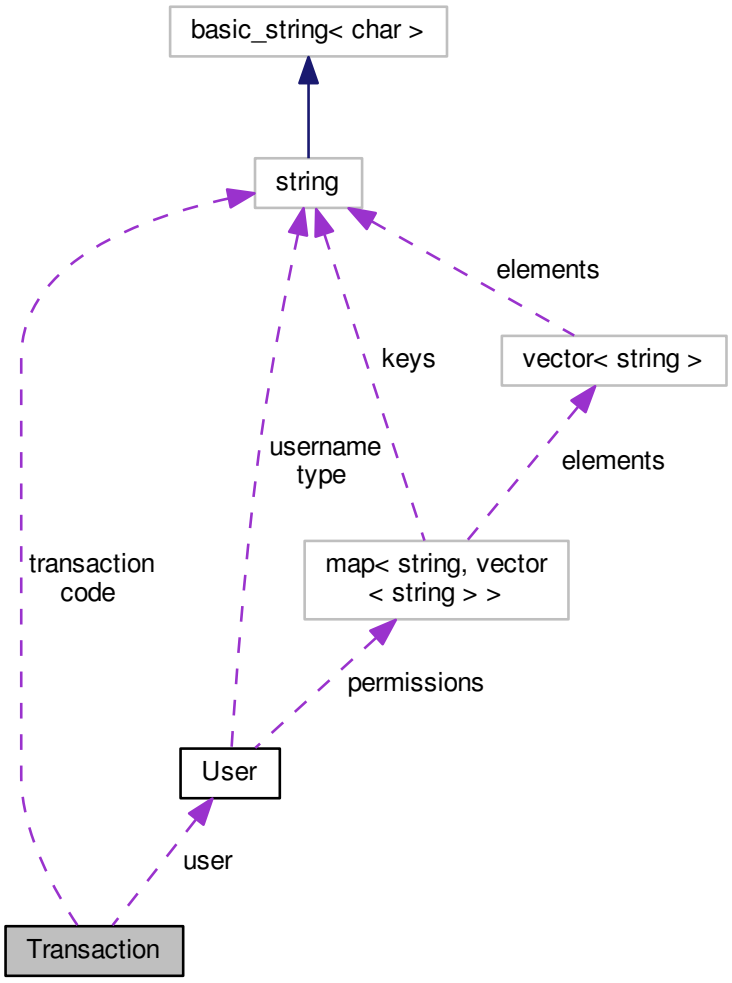
Transaction Class.

`#include <Transaction.hpp>`

Inheritance diagram for Transaction:



Collaboration diagram for Transaction:



Public Member Functions

- string get_transaction ()

  get_transaction Provides the stored formatted transaction string for any transaction.

Protected Member Functions

- string format (string code, string username, string type, double credit)

  format Formats transactions which require the usage of a user name, type, and credit amount.
- string format (string code, string username, string type, string seller, double refund)

  format Formats transactions which require the usage of a user name, type, seller, and credit amount.

- string format (string code, string event, string seller, int volume, double price)

  format Formats transactions which require the usage of an event title, seller user name, ticket volume, and price.
- virtual void save_transaction ()

  save_transaction Virtual function signature for class abstraction in C++.

Protected Attributes

- string code
- string transaction
- User user

  user Stores the user who performed the transaction.

4.14.1 Detailed Description

Transaction Class.

Acts as a superclass for the other transaction classes.

4.14.2 Member Function Documentation

4.14.2.1 string format (string *code,* string *username,* string *type,* double *credit*) [protected]

format Formats transactions which require the usage of a user name, type, and credit amount.

Parameters

code	The transaction code for the transaction.
username	The username specified for the transaction.
type	The type of user for the transaction.
credit	The credit amount for the transaction.

Returns

Returns a formatted transaction string containing all arguments.

4.14.2.2 string format (string *code,* string *username,* string *type,* string *seller,* double *refund*) [protected]

format Formats transactions which require the usage of a user name, type, seller, and credit amount.

Parameters

code	The transaction code for the transaction.
username	The user name specified for the transaction.
type	The type of user for the transaction.
seller	The seller user name specified for the transaction.
refund	The credit amount for the transaction.

Returns

Returns a formatted transaction string containing all arguments.

4.14.2.3 string format (string *code,* string *event,* string *seller,* int *volume,* double *price*) [protected]

format Formats transactions which require the usage of an event title, seller user name, ticket volume, and price.

Parameters

code	The transaction code for the transaction.
event	The event title specified for the transaction.
seller	The seller user name specified for the transaction.
volume	The number of tickets in the transaction.
price	The price of the tickets in the transaction, in dollars.

Returns

Returns a formatted transaction string containing all arguments.

4.14.2.4 string get_transaction ( )

get_transaction Provides the stored formatted transaction string for any transaction.

Returns

Returns the formatted transaction string.

4.14.2.5 virtual void save_transaction ( ) [protected], [virtual]

save_transaction Virtual function signature for class abstraction in C++.

Reimplemented in Refund, Sell, Create, Buy, AddCredit, Delete, Login, and Logout.

4.14.3 Member Data Documentation

4.14.3.1 string code [protected]

4.14.3.2 string transaction [protected]

4.14.3.3 User user [protected]

user Stores the user who performed the transaction.

code Stores the transaction code. transaction Stores the formatted transaction string.

