Fusion

Kamil Rusin

Aug 17, 2019

CONTENTS:

1 Indices and tables 1

2 Docs 3

Index 7

CHAPTER ONE

INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

TWO

DOCS

class Game

This class represents a game. It creates a common context for all joined clients.

Public Types

enum Team

This enum contains the teams' identifiers.

Values:

kRandom = 0

This indicates that a WebSocketSession should be assigned to a random team.

kFirst = 1

This identifies the first team in the game.

kSecond = 2

This identifies the second team in the game.

using join_result_t = std::optional<std::tuple<system::IncomingPackageDelegate&, json::JSON, std::size_t>>

This is the return type of the Join method.

Public Functions

Game(**const** *Game* *&other*)

Explicitly deleted copy constructor. It's deleted due to presence of unique_ptr in class hierarchy.

Parameters

- `[in] other`: Copied object.

Game &**operator=**(**const** *Game* *&other*)

Explicitly deleted copy operator. It's deleted due to presence of unique_ptr in class hierarchy.

Return Reference to `this` object.

Parameters

- `[in] other`: Copied object.

Game()

This constructor creates the asynchronous reading delegate.

void **SetLogger**(LoggerManager::Logger *logger*)
Sets the logger of this instance. This method sets the logger of this instance to the given one.

Parameters

- `logger`: [in] The given logger.

LoggerManager::Logger **GetLogger**() **const**
Returns this instance's logger. This method returns the logger of this instance.

Return The logger of this instance is returned. If the logger has not been set this method returns std::nullptr.

Game::*join_result_t* **Join**(WebSocketSession **session*, **const** std::string *&nick*, *Team team* = *Team*::kRandom)
This method joins the client to this game and adds its session to the proper team. If the joining was successful it returns a pair of a new incoming package delegate and a JSON object containing information about the current state of the game, otherwise the returned object is in its invalid state.

Return If the joining was successful pair of a new incoming package delegate and a JSON object containing information about the current state of the game is returned, otherwise the returned object is in its invalid state.

Note If a client has already joined to this game, the method does nothing and returns an invalid state object.

Parameters

- `[in] session`: This is the WebSocket session connected to a client.
- `[in] nick`: This is the nick of the new player.
- `[in] team`: This identifies the team, to which the client will be assigned. The default value indicates that the client will be assigned to a random team.

bool **Leave**(WebSocketSession **session*)
This method removes the given session from this game. It returns a indication whether or not the session has been removed.

Return A indication whether or not the session has been removed is returned.

Note If the session has not been assigned to this game, the method does nothing.

Parameters

- `[in] session`: The session to be removed from this game.

void **BroadcastPackage**(**const** std::shared_ptr<system::Package> *&package*)
This method broadcasts the given package to all clients connected to this game.

Parameters

- `[in] package`: The package to be broadcasted.

std::size_t **GetPlayersCount**() **const**
This method returns the amount of players in this game.

Return The amount of players in this game is returned.

Public Static Attributes

`constexpr` size_t **`kMaxPlayersPerTeam`** = 5
This constant contains the number of players that can be assigned to a team.

INDEX

F

`fusion_server::Game` (*C++ class*), 3
`fusion_server::Game::BroadcastPackage` (*C++ function*), 4
`fusion_server::Game::Game` (*C++ function*), 3
`fusion_server::Game::GetLogger` (*C++ function*), 4
`fusion_server::Game::GetPlayersCount` (*C++ function*), 4
`fusion_server::Game::Join` (*C++ function*), 4
`fusion_server::Game::join_result_t` (*C++ type*), 3
`fusion_server::Game::kFirst` (*C++ enumerator*), 3
`fusion_server::Game::kMaxPlayersPerTeam` (*C++ member*), 5
`fusion_server::Game::kRandom` (*C++ enumerator*), 3
`fusion_server::Game::kSecond` (*C++ enumerator*), 3
`fusion_server::Game::Leave` (*C++ function*), 4
`fusion_server::Game::operator=` (*C++ function*), 3
`fusion_server::Game::SetLogger` (*C++ function*), 3
`fusion_server::Game::Team` (*C++ enum*), 3