Samchon Framework for CPP

1.0.0

Generated by Doxygen 1.8.10

Thu Oct 29 2015 09:38:46

Contents

1 Samchon Framework **1**

2 Test List **3**

3 Todo List **5**

4 Deprecated List **7**

5 Bug List **9**

6 Namespace Index **11**

6.1 Namespace List 11

7 Hierarchical Index **13**

7.1 Class Hierarchy 13

8 Class Index **17**

8.1 Class List 17

9 Namespace Documentation **23**

9.1 samchon Namespace Reference 23

9.1.1 Detailed Description 24

9.1.2 Typedef Documentation 24

9.1.2.1 Dictionary 24

9.2 samchon::example::chat_service Namespace Reference 25

9.2.1 Detailed Description 25

9.3 samchon::example::console_chat_server Namespace Reference 26

9.3.1 Detailed Description 26

9.4 samchon::example::tsp Namespace Reference 27

9.4.1 Detailed Description 27

9.5 samchon::library Namespace Reference 30

9.5.1 Detailed Description 32

9.5.2 Typedef Documentation 37

9.5.2.1 CriticalDictionary 37

9.5.2.2 CriticalList . . . 37
9.5.2.3 CriticalMap . . . 38
9.5.2.4 CriticalSet . . . 38
9.5.2.5 CriticalVector . . . 38
9.5.2.6 XMLList . . . 39
9.6 samchon::namtree Namespace Reference . . . 39
9.6.1 Detailed Description . . . 39
9.7 samchon::protocol Namespace Reference . . . 43
9.7.1 Detailed Description . . . 45
9.7.2 Typedef Documentation . . . 54
9.7.2.1 SharedEntityArray . . . 54
9.7.2.2 SharedEntityList . . . 55
9.7.2.3 UniqueEntityArray . . . 55
9.7.2.4 UniqueEntityList . . . 56
9.8 samchon::protocol::master Namespace Reference . . . 56
9.8.1 Detailed Description . . . 58
9.9 samchon::protocol::service Namespace Reference . . . 60
9.9.1 Detailed Description . . . 61
9.10 samchon::protocol::slave Namespace Reference . . . 70
9.10.1 Detailed Description . . . 70

10 Class Documentation **73**
10.1 samchon::library::Base64 Class Reference . . . 73
10.1.1 Detailed Description . . . 73
10.1.2 Member Function Documentation . . . 74
10.1.2.1 encode(const ByteArray &) -> std::string . . . 74
10.1.2.2 decode(const std::string &) -> ByteArray . . . 75
10.2 samchon::ByteArray Class Reference . . . 75
10.2.1 Detailed Description . . . 76
10.2.2 Constructor & Destructor Documentation . . . 77
10.2.2.1 ByteArray() . . . 77
10.2.2.2 ByteArray(const ByteArray &) . . . 77
10.2.2.3 ByteArray(ByteArray &&) . . . 77
10.2.3 Member Function Documentation . . . 77
10.2.3.1 getPosition() const -> size_t . . . 77
10.2.3.2 setPosition(size_t) . . . 77
10.2.3.3 reverse(const _Ty &val) -> _Ty . . . 77
10.2.3.4 read() const -> _Ty . . . 78
10.2.3.5 readReversely() const -> _Ty . . . 78
10.2.3.6 write(const _Ty &val) . . . 78

10.2.3.7 writeReversely(const _Ty &val) . . . 79
10.2.3.8 compress() const -> ByteArray . . . 79
10.2.3.9 decompress() const -> ByteArray . . . 79
10.2.4 Member Data Documentation . . . 80
10.2.4.1 position . . . 80
10.3 samchon::library::CaseGenerator Class Reference . . . 80
10.3.1 Detailed Description . . . 81
10.3.2 Constructor & Destructor Documentation . . . 89
10.3.2.1 CaseGenerator(size_t n, size_t r) . . . 89
10.3.3 Member Function Documentation . . . 89
10.3.3.1 size() const -> size_t . . . 89
10.3.3.2 operator[](size_t) const -> std::vector< size_t >=0 . . . 90
10.3.3.3 at(size_t) const -> std::vector< size_t > . . . 90
10.3.3.4 n() const -> size_t . . . 90
10.3.3.5 r() const -> size_t . . . 90
10.3.3.6 toMatrix() const -> std::vector< std::vector< size_t >> . . . 91
10.3.4 Member Data Documentation . . . 91
10.3.4.1 n_ . . . 91
10.3.4.2 r_ . . . 91
10.3.4.3 size_ . . . 91
10.4 samchon::library::Charset Class Reference . . . 92
10.4.1 Detailed Description . . . 92
10.4.2 Member Function Documentation . . . 93
10.4.2.1 toMultibyte(const std::string &) -> std::string . . . 93
10.4.2.2 toMultibyte(const std::wstring &) -> std::string . . . 94
10.4.2.3 toUTF8(const std::string &) -> std::string . . . 94
10.4.2.4 toUTF8(const std::wstring &) -> std::string . . . 94
10.4.2.5 toUnicode(const std::string &, int) -> std::wstring . . . 95
10.5 samchon::example::chat_service::ChatClient Class Reference . . . 95
10.5.1 Detailed Description . . . 96
10.5.2 Constructor & Destructor Documentation . . . 96
10.5.2.1 ChatClient(protocol::service::User *) . . . 96
10.5.3 Member Function Documentation . . . 96
10.5.3.1 createService(const std::string &) -> protocol::service::Service *override . . . 96
10.6 samchon::example::console_chat_server::ChatClient Class Reference . . . 97
10.6.1 Detailed Description . . . 98
10.6.2 Constructor & Destructor Documentation . . . 98
10.6.2.1 ChatClient(ChatServer *, protocol::Socket *) . . . 98
10.6.2.2 ~ChatClient() . . . 98
10.6.3 Member Function Documentation . . . 99

10.6.3.1 replyData(std::shared_ptr< protocol::Invoke >) override . . . 99
10.6.4 Member Data Documentation . . . 99
10.6.4.1 server . . . 99
10.7 samchon::example::chat_service::ChatMessage Class Reference . . . 99
10.7.1 Detailed Description . . . 100
10.7.2 Constructor & Destructor Documentation . . . 101
10.7.2.1 ChatMessage() . . . 101
10.7.3 Member Function Documentation . . . 101
10.7.3.1 TAG() const -> std::string . . . 101
10.7.3.2 construct(std::shared_ptr< library::XML >) . . . 102
10.7.3.3 getListener() const -> std::string . . . 102
10.7.3.4 toXML() const -> std::shared_ptr< library::XML > . . . 102
10.7.4 Member Data Documentation . . . 103
10.7.4.1 orator . . . 103
10.7.4.2 listener . . . 103
10.7.4.3 message . . . 103
10.8 samchon::example::chat_service::ChatRoom Class Reference . . . 103
10.8.1 Detailed Description . . . 104
10.8.2 Constructor & Destructor Documentation . . . 105
10.8.2.1 ChatRoom(ChatRoomArray *, const std::string &, ChatUser *) . . . 105
10.8.3 Member Function Documentation . . . 105
10.8.3.1 TAG() const -> std::string . . . 105
10.8.3.2 registerClient(ChatService *) . . . 106
10.8.3.3 eraseClient(ChatService *) . . . 106
10.8.3.4 toXML() const -> std::shared_ptr< library::XML > override . . . 107
10.8.4 Member Data Documentation . . . 108
10.8.4.1 roomArray . . . 108
10.8.4.2 name . . . 108
10.8.4.3 host . . . 108
10.8.4.4 participants . . . 108
10.9 samchon::example::chat_service::ChatRoomArray Class Reference . . . 109
10.9.1 Detailed Description . . . 110
10.9.2 Constructor & Destructor Documentation . . . 110
10.9.2.1 ChatRoomArray(ChatServer *) . . . 110
10.9.3 Member Function Documentation . . . 110
10.9.3.1 CHILD_TAG() const -> std::string . . . 111
10.9.3.2 notify() . . . 111
10.9.3.3 toXML() const -> std::shared_ptr< library::XML > . . . 112
10.9.3.4 toInvoke() const -> std::shared_ptr< protocol::Invoke > . . . 112
10.9.4 Member Data Documentation . . . 112

10.9.4.1 server . . . 113
10.10 samchon::example::console_chat_server::ChatServer Class Reference . . . 113
10.10.1 Detailed Description . . . 114
10.10.2 Constructor & Destructor Documentation . . . 114
10.10.2.1 ChatServer() . . . 114
10.10.3 Member Function Documentation . . . 114
10.10.3.1 eraseClient(ChatClient *client) . . . 114
10.10.3.2 PORT() const -> int override . . . 114
10.10.3.3 addClient(protocol::Socket *socket) override . . . 115
10.10.3.4 replyData(std::shared_ptr< protocol::Invoke >) override . . . 115
10.10.3.5 sendData(std::shared_ptr< protocol::Invoke >) override . . . 116
10.10.4 Member Data Documentation . . . 116
10.10.4.1 clientSet . . . 116
10.10.4.2 rwMutex . . . 116
10.11 samchon::example::chat_service::ChatServer Class Reference . . . 117
10.11.1 Detailed Description . . . 117
10.11.2 Constructor & Destructor Documentation . . . 118
10.11.2.1 ChatServer() . . . 118
10.11.3 Member Function Documentation . . . 118
10.11.3.1 PORT() const -> int override . . . 118
10.11.3.2 createUser() -> protocol::service::User *override . . . 118
10.11.3.3 getChatRoomArray() const -> ChatRoomArray * . . . 119
10.11.4 Member Data Documentation . . . 119
10.11.4.1 roomArray . . . 119
10.12 samchon::example::chat_service::ChatService Class Reference . . . 119
10.12.1 Detailed Description . . . 119
10.12.2 Constructor & Destructor Documentation . . . 120
10.12.2.1 ChatService(protocol::service::Client *) . . . 120
10.12.3 Member Data Documentation . . . 120
10.12.3.1 room . . . 120
10.13 samchon::example::chat_service::ChatUser Class Reference . . . 121
10.13.1 Detailed Description . . . 121
10.13.2 Constructor & Destructor Documentation . . . 122
10.13.2.1 ChatUser(protocol::service::Server *) . . . 122
10.13.3 Member Function Documentation . . . 122
10.13.3.1 createClient() -> protocol::service::Client *override . . . 122
10.14 samchon::protocol::service::Client Class Reference . . . 123
10.14.1 Detailed Description . . . 124
10.14.2 Constructor & Destructor Documentation . . . 124
10.14.2.1 Client(User *) . . . 124

10.14.3 Member Function Documentation . . . 125
10.14.3.1 createService(const std::string &) -> Service *=0 . . . 125
10.14.3.2 getUser() const -> User * . . . 125
10.14.3.3 getService() const -> Service * . . . 125
10.14.3.4 getNo() const -> size_t . . . 125
10.14.3.5 sendData(std::shared_ptr< Invoke >) . . . 126
10.14.3.6 replyData(std::shared_ptr< Invoke >) . . . 127
10.14.3.7 constructService(const std::string &) . . . 128
10.14.4 Member Data Documentation . . . 129
10.14.4.1 user . . . 129
10.14.4.2 no . . . 129
10.14.4.3 service . . . 129
10.15 samchon::library::CombinedPermutationGenerator Class Reference . . . 130
10.15.1 Detailed Description . . . 130
10.15.2 Constructor & Destructor Documentation . . . 131
10.15.2.1 CombinedPermutationGenerator(size_t n, size_t r) . . . 131
10.15.3 Member Function Documentation . . . 132
10.15.3.1 operator[](size_t) const -> std::vector< size_t > override . . . 132
10.16 samchon::library::CriticalAllocator< _Ty > Class Template Reference . . . 132
10.16.1 Detailed Description . . . 133
10.16.2 Member Function Documentation . . . 134
10.16.2.1 getMutex() -> RWMutex & . . . 134
10.16.2.2 construct(_U *ptr, _Args &&...args) . . . 135
10.16.2.3 destroy(_U *ptr) . . . 135
10.16.2.4 allocate(size_type n, std::allocator< void >::const_pointer hint=NULL) -> pointer 136
10.16.2.5 deallocate(pointer ptr, size_type size) . . . 136
10.16.3 Member Data Documentation . . . 137
10.16.3.1 mtx . . . 137
10.17 samchon::library::Date Class Reference . . . 137
10.17.1 Detailed Description . . . 138
10.17.2 Constructor & Destructor Documentation . . . 139
10.17.2.1 Date() . . . 139
10.17.2.2 Date(const super &) . . . 139
10.17.2.3 Date(super &&) . . . 140
10.17.2.4 Date(int year, int month, int date) . . . 140
10.17.2.5 Date(const WeakString &) . . . 140
10.17.2.6 Date(long long linuxTime) . . . 141
10.17.3 Member Function Documentation . . . 141
10.17.3.1 calcSeconds(int year, int month, int date) -> long long . . . 141
10.17.3.2 calcLastDates(int year) -> std::array< int, 12 > . . . 142

10.17.3.3 set(const std::string &) . . . 142
10.17.3.4 set(const WeakString &) . . . 143
10.17.3.5 set(int year, int month, int date) . . . 144
10.17.3.6 set(long long linuxTime) . . . 145
10.17.3.7 setYear(int) . . . 146
10.17.3.8 setMonth(int) . . . 147
10.17.3.9 setDate(int) . . . 148
10.17.3.10 addYear(int) . . . 149
10.17.3.11 addMonth(int) . . . 149
10.17.3.12 addWeek(int) . . . 150
10.17.3.13 addDate(int) . . . 151
10.17.3.14 getYear() const -> int . . . 151
10.17.3.15 getMonth() const -> int . . . 153
10.17.3.16 getDate() const -> int . . . 154
10.17.3.17 getDay() const -> int . . . 155
10.17.3.18 toTM() const -> struct::tm . . . 156
10.17.3.19 toLinuxTime() const -> long long . . . 157
10.17.3.20 toString() const -> std::string . . . 158
10.17.4 Member Data Documentation . . . 159
10.17.4.1 TP_1970 . . . 159
10.18 samchon::library::Datetime Class Reference . . . 159
10.18.1 Detailed Description . . . 161
10.18.2 Constructor & Destructor Documentation . . . 162
10.18.2.1 Datetime() . . . 162
10.18.2.2 Datetime(const std::chrono::system_clock::time_point &) . . . 162
10.18.2.3 Datetime(std::chrono::system_clock::time_point &&) . . . 162
10.18.2.4 Datetime(int year, int month, int date, int hour=0, int min=0, int sec=0) . . . 162
10.18.2.5 Datetime(const WeakString &) . . . 163
10.18.2.6 Datetime(long long linuxTime) . . . 163
10.18.3 Member Function Documentation . . . 164
10.18.3.1 set(const WeakString &) override . . . 164
10.18.3.2 set(int year, int month, int date, int hours=0, int minutes=0, int seconds=0) . . . 166
10.18.3.3 set(long long linuxTime) . . . 167
10.18.3.4 setYear(int) . . . 167
10.18.3.5 setMonth(int) . . . 168
10.18.3.6 setDate(int) . . . 169
10.18.3.7 setHour(int) . . . 170
10.18.3.8 setMinute(int) . . . 171
10.18.3.9 setSecond(int) . . . 171
10.18.3.10 addYear(int) override . . . 172

10.18.3.11 addMonth(int) override . . . 173
10.18.3.12 addWeek(int) override . . . 173
10.18.3.13 addDate(int) override . . . 174
10.18.3.14 addHour(int) . . . 175
10.18.3.15 addMinute(int) . . . 175
10.18.3.16 addSecond(int) . . . 176
10.18.3.17 getHour() const -> int . . . 176
10.18.3.18 getMinute() const -> int . . . 178
10.18.3.19 getSecond() const -> int . . . 179
10.18.3.20 toString() const -> std::string override . . . 181
10.19 samchon::protocol::master::DistributedClient Class Reference . . . 181
10.19.1 Detailed Description . . . 182
10.19.2 Constructor & Destructor Documentation . . . 184
10.19.2.1 DistributedClient() . . . 184
10.20 samchon::protocol::master::DistributedClientArray Class Reference . . . 184
10.20.1 Detailed Description . . . 184
10.20.2 Constructor & Destructor Documentation . . . 186
10.20.2.1 DistributedClientArray() . . . 186
10.20.3 Member Function Documentation . . . 186
10.20.3.1 construct(std::shared_ptr< library::XML >) override . . . 186
10.20.3.2 start() override . . . 187
10.20.3.3 toXML() const -> std::shared_ptr< library::XML > override . . . 187
10.21 samchon::protocol::master::DistributedClientArrayMediator Class Reference . . . 188
10.21.1 Detailed Description . . . 188
10.21.2 Constructor & Destructor Documentation . . . 191
10.21.2.1 DistributedClientArrayMediator() . . . 191
10.21.3 Member Function Documentation . . . 191
10.21.3.1 construct(std::shared_ptr< library::XML >) override . . . 191
10.21.3.2 start() override . . . 192
10.21.3.3 toXML() const -> std::shared_ptr< library::XML > override . . . 192
10.22 samchon::protocol::master::DistributedClientServerMediator Class Reference . . . 193
10.22.1 Detailed Description . . . 193
10.22.2 Constructor & Destructor Documentation . . . 196
10.22.2.1 DistributedClientServerMediator() . . . 196
10.23 samchon::protocol::master::DistributedServer Class Reference . . . 196
10.23.1 Detailed Description . . . 196
10.23.2 Constructor & Destructor Documentation . . . 198
10.23.2.1 DistributedServer() . . . 198
10.23.3 Member Function Documentation . . . 199
10.23.3.1 construct(std::shared_ptr< library::XML >) override . . . 199

10.23.3.2 toXML() const -> std::shared_ptr< library::XML > override . . . 199
10.24 samchon::protocol::master::DistributedServerArray Class Reference . . . 200
10.24.1 Detailed Description . . . 200
10.24.2 Constructor & Destructor Documentation . . . 202
10.24.2.1 DistributedServerArray() . . . 202
10.24.3 Member Function Documentation . . . 202
10.24.3.1 start() override . . . 202
10.25 samchon::protocol::master::DistributedServerArrayMediator Class Reference . . . 203
10.25.1 Detailed Description . . . 203
10.25.2 Constructor & Destructor Documentation . . . 206
10.25.2.1 DistributedServerArrayMediator() . . . 206
10.25.3 Member Function Documentation . . . 206
10.25.3.1 start() override . . . 206
10.26 samchon::protocol::master::DistributedSlaveServerMediator Class Reference . . . 207
10.26.1 Detailed Description . . . 207
10.26.2 Constructor & Destructor Documentation . . . 210
10.26.2.1 DistributedSlaveServerMediator() . . . 210
10.27 samchon::protocol::master::DistributedSlaveSystemMediator Class Reference . . . 210
10.27.1 Detailed Description . . . 211
10.27.2 Constructor & Destructor Documentation . . . 213
10.27.2.1 DistributedSlaveSystemMediator() . . . 213
10.27.3 Member Function Documentation . . . 214
10.27.3.1 replyData(std::shared_ptr< Invoke >) override . . . 214
10.27.3.2 TAG() const -> std::string override . . . 214
10.27.4 Member Data Documentation . . . 214
10.27.4.1 master . . . 214
10.28 samchon::protocol::master::DistributedSystem Class Reference . . . 214
10.28.1 Detailed Description . . . 215
10.28.2 Constructor & Destructor Documentation . . . 217
10.28.2.1 DistributedSystem() . . . 217
10.28.3 Member Function Documentation . . . 218
10.28.3.1 construct(std::shared_ptr< library::XML >) override . . . 218
10.28.3.2 sendData(std::shared_ptr< Invoke >) override . . . 218
10.28.3.3 replyData(std::shared_ptr< Invoke >) override . . . 218
10.28.3.4 toXML() const -> std::shared_ptr< library::XML > override . . . 218
10.28.4 Member Data Documentation . . . 219
10.28.4.1 inProgress . . . 219
10.28.4.2 processed . . . 219
10.28.4.3 avgElapsedTime . . . 219
10.28.4.4 performance . . . 220

10.28.4.5 roleHistoryList . . . 220
10.28.4.6 invokeHistoryList . . . 220
10.29 samchon::protocol::master::DistributedSystemArray Class Reference . . . 220
10.29.1 Detailed Description . . . 221
10.29.2 Constructor & Destructor Documentation . . . 223
10.29.2.1 DistributedSystemArray() . . . 223
10.29.3 Member Function Documentation . . . 223
10.29.3.1 construct(std::shared_ptr< library::XML >) override . . . 223
10.29.3.2 createRole(std::shared_ptr< library::XML >) -> DistributedSystemRole *=0 . . . 225
10.29.3.3 start() override . . . 225
10.29.3.4 allocateRoles() . . . 225
10.29.3.5 getRole(const std::string &) const -> std::shared_ptr< DistributedSystemRole > 225
10.29.3.6 toXML() const -> std::shared_ptr< library::XML > override . . . 226
10.29.4 Member Data Documentation . . . 226
10.29.4.1 roleDictionary . . . 226
10.29.4.2 gaParameters . . . 226
10.30 samchon::protocol::master::DistributedSystemArrayMediator Class Reference . . . 227
10.30.1 Detailed Description . . . 227
10.30.2 Constructor & Destructor Documentation . . . 230
10.30.2.1 DistributedSystemArrayMediator() . . . 230
10.30.3 Member Function Documentation . . . 230
10.30.3.1 construct(std::shared_ptr< library::XML >) override . . . 230
10.30.3.2 createSlave() const -> DistributedSlaveSystemMediator *=0 . . . 232
10.30.3.3 start() override . . . 232
10.30.3.4 toXML() const -> std::shared_ptr< library::XML > override . . . 232
10.30.4 Member Data Documentation . . . 233
10.30.4.1 slave . . . 233
10.31 samchon::protocol::master::DistributedSystemRole Class Reference . . . 233
10.31.1 Detailed Description . . . 234
10.31.2 Constructor & Destructor Documentation . . . 235
10.31.2.1 DistributedSystemRole() . . . 235
10.31.3 Member Function Documentation . . . 235
10.31.3.1 getSystem() const -> ExternalSystem *=delete . . . 235
10.31.3.2 getPerformance() const -> double . . . 236
10.31.3.3 getAllocationHistoryList() const -> DSRoleHistoryList * . . . 236
10.31.3.4 getInvokeHistoryList() const -> DSInvokeHistoryList * . . . 236
10.31.3.5 sendData(std::shared_ptr< Invoke >) override . . . 236
10.31.3.6 toXML() const -> std::shared_ptr< library::XML > override . . . 236
10.31.4 Member Data Documentation . . . 236
10.31.4.1 allocatedSystems . . . 236

10.31.4.2 performance . . . 237
10.31.4.3 allocationHistoryList . . . 237
10.31.4.4 invokeHistoryList . . . 237
10.32 samchon::protocol::master::DSInvokeHistory Class Reference . . . 237
10.32.1 Detailed Description . . . 238
10.32.2 Constructor & Destructor Documentation . . . 240
10.32.2.1 DSInvokeHistory(master::DistributedSystem *, master::DistributedSystemRole *) 240
10.32.2.2 DSInvokeHistory(DistributedSystem *, DistributedSystemRole *) . . . 240
10.32.3 Member Function Documentation . . . 240
10.32.3.1 toXML() const -> std::shared_ptr< library::XML > override . . . 240
10.32.3.2 getSystem() const -> DistributedSystem * . . . 240
10.32.3.3 getRole() const -> DistributedSystemRole * . . . 241
10.32.4 Member Data Documentation . . . 241
10.32.4.1 system . . . 241
10.32.4.2 role . . . 241
10.32.4.3 system . . . 241
10.32.4.4 role . . . 241
10.33 samchon::protocol::master::DSInvokeHistoryList Class Reference . . . 241
10.33.1 Detailed Description . . . 242
10.33.2 Constructor & Destructor Documentation . . . 242
10.33.2.1 DSInvokeHistoryList() . . . 242
10.33.3 Member Function Documentation . . . 242
10.33.3.1 createChild(std::shared_ptr< library::XML >) -> DSInvokeHistory *override . . 242
10.34 samchon::protocol::Entity Class Reference . . . 243
10.34.1 Detailed Description . . . 243
10.34.2 Constructor & Destructor Documentation . . . 247
10.34.2.1 Entity() . . . 247
10.34.3 Member Function Documentation . . . 248
10.34.3.1 TAG() const -> std::string=0 . . . 248
10.34.3.2 construct(std::shared_ptr< library::XML >)=0 . . . 248
10.34.3.3 key() const -> std::string . . . 249
10.34.3.4 toXML() const -> std::shared_ptr< library::XML > . . . 249
10.35 samchon::protocol::EntityArray< _Ty > Class Template Reference . . . 251
10.35.1 Detailed Description . . . 252
10.35.2 Constructor & Destructor Documentation . . . 257
10.35.2.1 EntityArray() . . . 257
10.35.3 Member Function Documentation . . . 257
10.35.3.1 construct(std::shared_ptr< library::XML > xml) override . . . 257
10.35.3.2 toXML() const -> std::shared_ptr< library::XML > override . . . 258
10.36 samchon::protocol::EntityGroup< _Container, _ETy, _Ty > Class Template Reference . . . 259

10.36.1 Detailed Description . . . 259
10.36.2 Constructor & Destructor Documentation . . . 266
10.36.2.1 EntityGroup() . . . 266
10.36.3 Member Function Documentation . . . 267
10.36.3.1 construct(std::shared_ptr< library::XML > xml) . . . 267
10.36.3.2 createChild(std::shared_ptr< library::XML >) -> entity_type *=0 . . . 268
10.36.3.3 has(const std::string &key) const -> bool . . . 269
10.36.3.4 get(const std::string &key) -> value_type & . . . 270
10.36.3.5 get(const std::string &key) const -> const value_type & . . . 271
10.36.3.6 toXML() const -> std::shared_ptr< library::XML > . . . 271
10.37 samchon::protocol::EntityList< _Ty > Class Template Reference . . . 274
10.37.1 Detailed Description . . . 274
10.37.2 Constructor & Destructor Documentation . . . 279
10.37.2.1 EntityList() . . . 279
10.37.3 Member Function Documentation . . . 279
10.37.3.1 construct(std::shared_ptr< library::XML > xml) override . . . 279
10.37.3.2 toXML() const -> std::shared_ptr< library::XML > override . . . 280
10.38 samchon::library::ErrorEvent Class Reference . . . 281
10.38.1 Detailed Description . . . 281
10.38.2 Constructor & Destructor Documentation . . . 283
10.38.2.1 ErrorEvent(EventDispatcher *, const std::string &) . . . 283
10.38.3 Member Function Documentation . . . 283
10.38.3.1 getMessage() const -> std::string . . . 283
10.38.4 Member Data Documentation . . . 284
10.38.4.1 message . . . 284
10.39 samchon::library::Event Class Reference . . . 284
10.39.1 Detailed Description . . . 285
10.39.2 Constructor & Destructor Documentation . . . 286
10.39.2.1 Event(EventDispatcher *, int) . . . 286
10.39.3 Member Function Documentation . . . 287
10.39.3.1 getSource() const -> EventDispatcher * . . . 287
10.39.3.2 getType() const -> int . . . 287
10.39.4 Member Data Documentation . . . 287
10.39.4.1 source . . . 287
10.39.4.2 type . . . 287
10.40 samchon::library::EventDispatcher Class Reference . . . 287
10.40.1 Detailed Description . . . 288
10.40.2 Constructor & Destructor Documentation . . . 290
10.40.2.1 EventDispatcher() . . . 290
10.40.2.2 EventDispatcher(const EventDispatcher &eventDispatcher) . . . 291

10.40.2.3 EventDispatcher(EventDispatcher &&eventDispatcher) . . . 291
10.40.3 Member Function Documentation . . . 291
10.40.3.1 addEventListener(int, void(∗listener)(std::shared_ptr< Event >)) . . . 291
10.40.3.2 removeEventListener(int, void(∗listener)(std::shared_ptr< Event >)) . . . 291
10.40.3.3 dispatchEvent(std::shared_ptr< Event >) -> bool . . . 291
10.40.3.4 dispatchProgressEvent(size_t x, size_t size) -> bool . . . 292
10.40.4 Member Data Documentation . . . 292
10.40.4.1 eventSetMap . . . 292
10.40.4.2 mtx . . . 292
10.40.4.3 semaphore . . . 292
10.41 samchon::protocol::ExternalClient Class Reference . . . 293
10.41.1 Detailed Description . . . 293
10.41.2 Constructor & Destructor Documentation . . . 294
10.41.2.1 ExternalClient() . . . 294
10.41.3 Member Function Documentation . . . 295
10.41.3.1 start() override . . . 295
10.42 samchon::protocol::ExternalClientArray Class Reference . . . 295
10.42.1 Detailed Description . . . 296
10.42.2 Constructor & Destructor Documentation . . . 297
10.42.2.1 ExternalClientArray() . . . 297
10.42.3 Member Function Documentation . . . 298
10.42.3.1 construct(std::shared_ptr< library::XML >) override . . . 298
10.42.3.2 start() override . . . 299
10.42.3.3 PORT() const -> int . . . 299
10.42.3.4 MY_IP() const -> std::string . . . 299
10.42.3.5 addClient(Socket ∗) override . . . 300
10.42.3.6 toXML() const -> std::shared_ptr< library::XML > override . . . 300
10.42.4 Member Data Documentation . . . 301
10.42.4.1 myIP . . . 301
10.42.4.2 port . . . 301
10.43 samchon::protocol::ExternalServer Class Reference . . . 302
10.43.1 Detailed Description . . . 302
10.43.2 Constructor & Destructor Documentation . . . 303
10.43.2.1 ExternalServer() . . . 303
10.43.3 Member Function Documentation . . . 304
10.43.3.1 construct(std::shared_ptr< library::XML >) override . . . 304
10.43.3.2 start() override . . . 304
10.43.3.3 getIP() const -> std::string override . . . 305
10.43.3.4 getPort() const -> int override . . . 305
10.43.3.5 getMyIP() const -> std::string override . . . 305