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

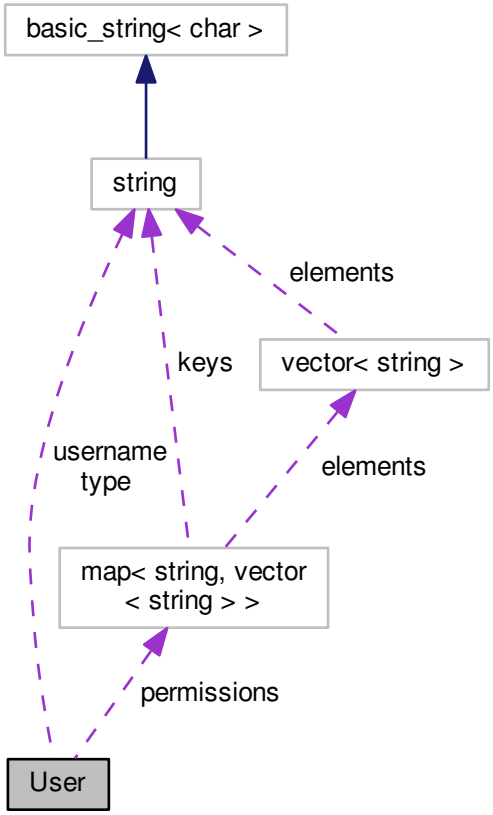
- Transaction.hpp

4.15 User Class Reference

User Class.

`#include <User.hpp>`

Collaboration diagram for User:



Public Member Functions

- User ()
- User (string username, string type, double credit)

  User The constructor for the class, requires all class attributes to be defined.
- double get_credit ()

  get_credit Access method for the user's credit amount.
- bool get_status ()

  get_status Returns the login status of the user.
- string get_type ()

  get_type Access method for the user's type.

- string get_username ()

  get_username Access method for the user's name attribute.
- bool has_permissions (string transaction)

  has_permissions Determines whether a user has permissions to execute a specified transaction.
- void login ()

  login Changes the user's login_status attribute to true.
- void logout ()

  logout Changes the user's login_status attribute to false.

Private Attributes

- double credit
- bool login_status
- map< string, vector< string > > permissions

  permissions Maps the user types to allowed transaction permissions.
- string type
- string username

  username Stores the user name string.

4.15.1 Detailed Description

User Class.

Used for storing user data, and provides functions for interacting with them.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 User ( )

4.15.2.2 User (string *username*, string *type*, double *credit*)

User The constructor for the class, requires all class attributes to be defined.

Parameters

username	The specified user name.
type	The user type for the account.
credit	The amount of credit for the user.

4.15.3 Member Function Documentation

4.15.3.1 double get_credit ( )

get_credit Access method for the user's credit amount.

Returns

Returns the user's credit amount, in dollars.

4.15.3.2 bool get_status ( )

get_status Returns the login status of the user.

Returns

Returns true if the user is logged in. False if not.

4.15.3.3 string get_type ( )

get_type Access method for the user's type.

Returns

Returns the user's type, as a string.

4.15.3.4 string get_username ( )

get_username Access method for the user's name attribute.

Returns

Returns the user's name as a string.

4.15.3.5 bool has_permissions (string *transaction*)

has_permissions Determines whether a user has permissions to execute a specified transaction.

Parameters

transaction	The specified transaction to determine permissions.

Returns

Returns true if the user has the appropriate permissions to execute the transaction, false if not.

4.15.3.6 void login ( )

login Changes the user's login_status attribute to true.

The user is now logged in.

4.15.3.7 void logout ( )

logout Changes the user's login_status attribute to false.

The user is now logged out.

4.15.4 Member Data Documentation

4.15.4.1 double credit [private]

4.15.4.2 bool login_status [private]

4.15.4.3 map<string, vector<string> > permissions [private]

Initial value:

```
= {
        {"AA", {"login", "logout", "create", "delete", "sell", "buy", "refund", "addcredit"}},
        {"FS", {"login", "logout", "sell", "buy", "addcredit"}},
        {"BS", {"login", "logout", "buy", "addcredit"}},
        {"SS", {"login", "logout", "sell", "addcredit"}}
    }
```

permissions Maps the user types to allowed transaction permissions.

4.15.4.4 string type [private]

4.15.4.5 string username [private]

username Stores the user name string.

type Stores the user type. credit Stores the user's credit amount. login_status Stores whether the user is logged in. True if logged in false, if not.

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

- User.hpp

4.16 Validate Class Reference

Validate Class.

`#include <Validate.hpp>`

Static Public Member Functions

- static bool atf_entry (string entry)

  atf_entry Validates available tickets file entries, and determines if they conform to the correct format and constraints.
- static bool cua_entry (string entry)

  cua_entry Validates current user accounts file entries, and determines if they conform to the correct format and constraints.
- static bool dollars (string amount, double &converted)

  dollars Accepts a price amount input as a string, validates it, and determines if it conforms to the correct format and constraints, then sets the converted parameter into the converted dollar amount.
- static bool title (string event)

  title Validates event titles, and determines if they conform to the correct format and constraints.
- static bool username (string username)

  username Validates user names, and determines if they conform to the correct format and constraints.
- static bool volume (string amount, int &converted)

  volume Accepts a ticket buy or sell amount as a string, validates it, and determines if it conforms to the correct format and constraints, then sets the converted parameter into the converted volume amount.

4.16.1 Detailed Description

Validate Class.

Provides functions to validate acceptable inputs and outputs for various other classes.

4.16.2 Member Function Documentation

4.16.2.1 static bool atf_entry (string *entry*) [static]

atf_entry Validates available tickets file entries, and determines if they conform to the correct format and constraints.