10.43.3.6 toXML() const -> std::shared_ptr< library::XML > override . . . 305
10.43.4 Member Data Documentation . . . 306
10.43.4.1 myIP . . . 306
10.44 samchon::protocol::ExternalServerArray Class Reference . . . 306
10.44.1 Detailed Description . . . 307
10.44.2 Member Function Documentation . . . 308
10.44.2.1 start() override . . . 308
10.45 samchon::protocol::ExternalSystem Class Reference . . . 309
10.45.1 Detailed Description . . . 310
10.45.2 Constructor & Destructor Documentation . . . 311
10.45.2.1 ExternalSystem() . . . 311
10.45.3 Member Function Documentation . . . 312
10.45.3.1 construct(std::shared_ptr< library::XML >) override . . . 312
10.45.3.2 start()=0 . . . 313
10.45.3.3 key() const -> std::string override . . . 313
10.45.3.4 replyData(std::shared_ptr< Invoke >) override . . . 314
10.45.3.5 TAG() const -> std::string override . . . 314
10.45.3.6 CHILD_TAG() const -> std::string override . . . 314
10.45.3.7 toXML() const -> std::shared_ptr< library::XML > override . . . 315
10.45.4 Member Data Documentation . . . 316
10.45.4.1 name . . . 316
10.45.4.2 ip . . . 316
10.45.4.3 port . . . 316
10.46 samchon::protocol::ExternalSystemArray Class Reference . . . 317
10.46.1 Detailed Description . . . 317
10.46.2 Constructor & Destructor Documentation . . . 318
10.46.2.1 ExternalSystemArray() . . . 318
10.46.3 Member Function Documentation . . . 318
10.46.3.1 start()=0 . . . 318
10.46.3.2 hasRole(const std::string &) const -> bool . . . 319
10.46.3.3 getRole(const std::string &) const -> std::shared_ptr< ExternalSystemRole > . . . 319
10.46.3.4 sendData(std::shared_ptr< Invoke >) override . . . 319
10.46.3.5 replyData(std::shared_ptr< Invoke >) override . . . 320
10.46.3.6 TAG() const -> std::string override . . . 320
10.46.3.7 CHILD_TAG() const -> std::string override . . . 320
10.47 samchon::protocol::ExternalSystemRole Class Reference . . . 321
10.47.1 Detailed Description . . . 322
10.47.2 Constructor & Destructor Documentation . . . 322
10.47.2.1 ExternalSystemRole(ExternalSystem *) . . . 322
10.47.3 Member Function Documentation . . . 323

10.47.3.1 construct(std::shared_ptr< library::XML >) override . . . 323
10.47.3.2 getSystem() const -> ExternalSystem * . . . 323
10.47.3.3 key() const -> std::string override . . . 323
10.47.3.4 hasSendListener(const std::string &) const -> bool . . . 323
10.47.3.5 hasReplyListener(const std::string &) const -> bool . . . 323
10.47.3.6 sendData(std::shared_ptr< Invoke >) override . . . 324
10.47.3.7 TAG() const -> std::string override . . . 325
10.47.3.8 toXML() const -> std::shared_ptr< library::XML > override . . . 325
10.47.4 Member Data Documentation . . . 327
10.47.4.1 system . . . 327
10.47.4.2 name . . . 327
10.47.4.3 replyListeners . . . 327
10.47.4.4 sendListeners . . . 328
10.48 samchon::library::FactorialGenerator Class Reference . . . 328
10.48.1 Detailed Description . . . 328
10.48.2 Constructor & Destructor Documentation . . . 329
10.48.2.1 FactorialGenerator(size_t size) . . . 329
10.49 samchon::protocol::FlashPolicyServer Class Reference . . . 329
10.49.1 Detailed Description . . . 330
10.49.2 Constructor & Destructor Documentation . . . 330
10.49.2.1 FlashPolicyServer() . . . 330
10.49.2.2 FlashPolicyServer(std::shared_ptr< library::XML >) . . . 330
10.49.3 Member Function Documentation . . . 330
10.49.3.1 openServer() . . . 330
10.49.3.2 accept(Socket *) . . . 331
10.49.4 Member Data Documentation . . . 331
10.49.4.1 policy . . . 331
10.50 samchon::library::FTByteFile Class Reference . . . 331
10.50.1 Detailed Description . . . 332
10.50.2 Constructor & Destructor Documentation . . . 333
10.50.2.1 FTByteFile(FTFolder *) . . . 333
10.50.3 Member Function Documentation . . . 333
10.50.3.1 getData() -> ByteArray . . . 333
10.50.4 Member Data Documentation . . . 334
10.50.4.1 data . . . 334
10.51 samchon::library::FTFactory Class Reference . . . 334
10.51.1 Detailed Description . . . 335
10.51.2 Constructor & Destructor Documentation . . . 335
10.51.2.1 FTFactory() . . . 335
10.51.3 Member Function Documentation . . . 335

10.51.3.1 createFile(FTFolder *, std::shared_ptr< XML >) -> FTFile *=0 . . . 335
10.51.3.2 registerInstance(FTInstance *) . . . 336
10.51.4 Member Data Documentation . . . 336
10.51.4.1 application . . . 336
10.51.4.2 category . . . 336
10.51.4.3 member . . . 336
10.51.4.4 instanceMap . . . 336
10.52 samchon::library::FTFile Class Reference . . . 337
10.52.1 Detailed Description . . . 337
10.52.2 Constructor & Destructor Documentation . . . 338
10.52.2.1 FTFile(FTFolder *) . . . 338
10.52.3 Member Function Documentation . . . 338
10.52.3.1 getExtension() const -> std::string . . . 338
10.52.3.2 toXML() const -> std::shared_ptr< library::XML > . . . 339
10.52.4 Member Data Documentation . . . 339
10.52.4.1 extension . . . 339
10.53 samchon::library::FTFolder Class Reference . . . 340
10.53.1 Detailed Description . . . 340
10.53.2 Constructor & Destructor Documentation . . . 341
10.53.2.1 FTFolder(FTFactory *, FTFolder *) . . . 341
10.53.3 Member Function Documentation . . . 341
10.53.3.1 CHILD_TAG() const -> std::string override . . . 341
10.53.3.2 toXML() const -> std::shared_ptr< library::XML > override . . . 342
10.53.4 Member Data Documentation . . . 342
10.53.4.1 factory . . . 342
10.54 samchon::library::FTInstance Class Reference . . . 343
10.54.1 Detailed Description . . . 343
10.54.2 Constructor & Destructor Documentation . . . 344
10.54.2.1 FTInstance(FTFolder *) . . . 344
10.54.3 Member Function Documentation . . . 344
10.54.3.1 TAG() const -> std::string . . . 344
10.54.3.2 key() const -> std::string . . . 345
10.54.3.3 getUID() const -> int . . . 345
10.54.3.4 getParent() const -> FTFolder * . . . 346
10.54.3.5 getName() const -> std::string . . . 346
10.54.3.6 getComment() const -> std::string . . . 346
10.54.3.7 toXML() const -> std::shared_ptr< library::XML > . . . 346
10.54.4 Member Data Documentation . . . 347
10.54.4.1 parent . . . 347
10.54.4.2 uid . . . 347

10.54.4.3 name . . . 347
10.54.4.4 comment . . . 347
10.55 samchon::library::FTTextFile Class Reference . . . 348
10.55.1 Detailed Description . . . 348
10.55.2 Constructor & Destructor Documentation . . . 349
10.55.2.1 FTTextFile(FTFolder *) . . . 349
10.55.3 Member Function Documentation . . . 349
10.55.3.1 getData() const -> std::string . . . 349
10.55.4 Member Data Documentation . . . 350
10.55.4.1 data . . . 350
10.56 samchon::example::tsp::GAParameters Struct Reference . . . 350
10.56.1 Detailed Description . . . 350
10.57 samchon::library::GAParameters Struct Reference . . . 351
10.57.1 Detailed Description . . . 351
10.57.2 Member Data Documentation . . . 351
10.57.2.1 generation . . . 351
10.57.2.2 population . . . 351
10.57.2.3 tournament . . . 351
10.57.2.4 mutationRate . . . 352
10.58 samchon::library::GAPopulation< GeneArray, Compare > Class Template Reference . . . 352
10.58.1 Detailed Description . . . 352
10.58.2 Constructor & Destructor Documentation . . . 354
10.58.2.1 GAPopulation(size_t size) . . . 354
10.58.2.2 GAPopulation(std::shared_ptr< GeneArray > geneArray, size_t size) . . . 354
10.58.3 Member Function Documentation . . . 354
10.58.3.1 fitTest() const -> std::shared_ptr< GeneArray > . . . 354
10.58.4 Member Data Documentation . . . 355
10.58.4.1 children . . . 355
10.59 samchon::library::GeneticAlgorithm< GeneArray, Compare > Class Template Reference . . . 355
10.59.1 Detailed Description . . . 356
10.59.2 Constructor & Destructor Documentation . . . 362
10.59.2.1 GeneticAlgorithm(bool unique, double mutationRate=0.015, size_t tournament=10) 362
10.59.3 Member Function Documentation . . . 363
10.59.3.1 evolveGeneArray(std::shared_ptr< GeneArray > geneArray, size_t population, size_t generation) const -> std::shared_ptr< GeneArray > . . . 363
10.59.3.2 evolvePopulation(std::shared_ptr< MyPopulation > population) const -> std↩ ::shared_ptr< MyPopulation > . . . 364
10.59.3.3 selection(std::shared_ptr< MyPopulation > population) const -> std::shared_↩ ptr< GeneArray > . . . 365
10.59.3.4 crossover(std::shared_ptr< GeneArray > &parent1, std::shared_ptr< GeneArray > &parent2) const -> std::shared_ptr< GeneArray > . . . 366

10.59.3.5 mutate(std::shared_ptr< GeneArray > geneArray) const . . . 367
10.59.4 Member Data Documentation . . . 368
10.59.4.1 unique . . . 368
10.59.4.2 mutationRate . . . 368
10.59.4.3 tournament . . . 369
10.60 samchon::example::tsp::GeometryPoint Class Reference . . . 369
10.60.1 Detailed Description . . . 369
10.60.2 Constructor & Destructor Documentation . . . 370
10.60.2.1 GeometryPoint(int) . . . 370
10.60.2.2 GeometryPoint(int, double, double) . . . 370
10.60.3 Member Function Documentation . . . 371
10.60.3.1 calcDistance(const GeometryPoint &) const -> double . . . 371
10.60.3.2 toString() const -> std::string . . . 372
10.60.4 Member Data Documentation . . . 372
10.60.4.1 uid . . . 372
10.60.4.2 longitude . . . 372
10.60.4.3 latitude . . . 372
10.61 samchon::library::HTTPLoader Class Reference . . . 372
10.61.1 Detailed Description . . . 373
10.62 samchon::protocol::IClient Class Reference . . . 374
10.62.1 Detailed Description . . . 374
10.62.2 Constructor & Destructor Documentation . . . 375
10.62.2.1 IClient() . . . 375
10.62.3 Member Function Documentation . . . 375
10.62.3.1 BUFFER_SIZE() const -> size_t . . . 375
10.62.3.2 listen() . . . 376
10.62.3.3 sendData(std::shared_ptr< Invoke >) . . . 376
10.62.3.4 _replyData(std::shared_ptr< Invoke >) . . . 377
10.62.4 Member Data Documentation . . . 377
10.62.4.1 socket . . . 377
10.62.4.2 sendMtx . . . 377
10.63 samchon::protocol::IEntityChain Class Reference . . . 377
10.63.1 Detailed Description . . . 378
10.63.2 Constructor & Destructor Documentation . . . 379
10.63.2.1 IEntityChain(Entity *) . . . 379
10.63.3 Member Data Documentation . . . 379
10.63.3.1 entity . . . 379
10.64 samchon::protocol::IEntityGroup Class Reference . . . 380
10.64.1 Detailed Description . . . 380
10.64.2 Constructor & Destructor Documentation . . . 380

10.64.2.1 IEntityGroup() . . . 380
10.64.3 Member Function Documentation . . . 381
10.64.3.1 CHILD_TAG() const -> std::string=0 . . . 381
10.65 samchon::protocol::IEntityPtrGroup< _Container > Class Template Reference . . . 382
10.65.1 Detailed Description . . . 382
10.66 samchon::protocol::IHTMLEntity Class Reference . . . 382
10.66.1 Detailed Description . . . 383
10.66.2 Constructor & Destructor Documentation . . . 386
10.66.2.1 IHTMLEntity() . . . 386
10.66.3 Member Function Documentation . . . 387
10.66.3.1 toTH(const _Ty &val, const _Args &...args) -> std::string . . . 387
10.66.3.2 toTR(const _Ty &val, const _Args &...args) -> std::string . . . 388
10.66.3.3 toHTML() const -> std::string=0 . . . 389
10.67 samchon::IndexPair< _Ty > Class Template Reference . . . 389
10.67.1 Detailed Description . . . 389
10.67.2 Member Function Documentation . . . 390
10.67.2.1 getIndex() const -> size_t . . . 390
10.67.2.2 getValue() -> _Ty & . . . 390
10.67.2.3 getValue() const -> const _Ty & . . . 391
10.68 samchon::example::packer::Instance Class Reference . . . 391
10.68.1 Detailed Description . . . 392
10.68.2 Constructor & Destructor Documentation . . . 392
10.68.2.1 Instance(const std::string &name, int price, int volume, int weight) . . . 392
10.68.3 Member Function Documentation . . . 393
10.68.3.1 getName() const -> std::string . . . 393
10.68.3.2 getPrice() const -> int . . . 393
10.68.3.3 getVolume() const -> int . . . 393
10.68.3.4 getWeight() const -> int . . . 394
10.68.3.5 toString() const -> std::string . . . 394
10.68.4 Member Data Documentation . . . 394
10.68.4.1 name . . . 394
10.68.4.2 price . . . 394
10.68.4.3 volume . . . 394
10.68.4.4 weight . . . 394
10.69 samchon::namtree::INTExplore Class Reference . . . 395
10.69.1 Detailed Description . . . 395
10.69.2 Constructor & Destructor Documentation . . . 397
10.69.2.1 INTExplore() . . . 397
10.69.3 Member Function Documentation . . . 397
10.69.3.1 construct(std::shared_ptr< library::XML >) . . . 397

10.69.3.2 getMinimum() const -> double . . . 397
10.69.3.3 getMaximum() const -> double . . . 398
10.69.3.4 getSection() const -> unsigned int . . . 398
10.69.3.5 getPrecision() const -> int . . . 398
10.69.3.6 toXML() const -> std::shared_ptr< library::XML > . . . 398
10.69.4 Member Data Documentation . . . 399
10.69.4.1 minimum . . . 399
10.69.4.2 maximum . . . 399
10.69.4.3 section . . . 399
10.69.4.4 precision . . . 399
10.70 samchon::protocol::Invoke Class Reference . . . 399
10.70.1 Detailed Description . . . 401
10.70.2 Constructor & Destructor Documentation . . . 403
10.70.2.1 Invoke() . . . 403
10.70.2.2 Invoke(const std::string &listener) . . . 403
10.70.2.3 Invoke(const std::string &listener, const _Ty &val, const _Args &...args) . . . 403
10.70.3 Member Function Documentation . . . 403
10.70.3.1 construct(std::shared_ptr< library::XML >) override . . . 403
10.70.3.2 createChild(std::shared_ptr< library::XML >) -> InvokeParameter *override . . . 404
10.70.3.3 getListener() const -> std::string . . . 404
10.70.3.4 setListener(const std::string &) . . . 405
10.70.3.5 TAG() const -> std::string override . . . 405
10.70.3.6 CHILD_TAG() const -> std::string override . . . 405
10.70.3.7 toXML() const -> std::shared_ptr< library::XML > override . . . 405
10.70.3.8 toSQL() const -> std::string . . . 406
10.70.4 Member Data Documentation . . . 406
10.70.4.1 listener . . . 406
10.71 samchon::protocol::InvokeHistory Class Reference . . . 406
10.71.1 Detailed Description . . . 408
10.71.2 Constructor & Destructor Documentation . . . 409
10.71.2.1 InvokeHistory() . . . 409
10.71.2.2 InvokeHistory(std::shared_ptr< Invoke >) . . . 409
10.71.3 Member Function Documentation . . . 409
10.71.3.1 construct(std::shared_ptr< library::XML >) override . . . 409
10.71.3.2 notifyEnd() . . . 409
10.71.3.3 key() const -> std::string override . . . 410
10.71.3.4 getUID() const -> size_t . . . 410
10.71.3.5 getListener() const -> std::string . . . 410
10.71.3.6 getStartTime() const -> library::Datetime . . . 410
10.71.3.7 getEndTime() const -> library::Datetime . . . 411

10.71.3.8 calcElapsedTime() const -> long long . . . 411
10.71.3.9 TAG() const -> std::string override . . . 411
10.71.3.10 toXML() const -> std::shared_ptr< library::XML > override . . . 411
10.71.3.11 toInvoke() const -> std::shared_ptr< Invoke > . . . 412
10.71.4 Member Data Documentation . . . 413
10.71.4.1 uid . . . 413
10.71.4.2 listener . . . 413
10.71.4.3 startTime . . . 414
10.71.4.4 endTime . . . 414
10.72 samchon::protocol::InvokeHistoryArray Class Reference . . . 414
10.72.1 Detailed Description . . . 415
10.72.2 Constructor & Destructor Documentation . . . 415
10.72.2.1 InvokeHistoryArray() . . . 415
10.72.3 Member Function Documentation . . . 415
10.72.3.1 createChild(std::shared_ptr< library::XML >) -> InvokeHistory *override . . . 415
10.72.3.2 TAG() const -> std::string . . . 415
10.72.3.3 CHILD_TAG() const -> std::string . . . 416
10.73 samchon::protocol::InvokeParameter Class Reference . . . 416
10.73.1 Detailed Description . . . 418
10.73.2 Constructor & Destructor Documentation . . . 419
10.73.2.1 InvokeParameter(const std::string &, const std::string &, const std::string &) . . . 419
10.73.2.2 InvokeParameter(const std::string &name, const _Ty &val) . . . 419
10.73.2.3 InvokeParameter(const std::string &, std::string &&) . . . 421
10.73.2.4 InvokeParameter(const std::string &, ByteArray &&) . . . 421
10.73.2.5 InvokeParameter() . . . 421
10.73.3 Member Function Documentation . . . 421
10.73.3.1 construct(std::shared_ptr< library::XML >) override . . . 421
10.73.3.2 key() const -> std::string override . . . 421
10.73.3.3 getName() const -> std::string . . . 421
10.73.3.4 getType() const -> std::string . . . 422
10.73.3.5 getValue() const -> _Ty . . . 422
10.73.3.6 getvalueAsXML() const -> std::shared_ptr< library::XML > . . . 422
10.73.3.7 referValue() const -> const _Ty & . . . 422
10.73.3.8 moveValue() -> _Ty . . . 422
10.73.3.9 TAG() const -> std::string override . . . 422
10.73.3.10 toXML() const -> std::shared_ptr< library::XML > override . . . 423
10.73.3.11 toSQL() const -> std::string . . . 423
10.73.4 Member Data Documentation . . . 423
10.73.4.1 name . . . 423
10.73.4.2 type . . . 424

10.73.4.3 str . . . 424
10.73.4.4 xml . . . 424
10.73.4.5 byteArray . . . 424
10.74 samchon::library::IOperator< _Ty > Class Template Reference . . . 424
10.74.1 Detailed Description . . . 425
10.74.2 Constructor & Destructor Documentation . . . 425
10.74.2.1 IOperator() . . . 425
10.74.3 Member Function Documentation . . . 425
10.74.3.1 operator<(const IOperator &) const -> bool=0 . . . 425
10.74.3.2 operator==(const IOperator &) const -> bool=0 . . . 425
10.75 samchon::protocol::IProtocol Class Reference . . . 426
10.75.1 Detailed Description . . . 426
10.75.2 Constructor & Destructor Documentation . . . 427
10.75.2.1 IProtocol() . . . 427
10.75.3 Member Function Documentation . . . 427
10.75.3.1 replyData(std::shared_ptr< Invoke >) . . . 427
10.75.3.2 sendData(std::shared_ptr< Invoke >) . . . 428
10.76 samchon::protocol::service::IPUserPair Class Reference . . . 428
10.76.1 Detailed Description . . . 429
10.76.2 Constructor & Destructor Documentation . . . 430
10.76.2.1 IPUserPair(Server *server, const std::string &) . . . 430
10.76.3 Member Function Documentation . . . 430
10.76.3.1 getSessionID(Socket *, size_t) -> std::string . . . 430
10.76.3.2 issueSessionID(size_t) const -> std::string . . . 431
10.76.4 Member Data Documentation . . . 431
10.76.4.1 server . . . 431
10.76.4.2 ip . . . 432
10.76.4.3 userSet . . . 432
10.77 samchon::protocol::IServer Class Reference . . . 432
10.77.1 Detailed Description . . . 432
10.77.2 Constructor & Destructor Documentation . . . 435
10.77.2.1 IServer() . . . 435
10.77.3 Member Function Documentation . . . 436
10.77.3.1 MY_IP() const -> std::string . . . 436
10.77.3.2 PORT() const -> int=NULL . . . 436
10.77.3.3 open() . . . 436
10.77.3.4 close() . . . 436
10.77.3.5 addClient(Socket *)=0 . . . 437
10.77.4 Member Data Documentation . . . 437
10.77.4.1 acceptor . . . 437

10.78 samchon::protocol::ISQLEntity Class Reference . . . 437
10.78.1 Detailed Description . . . 437
10.78.2 Constructor & Destructor Documentation . . . 438
10.78.2.1 ISQLEntity() . . . 438
10.78.3 Member Function Documentation . . . 438
10.78.3.1 load(std::shared_ptr< library::SQLStatement > stmt) . . . 438
10.78.3.2 archive(std::shared_ptr< library::SQLStatement > stmt) . . . 439
10.78.3.3 toSQL() const -> std::string . . . 439
10.79 samchon::protocol::IWebClient Class Reference . . . 439
10.79.1 Detailed Description . . . 440
10.79.2 Constructor & Destructor Documentation . . . 441
10.79.2.1 IWebClient() . . . 441
10.79.3 Member Function Documentation . . . 441
10.79.3.1 listen() override . . . 441
10.79.3.2 sendData(std::shared_ptr< Invoke >) override . . . 441
10.80 samchon::protocol::IWebServer Class Reference . . . 442
10.80.1 Detailed Description . . . 442
10.80.2 Constructor & Destructor Documentation . . . 445
10.80.2.1 IWebServer() . . . 445
10.80.3 Member Function Documentation . . . 445
10.80.3.1 open() override . . . 446
10.81 samchon::example::chat_service::ListService Class Reference . . . 446
10.81.1 Detailed Description . . . 446
10.81.2 Constructor & Destructor Documentation . . . 447
10.81.2.1 ListService(protocol::service::Client *) . . . 447
10.81.3 Member Function Documentation . . . 447
10.81.3.1 createRoom(const std::string &) . . . 447
10.82 samchon::Map< _Kty, _Ty, _Pr, _Alloc > Class Template Reference . . . 448
10.82.1 Detailed Description . . . 448
10.82.2 Member Function Documentation . . . 449
10.82.2.1 has(const _Kty &key) const -> bool . . . 449
10.82.2.2 get(const _Kty &key) -> _Ty & . . . 450
10.82.2.3 set(const _Kty &key, const _Ty &val) . . . 450
10.82.2.4 pop(const _Kty &key) -> _Ty . . . 451
10.83 samchon::library::Math Class Reference . . . 451
10.83.1 Detailed Description . . . 452
10.83.2 Member Function Documentation . . . 453
10.83.2.1 random() -> double . . . 453
10.83.2.2 degree_to_radian(double) -> double . . . 454
10.83.2.3 radian_to_degree(double) -> double . . . 454

10.83.2.4 minimum(const _Cont &container) -> IndexPair< _Ty > 454
10.83.2.5 maximum(const _Cont &container) -> IndexPair< _Ty > 454
10.83.2.6 mean(const _Cont &container) -> double 455
10.83.2.7 median(const _Cont &container) -> double 455
10.83.2.8 mode(const _Cont &container) -> _Ty 456
10.83.2.9 stdev_p(const _Cont &container) -> double 456
10.83.2.10 stdev_s(const _Cont &container) -> double 456
10.83.2.11 variance_p(const _Cont &container) -> double 457
10.83.2.12 variance_s(const _Cont &container) -> double 457
10.83.3 Member Data Documentation 458
10.83.3.1 E 458
10.83.3.2 PI 458
10.83.3.3 LN2 458
10.83.3.4 LN10 459
10.83.3.5 LOG2E 459
10.83.3.6 LOG10E 459
10.83.3.7 SQRT1_2 459
10.83.3.8 SQRT2 459
10.84 samchon::library::MySQLi Class Reference 459
10.84.1 Detailed Description 460
10.84.2 Constructor & Destructor Documentation 461
10.84.2.1 MySQLi(int port=3306) 461
10.85 samchon::namtree::NTCriteria Class Reference 462
10.85.1 Detailed Description 462
10.85.2 Member Function Documentation 465
10.85.2.1 construct(std::shared_ptr< library::XML >) override 465
10.85.2.2 createChild(std::shared_ptr< library::XML >) -> NTCriteria *override 465
10.85.2.3 TAG() const -> std::string override 466
10.85.2.4 CHILD_TAG() const -> std::string override 466
10.85.2.5 toXML() const -> std::shared_ptr< library::XML > override 466
10.86 samchon::namtree::NTEntityGroup Class Reference 467
10.86.1 Detailed Description 467
10.86.2 Constructor & Destructor Documentation 468
10.86.2.1 NTEntityGroup() 468
10.86.3 Member Function Documentation 468
10.86.3.1 begin() const -> std::shared_ptr< NTIterator >=0 468
10.86.3.2 end() const -> std::shared_ptr< NTIterator >=0 468
10.87 samchon::namtree::NTFactory Class Reference 468
10.87.1 Detailed Description 469
10.87.2 Constructor & Destructor Documentation 471

10.87.2.1 NTFactory(NTEntityGroup ∗) . . . 471
10.87.3 Member Function Documentation . . . 471
10.87.3.1 createCriteria(NTCriteria ∗, std::shared_ptr< library::XML >) -> NTCriteria ∗ . . . 471
10.87.3.2 createSide(std::shared_ptr< library::XML >) -> NTSide ∗ . . . 471
10.87.4 Member Data Documentation . . . 472
10.87.4.1 data . . . 472
10.87.4.2 sideFunctionMap . . . 472
10.88 samchon::namtree::NTFile Class Reference . . . 472
10.88.1 Detailed Description . . . 473
10.88.2 Constructor & Destructor Documentation . . . 474
10.88.2.1 NTFile(NTFactory ∗, library::FTFolder ∗) . . . 474
10.88.3 Member Function Documentation . . . 475
10.88.3.1 construct(std::shared_ptr< library::XML >) override . . . 475
10.88.3.2 getParameterArray() const -> NTParameterArray ∗ . . . 475
10.88.3.3 getOtherside() const -> NTFile ∗ . . . 475
10.88.3.4 getFunction() const -> SideFunction . . . 476
10.88.3.5 toXML() const -> std::shared_ptr< library::XML > override . . . 476
10.88.4 Member Data Documentation . . . 476
10.88.4.1 factory . . . 476
10.88.4.2 parameterArray . . . 476
10.88.4.3 otherside . . . 477
10.88.4.4 function . . . 477
10.89 samchon::namtree::NTIterator Class Reference . . . 477
10.89.1 Detailed Description . . . 478
10.89.2 Constructor & Destructor Documentation . . . 478
10.89.2.1 NTIterator(const NTEntityGroup ∗) . . . 478
10.89.3 Member Function Documentation . . . 479
10.89.3.1 operator--() const -> std::shared_ptr< NTIterator >=0 . . . 479
10.89.3.2 operator++() const -> std::shared_ptr< NTIterator >=0 . . . 479
10.89.3.3 operator==(const NTIterator &) const -> bool=0 . . . 479
10.89.3.4 operator<(const NTIterator &) const -> bool=0 . . . 479
10.89.4 Member Data Documentation . . . 479
10.89.4.1 data . . . 479
10.90 samchon::namtree::NTParameter Class Reference . . . 480
10.90.1 Detailed Description . . . 481
10.90.2 Constructor & Destructor Documentation . . . 481
10.90.2.1 NTParameter() . . . 481
10.90.3 Member Function Documentation . . . 481
10.90.3.1 construct(std::shared_ptr< library::XML >) override . . . 481

10.90.3.2 createChild(std::shared_ptr< library::XML >) -> NTParameterDetermined *override . . . 482
10.90.3.3 key() const -> std::string override . . . 482
10.90.3.4 getName() const -> std::string . . . 482
10.90.3.5 getInitialValue() const -> double . . . 483
10.90.3.6 TAG() const -> std::string override . . . 483
10.90.3.7 CHILD_TAG() const -> std::string override . . . 483
10.90.3.8 toXML() const -> std::shared_ptr< library::XML > override . . . 483
10.90.4 Member Data Documentation . . . 484
10.90.4.1 name . . . 484
10.90.4.2 initialValue . . . 484
10.91 samchon::namtree::NTParameterArray Class Reference . . . 484
10.91.1 Detailed Description . . . 485
10.91.2 Constructor & Destructor Documentation . . . 486
10.91.2.1 NTParameterArray() . . . 486
10.91.3 Member Function Documentation . . . 486
10.91.3.1 TAG() const -> std::string override . . . 486
10.91.3.2 CHILD_TAG() const -> std::string override . . . 487
10.91.3.3 createChild(std::shared_ptr< library::XML >) -> NTParameter *override . . . 487
10.92 samchon::namtree::NTParameterDetermined Class Reference . . . 487
10.92.1 Detailed Description . . . 488
10.92.2 Constructor & Destructor Documentation . . . 490
10.92.2.1 NTParameterDetermined() . . . 490
10.92.3 Member Function Documentation . . . 490
10.92.3.1 construct(std::shared_ptr< library::XML >) override . . . 490
10.92.3.2 key() const -> std::string override . . . 490
10.92.3.3 getLabel() const -> std::string . . . 490
10.92.3.4 getValue() const -> double . . . 490
10.92.3.5 TAG() const -> std::string override . . . 491
10.92.3.6 toXML() const -> std::shared_ptr< library::XML > override . . . 491
10.92.4 Member Data Documentation . . . 491
10.92.4.1 label . . . 491
10.92.4.2 value . . . 492
10.93 samchon::namtree::NTSide Class Reference . . . 492
10.93.1 Detailed Description . . . 492
10.93.2 Member Function Documentation . . . 494
10.93.2.1 TAG() const -> std::string override . . . 494
10.93.2.2 construct(std::shared_ptr< library::XML >) override . . . 494
10.93.2.3 toXML() const -> std::shared_ptr< library::XML > override . . . 494
10.94 samchon::example::packer::Packer Class Reference . . . 495

10.94.1 Detailed Description . . . 496
10.94.2 Constructor & Destructor Documentation . . . 497
10.94.2.1 Packer(std::vector< Product > *, std::vector< Wrapper > *) . . . 497
10.94.2.2 Packer(const Packer &) . . . 497
10.94.3 Member Function Documentation . . . 497
10.94.3.1 optimize() . . . 497
10.94.3.2 calcPrice() const -> int . . . 498
10.94.3.3 toString() const -> std::string . . . 498
10.94.4 Member Data Documentation . . . 498
10.94.4.1 productArray . . . 498
10.94.4.2 wrapperArray . . . 499
10.95 samchon::protocol::slave::ParallelClient Class Reference . . . 499
10.95.1 Detailed Description . . . 499
10.95.2 Constructor & Destructor Documentation . . . 501
10.95.2.1 ParallelClient() . . . 501
10.96 samchon::protocol::master::ParallelClient Class Reference . . . 501
10.96.1 Detailed Description . . . 501
10.96.2 Constructor & Destructor Documentation . . . 502
10.96.2.1 ParallelClient() . . . 502
10.97 samchon::protocol::master::ParallelClientArray Class Reference . . . 502
10.97.1 Detailed Description . . . 503
10.97.2 Constructor & Destructor Documentation . . . 504
10.97.2.1 ParallelClientArray() . . . 504
10.98 samchon::protocol::master::ParallelClientArrayMediator Class Reference . . . 505
10.98.1 Detailed Description . . . 505
10.98.2 Constructor & Destructor Documentation . . . 508
10.98.2.1 ParallelClientArrayMediator() . . . 508
10.98.3 Member Function Documentation . . . 508
10.98.3.1 construct(std::shared_ptr< library::XML >) override . . . 508
10.98.3.2 start() override . . . 509
10.98.3.3 toXML() const -> std::shared_ptr< library::XML > override . . . 510
10.99 samchon::protocol::master::ParallelServer Class Reference . . . 511
10.99.1 Detailed Description . . . 511
10.99.2 Constructor & Destructor Documentation . . . 512
10.99.2.1 ParallelServer() . . . 512
10.99.3 Member Function Documentation . . . 512
10.99.3.1 construct(std::shared_ptr< library::XML >) override . . . 512
10.99.3.2 toXML() const -> std::shared_ptr< library::XML > override . . . 513
10.100 samchon::protocol::slave::ParallelServer Class Reference . . . 513
10.100.1 Detailed Description . . . 514

10.100.2 Constructor & Destructor Documentation . . . 515
10.100.2.1 ParallelServer() . . . 515
10.101 samchon::protocol::master::ParallelServerArray Class Reference . . . 515
10.101.1 Detailed Description . . . 516
10.101.2 Constructor & Destructor Documentation . . . 517
10.101.2.1 ParallelServerArray() . . . 517
10.102 samchon::protocol::master::ParallelServerArrayMediator Class Reference . . . 517
10.102.1 Detailed Description . . . 518
10.102.2 Constructor & Destructor Documentation . . . 521
10.102.2.1 ParallelServerArrayMediator() . . . 521
10.102.3 Member Function Documentation . . . 521
10.102.3.1 start() override . . . 521
10.103 samchon::protocol::master::ParallelSlaveClientMediator Class Reference . . . 522
10.103.1 Detailed Description . . . 522
10.103.2 Constructor & Destructor Documentation . . . 525
10.103.2.1 ParallelSlaveClientMediator() . . . 525
10.104 samchon::protocol::master::ParallelSlaveServerMediator Class Reference . . . 526
10.104.1 Detailed Description . . . 526
10.104.2 Constructor & Destructor Documentation . . . 529
10.104.2.1 ParallelSlaveServerMediator() . . . 529
10.105 samchon::protocol::master::ParallelSlaveSystemMediator Class Reference . . . 529
10.105.1 Detailed Description . . . 530
10.105.2 Constructor & Destructor Documentation . . . 532
10.105.2.1 ParallelSlaveSystemMediator() . . . 532
10.105.3 Member Function Documentation . . . 533
10.105.3.1 replyPieceData(std::shared_ptr< Invoke >, size_t, size_t) override . . . 533
10.105.3.2 TAG() const -> std::string override . . . 534
10.105.4 Member Data Documentation . . . 534
10.105.4.1 master . . . 534
10.106 samchon::protocol::slave::ParallelSystem Class Reference . . . 535
10.106.1 Detailed Description . . . 535
10.106.2 Constructor & Destructor Documentation . . . 537
10.106.2.1 ParallelSystem() . . . 537
10.106.3 Member Function Documentation . . . 537
10.106.3.1 createChild(std::shared_ptr< library::XML >) -> ExternalSystemRole *override . . . 537
10.106.3.2 _replyData(std::shared_ptr< Invoke >) override . . . 537
10.106.3.3 replyPieceData(std::shared_ptr< Invoke >, size_t, size_t)=0 . . . 537
10.107 samchon::protocol::master::ParallelSystem Class Reference . . . 538
10.107.1 Detailed Description . . . 539
10.107.2 Constructor & Destructor Documentation . . . 539

10.107.2.1 ParallelSystem() . . . 539
10.107.3 Member Function Documentation . . . 540
10.107.3.1 construct(std::shared_ptr< library::XML >) override . . . 540
10.107.3.2 createChild(std::shared_ptr< library::XML >) -> ExternalSystemRole *override . . . 540
10.107.3.3 _replyData(std::shared_ptr< Invoke >) override . . . 541
10.107.3.4 sendPieceData(PRMasterHistory *, std::shared_ptr< Invoke >, size_t, size_t) . . . 541
10.107.3.5 toXML() const -> std::shared_ptr< library::XML > override . . . 541
10.107.4 Member Data Documentation . . . 542
10.107.4.1 systemArray . . . 542
10.107.4.2 historyArray . . . 542
10.107.4.3 progressArray . . . 542
10.107.4.4 performance . . . 542
10.108 samchon::protocol::master::ParallelSystemArray Class Reference . . . 543
10.108.1 Detailed Description . . . 543
10.108.2 Constructor & Destructor Documentation . . . 545
10.108.2.1 ParallelSystemArray() . . . 545
10.108.3 Member Function Documentation . . . 545
10.108.3.1 sendSegmentData(std::shared_ptr< Invoke >, size_t) . . . 545
10.108.3.2 sendPieceData(std::shared_ptr< Invoke >, size_t, size_t) . . . 545
10.108.3.3 notifyEnd(PRMasterHistory *) . . . 546
10.108.4 Member Data Documentation . . . 547
10.108.4.1 uid . . . 547
10.108.4.2 historyArray . . . 547
10.108.4.3 progressArray . . . 547
10.109 samchon::protocol::master::ParallelSystemArrayMediator Class Reference . . . 547
10.109.1 Detailed Description . . . 548
10.109.2 Constructor & Destructor Documentation . . . 550
10.109.2.1 ParallelSystemArrayMediator() . . . 550
10.109.3 Member Function Documentation . . . 550
10.109.3.1 construct(std::shared_ptr< library::XML >) override . . . 550
10.109.3.2 start() override . . . 551
10.109.3.3 notifyEnd(PRMasterHistory *) . . . 552
10.109.3.4 toXML() const -> std::shared_ptr< library::XML > override . . . 552
10.109.4 Member Data Documentation . . . 553
10.109.4.1 slave . . . 553
10.110 samchon::library::PermutationGenerator Class Reference . . . 553
10.110.1 Detailed Description . . . 554
10.110.2 Constructor & Destructor Documentation . . . 555
10.110.2.1 PermutationGenerator(size_t n, size_t r) . . . 555
10.110.3 Member Function Documentation . . . 556

10.110.3.1 operator[](size_t) const -> std::vector< size_t > override . . . 556
10.111 samchon::protocol::master::PRInvokeHistory Class Reference . . . 556
10.111.1 Detailed Description . . . 557
10.111.2 Constructor & Destructor Documentation . . . 558
10.111.2.1 PRInvokeHistory() . . . 558
10.111.2.2 PRInvokeHistory(PRMasterHistory *, ParallelSystem *, std::shared_ptr< Invoke >) . . . 558
10.111.3 Member Function Documentation . . . 558
10.111.3.1 construct(std::shared_ptr< library::XML >) override . . . 558
10.111.3.2 notifyEnd() override . . . 559
10.111.3.3 getIndex() const -> size_t . . . 559
10.111.3.4 getSize() const -> size_t . . . 559
10.111.3.5 calcAverageElapsedTime() const -> double . . . 559
10.111.3.6 toXML() const -> std::shared_ptr< library::XML > override . . . 560
10.111.4 Member Data Documentation . . . 560
10.111.4.1 masterHistory . . . 560
10.111.4.2 system . . . 560
10.111.4.3 index . . . 561
10.111.4.4 size . . . 561
10.112 samchon::protocol::master::PRInvokeHistoryArray Class Reference . . . 561
10.112.1 Detailed Description . . . 562
10.112.2 Constructor & Destructor Documentation . . . 562
10.112.2.1 PRInvokeHistoryArray() . . . 562
10.112.3 Member Function Documentation . . . 562
10.112.3.1 createChild(std::shared_ptr< library::XML >) -> InvokeHistory *override . . . 562
10.113 samchon::protocol::master::PRMasterHistory Class Reference . . . 562
10.113.1 Detailed Description . . . 563
10.113.2 Constructor & Destructor Documentation . . . 564
10.113.2.1 PRMasterHistory(PRMasterHistoryArray *, std::shared_ptr< Invoke >, size_t, size_t) . . . 564
10.113.3 Member Function Documentation . . . 564
10.113.3.1 notifyEnd() override . . . 564
10.113.4 Member Data Documentation . . . 565
10.113.4.1 master . . . 565
10.113.4.2 historyArray . . . 565
10.113.4.3 completed . . . 565
10.114 samchon::protocol::master::PRMasterHistoryArray Class Reference . . . 565
10.114.1 Detailed Description . . . 565
10.114.2 Constructor & Destructor Documentation . . . 566
10.114.2.1 PRMasterHistoryArray(ParallelSystemArray *) . . . 566

10.114.3 Member Data Documentation . . . 566
10.114.3.1 master . . . 566
10.115 Samchon::example::packer::Product Class Reference . . . 566
10.115.1 Detailed Description . . . 566
10.115.2 Constructor & Destructor Documentation . . . 567
10.115.2.1 Product(const std::string &name, int price, int volume, int weight) . . . 567
10.115.3 Member Function Documentation . . . 567
10.115.3.1 toString() const -> std::string override . . . 567
10.116 Samchon::library::ProgressEvent Class Reference . . . 568
10.116.1 Detailed Description . . . 569
10.116.2 Constructor & Destructor Documentation . . . 569
10.116.2.1 ProgressEvent(EventDispatcher *, size_t, size_t) . . . 569
10.116.3 Member Function Documentation . . . 569
10.116.3.1 getNumerator() const -> size_t . . . 569
10.116.3.2 getDenominator() const -> size_t . . . 569
10.116.3.3 getPercent() const -> double . . . 569
10.116.4 Member Data Documentation . . . 569
10.116.4.1 numerator . . . 569
10.116.4.2 denominator . . . 570
10.117 Samchon::library::RWMutex Class Reference . . . 570
10.117.1 Detailed Description . . . 570
10.117.2 Constructor & Destructor Documentation . . . 571
10.117.2.1 RWMutex() . . . 571
10.117.3 Member Function Documentation . . . 571
10.117.3.1 readLock() const . . . 571
10.117.3.2 readUnlock() const . . . 572
10.117.3.3 writeLock() . . . 572
10.117.3.4 writeUnlock() . . . 573
10.118 Samchon::example::tsp::Scheduler Class Reference . . . 573
10.118.1 Detailed Description . . . 574
10.118.2 Constructor & Destructor Documentation . . . 574
10.118.2.1 Scheduler(std::shared_ptr< Travel >, const struct GAParameters &) . . . 574
10.118.3 Member Function Documentation . . . 574
10.118.3.1 optimize() -> std::shared_ptr< Travel > . . . 574
10.118.4 Member Data Documentation . . . 575
10.118.4.1 travel . . . 575
10.118.4.2 gaParameters . . . 575
10.119 Samchon::library::Semaphore Class Reference . . . 575
10.119.1 Detailed Description . . . 576
10.119.2 Constructor & Destructor Documentation . . . 577

10.119.2.1 Semaphore(size_t=2) . . . 577
10.119.3 Member Function Documentation . . . 577
10.119.3.1 setSize(size_t) . . . 577
10.119.3.2 size() const -> size_t . . . 577
10.119.3.3 acquiredSize() const -> size_t . . . 577
10.119.3.4 acquire() . . . 577
10.119.3.5 tryAcquire() -> bool . . . 578
10.119.3.6 release() . . . 578
10.119.4 Member Data Documentation . . . 579
10.119.4.1 size_ . . . 579
10.119.4.2 acquired . . . 579
10.119.4.3 mtx . . . 579
10.120 Samchon::protocol::service::Server Class Reference . . . 579
10.120.1 Detailed Description . . . 580
10.120.2 Constructor & Destructor Documentation . . . 581
10.120.2.1 Server() . . . 581
10.120.3 Member Function Documentation . . . 581
10.120.3.1 NAME() const -> std::string=0 . . . 581
10.120.3.2 size() const -> size_t . . . 581
10.120.3.3 begin() const -> const_iterator . . . 582
10.120.3.4 end() const -> const_iterator . . . 582
10.120.3.5 createUser() -> User *=0 . . . 582
10.120.3.6 addClient(Socket *) override . . . 583
10.120.4 Member Data Documentation . . . 583
10.120.4.1 sqli . . . 583
10.120.4.2 pMap . . . 583
10.120.4.3 sequence . . . 584
10.121 Samchon::protocol::ServerConnector Class Reference . . . 584
10.121.1 Detailed Description . . . 585
10.121.2 Constructor & Destructor Documentation . . . 585
10.121.2.1 ServerConnector() . . . 585
10.121.3 Member Function Documentation . . . 585
10.121.3.1 getIP() const -> std::string=0 . . . 586
10.121.3.2 getPort() const -> int=0 . . . 586
10.121.3.3 getMyIP() const -> std::string . . . 586
10.121.3.4 connect() . . . 586
10.121.4 Member Data Documentation . . . 587
10.121.4.1 ioService . . . 587
10.121.4.2 endPoint . . . 587
10.121.4.3 localEndPoint . . . 588

10.122 Samchon::protocol::service::Service Class Reference . . . 588
10.122.1 Detailed Description . . . 589
10.122.2 Constructor & Destructor Documentation . . . 589
10.122.2.1 Service(Client ∗) . . . 589
10.122.3 Member Function Documentation . . . 589
10.122.3.1 REQUIRE_AUTHORITY() const -> int=NULL . . . 589
10.122.3.2 getClient() const -> Client ∗ . . . 590
10.122.3.3 sendData(std::shared_ptr< Invoke >) . . . 590
10.122.4 Member Data Documentation . . . 590
10.122.4.1 client . . . 590
10.122.4.2 name . . . 590
10.123 Samchon::protocol::service::ServiceKeeper Class Reference . . . 591
10.123.1 Detailed Description . . . 591
10.123.2 Constructor & Destructor Documentation . . . 592
10.123.2.1 ServiceKeeper(const User ∗user, const Client ∗client) . . . 592
10.123.2.2 ServiceKeeper(const ServiceKeeper &) . . . 592
10.123.2.3 ServiceKeeper(ServiceKeeper &&) . . . 592
10.123.3 Member Data Documentation . . . 593
10.123.3.1 user . . . 593
10.123.3.2 client . . . 593
10.124 Samchon::library::SharedAcquire Class Reference . . . 593
10.124.1 Detailed Description . . . 594
10.124.2 Constructor & Destructor Documentation . . . 595
10.124.2.1 SharedAcquire(Semaphore &, bool=true) . . . 595
10.124.2.2 SharedAcquire(const SharedAcquire &) . . . 595
10.124.2.3 SharedAcquire(SharedAcquire &&) . . . 595
10.124.2.4 ∼SharedAcquire() . . . 595
10.124.3 Member Function Documentation . . . 595
10.124.3.1 acquire() . . . 595
10.124.3.2 release() . . . 596
10.124.4 Member Data Documentation . . . 596
10.124.4.1 semaphore . . . 596
10.124.4.2 reference . . . 596
10.124.4.3 isLocked . . . 596
10.125 Samchon::library::SharedReadLock Class Reference . . . 596
10.125.1 Detailed Description . . . 597
10.125.2 Constructor & Destructor Documentation . . . 598
10.125.2.1 SharedReadLock(const RWMutex &, bool=true) . . . 598
10.125.2.2 SharedReadLock(const SharedReadLock &) . . . 599
10.125.2.3 SharedReadLock(SharedReadLock &&) . . . 599

10.125.2.4 ~SharedReadLock() . . . 599
10.125.3 Member Function Documentation . . . 599
10.125.3.1 lock() const . . . 599
10.125.3.2 unlock() const . . . 599
10.125.4 Member Data Documentation . . . 599
10.125.4.1 semaphore . . . 599
10.125.4.2 reference . . . 600
10.125.4.3 isLocked . . . 600
10.126 Samchon::library::SharedWriteLock Class Reference . . . 600
10.126.1 Detailed Description . . . 601
10.126.2 Constructor & Destructor Documentation . . . 602
10.126.2.1 SharedWriteLock(RWMutex &, bool=true) . . . 602
10.126.2.2 SharedWriteLock(const SharedWriteLock &) . . . 602
10.126.2.3 SharedWriteLock(SharedWriteLock &&) . . . 602
10.126.2.4 ~SharedWriteLock() . . . 602
10.126.3 Member Function Documentation . . . 602
10.126.3.1 lock() . . . 602
10.126.3.2 unlock() . . . 603
10.126.4 Member Data Documentation . . . 603
10.126.4.1 semaphore . . . 603
10.126.4.2 reference . . . 603
10.126.4.3 isLocked . . . 603
10.127 Samchon::protocol::slave::SlaveClient Class Reference . . . 603
10.127.1 Detailed Description . . . 604
10.127.2 Constructor & Destructor Documentation . . . 605
10.127.2.1 SlaveClient() . . . 605
10.128 Samchon::protocol::slave::SlaveServer Class Reference . . . 605
10.128.1 Detailed Description . . . 605
10.128.2 Constructor & Destructor Documentation . . . 607
10.128.2.1 SlaveServer() . . . 607
10.129 Samchon::protocol::slave::SlaveSystem Class Reference . . . 607
10.129.1 Detailed Description . . . 607
10.129.2 Constructor & Destructor Documentation . . . 608
10.129.2.1 SlaveSystem() . . . 608
10.129.3 Member Function Documentation . . . 608
10.129.3.1 sendData(std::shared_ptr< Invoke >) override . . . 608
10.129.3.2 _replyData(std::shared_ptr< Invoke >) override . . . 609
10.130 Samchon::SmartPointer< _Ty > Class Template Reference . . . 610
10.130.1 Detailed Description . . . 611
10.130.2 Constructor & Destructor Documentation . . . 611

10.130.2.1 SmartPointer() . . . 611
10.130.2.2 SmartPointer(const _Ty *ptr) . . . 611
10.130.2.3 SmartPointer(const SmartPointer &smartPointer) . . . 611
10.130.2.4 SmartPointer(SmartPointer &&smartPointer) . . . 611
10.130.2.5 ~SmartPointer() . . . 612
10.130.3 Member Function Documentation . . . 612
10.130.3.1 reset(const _Ty *ptr) . . . 612
10.130.3.2 get() const -> _Ty * . . . 612
10.130.3.3 operator->() const -> _Ty * . . . 613
10.130.3.4 operator*() const -> _Ty & . . . 613
10.130.4 Member Data Documentation . . . 613
10.130.4.1 useCountMap . . . 613
10.130.4.2 mtx . . . 613
10.130.4.3 ptr . . . 614
10.131 samchon::library::SQLi Class Reference . . . 614
10.131.1 Detailed Description . . . 615
10.131.2 Constructor & Destructor Documentation . . . 616
10.131.2.1 SQLi(const std::string &driver, int port) . . . 616
10.131.3 Member Function Documentation . . . 617
10.131.3.1 connect(const std::string &ip, const std::string &db, const std::string &id, const std::string &pwd) . . . 617
10.131.3.2 disconnect() . . . 617
10.131.3.3 createStatement() -> std::shared_ptr< SQLStatement > . . . 617
10.131.4 Member Data Documentation . . . 617
10.131.4.1 driver . . . 617
10.131.4.2 port . . . 617
10.131.4.3 hdbc . . . 618
10.131.4.4 stmt . . . 618
10.131.4.5 stmtMutex . . . 618
10.132 samchon::library::SQLStatement Class Reference . . . 618
10.132.1 Detailed Description . . . 619
10.132.2 Constructor & Destructor Documentation . . . 620
10.132.2.1 SQLStatement(SQLi *sqli) . . . 620
10.132.3 Member Function Documentation . . . 621
10.132.3.1 free() . . . 621
10.132.3.2 refresh() . . . 621
10.132.3.3 prepare(const std::string &sql, const _Ty &val, const _Args &...args) . . . 621
10.132.3.4 execute() . . . 621
10.132.3.5 executeDirectly(const std::string &) . . . 621
10.132.3.6 fetch() const -> bool . . . 622

10.132.3.7 next() const -> bool . . . 622
10.132.3.8 size() const -> size_t . . . 622
10.132.3.9 at(size_t index) const -> _Ty . . . 622
10.132.3.10 get(const std::string &) const -> _Ty . . . 623
10.132.3.11 toXML() const -> std::shared_ptr< XML > . . . 623
10.132.4 Member Data Documentation . . . 623
10.132.4.1 sqli . . . 623
10.132.4.2 hstmt . . . 623
10.132.4.3 bindParameterCount . . . 623
10.132.4.4 bindParameterBASizeMap . . . 623
10.133 Samchon::library::StringUtil Class Reference . . . 624
10.133.1 Detailed Description . . . 625
10.133.2 Member Function Documentation . . . 625
10.133.2.1 substitute(const std::string &format, const _Ty &val, const _Args &...args) -> std::string . . . 626
10.133.2.2 substituteSQL(const std::string &format, const _Ty &value, const _Args &...args) -> std::string . . . 627
10.133.2.3 isNumeric(const std::string &str) -> bool . . . 627
10.133.2.4 toNumber(const std::string &str) -> double . . . 628
10.133.2.5 numberFormat(double val, int precision=2) -> std::string . . . 628
10.133.2.6 percentFormat(double val, int precision=2) -> std::string . . . 628
10.133.2.7 colorNumberFormat(double value, int precision=2, double delimiter=0.0) -> std::string . . . 628
10.133.2.8 colorPercentFormat(double value, int precision=2, double delimiter=0.0) -> std::string . . . 629
10.133.2.9 trim(const std::string &val, const std::vector< std::string > &delims) -> std::string 629
10.133.2.10 ltrim(const std::string &val, const std::vector< std::string > &delims) -> std::string 629
10.133.2.11 rtrim(const std::string &val, const std::vector< std::string > &delims) -> std::string 629
10.133.2.12 finds(const std::string &str, const std::vector< std::string > &delims, size_t startIndex=0) -> IndexPair< std::string > . . . 630
10.133.2.13 finds(const std::string &str, const std::vector< std::string > &delims, size_t endIndex=SIZE_MAX) -> IndexPair< std::string > . . . 630
10.133.2.14 substring(const std::string &str, size_t startIndex, size_t endIndex=SIZE_MAX) -> std::string . . . 631
10.133.2.15 between(const std::string &str, const std::string &start="""", const std::string &end="""") -> std::string . . . 631
10.133.2.16 addTab(const std::string &str, size_t n=1) -> std::string . . . 632
10.133.2.17 split(const std::string &str, const std::string &delim) -> std::vector< std::string > 632
10.133.2.18 betweens(const std::string &str, const std::string &start="""", const std::string &end="""") -> std::vector< std::string > . . . 633
10.133.2.19 toLowerCase(const std::string &str) -> std::string . . . 633
10.133.2.20 toUpperCase(const std::string &str) -> std::string . . . 633

10.133.2.21replaceAll(const std::string &str, const std::string &before, const std::string &after) -> std::string . . . 634
10.133.2.22replaceAll(const std::string &str, const std::vector< std::pair< std::string, std::string >> &pairs) -> std::string . . . 634
10.134 samchon::protocol::SystemRole Class Reference . . . 634
10.134.1 Detailed Description . . . 635
10.134.2 Constructor & Destructor Documentation . . . 635
10.134.2.1 SystemRole() . . . 635
10.134.3 Member Function Documentation . . . 635
10.134.3.1 construct(std::shared_ptr< library::XML >) override . . . 635
10.134.3.2 key() const -> std::string override . . . 635
10.134.3.3 hasListener(const std::string &) const -> bool . . . 636
10.134.3.4 TAG() const -> std::string override . . . 636
10.134.3.5 toXML() const -> std::shared_ptr< library::XML > override . . . 636
10.134.4 Member Data Documentation . . . 637
10.134.4.1 name . . . 637
10.134.4.2 listeners . . . 637
10.135 samchon::example::tsp::Travel Class Reference . . . 637
10.135.1 Detailed Description . . . 637
10.135.2 Constructor & Destructor Documentation . . . 638
10.135.2.1 Travel() . . . 638
10.135.2.2 Travel(const Travel &) . . . 638
10.135.2.3 Travel(Travel &&) . . . 638
10.135.3 Member Function Documentation . . . 639
10.135.3.1 calcDistance() const -> double . . . 639
10.135.3.2 operator<(const Travel &) const -> bool . . . 639
10.135.3.3 toString() const -> std::string . . . 639
10.135.4 Member Data Documentation . . . 640
10.135.4.1 distance . . . 640
10.136 samchon::library::TSQLi Class Reference . . . 640
10.136.1 Detailed Description . . . 641
10.136.2 Constructor & Destructor Documentation . . . 642
10.136.2.1 TSQLi(int port=1433) . . . 642
10.136.3 Member Function Documentation . . . 642
10.136.3.1 createStatement() -> std::shared_ptr< SQLStatement > . . . 642
10.137 samchon::library::TSQLStatement Class Reference . . . 642
10.137.1 Detailed Description . . . 643
10.137.2 Member Function Documentation . . . 644
10.137.2.1 toXML() const -> std::shared_ptr< XML > override . . . 644
10.138 samchon::library::UniqueAcquire Class Reference . . . 644

10.138.1 Detailed Description . . . 645
10.138.2 Constructor & Destructor Documentation . . . 646
10.138.2.1 UniqueAcquire(Semaphore &, bool=true) . . . 646
10.138.2.2 UniqueAcquire(const UniqueAcquire &)=delete . . . 647
10.138.2.3 UniqueAcquire(UniqueAcquire &&) . . . 647
10.138.2.4 ~UniqueAcquire() . . . 647
10.138.3 Member Function Documentation . . . 647
10.138.3.1 acquire() . . . 647
10.138.3.2 release() . . . 648
10.138.4 Member Data Documentation . . . 648
10.138.4.1 semaphore . . . 648
10.138.4.2 isLocked . . . 648
10.139 Samchon::library::UniqueReadLock Class Reference . . . 649
10.139.1 Detailed Description . . . 649
10.139.2 Constructor & Destructor Documentation . . . 650
10.139.2.1 UniqueReadLock(const RWMutex &, bool=true) . . . 650
10.139.2.2 UniqueReadLock(const UniqueReadLock &)=delete . . . 651
10.139.2.3 UniqueReadLock(UniqueReadLock &&) . . . 651
10.139.2.4 ~UniqueReadLock() . . . 651
10.139.3 Member Function Documentation . . . 651
10.139.3.1 lock() const . . . 651
10.139.3.2 unlock() const . . . 652
10.139.4 Member Data Documentation . . . 653
10.139.4.1 mtx . . . 653
10.139.4.2 isLocked . . . 653
10.140 Samchon::library::UniqueWriteLock Class Reference . . . 653
10.140.1 Detailed Description . . . 654
10.140.2 Constructor & Destructor Documentation . . . 655
10.140.2.1 UniqueWriteLock(RWMutex &, bool=true) . . . 655
10.140.2.2 UniqueWriteLock(const UniqueWriteLock &)=delete . . . 655
10.140.2.3 UniqueWriteLock(UniqueWriteLock &&) . . . 655
10.140.2.4 ~UniqueWriteLock() . . . 655
10.140.3 Member Function Documentation . . . 656
10.140.3.1 lock() . . . 656
10.140.3.2 unlock() . . . 656
10.140.4 Member Data Documentation . . . 657
10.140.4.1 mtx . . . 657
10.140.4.2 isLocked . . . 657
10.141 Samchon::library::URLVariables Class Reference . . . 657
10.141.1 Detailed Description . . . 658

10.141.2 Constructor & Destructor Documentation . . . 658
10.141.2.1 URLVariables() . . . 658
10.141.2.2 URLVariables(const WeakString &flashVars) . . . 658
10.141.3 Member Function Documentation . . . 659
10.141.3.1 encode(const WeakString &) -> std::string . . . 659
10.141.3.2 decode(const WeakString &) -> std::string . . . 660
10.141.3.3 toString() const -> std::string . . . 660
10.142 Samchon::protocol::service::User Class Reference . . . 660
10.142.1 Detailed Description . . . 662
10.142.2 Constructor & Destructor Documentation . . . 662
10.142.2.1 User(Server *) . . . 662
10.142.3 Member Function Documentation . . . 663
10.142.3.1 getServer() const -> Server * . . . 663
10.142.3.2 getSemaphore() const -> library::Semaphore * . . . 663
10.142.3.3 getID() const -> std::string . . . 663
10.142.3.4 getAuthority() const -> int . . . 664
10.142.3.5 createClient() -> Client *=0 . . . 664
10.142.3.6 addClient(Socket *) . . . 664
10.142.3.7 eraseClient(size_t) . . . 665
10.142.3.8 goLogin(std::shared_ptr< Invoke >) . . . 666
10.142.3.9 goJoin(std::shared_ptr< Invoke >) . . . 666
10.142.3.10 goLogout() . . . 667
10.142.3.11 toLogin(std::shared_ptr< Invoke >) -> bool=0 . . . 668
10.142.3.12 toJoin(std::shared_ptr< Invoke >) -> bool=0 . . . 668
10.142.3.13 sendData(std::shared_ptr< Invoke >) override . . . 668
10.142.3.14 replyData(std::shared_ptr< Invoke >) override . . . 669
10.142.4 Member Data Documentation . . . 670
10.142.4.1 server . . . 670
10.142.4.2 sessionID . . . 670
10.142.4.3 mtx . . . 670
10.142.4.4 semaphore . . . 670
10.142.4.5 id . . . 671
10.142.4.6 authority . . . 671
10.143 Samchon::WeakString Class Reference . . . 672
10.143.1 Detailed Description . . . 673
10.143.2 Constructor & Destructor Documentation . . . 674
10.143.2.1 WeakString() . . . 674
10.143.2.2 WeakString(const char *data, size_t size) . . . 674
10.143.2.3 WeakString(const char *begin, const char *end) . . . 675
10.143.2.4 WeakString(const char *data) . . . 675