Parameters

entry	An entry in the available tickets file.

Returns

Returns true if the entry is valid, false if not.

4.16.2.2 static bool cua_entry (string *entry*) [static]

cua_entry Validates current user accounts file entries, and determines if they conform to the correct format and constraints.

Parameters

entry	An entry in the current user accounts file.

Returns

Returns true if the entry is valid, false if not.

4.16.2.3 static bool dollars (string *amount,* double & *converted*) [static]

dollars Accepts a price amount input as a string, validates it, and determines if it conforms to the correct format and constraints, then sets the converted parameter into the converted dollar amount.

Parameters

amount	The amount specified, unformatted, in dollars.
converted	The variable memory space to output the formatted dollar amount to.

Returns

Returns true if the amount specified is a valid amount, false if not.

4.16.2.4 static bool title (string *event*) [static]

title Validates event titles, and determines if they conform to the correct format and constraints.

Parameters

event	The event title to validate.

Returns

Returns true if the event title is valid, false if not.

4.16.2.5 static bool username (string *username*) [static]

username Validates user names, and determines if they conform to the correct format and constraints.

Parameters

username	The user name to validate.

Returns

Returns true if the user name is valid, false if not.

4.16.2.6 static bool volume (string *amount,* int & *converted*) [static]

volume Accepts a ticket buy or sell amount as a string, validates it, and determines if it conforms to the correct format and constraints, then sets the converted parameter into the converted volume amount.

Parameters

amount	The amount of tickets for buy/sell.
converted	The variable memory space to output the formatted volume to.

Returns

Returns true if the amount specified is a valid amount, false if not.

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

- Validate.hpp

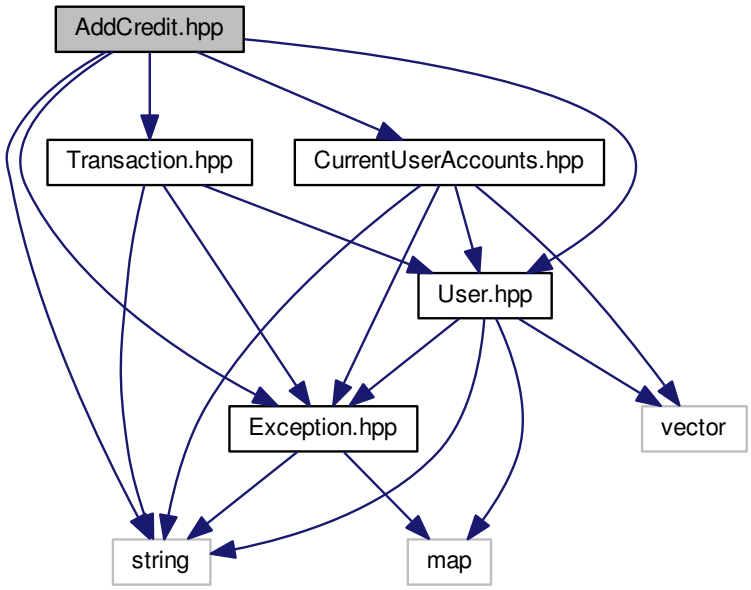
Chapter 5

File Documentation

5.1 AddCredit.hpp File Reference

```
#include "Transaction.hpp"
#include "CurrentUserAccounts.hpp"
#include "User.hpp"
#include "Exception.hpp"
#include <string>
```

Include dependency graph for AddCredit.hpp:



Classes

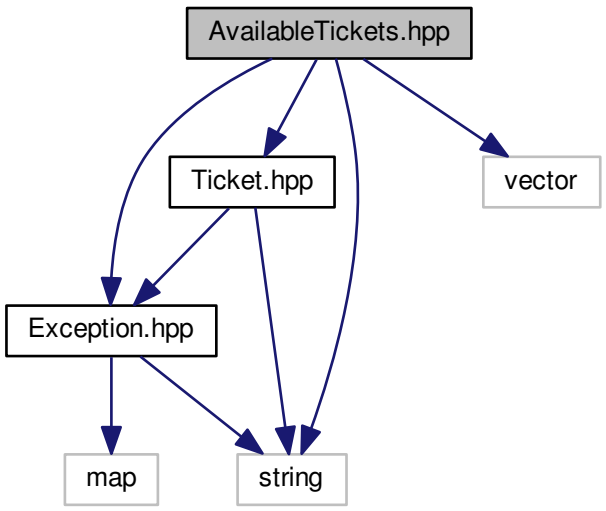
- class AddCredit

AddCredit Class.

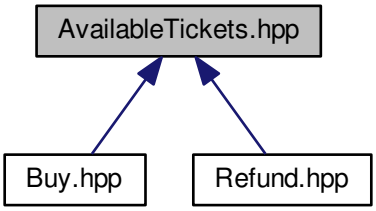
5.2 AvailableTickets.hpp File Reference

```
#include "Ticket.hpp"
#include "Exception.hpp"
#include <string>
#include <vector>
```

Include dependency graph for AvailableTickets.hpp:



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



Classes

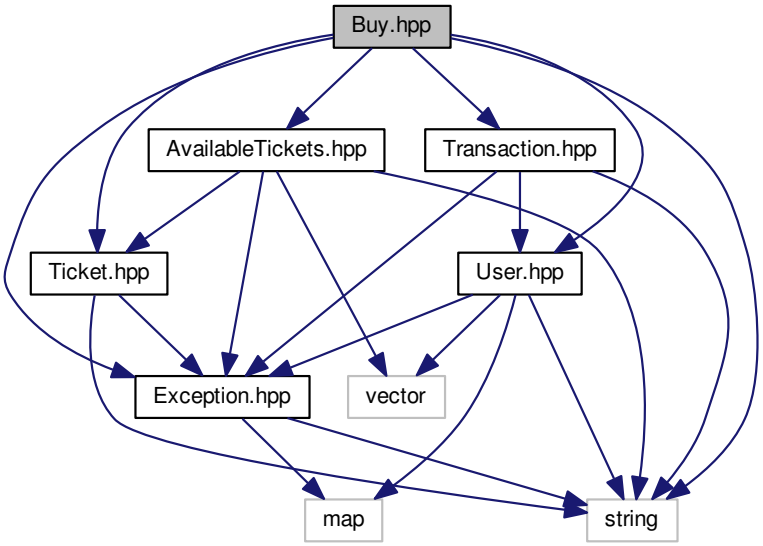
- class AvailableTickets

  AvailableTickets Class.

5.3 Buy.hpp File Reference

```
#include "Transaction.hpp"
#include "User.hpp"
#include "Ticket.hpp"
#include "AvailableTickets.hpp"
#include "Exception.hpp"
#include <string>
```

Include dependency graph for Buy.hpp:



Classes

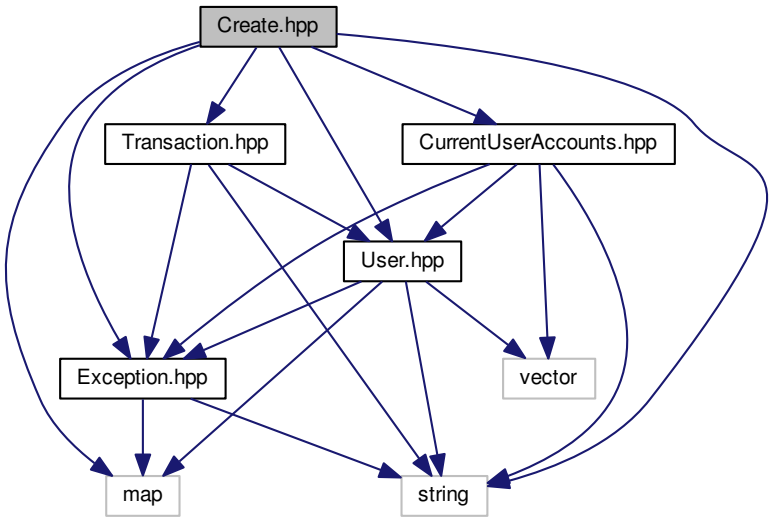
- class Buy

  Buy Class.

5.4 Create.hpp File Reference

```
#include "Transaction.hpp"
#include "User.hpp"
#include "CurrentUserAccounts.hpp"
#include "Exception.hpp"
#include <string>
#include <map>
```

Include dependency graph for Create.hpp:



Classes

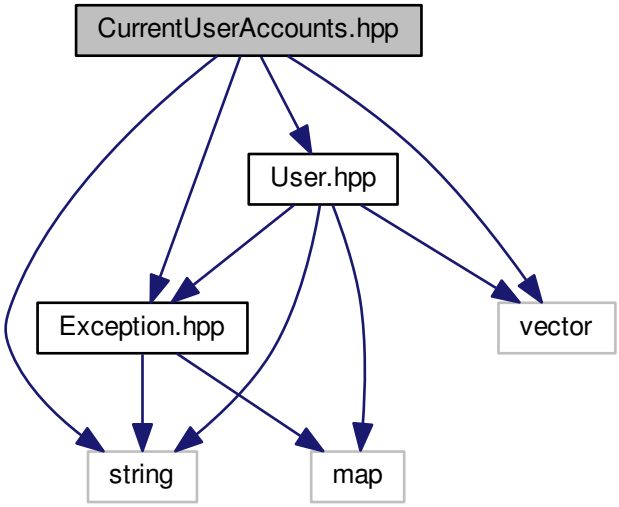
- class Create

  Create Class.

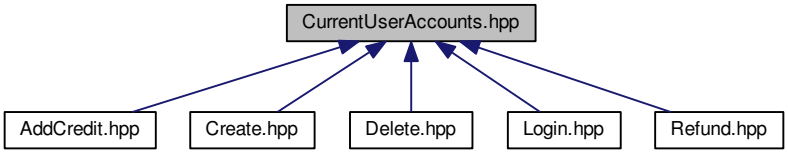
5.5 CurrentUserAccounts.hpp File Reference

```
#include "User.hpp"
#include "Exception.hpp"
#include <string>
#include <vector>
```

Include dependency graph for CurrentUserAccounts.hpp:



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



Classes

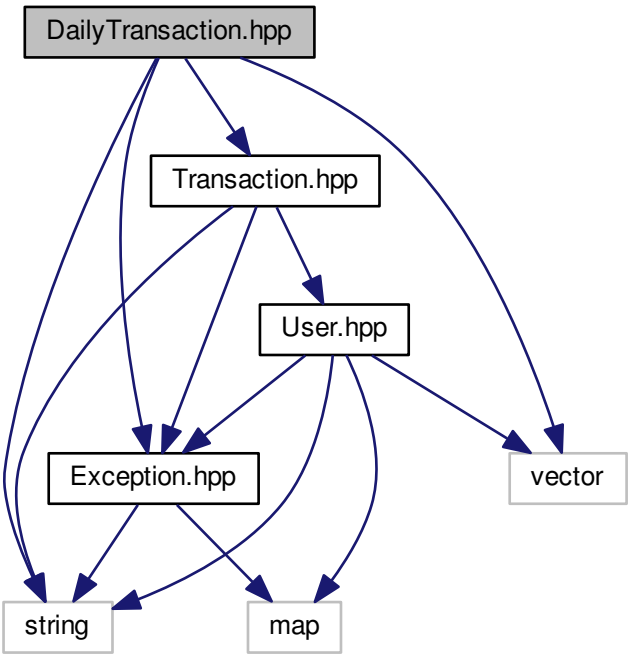
- class CurrentUserAccounts

  CurrentUserAccounts Class.