10.143.2.5 WeakString(const char &ch) . . . 676
10.143.2.6 WeakString(std::initializer_list< char > &il) . . . 676
10.143.2.7 WeakString(const std::string &str) . . . 676
10.143.3 Member Function Documentation . . . 676
10.143.3.1 data() const -> const char * . . . 676
10.143.3.2 size() const -> size_t . . . 677
10.143.3.3 empty() const -> bool . . . 677
10.143.3.4 at(size_t index) const -> const char & . . . 678
10.143.3.5 operator[](size_t index) const -> const char & . . . 678
10.143.3.6 find(const WeakString &delim, size_t startIndex=NULL) const -> size_t . . . 678
10.143.3.7 rfind(const WeakString &delim, size_t endIndex=SIZE_MAX) const -> size_t . . . 679
10.143.3.8 finds(const std::vector< std::string > &delims, size_t startIndex=0) const -> IndexPair< WeakString > . . . 680
10.143.3.9 rfinds(const std::vector< std::string > &delims, size_t endIndex=SIZE_MAX) const -> IndexPair< WeakString > . . . 680
10.143.3.10 substr(size_t startIndex, size_t endIndex=SIZE_MAX) const -> WeakString . . . 681
10.143.3.11 substring(size_t startIndex, size_t size=SIZE_MAX) const -> WeakString . . . 681
10.143.3.12 between(const WeakString &start={}, const WeakString &end={}) const -> WeakString . . . 682
10.143.3.13 split(const WeakString &delim) const -> std::vector< WeakString > . . . 683
10.143.3.14 betweens(const WeakString &start={}, const WeakString &end={}) const -> std::vector< WeakString > . . . 684
10.143.3.15 trim(const std::vector< std::string > &delims) const -> WeakString . . . 685
10.143.3.16 ltrim(const std::vector< std::string > &delims) const -> WeakString . . . 685
10.143.3.17 rtrim(const std::vector< std::string > &delims) const -> WeakString . . . 685
10.143.3.18 replace(const WeakString &before, const WeakString &after) const -> std::string 686
10.143.3.19 replaceAll(const WeakString &before, const WeakString &after) const -> std::string 686
10.143.3.20 replaceAll(const std::vector< std::pair< std::string, std::string >> &pairs) const -> std::string . . . 686
10.143.3.21 toLowerCase() const -> std::string . . . 687
10.143.3.22 toUpperCase() const -> std::string . . . 688
10.143.3.23 str() const -> std::string . . . 688
10.143.4 Member Data Documentation . . . 688
10.143.4.1 SPACE_ARRAY . . . 688
10.143.4.2 npos . . . 689
10.143.4.3 data_ . . . 689
10.143.4.4 size_ . . . 689
10.144 samchon::protocol::WebServerConnector Class Reference . . . 689
10.144.1 Detailed Description . . . 690
10.144.2 Constructor & Destructor Documentation . . . 691
10.144.2.1 WebServerConnector() . . . 691

10.144.3 Member Function Documentation . . . 691
10.144.3.1 connect() override . . . 691
10.145 Samchon::example::packer::Wrapper Class Reference . . . 691
10.145.1 Detailed Description . . . 692
10.145.2 Constructor & Destructor Documentation . . . 693
10.145.2.1 Wrapper(const std::string &name, int price, int volume, int weight) . . . 693
10.145.2.2 Wrapper(const Wrapper &wrapper) . . . 693
10.145.3 Member Function Documentation . . . 693
10.145.3.1 tryInsert(Product *) -> bool . . . 693
10.145.3.2 toString() const -> std::string override . . . 694
10.146 Samchon::example::packer::WrapperArray Class Reference . . . 694
10.146.1 Detailed Description . . . 695
10.146.2 Constructor & Destructor Documentation . . . 696
10.146.2.1 WrapperArray(Wrapper *) . . . 696
10.146.3 Member Function Documentation . . . 696
10.146.3.1 tryInsert(Product *) -> bool . . . 696
10.146.3.2 optimize() . . . 697
10.146.3.3 calcPrice() const -> int . . . 698
10.146.3.4 toString() const -> std::string . . . 698
10.146.4 Member Data Documentation . . . 698
10.146.4.1 reserved . . . 698
10.146.4.2 sample . . . 698
10.147 Samchon::library::XML Class Reference . . . 699
10.147.1 Detailed Description . . . 700
10.147.2 Constructor & Destructor Documentation . . . 702
10.147.2.1 XML() . . . 702
10.147.2.2 XML(const XML &) . . . 702
10.147.2.3 XML(XML &&) . . . 703
10.147.2.4 XML(WeakString) . . . 703
10.147.2.5 XML(XML *, WeakString &) . . . 704
10.147.3 Member Function Documentation . . . 704
10.147.3.1 push_back(const WeakString &) . . . 704
10.147.3.2 push_back(const std::shared_ptr< XML >) . . . 705
10.147.3.3 addAllProperty(const std::shared_ptr< XML >) . . . 705
10.147.3.4 setTag(const std::string &) . . . 705
10.147.3.5 setValue(const _Ty &val) . . . 705
10.147.3.6 setProperty(const std::string &name, const _Ty &val) . . . 706
10.147.3.7 eraseProperty(const std::string &) . . . 706
10.147.3.8 clearProperties() . . . 706
10.147.3.9 getTag() const -> std::string . . . 707

10.147.3.10 getValue() const -> _Ty 707
10.147.3.11 getProperty(const std::string &key) const -> _Ty 707
10.147.3.12 hasProperty(const std::string &) const -> bool 707
10.147.3.13 getPropertyMap() const -> const Map< std::string, std::string > & 707
10.147.3.14 toString(size_t level=0) const -> std::string 707
10.147.4 Member Data Documentation 708
10.147.4.1 tag 708
10.147.4.2 value 708
10.147.4.3 propertyMap 708

Index **709**

Chapter 1

Samchon Framework

Samchon Framework is a framework for

- Providing libraries.
- Realizing cloud server(service).
- Constituting Distributed, Parallel processing system.
- Realizing composite and logical retrivalance model; Nam-Tree.

Related Homepages

- `Homepage of Samchon Framework`
- `Github - Samchon Framework`
- Examples
  - `Samchon Simulation Cloud`
  - `Hansung Timetable v3.0` (id: guest, password: N/A)

Related Documents

- Guidance for Developers
  - `English`
  - `Korean`
- Architecture Designs
  - `Class Diagram of CPP`
  - `Class Diagram of JS(TypeScript and Flex(JS))`
  - `Sequence Diagram`
  - `Entity Relationship Diagram`
- Another API Documents
  - `C++ as PDF file`
  - `Flex`
  - `TypeScript`

Author

Jeongho Nam

- E-mail: `samchon@samchon.org`
- Mobile: 82-10-3627-0016

Copyright

BSD license

Chapter 2

Test List

Class samchon::library::EventDispatcher

Chapter 3

Todo List

Class samchon::library::EventDispatcher

Class samchon::protocol::master::DistributedSystem

Class samchon::protocol::master::ParallelSystemArrayMediator

Sends PRMasterHistory to its origin master.

Chapter 4

Deprecated List

Class samchon::library::Event

Class samchon::library::EventDispatcher

Chapter 5

Bug List

Class samchon::library::EventDispatcher

Chapter 6

Namespace Index

6.1 Namespace List

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

samchon
Top level namespace of products built from samchon . . . . . ??
samchon::example::chat_service
A chat service inherited from cloud service . . . . . ??
samchon::example::console_chat_server
A chat server . . . . . ??
samchon::example::tsp
A packer with number of wrapper type and products . . . . . ??
samchon::library
Package of libraries . . . . . ??
samchon::namtree
Package of Nam-Tree . . . . . ??
samchon::protocol
Package of network protocol and libraries . . . . . ??
samchon::protocol::master
Package for external system, within the framework of master . . . . . ??
samchon::protocol::service
Package of cloud service as a server . . . . . ??
samchon::protocol::slave
Package of external system, within the framework of slave . . . . . ??

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

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

samchon::library::Base64 . . . 73
samchon::ByteArray . . . 75
samchon::library::CaseGenerator . . . 80
  samchon::library::CombinedPermutationGenerator . . . 130
  samchon::library::PermutationGenerator . . . 553
    samchon::library::FactorialGenerator . . . 328
samchon::library::Charset . . . 92
samchon::library::CriticalAllocator< _Ty > . . . 132
samchon::library::Date . . . 137
  samchon::library::Datetime . . . 159
samchon::protocol::Entity . . . 243
  samchon::protocol::EntityGroup< _Container, Entity * > . . . 259
    samchon::protocol::IEntityPtrGroup< _Container > . . . 382
  samchon::protocol::EntityGroup< _Container, std::shared_ptr< Entity > > . . . 259
  samchon::protocol::EntityGroup< _Container, std::unique_ptr< Entity > > . . . 259
  samchon::example::chat_service::ChatMessage . . . 99
  samchon::example::chat_service::ChatRoom . . . 103
  samchon::library::FTInstance . . . 343
    samchon::library::FTFile . . . 337
      samchon::library::FTByteFile . . . 331
      samchon::library::FTTextFile . . . 348
      samchon::namtree::NTFile . . . 472
    samchon::library::FTFolder . . . 340
  samchon::namtree::INTExplore . . . 395
    samchon::namtree::NTFile . . . 472
    samchon::namtree::NTParameter . . . 480
  samchon::namtree::NTParameterDetermined . . . 487
  samchon::namtree::NTSide . . . 492
  samchon::protocol::EntityArray< _Ty > . . . 251
  samchon::protocol::EntityGroup< _Container, _ETy, _Ty > . . . 259
    samchon::library::FTFolder . . . 340
    samchon::namtree::NTCriteria . . . 462
    samchon::namtree::NTParameter . . . 480
    samchon::namtree::NTParameterArray . . . 484
    samchon::protocol::ExternalSystem . . . 309
      samchon::protocol::ExternalClient . . . 293

samchon::protocol::master::DistributedClient . . . 181
samchon::protocol::master::ParallelClient . . . 501
samchon::protocol::slave::SlaveClient . . . 603
samchon::protocol::master::DistributedClientServerMediator . . . 193
samchon::protocol::slave::ParallelClient . . . 499
samchon::protocol::master::ParallelSlaveClientMediator . . . 522
samchon::protocol::ExternalServer . . . 302
samchon::protocol::master::DistributedServer . . . 196
samchon::protocol::master::ParallelServer . . . 511
samchon::protocol::slave::SlaveServer . . . 605
samchon::protocol::master::DistributedSlaveServerMediator . . . 207
samchon::protocol::slave::ParallelServer . . . 513
samchon::protocol::master::ParallelSlaveServerMediator . . . 526
samchon::protocol::master::DistributedSystem . . . 214
samchon::protocol::master::DistributedClient . . . 181
samchon::protocol::master::DistributedServer . . . 196
samchon::protocol::master::ParallelSystem . . . 538
samchon::protocol::master::ParallelClient . . . 501
samchon::protocol::master::ParallelServer . . . 511
samchon::protocol::slave::SlaveSystem . . . 607
samchon::protocol::master::DistributedSlaveSystemMediator . . . 210
samchon::protocol::master::DistributedClientServerMediator . . . 193
samchon::protocol::master::DistributedSlaveServerMediator . . . 207
samchon::protocol::slave::ParallelSystem . . . 535
samchon::protocol::master::ParallelSlaveSystemMediator . . . 529
samchon::protocol::master::ParallelSlaveClientMediator . . . 522
samchon::protocol::master::ParallelSlaveServerMediator . . . 526
samchon::protocol::slave::ParallelClient . . . 499
samchon::protocol::slave::ParallelServer . . . 513
samchon::protocol::slave::SlaveClient . . . 603
samchon::protocol::slave::SlaveServer . . . 605
samchon::protocol::ExternalSystemArray . . . 317
samchon::protocol::ExternalClientArray . . . 295
samchon::protocol::master::DistributedClientArray . . . 184
samchon::protocol::master::DistributedClientArrayMediator . . . 188
samchon::protocol::master::ParallelClientArray . . . 502
samchon::protocol::master::ParallelClientArrayMediator . . . 505
samchon::protocol::ExternalServerArray . . . 306
samchon::protocol::master::DistributedServerArray . . . 200
samchon::protocol::master::DistributedServerArrayMediator . . . 203
samchon::protocol::master::ParallelServerArray . . . 515
samchon::protocol::master::ParallelServerArrayMediator . . . 517
samchon::protocol::master::DistributedSystemArray . . . 220
samchon::protocol::master::DistributedClientArray . . . 184
samchon::protocol::master::DistributedServerArray . . . 200
samchon::protocol::master::DistributedSystemArrayMediator . . . 227
samchon::protocol::master::DistributedClientArrayMediator . . . 188
samchon::protocol::master::DistributedServerArrayMediator . . . 203
samchon::protocol::master::ParallelSystemArray . . . 543
samchon::protocol::master::ParallelClientArray . . . 502
samchon::protocol::master::ParallelServerArray . . . 515
samchon::protocol::master::ParallelSystemArrayMediator . . . 547
samchon::protocol::master::ParallelClientArrayMediator . . . 505
samchon::protocol::master::ParallelServerArrayMediator . . . 517
samchon::protocol::Invoke . . . 399
samchon::protocol::InvokeHistoryArray . . . 414

samchon::protocol::master::PRInvokeHistoryArray . . . 561
samchon::protocol::master::PRMasterHistoryArray . . . 565
samchon::protocol::master::DSInvokeHistoryList . . . 241
samchon::protocol::EntityList< _Ty > . . . 274
samchon::protocol::ExternalSystemRole . . . 321
samchon::protocol::master::DistributedSystemRole . . . 233
samchon::protocol::InvokeHistory . . . 406
samchon::protocol::master::DSInvokeHistory . . . 237
samchon::protocol::master::PRInvokeHistory . . . 556
samchon::protocol::master::PRMasterHistory . . . 562
samchon::protocol::InvokeParameter . . . 416
samchon::protocol::SystemRole . . . 634
samchon::library::Event . . . 284
samchon::library::ErrorEvent . . . 281
samchon::library::ProgressEvent . . . 568
samchon::library::EventDispatcher . . . 287
samchon::protocol::FlashPolicyServer . . . 329
samchon::library::FTFactory . . . 334
samchon::namtree::NTFactory . . . 468
samchon::example::tsp::GAParameters . . . 350
samchon::library::GAParameters . . . 351
samchon::library::GAPopulation< GeneArray, Compare > . . . 352
samchon::library::GeneticAlgorithm< GeneArray, Compare > . . . 355
samchon::example::tsp::GeometryPoint . . . 369
samchon::library::HTTPLoader . . . 372
samchon::protocol::IEntityChain . . . 377
samchon::protocol::IEntityGroup . . . 380
samchon::protocol::EntityGroup< _Container, Entity * > . . . 259
samchon::protocol::EntityGroup< _Container, std::shared_ptr< Entity > > . . . 259
samchon::protocol::EntityGroup< _Container, std::unique_ptr< Entity > > . . . 259
samchon::example::chat_service::ChatRoomArray . . . 109
samchon::protocol::EntityArray< _Ty > . . . 251
samchon::protocol::EntityGroup< _Container, _ETy, _Ty > . . . 259
samchon::protocol::EntityList< _Ty > . . . 274
samchon::protocol::IHTMLEntity . . . 382
samchon::IndexPair< _Ty > . . . 389
samchon::example::packer::Instance . . . 391
samchon::example::packer::Product . . . 566
samchon::example::packer::Wrapper . . . 691
samchon::library::IOperator< _Ty > . . . 424
samchon::library::IOperator< NTIterator > . . . 424
samchon::namtree::NTIterator . . . 477
samchon::protocol::IProtocol . . . 426
samchon::example::chat_service::ChatRoom . . . 103
samchon::example::console_chat_server::ChatServer . . . 113
samchon::protocol::ExternalSystemArray . . . 317
samchon::protocol::ExternalSystemRole . . . 321
samchon::protocol::IClient . . . 374
samchon::example::console_chat_server::ChatClient . . . 97
samchon::protocol::ExternalSystem . . . 309
samchon::protocol::IWebClient . . . 439
samchon::protocol::WebServerConnector . . . 689
samchon::protocol::ServerConnector . . . 584
samchon::protocol::ExternalServer . . . 302
samchon::protocol::WebServerConnector . . . 689
samchon::protocol::service::Client . . . 123

samchon::example::chat_service::ChatClient . . . 95
samchon::protocol::service::Service . . . 588
samchon::example::chat_service::ChatService . . . 119
samchon::example::chat_service::ListService . . . 446
samchon::protocol::service::User . . . 660
samchon::example::chat_service::ChatUser . . . 121
samchon::protocol::SystemRole . . . 634
samchon::protocol::service::IPUserPair . . . 428
samchon::protocol::IServer . . . 432
samchon::example::console_chat_server::ChatServer . . . 113
samchon::protocol::ExternalClientArray . . . 295
samchon::protocol::IWebServer . . . 442
samchon::protocol::service::Server . . . 579
samchon::example::chat_service::ChatServer . . . 117
samchon::protocol::ISQLEntity . . . 437
samchon::Map< _Kty, _Ty, _Pr, _Alloc > . . . 448
samchon::protocol::service::Server . . . 579
samchon::Map< int, samchon::library::FTInstance * > . . . 448
samchon::Map< SideFunction > . . . 448
samchon::Map< size_t, SmartPointer< Client > > . . . 448
samchon::protocol::service::User . . . 660
samchon::Map< size_t, SQL_SIZE_T > . . . 448
samchon::Map< std::shared_ptr< samchon::protocol::master::DistributedSystemRole > > . . . 448
samchon::Map< std::shared_ptr< samchon::protocol::service::IPUserPair > > . . . 448
samchon::Map< std::string, std::shared_ptr< XMLList > > . . . 448
samchon::library::XML . . . 699
samchon::Map< std::string, std::string > . . . 448
samchon::library::URLVariables . . . 657
samchon::library::Math . . . 451
samchon::namtree::NTEntityGroup . . . 467
samchon::example::packer::Packer . . . 495
samchon::library::RWMutex . . . 570
samchon::example::tsp::Scheduler . . . 573
samchon::library::Semaphore . . . 575
samchon::protocol::service::ServiceKeeper . . . 591
samchon::library::SharedAcquire . . . 593
samchon::library::SharedReadLock . . . 596
samchon::library::SharedWriteLock . . . 600
samchon::SmartPointer< _Ty > . . . 610
samchon::SmartPointer< samchon::protocol::service::Client > . . . 610
samchon::SmartPointer< samchon::protocol::service::User > . . . 610
samchon::library::SQLi . . . 614
samchon::library::MySQLi . . . 459
samchon::library::TSQLi . . . 640
samchon::library::SQLStatement . . . 618
samchon::library::TSQLStatement . . . 642
samchon::library::StringUtil . . . 624
samchon::example::tsp::Travel . . . 637
samchon::library::UniqueAcquire . . . 644
samchon::library::UniqueReadLock . . . 649
samchon::library::UniqueWriteLock . . . 653
samchon::WeakString . . . 672
samchon::example::packer::WrapperArray . . . 694

Chapter 8

Class Index

8.1 Class List

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

samchon::library::Base64
Utility class for base64 format's en-decoding . . . . . ??
samchon::ByteArray
Binary data class . . . . . ??
samchon::library::CaseGenerator
Case generator . . . . . ??
samchon::library::Charset
A utility class supporting conversion between multiple character-sets . . . . . ??
samchon::example::chat_service::ChatClient
A client . . . . . ??
samchon::example::console_chat_server::ChatClient
A chat client . . . . . ??
samchon::example::chat_service::ChatMessage
A chat message . . . . . ??
samchon::example::chat_service::ChatRoom
A chat room . . . . . ??
samchon::example::chat_service::ChatRoomArray
A manager of chat rooms . . . . . ??
samchon::example::console_chat_server::ChatServer
A chat server . . . . . ??
samchon::example::chat_service::ChatServer
A chat server . . . . . ??
samchon::example::chat_service::ChatService
A chat service . . . . . ??
samchon::example::chat_service::ChatUser
An user in chat server . . . . . ??
samchon::protocol::service::Client
A network boundary with the client in an User . . . . . ??
samchon::library::CombinedPermutationGenerator
A combined-permutation case generator . . . . . ??
samchon::library::CriticalAllocator< _Ty >
An allocator ensuring concurrency . . . . . ??
samchon::library::Date
Date (year, month, day in month) . . . . . ??
samchon::library::Datetime
Date and time . . . . . ??
samchon::protocol::master::DistributedClient
A network driver for a distributed client . . . . . ??

samchon::protocol::master::DistributedClientArray
An array of DistributedClient(s) . . . . . ??
samchon::protocol::master::DistributedClientArrayMediator
A mediator of distributed system between master and slaves . . . . . ??
samchon::protocol::master::DistributedClientServerMediator
A distributed slave client for mediation . . . . . ??
samchon::protocol::master::DistributedServer
A network driver for a distributed server . . . . . ??
samchon::protocol::master::DistributedServerArray
An array of DistributedServer(s) . . . . . ??
samchon::protocol::master::DistributedServerArrayMediator
A mediator of distributed system between master and slaves . . . . . ??
samchon::protocol::master::DistributedSlaveServerMediator
A distributed slave server for mediation . . . . . ??
samchon::protocol::master::DistributedSlaveSystemMediator
A distributed slave system for mediation . . . . . ??
samchon::protocol::master::DistributedSystem
A network driver for a distributed system . . . . . ??
samchon::protocol::master::DistributedSystemArray
An array of distributed system drivers . . . . . ??
samchon::protocol::master::DistributedSystemArrayMediator
A mediator of distributed system between master and slaves . . . . . ??
samchon::protocol::master::DistributedSystemRole
A role of distributed processing system . . . . . ??
samchon::protocol::master::DSInvokeHistory
A reported history of an Invoke message . . . . . ??
samchon::protocol::master::DSInvokeHistoryList
A list of DSInvokeHistory . . . . . ??
samchon::protocol::Entity
An entity, a standard data class . . . . . ??
samchon::protocol::EntityArray< _Ty >
An Entity and a static array containing Entity objects . . . . . ??
samchon::protocol::EntityGroup< _Container, _ETy, _Ty >
An Entity and a container of children Entity objects . . . . . ??
samchon::protocol::EntityList< _Ty >
An Entity and a static list containing Entity objects . . . . . ??
samchon::library::ErrorEvent
Event representing an error . . . . . ??
samchon::library::Event
Represent an event running on background . . . . . ??
samchon::library::EventDispatcher
Abstract class for dispatching Event . . . . . ??
samchon::protocol::ExternalClient
A network driver for an external client . . . . . ??
samchon::protocol::ExternalClientArray
An array of ExternalClient(s) . . . . . ??
samchon::protocol::ExternalServer
A network driver for an external server . . . . . ??
samchon::protocol::ExternalServerArray
An array of ExternalServer(s) . . . . . ??
samchon::protocol::ExternalSystem
A network driver for an external system . . . . . ??
samchon::protocol::ExternalSystemArray
An array of ExternalSystem(s) . . . . . ??
samchon::protocol::ExternalSystemRole
A role belongs to an external system . . . . . ??
samchon::library::FactorialGenerator
Factorical case generator . . . . . ??

samchon::protocol::FlashPolicyServer
A flash policy server . . . . . ??
samchon::library::FTByteFile
A binary file . . . . . ??
samchon::library::FTFactory
A Factory and manager for file-tree . . . . . ??
samchon::library::FTFile
A file . . . . . ??
samchon::library::FTFolder
A folder . . . . . ??
samchon::library::FTInstance
An interface of file-tree . . . . . ??
samchon::library::FTTextFile
A text file . . . . . ??
samchon::example::tsp::GAParameters
Parameters for Genetic-Algorithm . . . . . ??
samchon::library::GAParameters
A structure containing parameters of genetic algorithm . . . . . ??
samchon::library::GAPopulation< GeneArray, Compare >
A population of a generation in G.A . . . . . ??
samchon::library::GeneticAlgorithm< GeneArray, Compare >
A genetic algorithm class . . . . . ??
samchon::example::tsp::GeometryPoint
A geometry coordinates (point) . . . . . ??
samchon::library::HTTPLoader
A http, web-page loader . . . . . ??
samchon::protocol::IClient
An interface for a client . . . . . ??
samchon::protocol::IEntityChain
A chain of entity . . . . . ??
samchon::protocol::IEntityGroup
An iternface for entity group . . . . . ??
samchon::protocol::IEntityPtrGroup< _Container > . . . . . ??
samchon::protocol::IHTMLEntity
An interface supporting conversion to html . . . . . ??
samchon::IndexPair< _Ty >
A pair of index and its value(_Ty) . . . . . ??
samchon::example::packer::Instance
A physical instance . . . . . ??
samchon::namtree::INTExplore
An interface for exploration . . . . . ??
samchon::protocol::Invoke
Standard message of network I/O . . . . . ??
samchon::protocol::InvokeHistory
A history of an Invoke message . . . . . ??
samchon::protocol::InvokeHistoryArray
An array of InvokeHistory . . . . . ??
samchon::protocol::InvokeParameter
A parameter of an Invoke . . . . . ??
samchon::library::IOperator< _Ty >
Interface for comparision operator . . . . . ??
samchon::protocol::IProtocol
An interface of Invoke message chain . . . . . ??
samchon::protocol::service::IPUserPair
Relationship between IP address and User for issuing session . . . . . ??
samchon::protocol::IServer
An interface of a physical server . . . . . ??

samchon::protocol::ISQLEntity
An interface supporting DB-I/O . . . . . ??
samchon::protocol::IWebClient
An interface for a web-client . . . . . ??
samchon::protocol::IWebServer
An interface for a physical server following web-socket . . . . . ??
samchon::example::chat_service::ListService
A list service of chat rooms . . . . . ??
samchon::Map< _Kty, _Ty, _Pr, _Alloc >
Map . . . . . ??
samchon::library::Math
Utility of Math . . . . . ??
samchon::library::MySQLi
A SQL interface for My-SQL . . . . . ??
samchon::namtree::NTCriteria
Criteria, a conditional expression with weight . . . . . ??
samchon::namtree::NTEntityGroup
A historical, studying data . . . . . ??
samchon::namtree::NTFactory
A factory for Nam-Tree objects . . . . . ??
samchon::namtree::NTFile
A file archiving metadata of a function . . . . . ??
samchon::namtree::NTIterator
Iterator of historical data . . . . . ??
samchon::namtree::NTParameter
A metadata of a parameter in a function . . . . . ??
samchon::namtree::NTParameterArray
An Array of NTParameter . . . . . ??
samchon::namtree::NTParameterDetermined
A pre-determined value of a parameter . . . . . ??
samchon::namtree::NTSide
A side of a conditional expresson . . . . . ??
samchon::example::packer::Packer
A packer planning the best packaging . . . . . ??
samchon::protocol::slave::ParallelClient
A slave parallel client . . . . . ??
samchon::protocol::master::ParallelClient
A network driver for a parallel client . . . . . ??
samchon::protocol::master::ParallelClientArray
An array of ParallelClient(s) . . . . . ??
samchon::protocol::master::ParallelClientArrayMediator
A mediator of parallel system between master and slaves . . . . . ??
samchon::protocol::master::ParallelServer
A network driver for a parallel server . . . . . ??
samchon::protocol::slave::ParallelServer
A slave parallel server . . . . . ??
samchon::protocol::master::ParallelServerArray
An array of ParallelServer(s) . . . . . ??
samchon::protocol::master::ParallelServerArrayMediator
A mediator of parallel system between master and slaves . . . . . ??
samchon::protocol::master::ParallelSlaveClientMediator
A parallel slave client for mediation . . . . . ??
samchon::protocol::master::ParallelSlaveServerMediator
A parallel slave server for mediation . . . . . ??
samchon::protocol::master::ParallelSlaveSystemMediator
A parallel slave system for mediation . . . . . ??
samchon::protocol::slave::ParallelSystem
A slave parallel system . . . . . ??

samchon::protocol::master::ParallelSystem
A network driver for a parallel system . . . . . ??
samchon::protocol::master::ParallelSystemArray
An array of parallel system drivers . . . . . ??
samchon::protocol::master::ParallelSystemArrayMediator
A mediator of parallel system between master and slaves . . . . . ??
samchon::library::PermutationGenerator
A permutation case generator . . . . . ??
samchon::protocol::master::PRInvokeHistory
A reported history of an Invoke message . . . . . ??
samchon::protocol::master::PRInvokeHistoryArray
An array of PRInvokeHistory . . . . . ??
samchon::protocol::master::PRMasterHistory
A history log of an Invoke message on a master . . . . . ??
samchon::protocol::master::PRMasterHistoryArray
An array of invoke histories of master . . . . . ??
samchon::example::packer::Product
A product, merchandise . . . . . ??
samchon::library::ProgressEvent
Event representing a progress . . . . . ??
samchon::library::RWMutex
Rw_mutex . . . . . ??
samchon::example::tsp::Scheduler
A scheduler . . . . . ??
samchon::library::Semaphore
A semaphore . . . . . ??
samchon::protocol::service::Server
A server for (cloud) service . . . . . ??
samchon::protocol::ServerConnector
A server connector for a physical client . . . . . ??
samchon::protocol::service::Service
A service . . . . . ??
samchon::protocol::service::ServiceKeeper
A keeper blocking destrunction of User and Client . . . . . ??
samchon::library::SharedAcquire
Shared acquire from a Semaphore . . . . . ??
samchon::library::SharedReadLock
Shared lock from a RWMutex . . . . . ??
samchon::library::SharedWriteLock
Shared lock from a RWMutex . . . . . ??
samchon::protocol::slave::SlaveClient
A slave client . . . . . ??
samchon::protocol::slave::SlaveServer
A slave server . . . . . ??
samchon::protocol::slave::SlaveSystem
A slave system . . . . . ??
samchon::SmartPointer< _Ty >
Global shared pointer
??
samchon::library::SQLi
A SQL interface; DBMS connector . . . . . ??
samchon::library::SQLStatement
A sql statement . . . . . ??
samchon::library::StringUtil
Utility class for string . . . . . ??
samchon::protocol::SystemRole
A role belongs to a system . . . . . ??

samchon::example::tsp::Travel
Represent a travel containning Point(s) . . . . . ??
samchon::library::TSQLi
A SQL interface for T-SQL . . . . . ??
samchon::library::TSQLStatement
A t-sql statement . . . . . ??
samchon::library::UniqueAcquire
Unique acquire from a Semaphore . . . . . ??
samchon::library::UniqueReadLock
Unique lock for reading . . . . . ??
samchon::library::UniqueWriteLock
Unique lock for writing . . . . . ??
samchon::library::URLVariables
URLVariables class is for representing variables of HTTP . . . . . ??
samchon::protocol::service::User
User containing Client(s) with session-id . . . . . ??
samchon::WeakString
A string class only references characeters, reference only . . . . . ??
samchon::protocol::WebServerConnector
A web-socket server connector . . . . . ??
samchon::example::packer::Wrapper
A wrapper can contain products . . . . . ??
samchon::example::packer::WrapperArray
An array of wrapper with same category (name) . . . . . ??
samchon::library::XML
XML is a class representing xml object . . . . . ??