5.6 DailyTransaction.hpp File Reference

```
#include "Transaction.hpp"
#include "Exception.hpp"
#include <string>
#include <vector>
```

Include dependency graph for DailyTransaction.hpp:



Classes

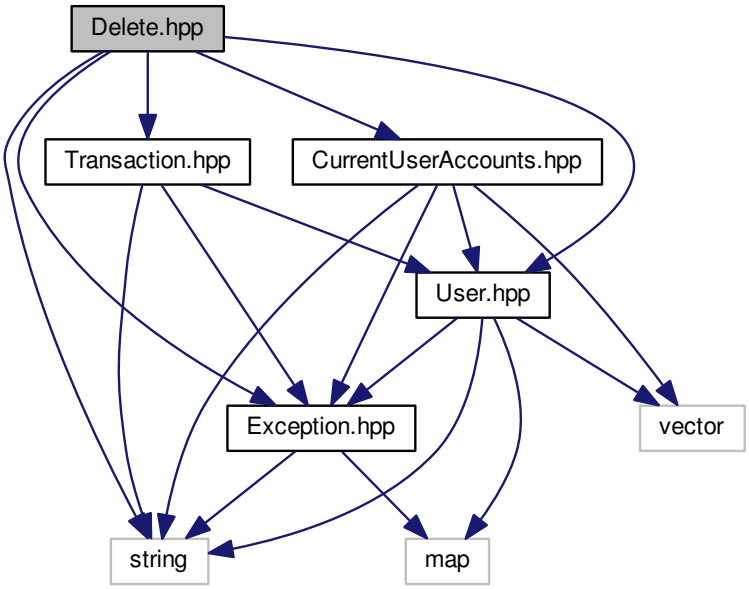
- class DailyTransaction

  DailyTransaction Class.

5.7 Delete.hpp File Reference

```
#include "Transaction.hpp"
#include "User.hpp"
#include "CurrentUserAccounts.hpp"
#include "Exception.hpp"
#include <string>
```

Include dependency graph for Delete.hpp:



Classes

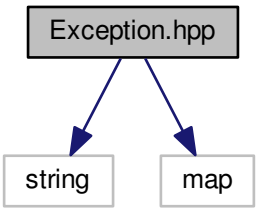
- class Delete

  Delete Class.

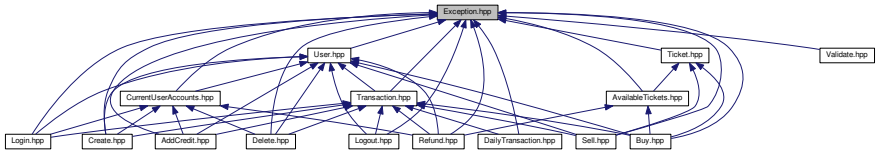
5.8 Exception.hpp File Reference

```
#include <string>
#include <map>
```

Include dependency graph for Exception.hpp:



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



Classes

- class Exception

  Exception Class.

Enumerations

- enum exception_codes {
  ALREADY_LOGIN, DELETE_SELF, INVALID_TRANSACTION, INVALID_PRIV,
  MUST_LOGIN, ATF_NOT_FOUND, CUA_NOT_FOUND, CORRUPT_ATF,
  CORRUPT_CUA, DTF_WRITE_ERROR, INVALID_USER, UNKNOWN_USER,
  INVALID_USER_TYPE, INVALID_USER_LENGTH, INVALID_USER_RESERVED, INVALID_USER_EXISTS,
  SELLER_IS_SELF, TITLE_NOT_FOUND, SELLER_NOT_FOUND, TICKET_NOT_FOUND,
  INVALID_TITLE, INVALID_TITLE_LENGTH, INVALID_TITLE_RESERVED, INVALID_AMOUNT,
  NEGATIVE_AMOUNT, CREDIT_AMOUNT_OVERFLOW, SALE_PRICE_OVERFLOW, USER_CREDIT_OVERFLOW,
  INVALID_TICKET_VOLUME, TICKET_VOLUME_NEGATIVE, TICKET_VOLUME_OVERFLOW, TICKET_VOLUME_USER_MAX,
  ONE_SELL_PER_SESSION, PURCHASE_CANCELED, INVALID_CONFIRMATION, NOT_YET_IMPLEMENTED }

  Exception codes used for various error messages throughout the program.

5.8.1 Enumeration Type Documentation

5.8.1.1 enum exception_codes

Exception codes used for various error messages throughout the program.

Enumerator

ALREADY_LOGIN ALREADY_LOGIN User is already logged in.

DELETE_SELF DELETE_SELF Cannot delete a user account that is logged in.

INVALID_TRANSACTION INVALID_TRANSACTION Generic invalid transaction.

INVALID_PRIV INVALID_PRIV Insufficient privileges to execute transaction.