Chapter 9

Namespace Documentation

9.1 samchon Namespace Reference

Top level namespace of products built from samchon.

Namespaces

- library

  Package of libraries.
- namtree

  Package of Nam-Tree.
- protocol

  Package of network protocol and libraries.

Classes

- class ByteArray

  Binary data class.
- class IndexPair

  A pair of index and its value(_Ty)
- class Map

  Map.
- class SmartPointer

  Global shared pointer

  .
- class WeakString

  A string class only references characeters, reference only.

Typedefs

- template<typename _Ty , typename _Pr = std::less<std::string>, typename _Alloc = std::allocator<std::pair<const std::string, _↩ Ty>>>

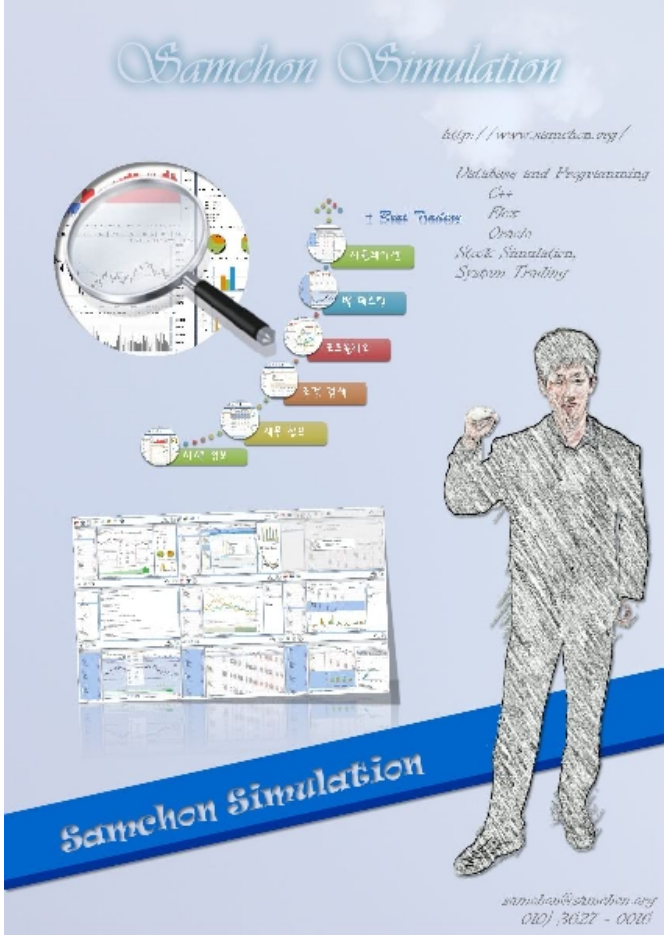
  using Dictionary = Map< std::string, _Ty, _Pr, _Alloc >

  A Map key is string.

9.1.1 Detailed Description

Top level namespace of products built from samchon.

- http://samchon.org/
- https://github.com/samchon/



Author

Jeongho Nam

9.1.2 Typedef Documentation

9.1.2.1 template<typename _Ty , typename _Pr = std::less<std::string>, typename _Alloc = std::allocator<std::pair<const std::string, _Ty>>> using samchon::Dictionary = typedef Map<std::string, _Ty, _Pr, _Alloc>

A Map key is string.

See also

library::Map

Author

Jeongho Nam

Definition at line 14 of file Dictionary.hpp.

9.2 samchon::example::chat_service Namespace Reference

A chat service inherited from cloud service.

Classes

- class ChatClient

  A client.

- class ChatMessage

  A chat message.

- class ChatRoom

  A chat room.

- class ChatRoomArray

  A manager of chat rooms.

- class ChatServer

  A chat server.

- class ChatService

  A chat service.

- class ChatUser

  An user in chat server.

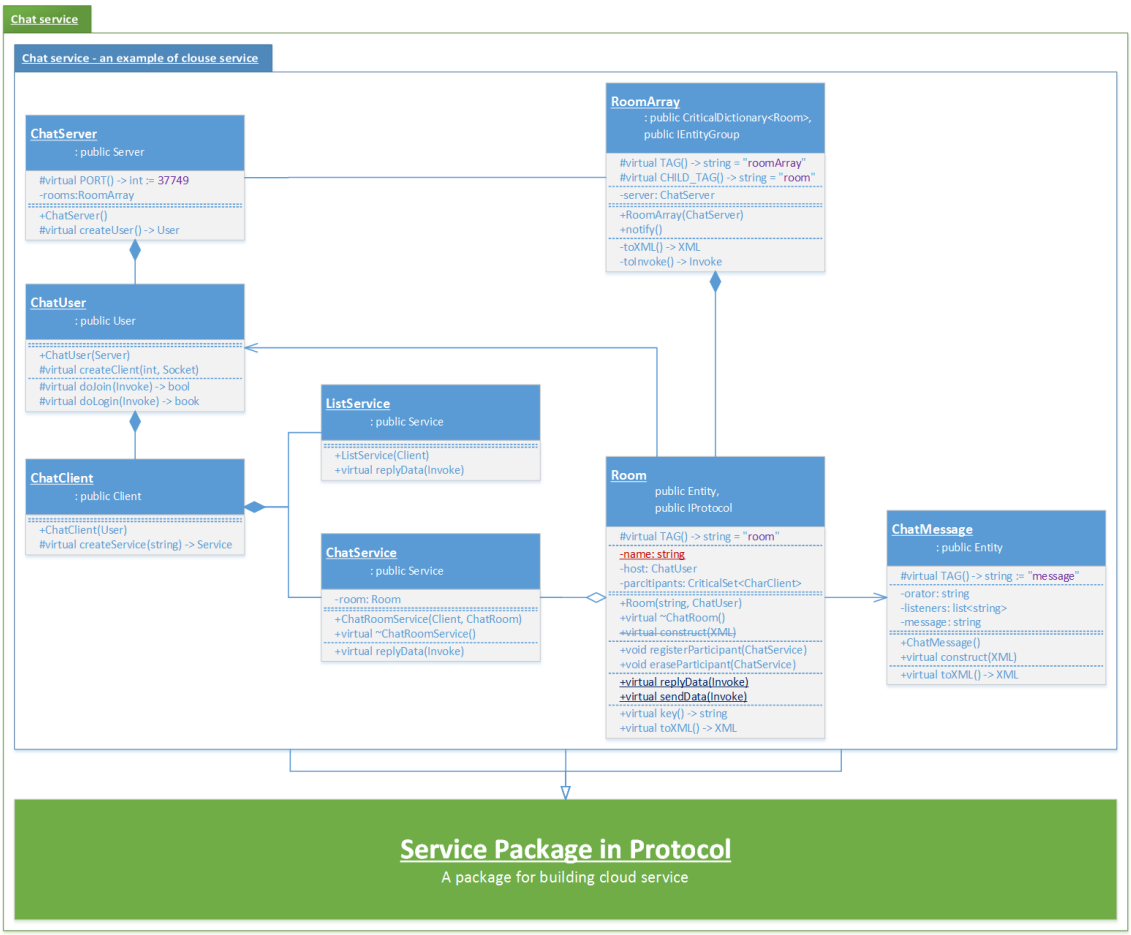
- class ListService

  A list service of chat rooms.

9.2.1 Detailed Description

A chat service inherited from cloud service.

An example for realizing cloud service and using critical allocators.



Author

Jeongho Nam

9.3 samchon::example::console_chat_server Namespace Reference

A chat server.

Classes

- class ChatClient

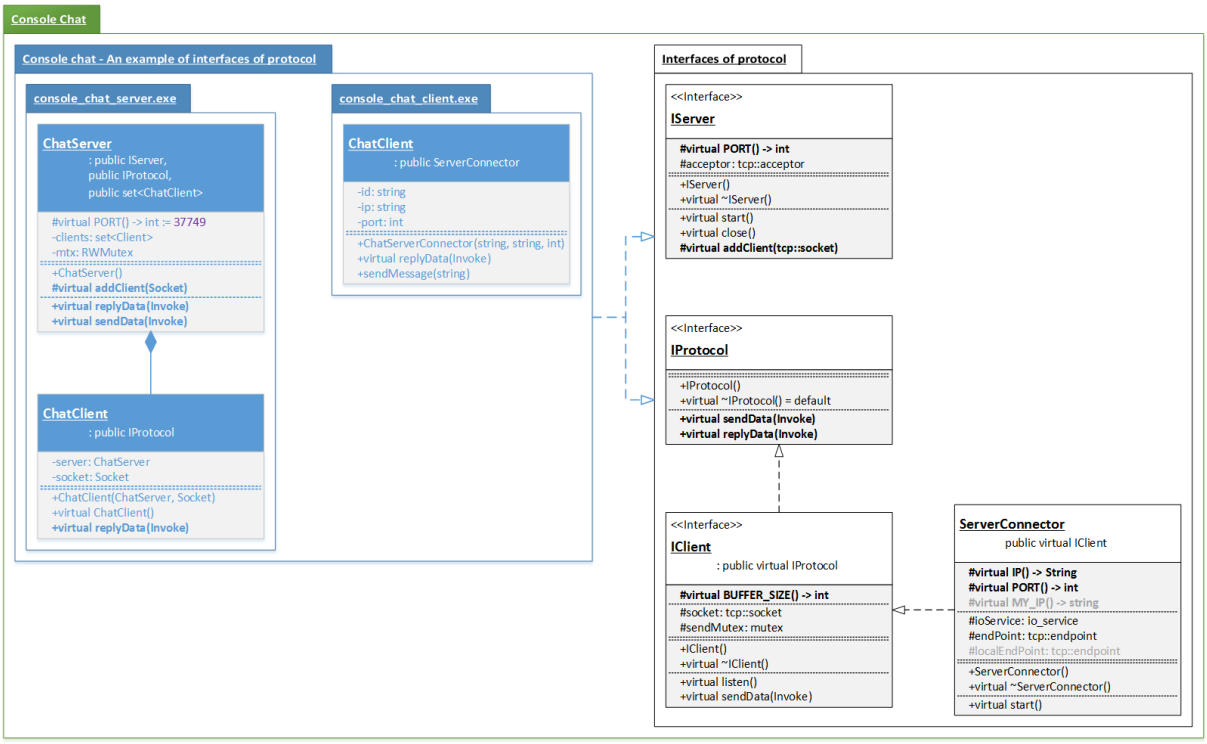
  A chat client.
- class ChatServer

  A chat server.

9.3.1 Detailed Description

A chat server.

A simple chat server operates on a console.



Author

Jeongho Nam

9.4 samchon::example::tsp Namespace Reference

A packer with number of wrapper type and products.

Classes

- struct GAParameters

  Parameters for Genetic-Algorithm.
- class GeometryPoint

  A geometry coordinates (point)
- class Scheduler

  A scheduler.
- class Travel

  Represent a travel containning Point(s)

9.4.1 Detailed Description

A packer with number of wrapper type and products.

Traveling Salesman Problem Solver.

Retrieves the solution of packaging by combination permuation and factorial case.

D:\OneDrive\Project\Samchon\framework\cpp\x64\Release\packer.exe

```
$480
Category - Large
        Wrapper Large: $100, 200cm^3, 1000g
                Product Umbrella: $4000, 200cm^3, 1000g
        Wrapper Large: $100, 200cm^3, 1000g
                Product Eraser: $500, 10cm^3, 70g
                Product Pencil: $400, 30cm^3, 35g
                Product Notebook-PC: $800000, 150cm^3, 850g

Category - Medium
        Wrapper Medium: $70, 150cm^3, 500g
                Product Drink: $1000, 75cm^3, 250g
                Product Pencil: $400, 30cm^3, 35g
                Product Pencil: $400, 30cm^3, 35g
        Wrapper Medium: $70, 150cm^3, 500g
                Product Book: $8000, 150cm^3, 300g
        Wrapper Medium: $70, 150cm^3, 500g
                Product Book: $8000, 150cm^3, 300g
        Wrapper Medium: $70, 150cm^3, 500g
                Product Tablet-PC: $600000, 120cm^3, 450g

Category - Small


Press any key to continue . . .
```

D:\OneDrive\Project\Samchon\framework\cpp\x64\Release\packer.exe

```
$480
Category - Large
        Wrapper Large: $100, 200cm^3, 1000g
                Product Umbrella: $4000, 200cm^3, 1000g
        Wrapper Large: $100, 200cm^3, 1000g
                Product Eraser: $500, 10cm^3, 70g
                Product Pencil: $400, 30cm^3, 35g
                Product Notebook-PC: $800000, 150cm^3, 850g

Category - Medium
        Wrapper Medium: $70, 150cm^3, 500g
                Product Drink: $1000, 75cm^3, 250g
                Product Pencil: $400, 30cm^3, 35g
                Product Pencil: $400, 30cm^3, 35g
        Wrapper Medium: $70, 150cm^3, 500g
                Product Book: $8000, 150cm^3, 300g
        Wrapper Medium: $70, 150cm^3, 500g
                Product Book: $8000, 150cm^3, 300g
        Wrapper Medium: $70, 150cm^3, 500g
                Product Tablet-PC: $600000, 120cm^3, 450g

Category - Small


Press any key to continue . . .
```


See also

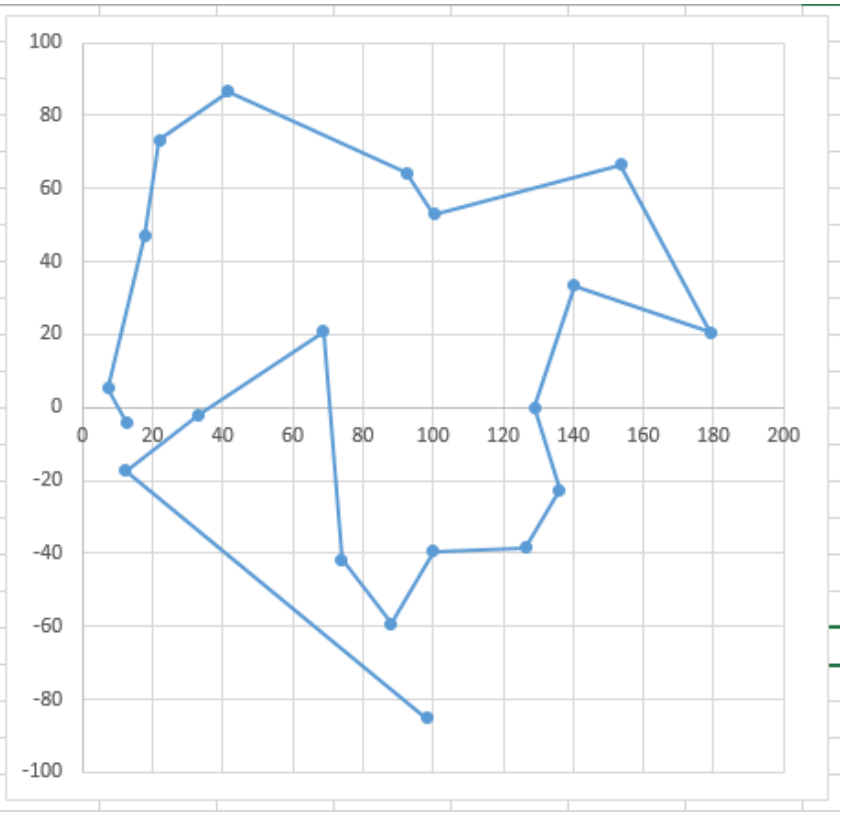
library::CaseGenerator

Author

Jeongho Nam

Deducts an optimal traveling scheduel by genetic algorithm.

Distance: 64,896.70 km		
uid	longitude	latitude
18	12.5233	-3.95529
15	7.23275	5.18342
7	17.7528	46.8665
2	22.0434	73.4133
20	41.6615	86.5372
13	92.427	64.3369
1	100.345	53.1067
9	153.521	66.5119
3	179.118	20.5073
10	140.113	33.3537
17	128.885	0.054607
4	136.012	-22.4915
6	126.352	-38.5031
16	99.9775	-39.3696
11	87.8859	-59.3663
14	73.8477	-41.5867
5	68.6563	20.8528
12	33.2162	-1.92823
8	12.1839	-17.3552
19	98.0967	-85.195



cpp/resultexample_tsp.png

Author

Jeongho Nam

9.5 samchon::library Namespace Reference

Package of libraries.

Classes

- class Base64

  Utility class for base64 format's en-decoding.
- class CaseGenerator

  Case generator.
- class Charset

  A utility class supporting conversion between multiple character-sets.
- class CombinedPermutationGenerator

  A combined-permutation case generator.
- class CriticalAllocator

  An allocator ensuring concurrency.
- class Date

  Date (year, month, day in month)
- class Datetime

Date and time.

- class ErrorEvent

  Event representing an error.
- class Event

  Represent an event running on background.
- class EventDispatcher

  Abstract class for dispatching Event.
- class FactorialGenerator

  Factorical case generator.
- class FTByteFile

  A binary file.
- class FTFactory

  A Factory and manager for file-tree.
- class FTFile

  A file.
- class FTFolder

  A folder.
- class FTInstance

  An interface of file-tree.
- class FTTextFile

  A text file.
- struct GAParameters

  A structure containing parameters of genetic algorithm.
- class GAPopulation

  A population of a generation in G.A.
- class GeneticAlgorithm

  A genetic algorithm class.
- class HTTPLoader

  A http, web-page loader.
- class IOperator

  Interface for comparision operator.
- class Math

  Utility of Math.
- class MySQLi

  A SQL interface for My-SQL.
- class PermutationGenerator

  A permutation case generator.
- class ProgressEvent

  Event representing a progress.
- class RWMutex

  rw_mutex
- class Semaphore

  A semaphore.
- class SharedAcquire

  Shared acquire from a Semaphore.
- class SharedReadLock

  Shared lock from a RWMutex.
- class SharedWriteLock

  Shared lock from a RWMutex.
- class SQLi

  A SQL interface; DBMS connector.

- class SQLStatement

  A sql statement.
- class StringUtil

  Utility class for string.
- class TSQLi

  A SQL interface for T-SQL.
- class TSQLStatement

  A t-sql statement.
- class UniqueAcquire

  Unique acquire from a Semaphore.
- class UniqueReadLock

  Unique lock for reading.
- class UniqueWriteLock

  Unique lock for writing.
- class URLVariables

  URLVariables class is for representing variables of HTTP.
- class XML

  XML is a class representing xml object.

Typedefs

- template<typename _Ty , typename _Pr = std::less<std::string>>
  using CriticalDictionary = CriticalMap< std::string, _Ty, _Pr >

  A Dictionary ensures concurrency.
- template<typename _Ty >
  using CriticalList = std::list< _Ty, CriticalAllocator< _Ty >>

  A std::list ensures concurrency.
- template<typename _Kty , typename _Ty , typename _Pr = std::less<_Kty>>
  using CriticalMap = Map< _Kty, _Ty, _Pr, CriticalAllocator< std::pair< const std::string, _Ty >>>

  A Map ensures concurrency.
- template<typename _Ty , typename _Pr = std::less<_Ty>>
  using CriticalSet = Set< _Ty, _Pr, CriticalAllocator< _Ty >>

  A Set ensures concurrency.
- template<typename _Ty >
  using CriticalVector = std::vector< _Ty, CriticalAllocator< _Ty >>

  A std::vector ensures concurrency.
- typedef std::vector< std::shared_ptr< XML > > XMLList

  A list of XML, tags are same.

9.5.1 Detailed Description

Package of libraries.

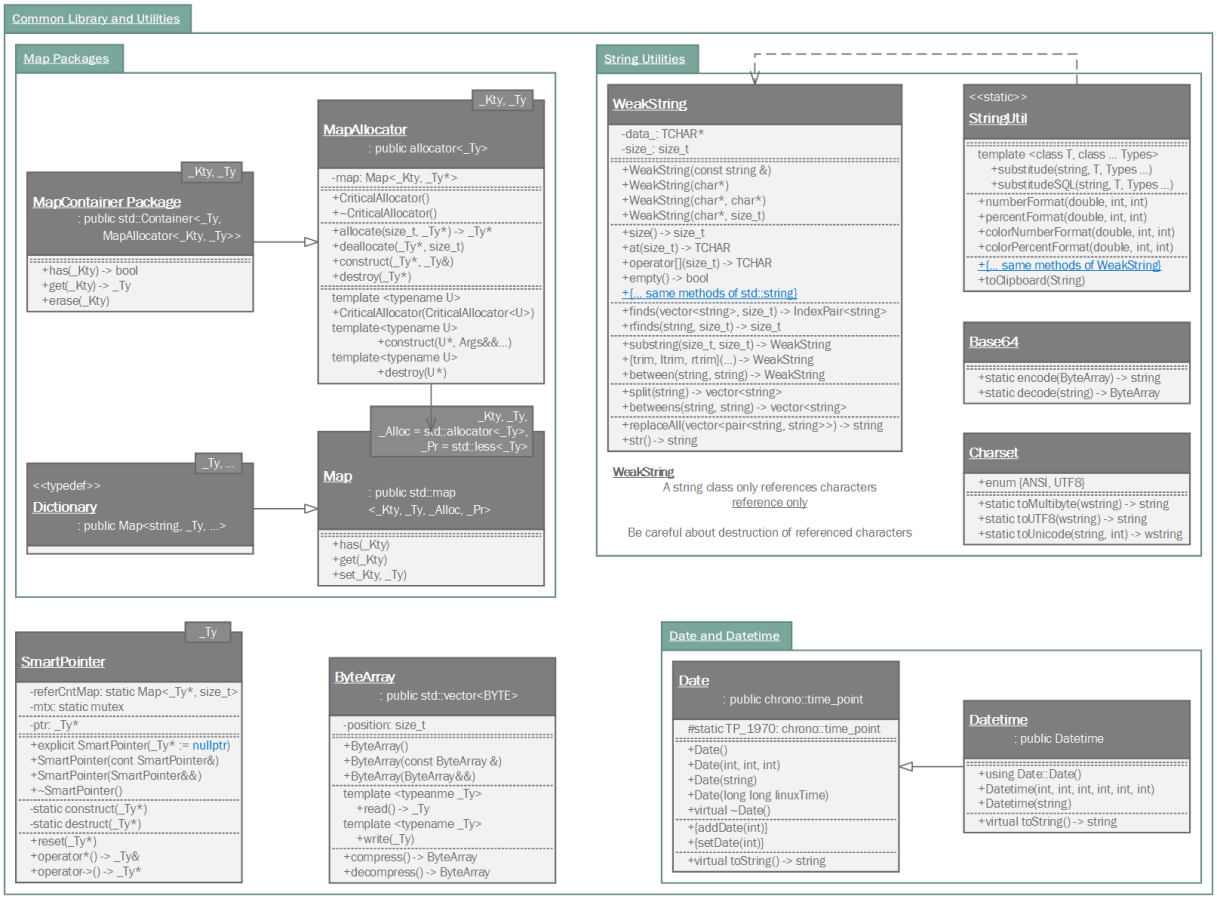
The library package supports some classes for package of protocol and nam-tree.

Furthermore, library package provides some utility libraries, which are not being supported by standard library (STL). Those can be helpful for users and those are like STL, independent from operating systems, so that can compile in any operating system.

The library package can be divided into some sectinos.

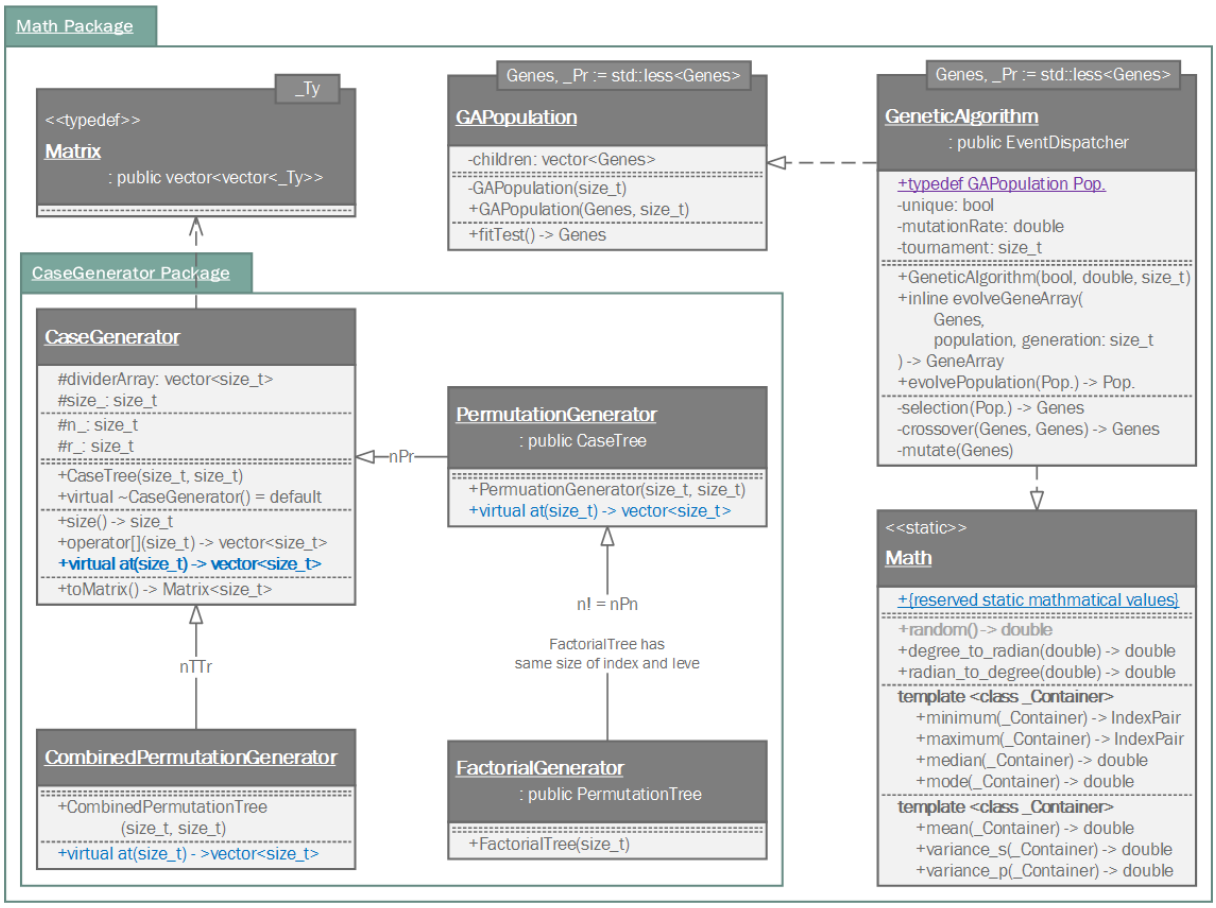
- **Common libraries.**

  Common containers with utility methods.



- **Mathmatical libaries.**

  Utilty classes for mathmatical methods, or libraries of case generator and genetic algorithm. In those mathmatical part, lots of examples are supported.

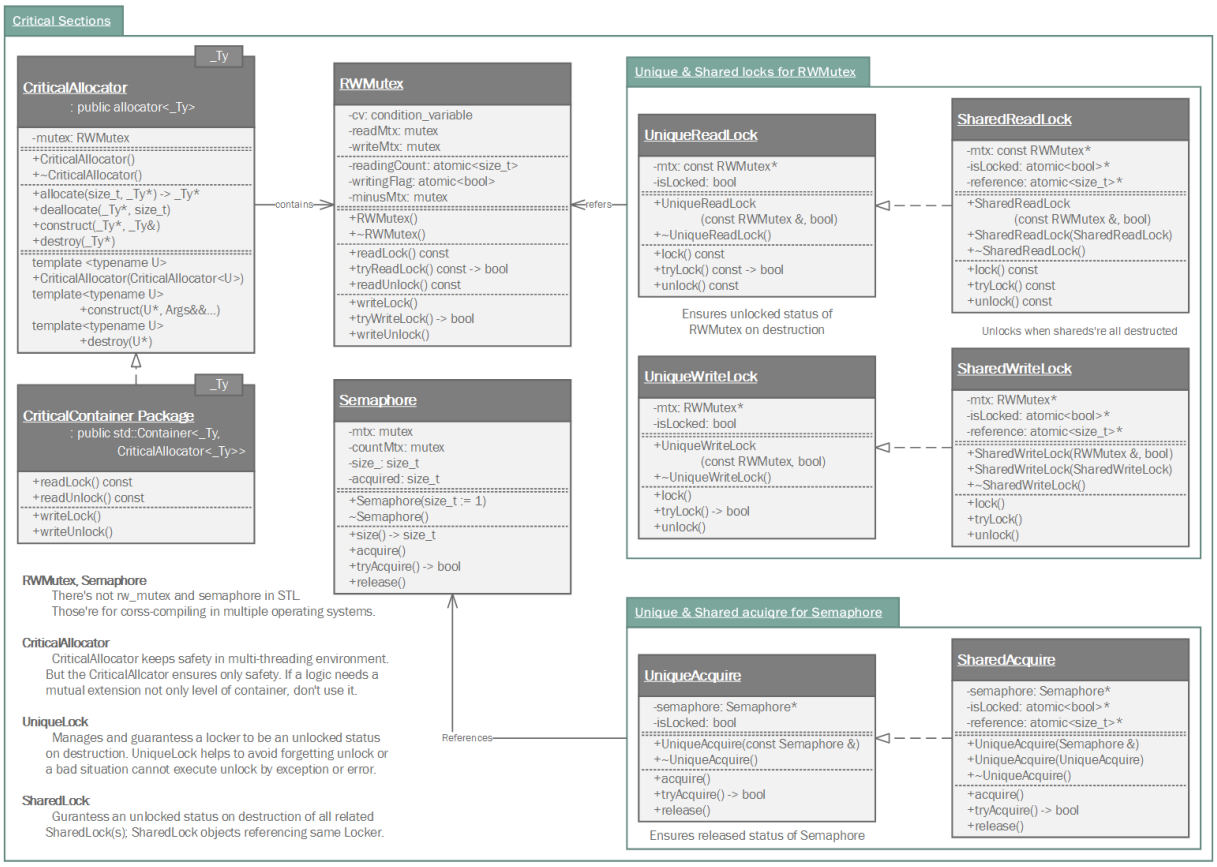


- **Critical section libraries.**

Classes handling critical section, like RWMutex and Semaphore which are not supported in STL yet.