MUST_LOGIN MUST_LOGIN User must be logged in.

ATF_NOT_FOUND ATF_NOT_FOUND Available tickets file not found.

CUA_NOT_FOUND CUA_NOT_FOUND Current user accounts file not found.

CORRUPT_ATF CORRUPT_ATF Available tickets file corrupted.

CORRUPT_CUA CORRUPT_CUA Current user accounts file corrupted.

DTF_WRITE_ERROR DTF_WRITE_ERROR Error writing the daily transaction file.

INVALID_USER INVALID_USER Invalid user specified.

UNKNOWN_USER UNKNOWN_USER User specified is unknown.

INVALID_USER_TYPE INVALID_USER_TYPE Invalid user type.

INVALID_USER_LENGTH INVALID_USER_LENGTH Invalid user name specified, length is invalid.

INVALID_USER_RESERVED INVALID_USER_RESERVED Invalid user name specified, contains reserved word.

INVALID_USER_EXISTS INVALID_USER_EXISTS Invalid user specified, user already exists.

SELLER_IS_SELF SELLER_IS_SELF Trying to purchase tickets from yourself.

TITLE_NOT_FOUND TITLE_NOT_FOUND Event title not found.

SELLER_NOT_FOUND SELLER_NOT_FOUND The seller was not found selling any tickets.

TICKET_NOT_FOUND TICKET_NOT_FOUND Ticket not found for sale by seller specified.

INVALID_TITLE INVALID_TITLE Invalid title specified.

INVALID_TITLE_LENGTH INVALID_TITLE_LENGTH Invalid title specified, length is invalid.

INVALID_TITLE_RESERVED INVALID_TITLE_RESERVED Invalid title specified, contains reserved word.

INVALID_AMOUNT INVALID_AMOUNT Invalid value specified, amount must be in dollars.

NEGATIVE_AMOUNT NEGATIVE_AMOUNT Invalid value specified, amount must be positive.

CREDIT_AMOUNT_OVERFLOW CREDIT_AMOUNT_OVERFLOW Invalid credit amount specified, amount cannot exceed 1000.00.

SALE_PRICE_OVERFLOW SALE_PRICE_OVERFLOW Invalid sale price specified, amount cannot exceed 999.99.

USER_CREDIT_OVERFLOW USER_CREDIT_OVERFLOW User credit cannot exceed 999999.99.

INVALID_TICKET_VOLUME INVALID_TICKET_VOLUME Invalid ticket volume specified, volume must be an integer value.

TICKET_VOLUME_NEGATIVE TICKET_VOLUME_NEGATIVE Invalid ticket volume specified, volume must be a positive value.

TICKET_VOLUME_OVERFLOW TICKET_VOLUME_OVERFLOW Ticket volume cannot exceed 100.

TICKET_VOLUME_USER_MAX TICKET_VOLUME_USER_MAX Non-admin users cannot purchase more than 4 tickets per session.

ONE_SELL_PER_SESSION ONE_SELL_PER_SESSION Only one sell transaction may occur per session.

PURCHASE_CANCELED PURCHASE_CANCELLED User canceled the ticket purchase.

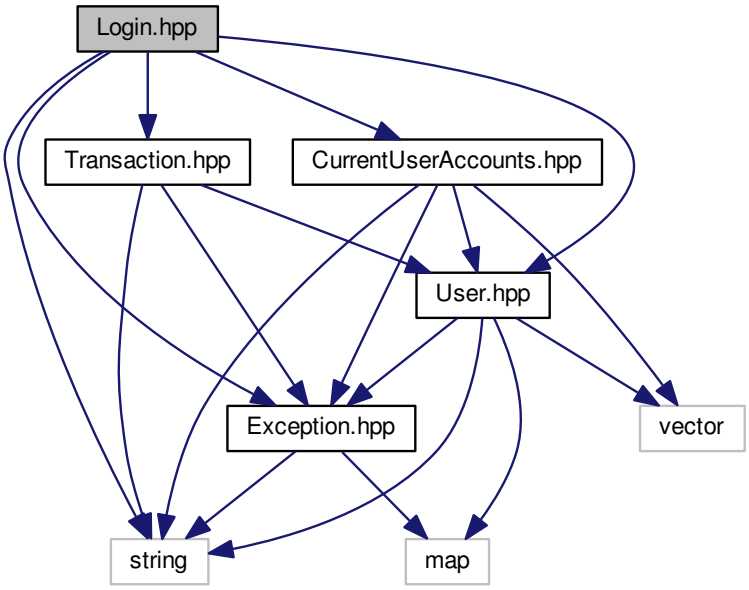
INVALID_CONFIRMATION INVALID_CONFIRMATION Confirmation must be 'yes' or 'no'.

NOT_YET_IMPLEMENTED NOT_YET_IMPLEMENTED Exception used for anything not implemented yet.