Of course, rw_mutex is already defined in linux C, semaphore is already defined in MFC. But they are dependent on their own operating system, so that cannot compile in multiple operating systems. It's the reason why Samchon Framework provides those classes.

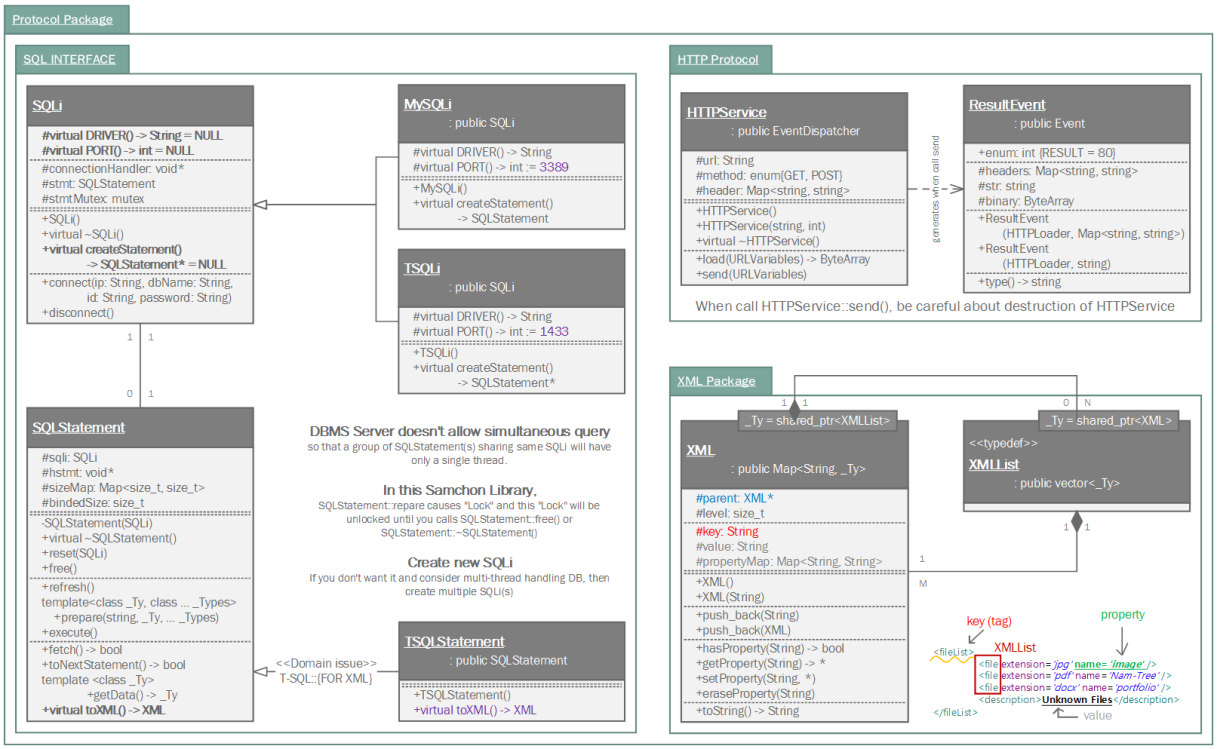
Furthermore, unique and shared lock of those critical section classes, which are managing locking and unlocking, and critical allocator are provided.



- **Data I/O libraries.**

  Provides libraries about data I/O.

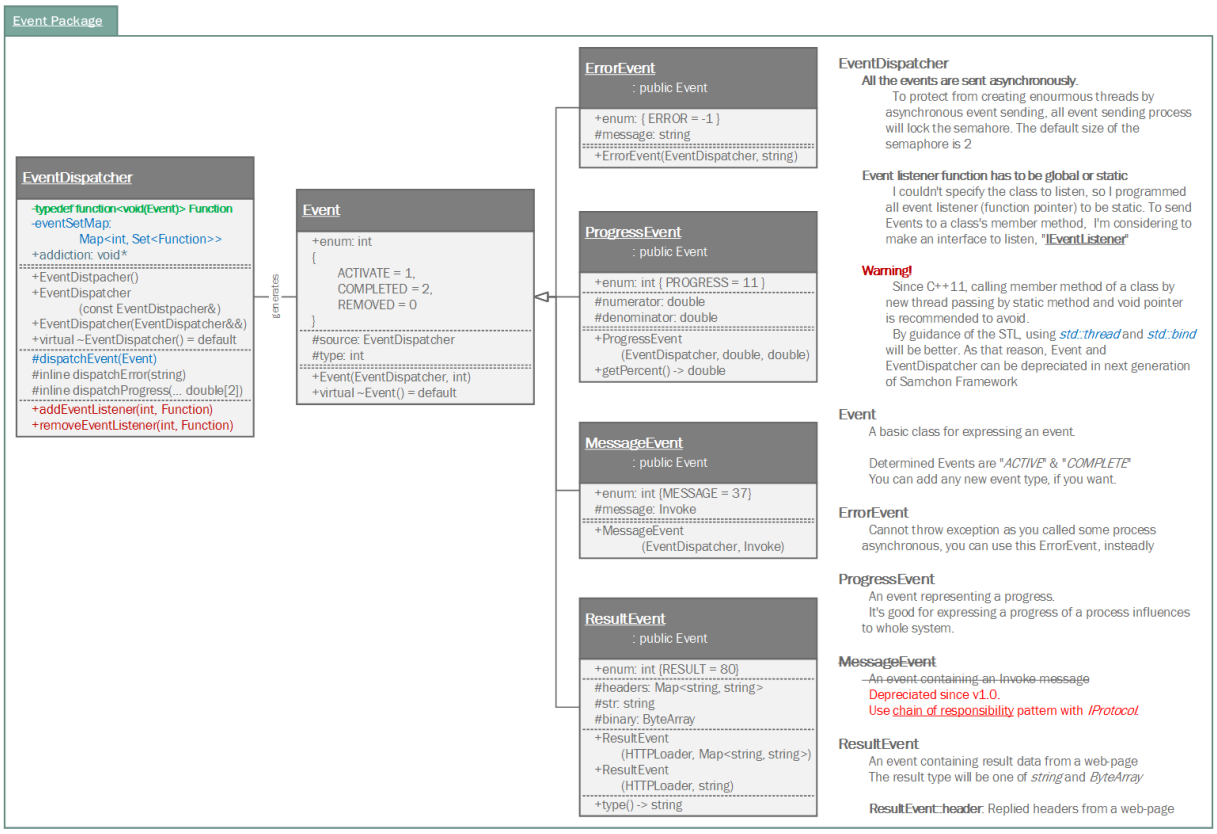
  SQLi and SQLStatement are the ODBC drivier designed to follow principles of OOP by adapter pattern. XML class is designed to follow composite relationship.



- **Event libraries.**

Libraries representing events and dispatching those events, which are running on background, with own exclusive thread.

But there's something to notice. Since C++11, calling member method of a class by new thread passing by static method and void pointer is recommended to avoid. By guidance of the STL, using std::thread and std::bind will be better. As that reason, Event and EventDispatcher can be deprecated in next generation of Samchon Framework.

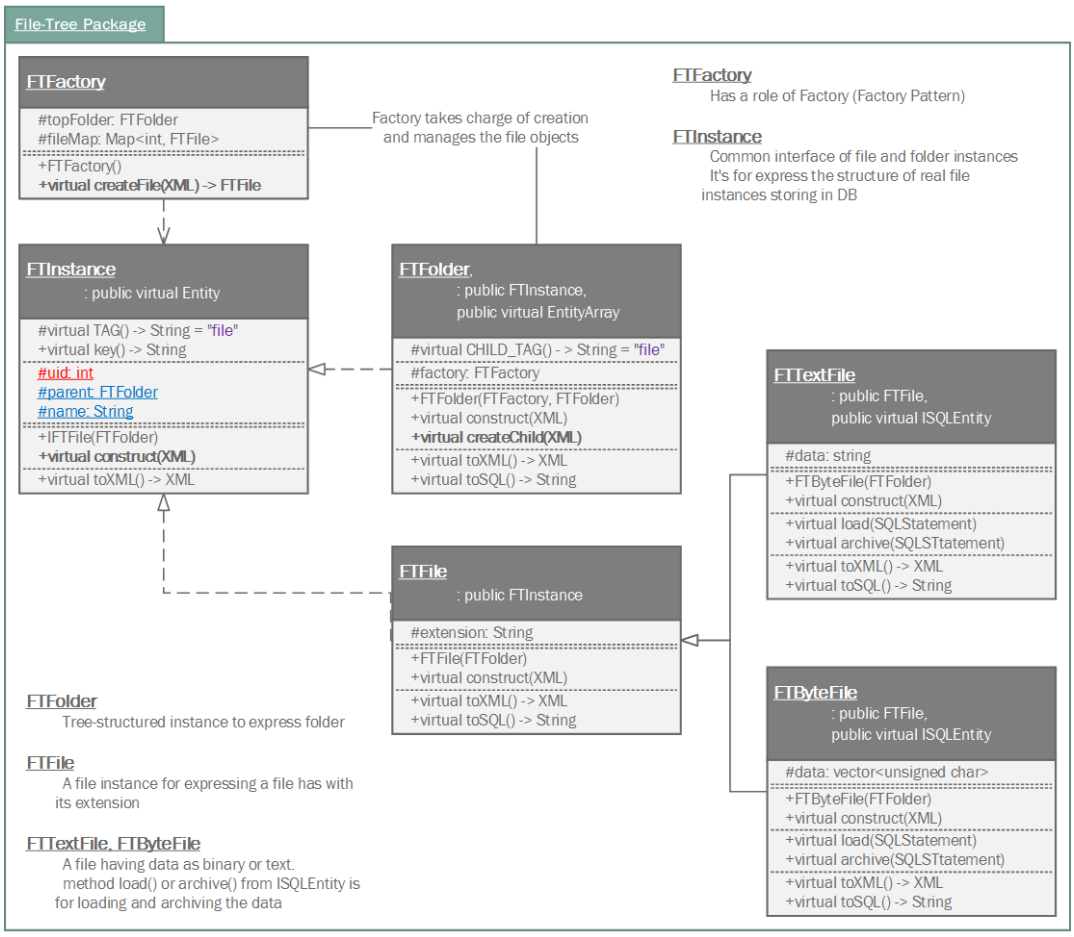


- **File-tree libraries.**

A module for expressing folder and file instances. The objects in file-tree module are realized by protocol::Entity.

They have recursive and hierarchical relationship and created by Factory class (FTFactory, factory pattern). Those objects can be archived in and loaded from Database. Using the pre-defined methods interaction with Databse, you can not only express real file and folder instances but also realize virtual file system.

Classes in the Module of file-tree are all abstract, but package nam-tree can be an example inheriting and having real model from those abstract classes.



Author

Jeongho Nam

9.5.2 Typedef Documentation

9.5.2.1 template<typename _Ty , typename _Pr = std::less<std::string>> using samchon::library::CriticalDictionary = typedef CriticalMap<std::string, _Ty, _Pr>

A Dictionary ensures concurrency.

See also

library::CriticalAllocator
samchon::library

Author

Jeongho Nam

Definition at line 16 of file CriticalDictionary.hpp.

9.5.2.2 template<typename _Ty > using samchon::library::CriticalList = typedef std::list<_Ty, CriticalAllocator<_Ty>>

A std::list ensures concurrency.

See also

library::CriticalAllocator
samchon::library

Author

Jeongho Nam

Definition at line 18 of file CriticalList.hpp.

9.5.2.3 template<typename _Kty , typename _Ty , typename _Pr = std::less<_Kty>> using samchon::library::CriticalMap = typedef Map<_Kty, _Ty, _Pr, CriticalAllocator<std::pair<const std::string, _Ty>>>

A Map ensures concurrency.

See also

library::CriticalAllocator
samchon::library

Author

Jeongho Nam

Definition at line 18 of file CriticalMap.hpp.

9.5.2.4 template<typename _Ty , typename _Pr = std::less<_Ty>> using samchon::library::CriticalSet = typedef Set<_Ty, _Pr, CriticalAllocator<_Ty>>

A Set ensures concurrency.

See also

library::CriticalAllocator
samchon::library

Author

Jeongho Nam

Definition at line 18 of file CriticalSet.hpp.

9.5.2.5 template<typename _Ty > using samchon::library::CriticalVector = typedef std::vector<_Ty, CriticalAllocator<_Ty>>

A std::vector ensures concurrency.

See also

library::CriticalAllocator
samchon::library

Author

Jeongho Nam

Definition at line 18 of file CriticalVector.hpp.

9.5.2.6 typedef std::vector<std::shared_ptr<XML> > samchon::library::XMLList

A list of XML, tags are same.

See also

samchon::library

Author

Jeongho Nam

Definition at line 19 of file XMLList.hpp.

9.6 samchon::namtree Namespace Reference

Package of Nam-Tree.

Classes

- class INTExplore

  An interface for exploration.
- class NTCriteria

  Criteria, a conditional expression with weight.
- class NTEntityGroup

  A historical, studying data.
- class NTFactory

  A factory for Nam-Tree objects.
- class NTFile

  A file archiving metadata of a function.
- class NTIterator

  Iterator of historical data.
- class NTParameter

  A metadata of a parameter in a function.
- class NTParameterArray

  An Array of NTParameter.
- class NTParameterDetermined

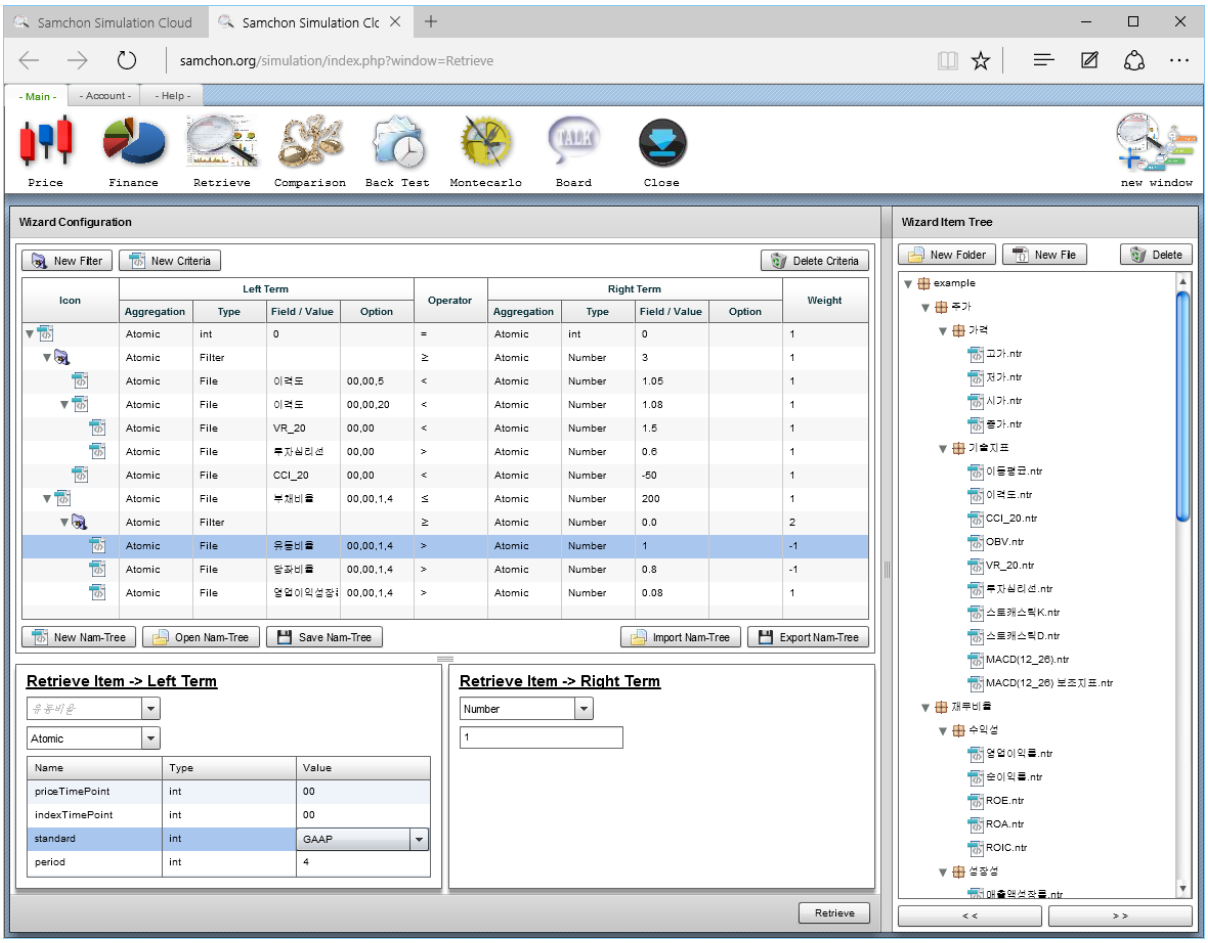
  A pre-determined value of a parameter.
- class NTSide

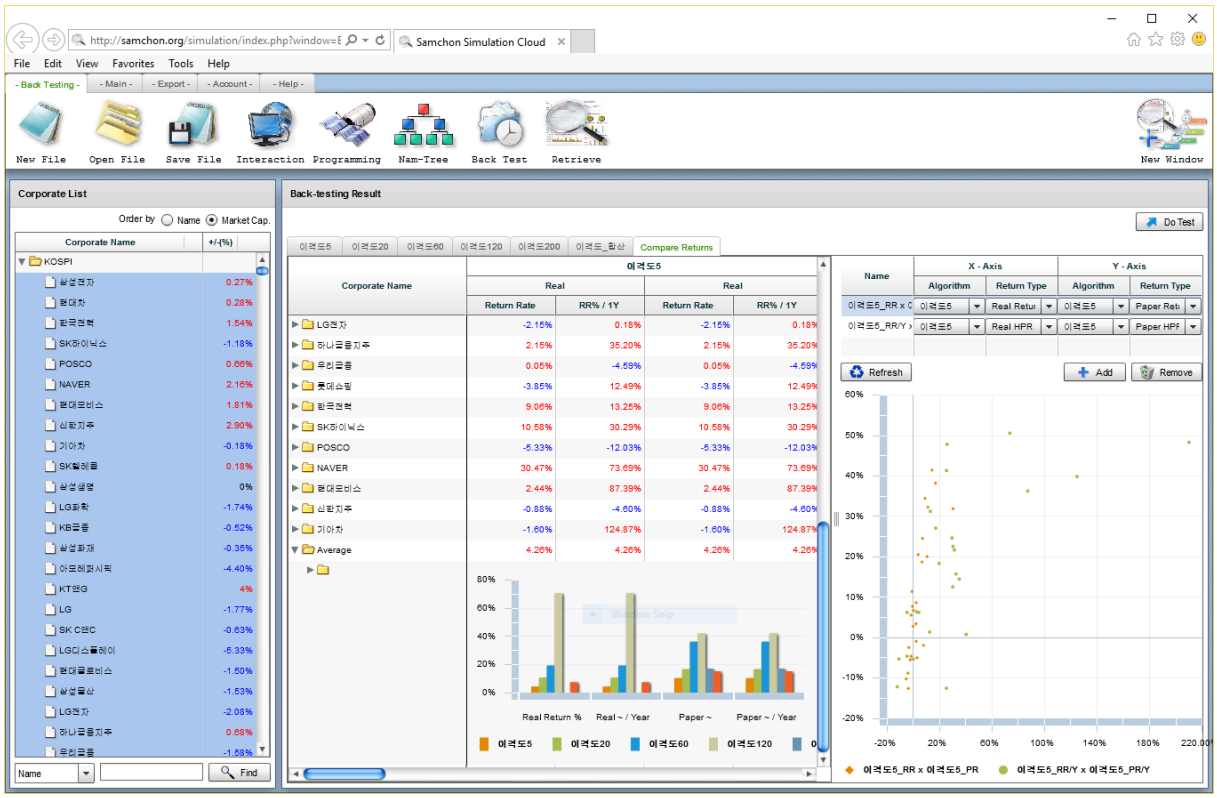
  A side of a conditional expresson.

9.6.1 Detailed Description

Package of Nam-Tree.

Nam-Tree is a module of tree-strucrued complicate logical condition model for realizing A.N.N. (Artificial Neural Network). You can make the ANN model with weight and bias and express not only logical condition by your hand, but also construct the logical condition by automaticaly with optimization of genetic algorithm and grid method.





The basic pricinple of Nam-Tree are follow:

1. A conditional expression with weight

   F(x) = (NTSide < NTSide ? 1: 0) X weight

   NTCrietria is made up for conditional expression.

   - When the expression is true, returns 1 ∗ weight
   - When the expression is false, returns 0

2. Hierarchical relationship

   - In vertical relationship: multiply (X)
   - In horizontal relationship: plus (+)

   With that rule, you can make enormous conditions. I can sure there's not any condition can'be be expressed by the model.

3. Making bias

   Just make a NTCriteria returns only true.

   It's the bias returns only weight.

4. Explore

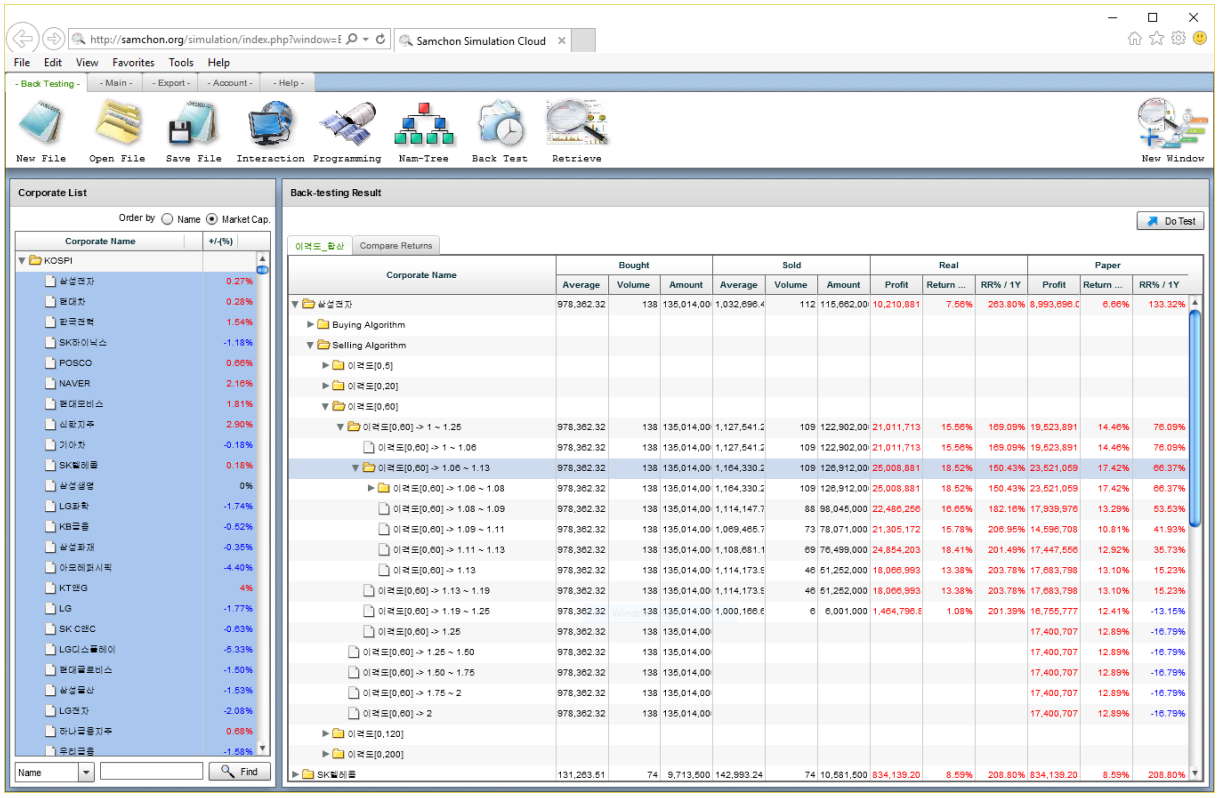
   (a) Exporing in a NTCriteria (optimize a side)

   Make a NTSide to be nullptr, then NTCriteria will explore the best value.

   Nam-Tree will calculate the condtions from minimum to maximum in INTExplore reach to the precision, by the method of multi-demensional grid.

   (b) Exploring parameter in NTSide

   If you set the parameterMNap to be empty, Nam-Tree will explore the best parameter until reach to the precision in INTExploreParameter from minimum to maximum.

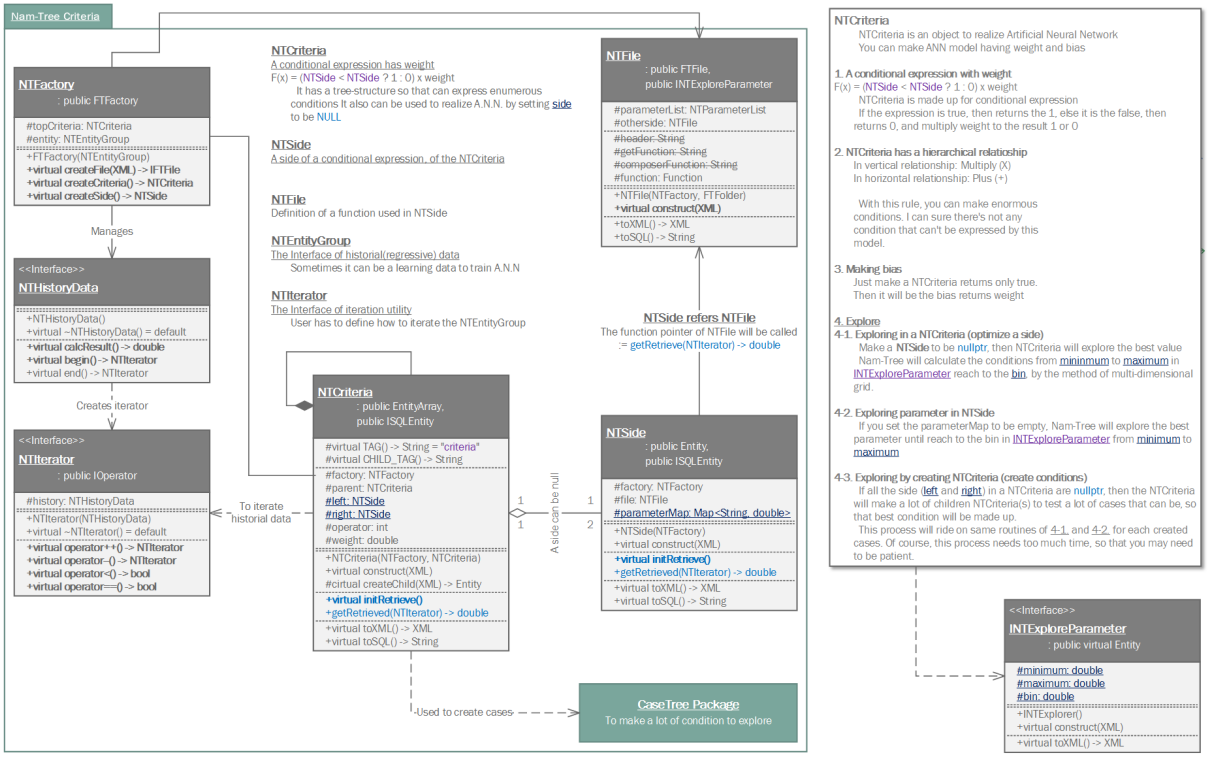


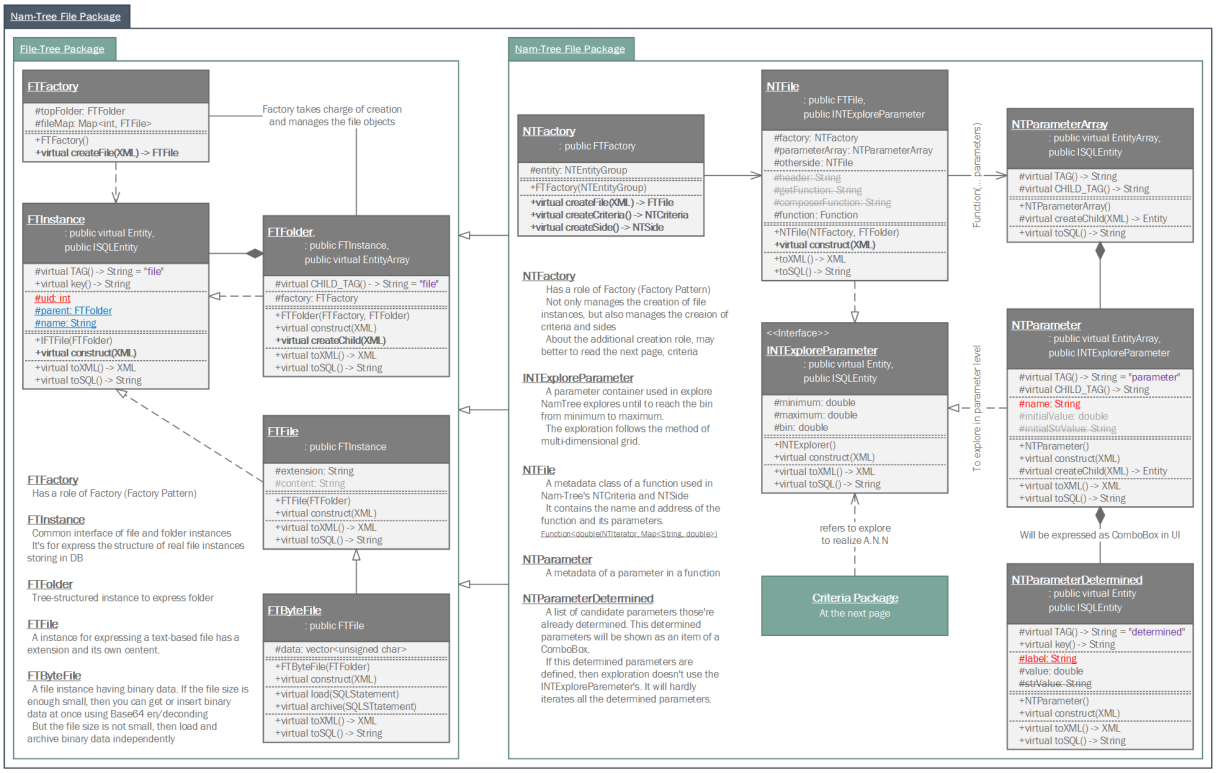
   (c) Exploring by creating NTCriteria

   (Creates lots of hierarchical conditions and tests them)

If all the side (left and right) in a NTCriteria are nullptr, the NTCriteria will make a lot of children NT↩ Criteria(s) to test lots of cases, so that derives the best condition set will be made up.