5.9 Login.hpp File Reference

```
#include "Transaction.hpp"
#include "User.hpp"
#include "CurrentUserAccounts.hpp"
#include "Exception.hpp"
#include <string>
```

Include dependency graph for Login.hpp:



Classes

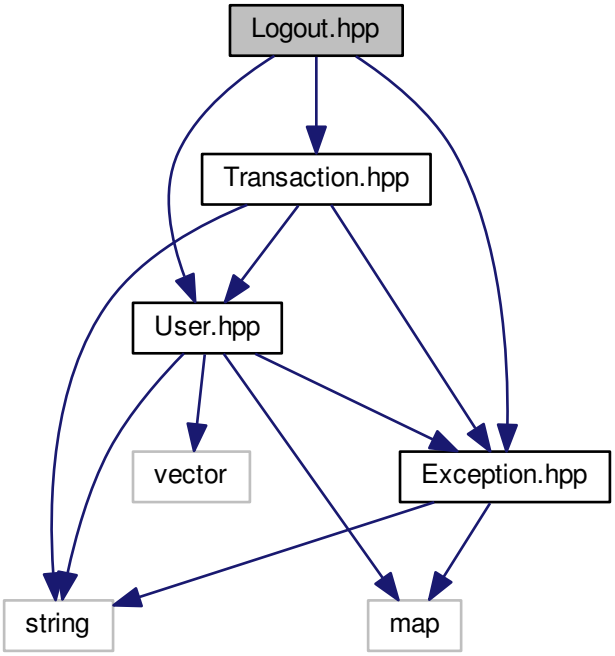
- class Login

  Login Class.

5.10 Logout.hpp File Reference

```
#include "Transaction.hpp"
#include "Exception.hpp"
#include "User.hpp"
```

Include dependency graph for Logout.hpp:



Classes

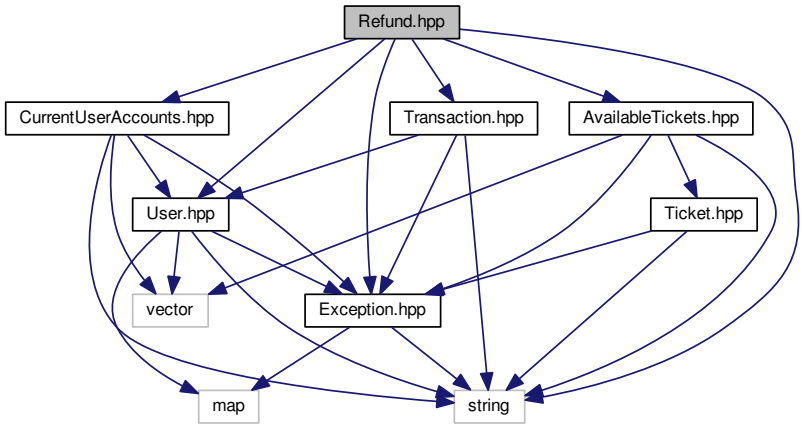
- class Logout

  Logout Class.

5.11 Refund.hpp File Reference

```
#include "Transaction.hpp"
#include "User.hpp"
#include "CurrentUserAccounts.hpp"
#include "AvailableTickets.hpp"
#include "Exception.hpp"
#include <string>
```

Include dependency graph for Refund.hpp:



Classes

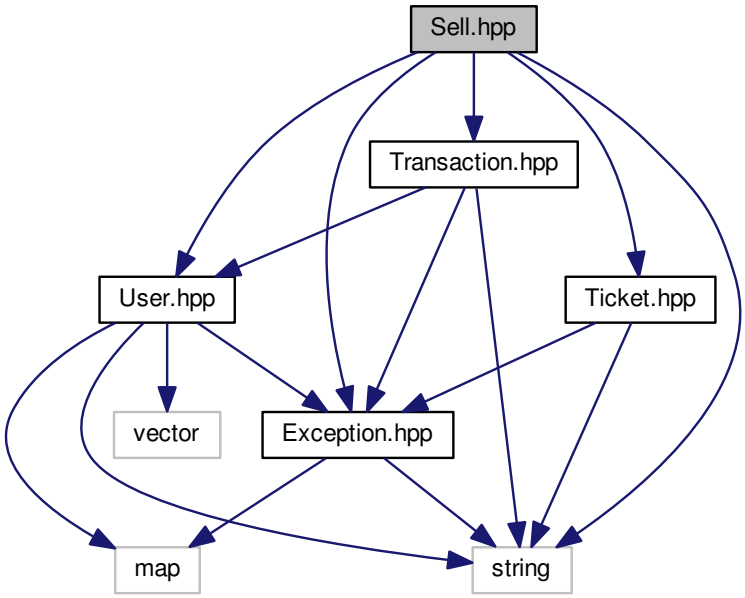
- class Refund

  Refund Class.

5.12 Sell.hpp File Reference

```
#include "Transaction.hpp"
#include "User.hpp"
#include "Ticket.hpp"
#include "Exception.hpp"
#include <string>
```

Include dependency graph for Sell.hpp:



Classes

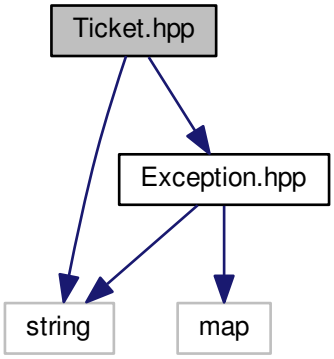
- class Sell

  Sell Class.