This process will ride on same routines of 4-1 and 4-2, for each created cases. Of course, this process needs too much time, so that you may need to be patient.





Note

Previous version of the Samchon Framework, NTFile class had a function script on the NTFile and Nam-Tree module compiled the script like Nam-Tree moudle of Flex standalone that is keeping the compiling method.

However, modern Samchon Framework's C++ NTFile class doesn't have a function script and does not compile. I don't know a way to compile a script in lots of operating systems, so I dropped the compiling method. Until update, NTFactory has function pointers in a Dictionary and NTFile reference them. Users must put own methods into the Dictionary by their hands.

I will solve the problem sooon. Until next generation of Samchon Framework, put your own function pointers to a Dictionary in NTFactory by your hand please. Sorry for my unripe skill on programming.

Author

Jeongho Nam

9.7 samchon::protocol Namespace Reference

Package of network protocol and libraries.

Namespaces

- master

  Package for external system, within the framework of master.
- service

  Package of cloud service as a server.
- slave

  Package of external system, within the framework of slave.

Classes

- class Entity

  An entity, a standard data class.
- class EntityArray

  An Entity and a static array containing Entity objects.
- class EntityGroup

  An Entity and a container of children Entity objects.
- class EntityList

  An Entity and a static list containing Entity objects.
- class ExternalClient

  A network driver for an external client.
- class ExternalClientArray

  An array of ExternalClient(s)
- class ExternalServer

  A network driver for an external server.
- class ExternalServerArray

  An array of ExternalServer(s)
- class ExternalSystem

  A network driver for an external system.
- class ExternalSystemArray

An array of ExternalSystem(s).

- class ExternalSystemRole

  A role belongs to an external system.
- class FlashPolicyServer

  A flash policy server.
- class IClient

  An interface for a client.
- class IEntityChain

  A chain of entity.
- class IEntityGroup

  An iternface for entity group.
- class IEntityPtrGroup
- class IHTMLEntity

  An interface supporting conversion to html.
- class Invoke

  Standard message of network I/O.
- class InvokeHistory

  A history of an Invoke message.
- class InvokeHistoryArray

  An array of InvokeHistory.
- class InvokeParameter

  A parameter of an Invoke.
- class IProtocol

  An interface of Invoke message chain.
- class IServer

  An interface of a physical server.
- class ISQLEntity

  An interface supporting DB-I/O.
- class IWebClient

  An interface for a web-client.
- class IWebServer

  An interface for a physical server following web-socket.
- class ServerConnector

  A server connector for a physical client.
- class SystemRole

  A role belongs to a system.
- class WebServerConnector

  A web-socket server connector.

Typedefs

- template<typename _Ty = Entity>
  using SharedEntityArray = EntityGroup< std::vector< std::shared_ptr< _Ty >>, _Ty, std::shared_ptr< _Ty > >

  An EntityGroup with vector container and children capsuled in shared pointers.
- template<typename _Ty = Entity>
  using SharedEntityList = EntityGroup< std::list< std::shared_ptr< _Ty >>, _Ty, std::shared_ptr< _Ty > >

  An EntityGroup with list container and children capsuled in shared pointers.
- template<typename _Ty = Entity>
  using UniqueEntityArray = EntityGroup< std::vector< std::unique_ptr< _Ty >>, _Ty, std::unique_ptr< _Ty > >

An EntityGroup with vector container and children capsuled in unique pointers.

- template<typename _Ty = Entity>
  using UniqueEntityList = EntityGroup< std::list< std::unique_ptr< _Ty >>, _Ty, std::unique_ptr< _Ty > >

  An EntityGroup with list container and children capsuled in unique pointers.

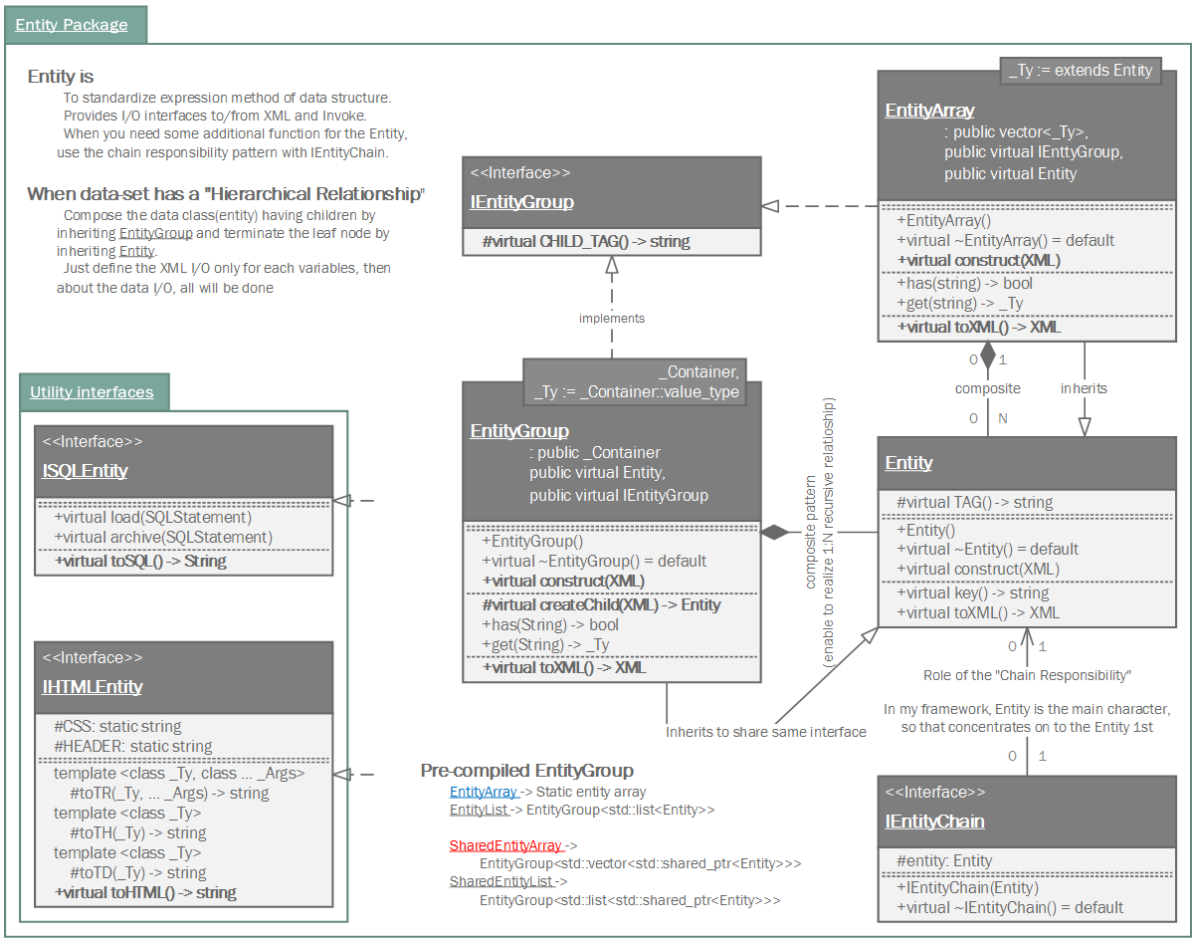
9.7.1 Detailed Description

Package of network protocol and libraries.

- **Entity - data classes with utility methods**

  In Samchon Framework, with entity, boundary and control, entity is the main. The entity package provides I/O interface to/from XML, Invoke and DBMS. If you need some additional function (role of boundary or control) for an entity, use chain of responsibility pattern starting from the entity.

  When data-set has a "hierarchical relationship", do not have the children objects by having a container member variable. Directly inherits container set. Composes the data class (entity) having children by inheriting EntityGroup and terminate the leaf node by inheriting Entity (single object).



- **Invoke - standard network I/O message**

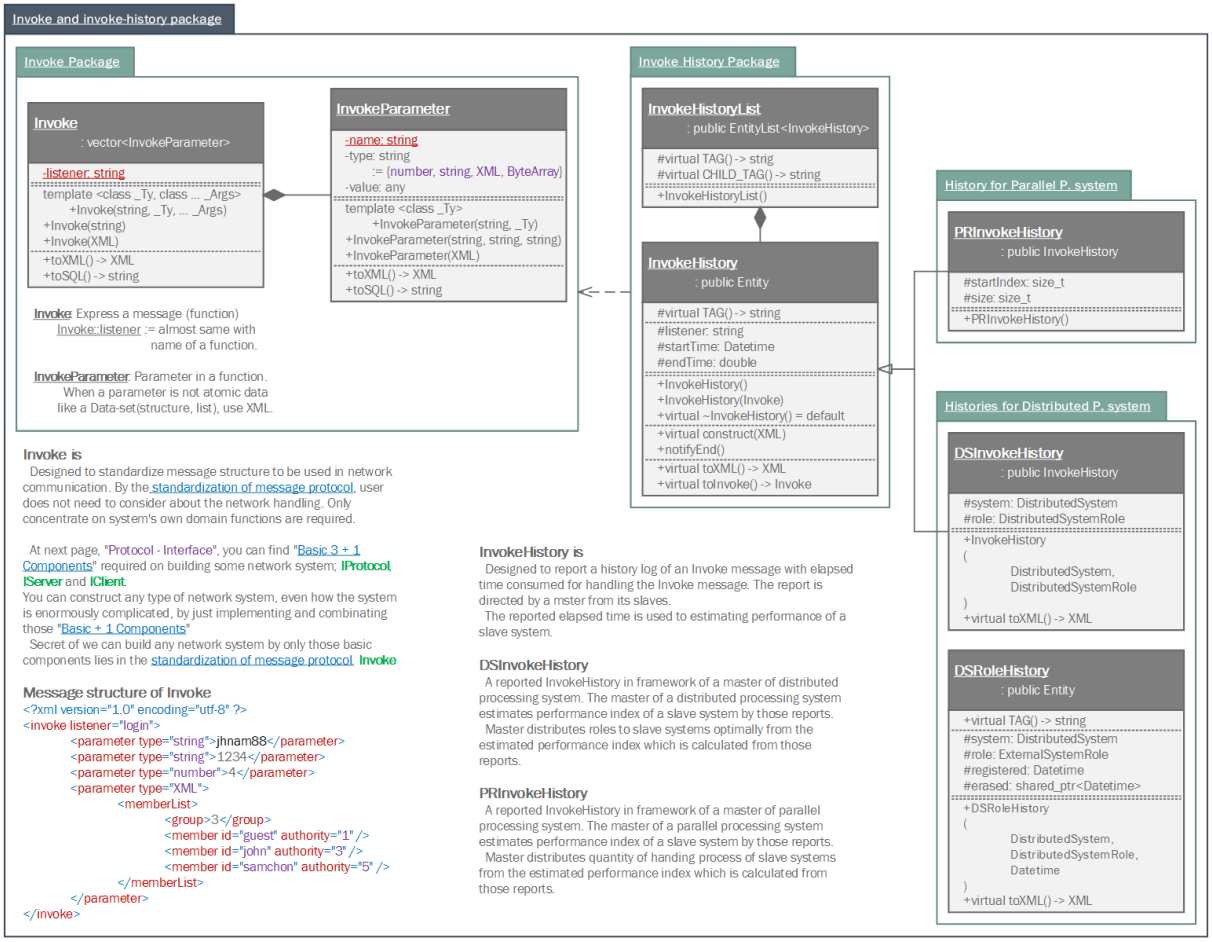
  Invoke is a standard message structure using network I/O in Samchon Framework.

  The Invoke message has an XML structure like the piacture in below. We can enjoy lots of benefits by the normalized and standardized message structure used in network I/O.

  The greatest advantage is that we can make any type of network system, even how the system is enourmously complicated. As network communication message is standardized, we only need to concentrate on logical

relationships between network systems. We can handle each network system like a object (class) in OOD. And those relationships can be easily designed by using design pattern.

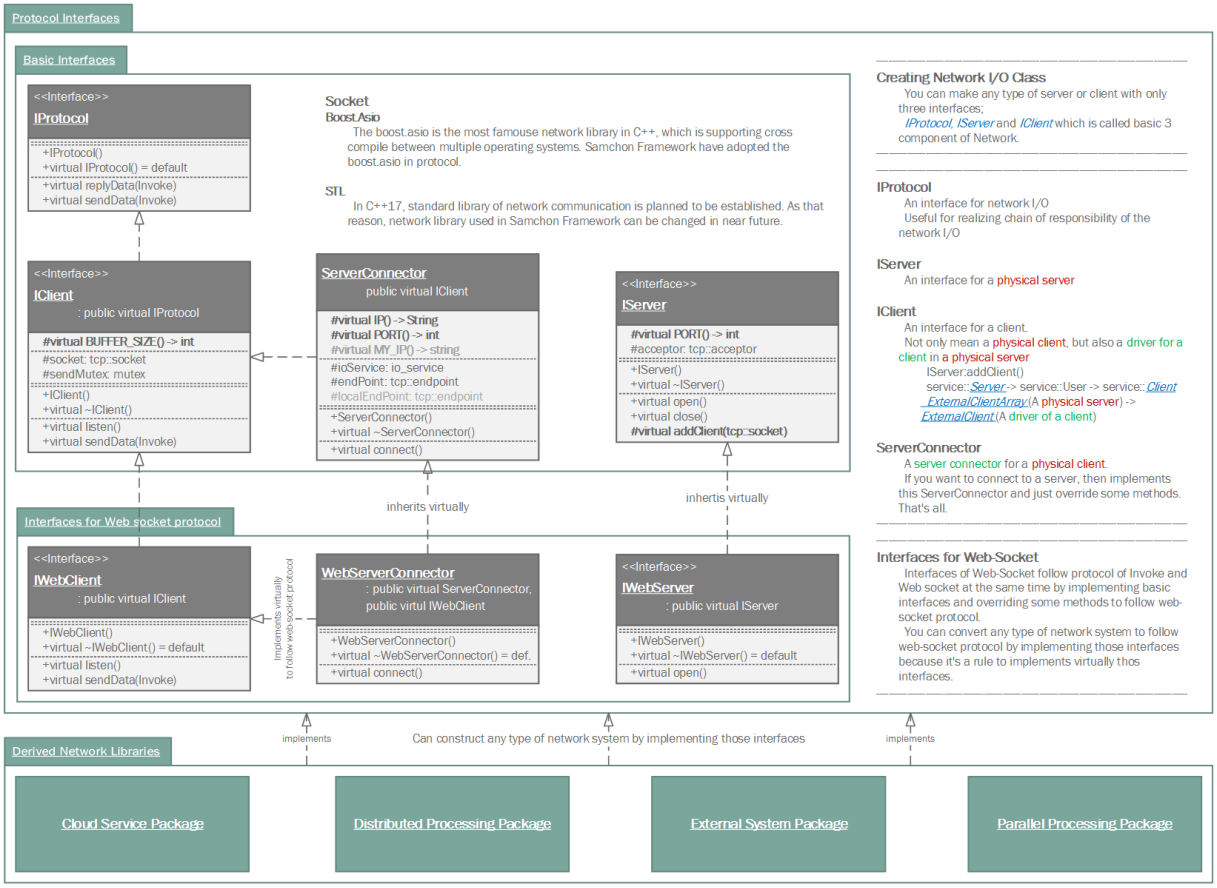
Furthermore, The module invoke not only provides parsing and converting from/to Invoke message, but also history classes can be used to archiving log or estimating performance of a system. The purpose of estimating performance a system, especially, used in module of distributed processing and parallel processing systems.

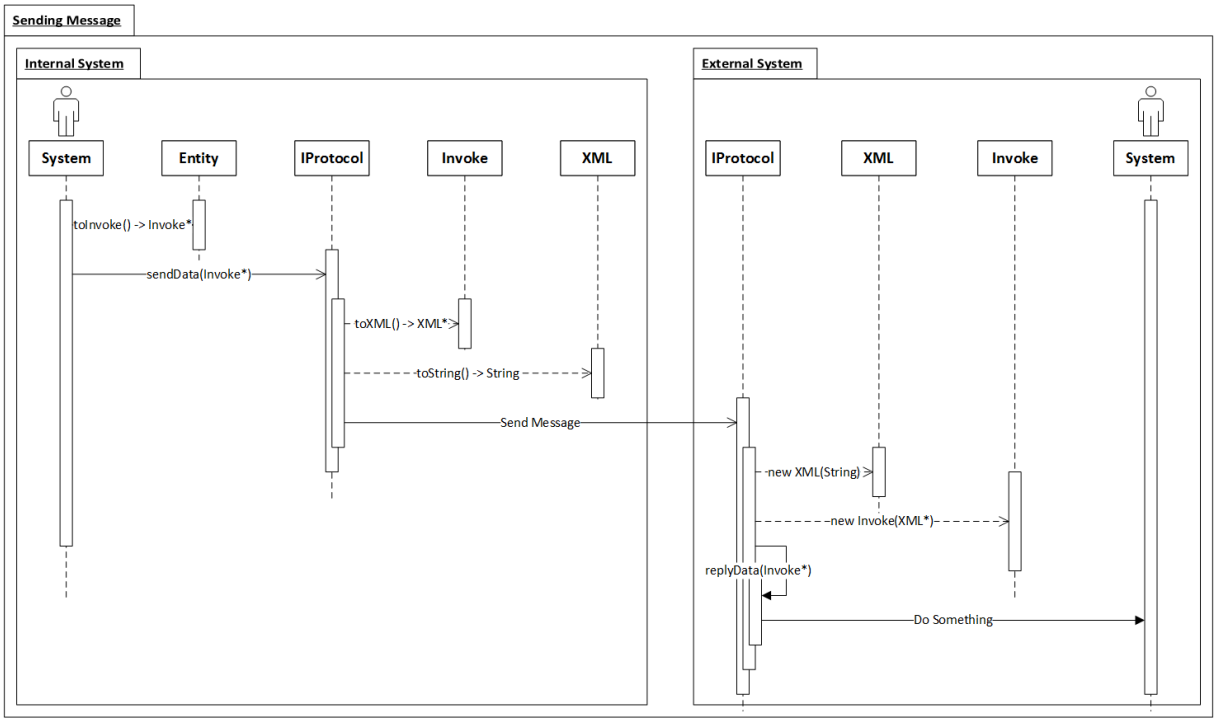


- **Intefaces**

You can make any type of network system with only three + one interfaces; IProtocol, IServer and IClient + ServerConnector which are called basic 3 + 1 components. As I've mentiond in previous invoke module, because of standardization of message of network I/O, we only need to concentrate on logical relationship like handling OOD objects

The basic 3 + 1 components are top level abstract interfaces for designing network system like handling OOD objects. When you see the protocol module from a different view, out of framework, you can find that all the other classes in protocol are another examples combinationing and utilizing those basic 3 + 1 components (interaces).





- **External System**

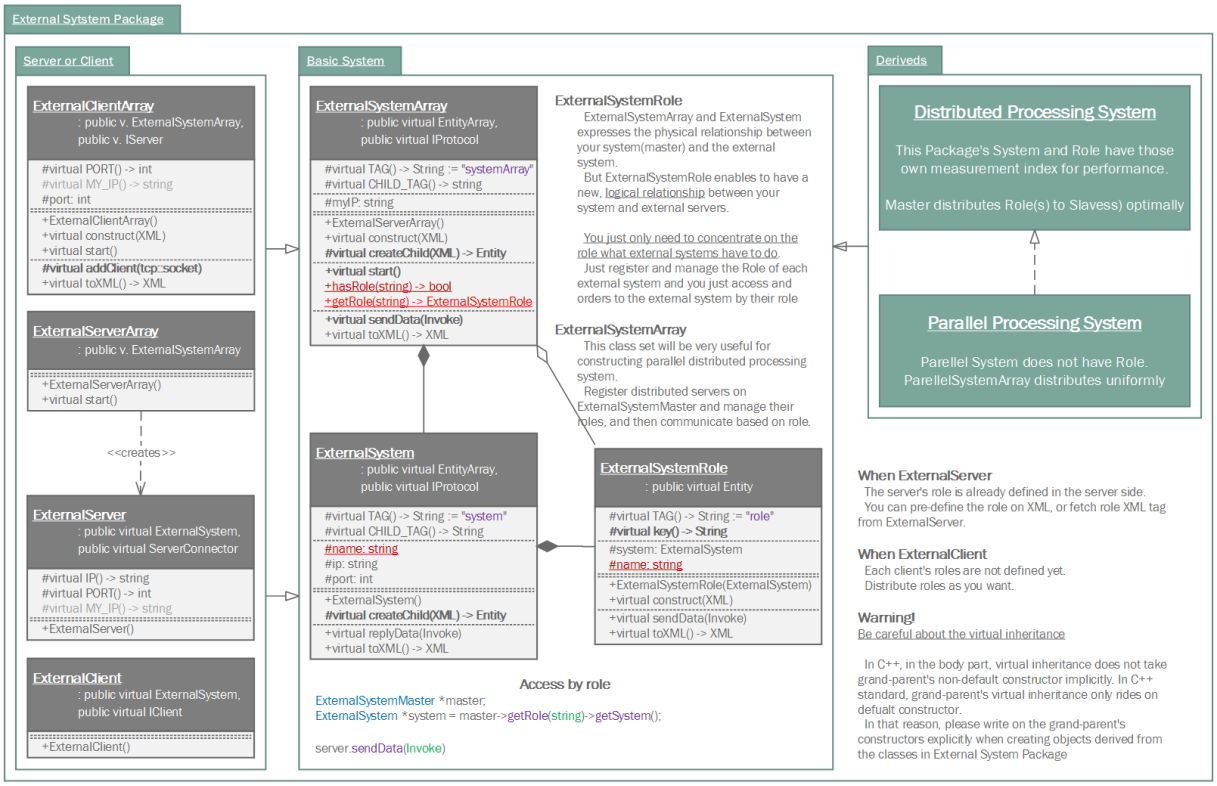
  Module *external_system* provides interfaces for interaction with external network system. Although, the module *external_system* acts boundary as main role, what you've to concentrate on is the entity. Samchon Framework takes responsibility of network communication and you only consider about relationship and role of each *external network systems*.

ExternalSystem objects are managed by ExternalSystemArray and the ExternalSystemArray can access to an ExternalSystemRole belongs to an ExternalSystem directly. When you send an Invoke message to ExternalSystemArray, the ExternalSystemArray finds matched ExternalSystemRole and the External↩ SystemRole shifts the network I/O responsibility to belonged ExternalSystem.

The relationship called as "Proxy Pattern". With the pattern, "Proxy", you can concentrate on roles irrespective of where each role is belonged to (you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem).

- ExternalSystemArray::sendData() -> ExternalSystemRole(Proxy)::sendData() -> ExternalSystem↩ ::sendData()
- ExternalSystem::replyData() -> ExternalSystemRole(Proxy)::replyData()

Whether using the "Proxy pattern" is on your mind in *external_system module* level. "Proxy pattern" is recommend to use in *external_system* module, but not forced. However, since *parallel_processing_system* module, you've to follow the pattern.

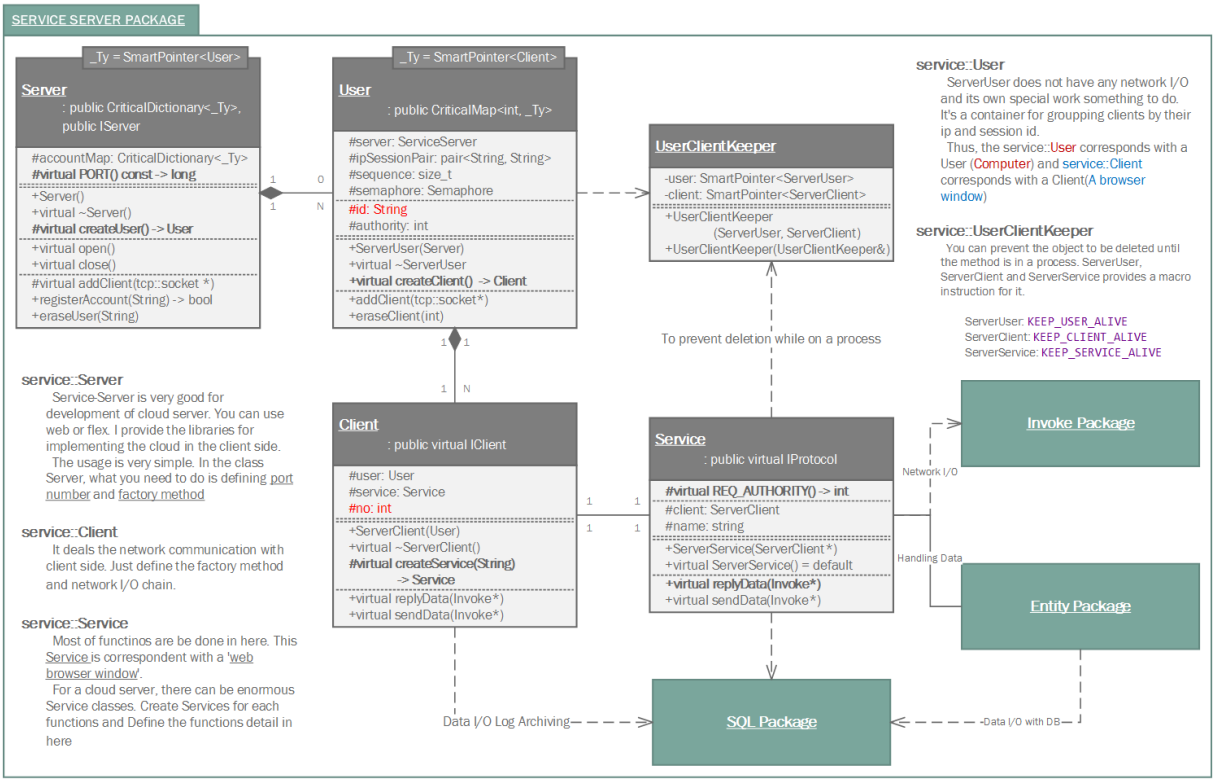


- **Packages in protocol**

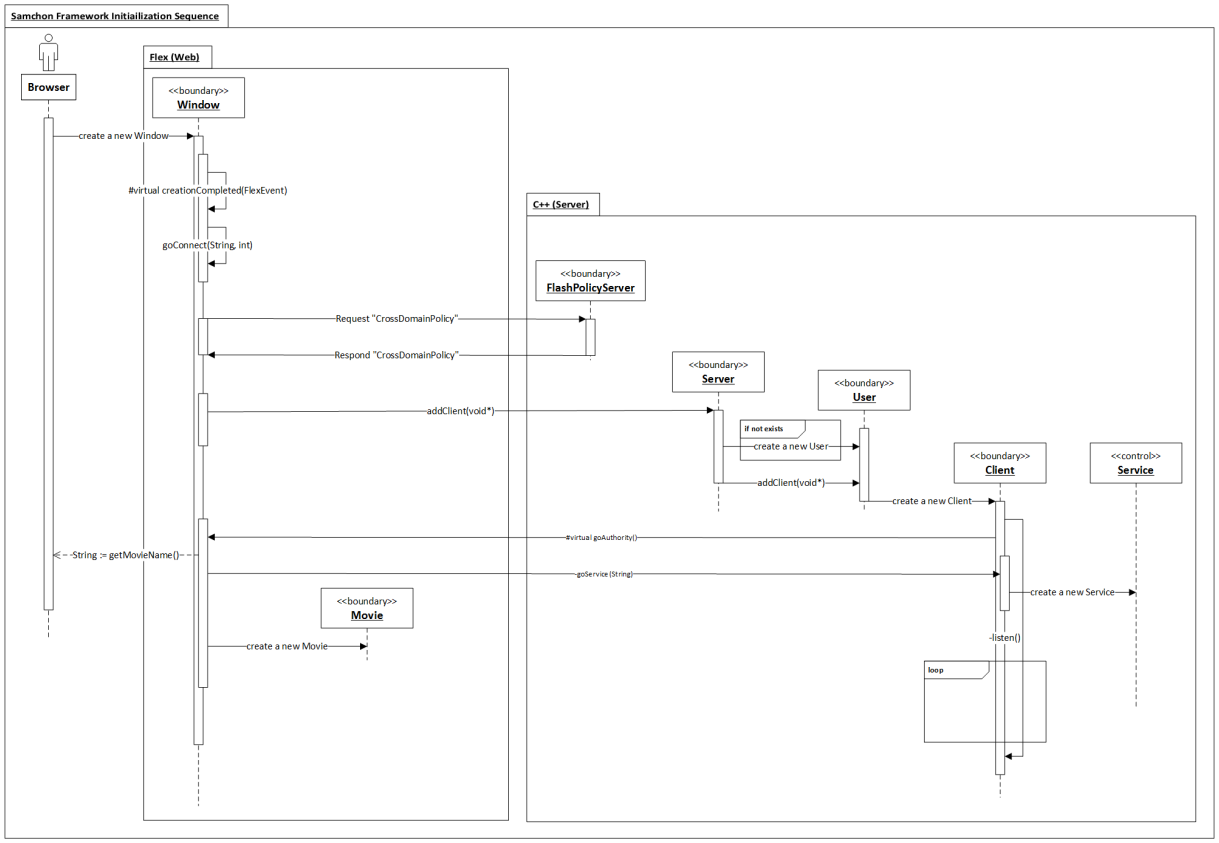
Cloud service

The "service" package is for realizing cloud service. In Samchon Framework, realizing cloud service, there're two classes representing physical server and client as service::Server and service::Client. Between the two classes representing boundaries, there is a class representing logical conception user as service::User. In the terminal node under the service::Client, service::Service class represents a control. The default protocol of the service package is not 'web socket'. The default only has message protocol of Invoke, does not have handshake or another pre-defined protocol. By the default protocol, you can connect to the cloud server (built from c++, by service package) as a client by a program or plug-in like Flex or C#. However you can't connect by web-browser only allowing socket protocol as web-socket. To build a cloud service to follow web-socket protocol, inherits derviced classes not from service::Server, service::Client, but from service::WebServer, service::WebClient or implements IWebServer, IWebClient which are derived from service::Server and service::Client. **Class Diagram**