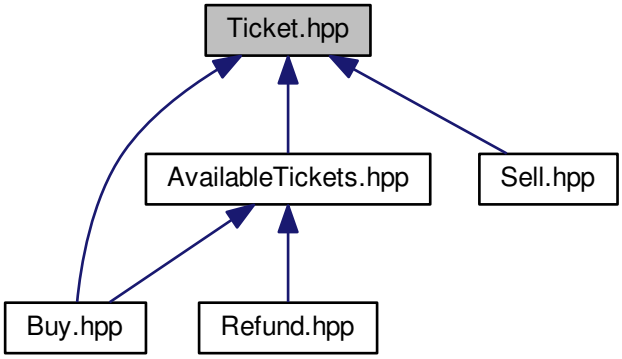
5.13 Ticket.hpp File Reference

```
#include "Exception.hpp"
#include <string>
```

Include dependency graph for Ticket.hpp:



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



Classes

- class Ticket

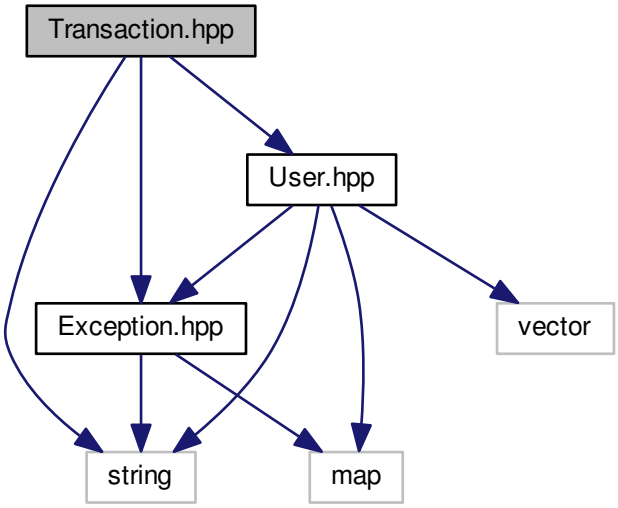
  Ticket Class.

5.14 Transaction.hpp File Reference

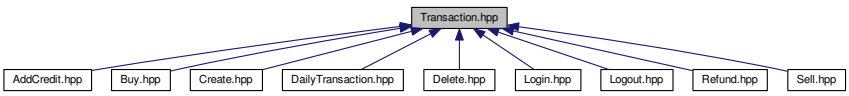
```
#include "User.hpp"
```

```
#include "Exception.hpp"
#include <string>
```

Include dependency graph for Transaction.hpp:



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



Classes

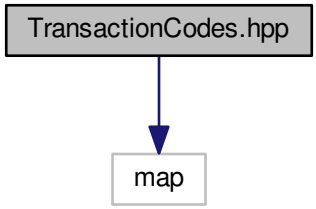
- class Transaction

  Transaction Class.

5.15 TransactionCodes.hpp File Reference

```
#include <map>
```

Include dependency graph for TransactionCodes.hpp:



Enumerations

- enum transaction_codes {
  _undefined, _login, _logout, _create,
  _delete, _sell, _buy, _refund,
  _addcredit }

  Transaction codes used for the transactions supported.

Variables

- static map< string,
  transaction_codes > map_code

  map_code Maps each transaction code to a transaction string.

5.15.1 Enumeration Type Documentation

5.15.1.1 enum transaction_codes

Transaction codes used for the transactions supported.

Enumerator

_undefined _undefined Used for undefined transactions.

_login _login Used for the login transaction.

_logout _logout Used for the logout transaction.

_create _create Used for the create transaction.

_delete _delete Used for the delete transaction.

_sell _sell Used for the sell transaction.

_buy _buy Used for the buy transaction.

_refund _refund Used for the refund transaction.

_addcredit _addcredit used for the add credit transaction.

5.15.2 Variable Documentation

5.15.2.1 map<string, transaction_codes> map_code [static]

Initial value:

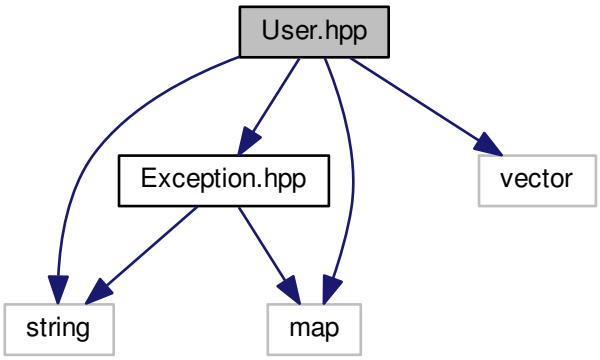
```
= {
            {"login", _login},
            {"logout", _logout},
            {"create", _create},
            {"delete", _delete},
            {"sell", _sell },
            {"buy", _buy},
            {"refund", _refund},
            {"addcredit", _addcredit}
}
```

map_code Maps each transaction code to a transaction string.

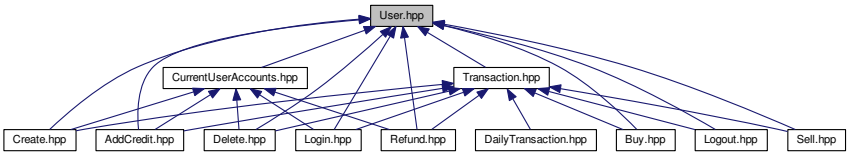
5.16 User.hpp File Reference

```
#include "Exception.hpp"
#include <string>
#include <vector>
#include <map>
```

Include dependency graph for User.hpp:



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



Classes

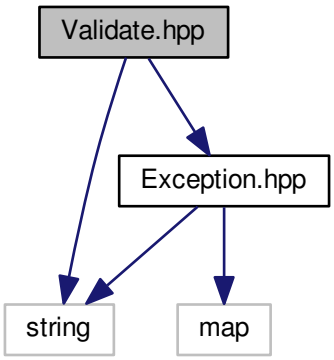
- class User

  User Class.

5.17 Validate.hpp File Reference

```
#include "Exception.hpp"
#include <string>
```

Include dependency graph for Validate.hpp:



Classes

- class Validate

  Validate Class.