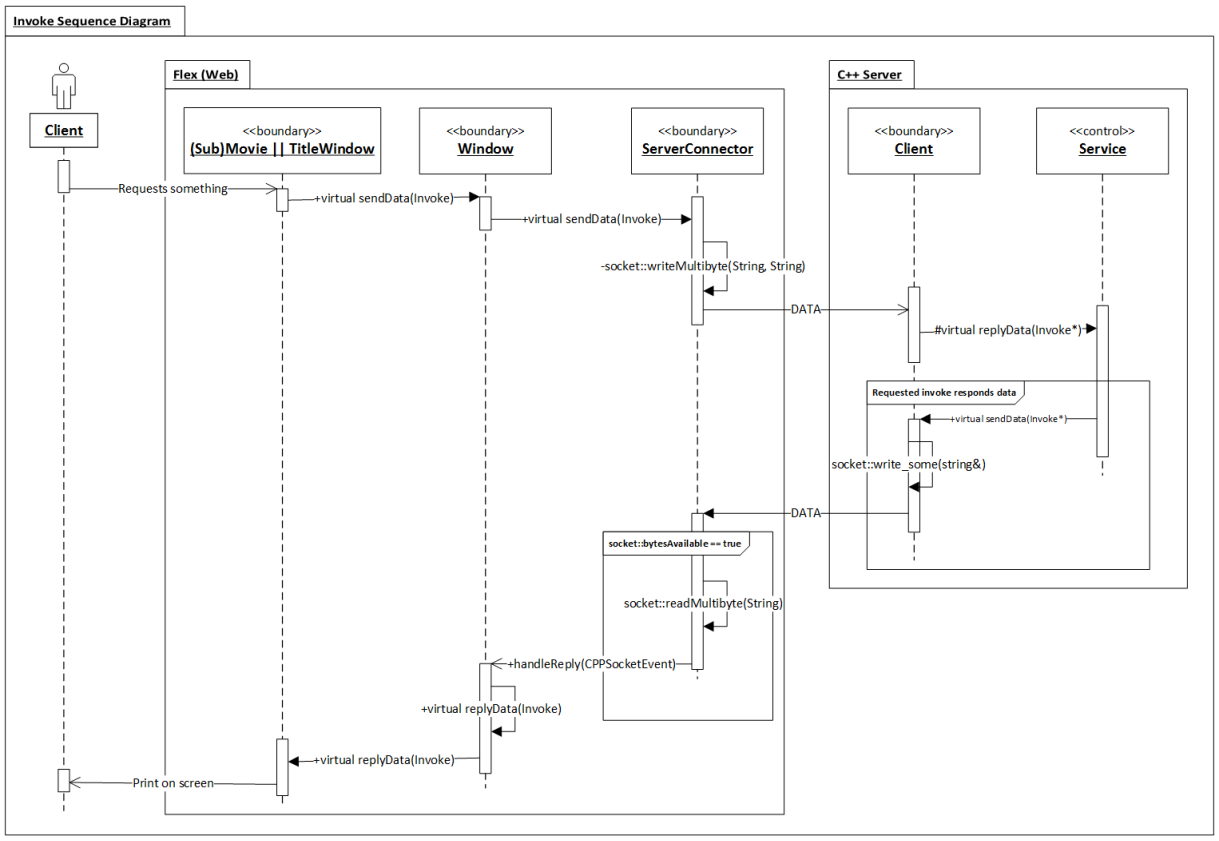
–



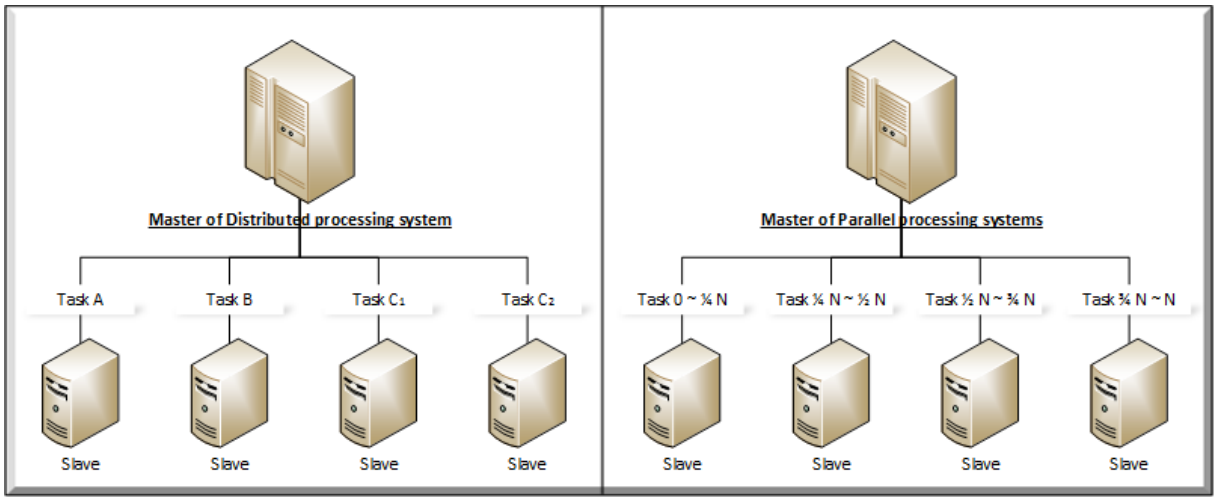
Sequence Diagram - A client's connection



Sequence Diagram - Network communication



– Master system



In master package, provides distributed and parallel processing system modules which are in framework of master. With classes in master package, you can realize any type of distributed or parallel processing system in master side, even how enormouse scale those have.

Distributed Processing System

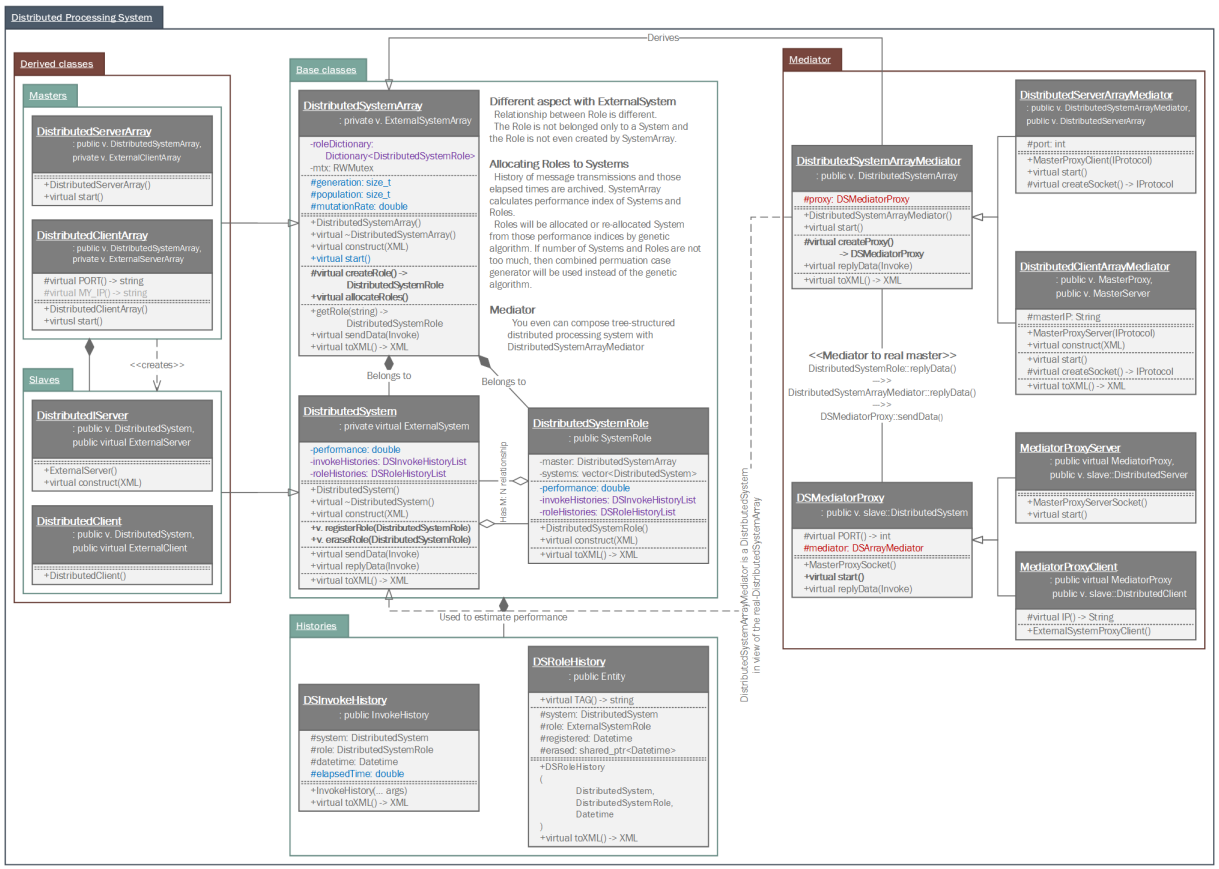
A distributed processing system of master, a master (master::DistributedSystemArray) allocates roles (master::DistributedSystemRole) to systems (master::DistributedSystem). User does not consider about which roles are allocated to which systems. User just accesses to a *role* and send or reply message from those roles.

* It's the proxy pattern have mentioned in external system
  ExternalSystem objects are managed by ExternalSystemArray and the ExternalSystemArray can access to a role belongs to a system directly. When you send an Invoke message to External↩SystemArray, the ExternalSystemArray finds matched ExternalSystemRole and the External↩SystemRole shifts the network I/O responsibility to belonged ExternalSystem. This relationship called "Proxy Pattern". By the pattern, "Proxy", you can concentrate on roles irrespective of where each role is belonged to.

  · ExternalSystemArray::sendData() -> ExternalSystemRole(Proxy)::sendData() -> External↩System::sendData()

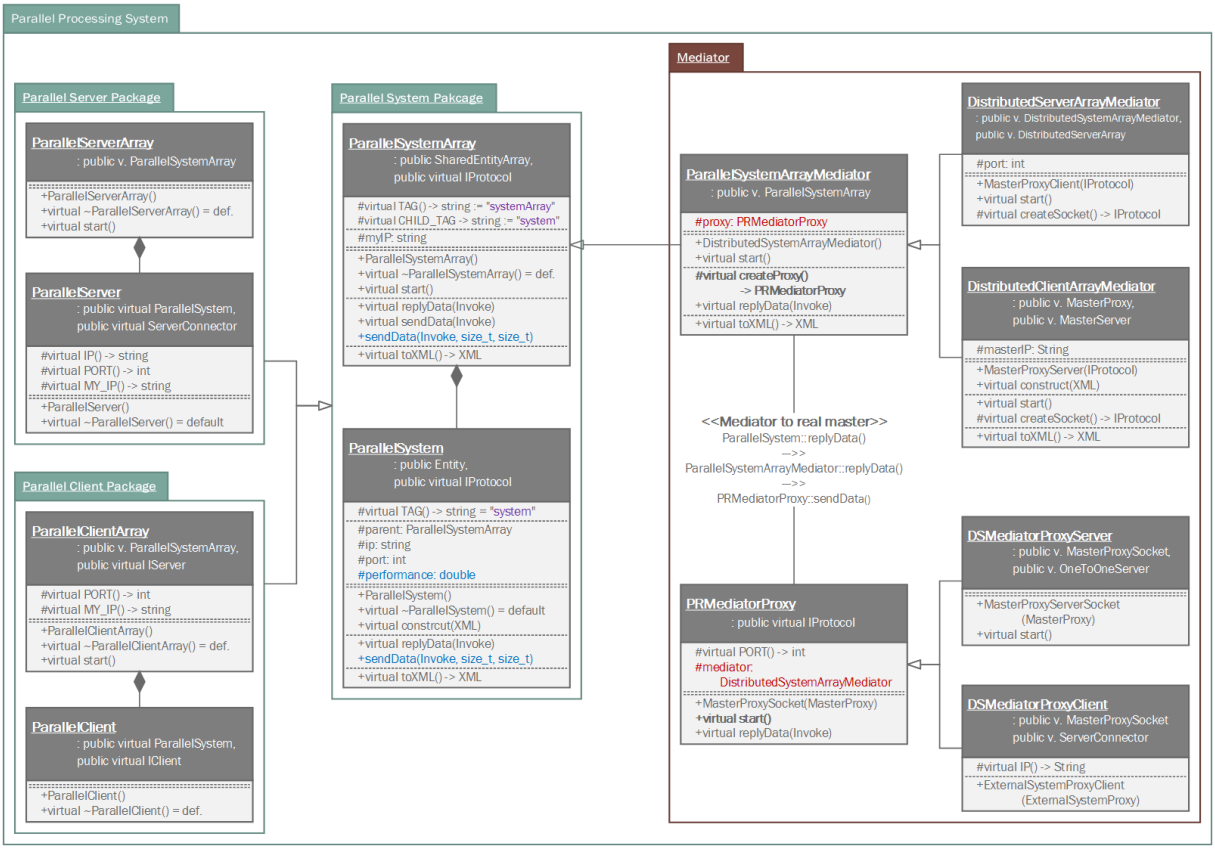
  · ExternalSystem::replyData() -> ExternalSystemRole(Proxy)::replyData()

However, unlike the case of managing roles from ExternalSystemArray, in DistributedSystemArray, roles can be allocated in multiple systems duplicately. And the roles can be moved to another systems. Those management and allocation of roles are determined by estimation of performance of each system, and required performance of each role.



Parallel Processing System

Unlike distributed processing system, the parallel processing system is not complicate. It has very simple logic. About requested processes, master(master::ParallelSystemArray) allocates each process to each slave system by their own performance index have estimated.



– **Slave system**

A package for composing distributed or parallel processing system in slave side.

Slave system get orders from master, processes the orders and report the result with its elapsed time for estimating performance of the slave system.

cpp/protocol_slave.png

Note

- Boost Libraries: http://www.boost.org/
- Boot.Asio: http://www.boost.org/doc/libs/1_59_0/doc/html/boost_asio.html

Author

Jeongho Nam

9.7.2 Typedef Documentation

9.7.2.1 template<typename _Ty = Entity> SharedEntityArray::SharedEntityArray

An EntityGroup with vector container and children capsuled in shared pointers.

Template Parameters

_Ty	A type of children Entity. Must be a class derived from an Entity or Entity itself.

For a case of inheritance of SharedEntityArray and the inherited SharedEntityArray also contains entity objects which are inherited from original child entity type, SharedEntityArray supports macros changing returned type of accessor methods.

- SHARED_ENTITY_ARRAY_ELEMENT_ACCESSOR_HEADER
- SHARED_ENTITY_ARRAY_ELEMENT_ACCESSOR_BODY

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 32 of file SharedEntityArray.hpp.

9.7.2.2 template<typename _Ty = Entity> using samchon::protocol::SharedEntityList = typedef EntityGroup < std::list<std::shared_ptr<_Ty>>, _Ty, std::shared_ptr<_Ty> >

An EntityGroup with list container and children capsuled in shared pointers.

Template Parameters

_Ty	A type of children Entity. Must be a class derived from an Entity or Entity itself.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 24 of file SharedEntityList.hpp.

9.7.2.3 template<typename _Ty = Entity> using samchon::protocol::UniqueEntityArray = typedef EntityGroup < std::vector<std::unique_ptr<_Ty>>, _Ty, std::unique_ptr<_Ty> >

An EntityGroup with vector container and children capsuled in unique pointers.

Template Parameters

_Ty	A type of children Entity. Must be a class derived from an Entity or Entity itself.

For a case of inheritance of UniqueEntityArray and the inherited UniqueEntityArray also contains entity objects which are inherited from original child entity type, UniqueEntityArray supports macros changing returned type of accessor methods.

- UNIQUE_ENTITY_ARRAY_ELEMENT_ACCESSOR_HEADER
- UNIQUE_ENTITY_ARRAY_ELEMENT_ACCESSOR_BODY

Note

std::unique_ptr doesn't allow copy construction. It allows only move construction. When inserts children objcets not by construct() method but by your hand, Be careful to insert deriving copy construction. You've use move constructor.

- *UniqueEntityArray<?>::push_back(move(std::unique_ptr<?>(new ?(...))));*

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 39 of file UniqueEntityArray.hpp.

9.7.2.4 template<typename _Ty = Entity> using samchon::protocol::UniqueEntityList = typedef EntityGroup < std::list<std::unique_ptr<_Ty>>, _Ty, std::unique_ptr<_Ty> >

An EntityGroup with list container and children capsuled in unique pointers.

Template Parameters

_Ty	A type of children Entity. Must be a class derived from an Entity or Entity itself.

Note

std::unique_ptr doesn't allow copy construction. It allows only move construction. When inserts children objcets not by construct() method but by your hand, Be careful to insert deriving copy construction. You've use move constructor.

- *UniqueEntityList<?>::push_back(move(std::unique_ptr<?>(new ?(...))));*

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 32 of file UniqueEntityList.hpp.

9.8 samchon::protocol::master Namespace Reference

Package for external system, within the framework of master.

Classes

- class DistributedClient

  A network driver for a distributed client.
- class DistributedClientArray

  An array of DistributedClient(s).
- class DistributedClientArrayMediator

  A mediator of distributed system between master and slaves.
- class DistributedClientServerMediator

  A distributed slave client for mediation.
- class DistributedServer

A network driver for a distributed server.

- class DistributedServerArray

  An array of DistributedServer(s).
- class DistributedServerArrayMediator

  A mediator of distributed system between master and slaves.
- class DistributedSlaveServerMediator

  A distributed slave server for mediation.
- class DistributedSlaveSystemMediator

  A distributed slave system for mediation.
- class DistributedSystem

  A network driver for a distributed system.
- class DistributedSystemArray

  An array of distributed system drivers.
- class DistributedSystemArrayMediator

  A mediator of distributed system between master and slaves.
- class DistributedSystemRole

  A role of distributed processing system.
- class DSInvokeHistory

  A reported history of an Invoke message.
- class DSInvokeHistoryList

  A list of DSInvokeHistory.
- class ParallelClient

  A network driver for a parallel client.
- class ParallelClientArray

  An array of ParallelClient(s).
- class ParallelClientArrayMediator

  A mediator of parallel system between master and slaves.
- class ParallelServer

  A network driver for a parallel server.
- class ParallelServerArray

  An array of ParallelServer(s).
- class ParallelServerArrayMediator

  A mediator of parallel system between master and slaves.
- class ParallelSlaveClientMediator

  A parallel slave client for mediation.
- class ParallelSlaveServerMediator

  A parallel slave server for mediation.
- class ParallelSlaveSystemMediator

  A parallel slave system for mediation.
- class ParallelSystem

  A network driver for a parallel system.
- class ParallelSystemArray

  An array of parallel system drivers.
- class ParallelSystemArrayMediator

  A mediator of parallel system between master and slaves.
- class PRInvokeHistory

  A reported history of an Invoke message.
- class PRInvokeHistoryArray

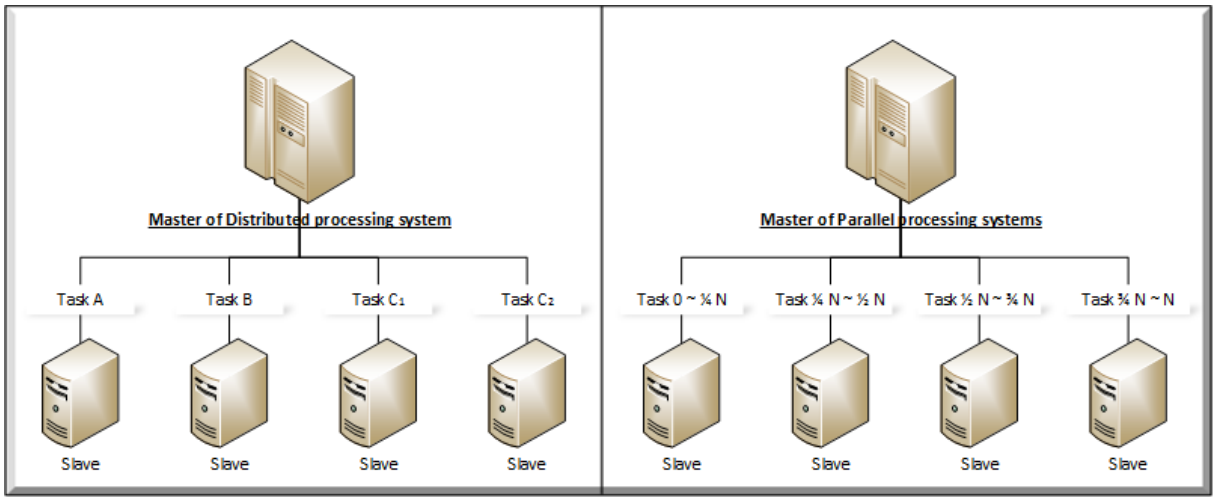
  An array of PRInvokeHistory.
- class PRMasterHistory

  A history log of an Invoke message on a master.
- class PRMasterHistoryArray

  An array of invoke histories of master.

9.8.1 Detailed Description

Package for external system, within the framework of master.



In master package, provides distributed and parallel processing system modules which are in framework of master. With classes in master package, you can realize any type of distributed or parallel processing system in master side, even how enormouse scale those have.

Distributed Processing System

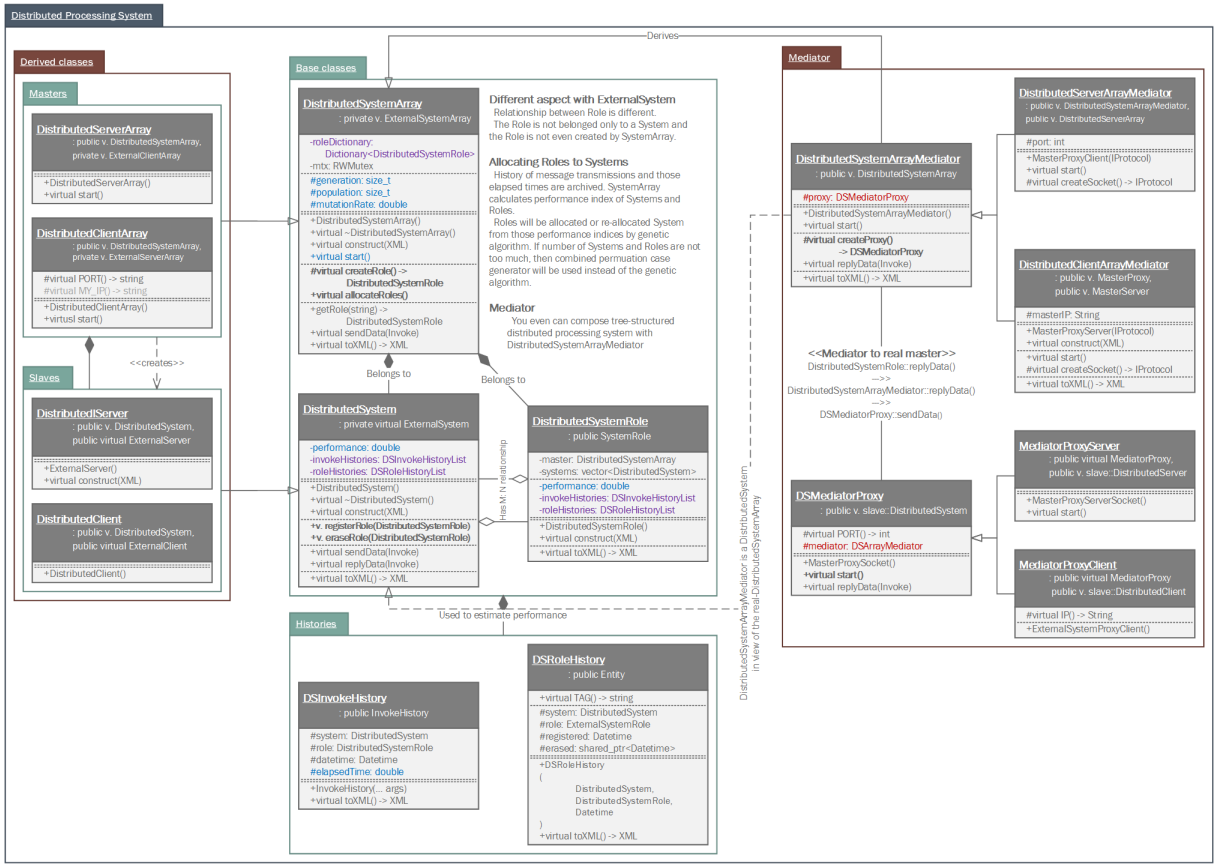
A distributed processing system of master, a master (master::DistributedSystemArray) allocates roles (master↩::DistributedSystemRole) to systems (master::DistributedSystem). User does not consider about which roles are allocated to which systems. User just accesses to a *role* and send or reply message from those roles.

- It's the proxy pattern have mentioned in external system

  ExternalSystem objects are managed by ExternalSystemArray and the ExternalSystemArray can access to a role belongs to a system directly. When you send an Invoke message to ExternalSystemArray, the ExternalSystemArray finds matched ExternalSystemRole and the ExternalSystemRole shifts the network I/O responsibility to belonged ExternalSystem. This relationship called "Proxy Pattern". By the pattern, "Proxy", you can concentrate on roles irrespective of where each role is belonged to.

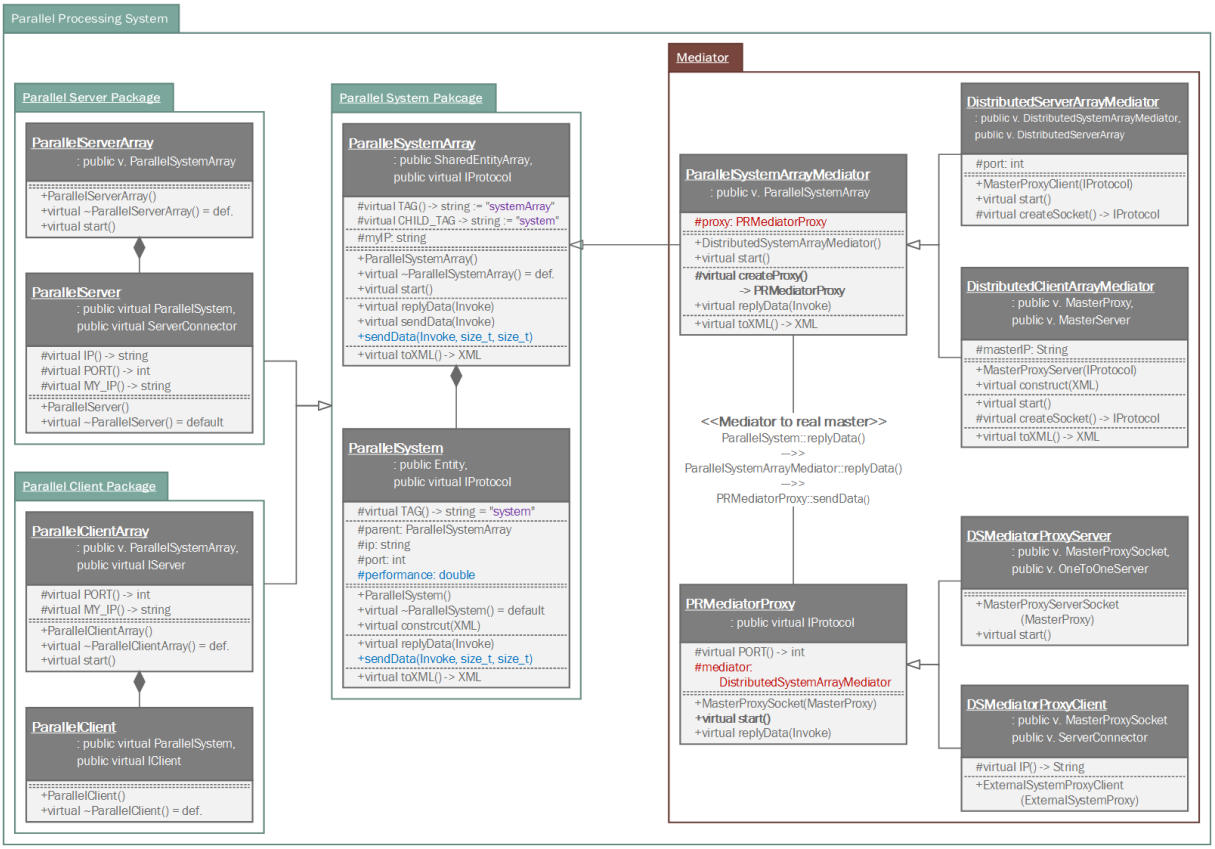
  - ExternalSystemArray::sendData() -> ExternalSystemRole(Proxy)::sendData() -> ExternalSystem↩::sendData()
  - ExternalSystem::replyData() -> ExternalSystemRole(Proxy)::replyData()

However, unlike the case of managing roles from ExternalSystemArray, in DistributedSystemArray, roles can be allocated in multiple systems duplicately. And the roles can be moved to another systems. Those management and allocation of roles are determined by estimation of performance of each system, and required performance of each role.



Parallel Processing System

Unlike distributed processing system, the parallel processing system is not complicate. It has very simple logic. About requested processes, master(master::ParallelSystemArray) allocates each process to each slave system by their own performance index have estimated.



Author

Jeongho Nam

9.9 samchon::protocol::service Namespace Reference

Package of cloud service as a server.

Classes

- class Client

  A network boundary with the client in an User.
- class IPUserPair

  Relationship between IP address and User for issuing session.
- class Server

  A server for (cloud) service.
- class Service

  A service.
- class ServiceKeeper

  A keeper blocking destrunction of User and Client.
- class User

  User containing Client(s) with session-id.

9.9.1 Detailed Description

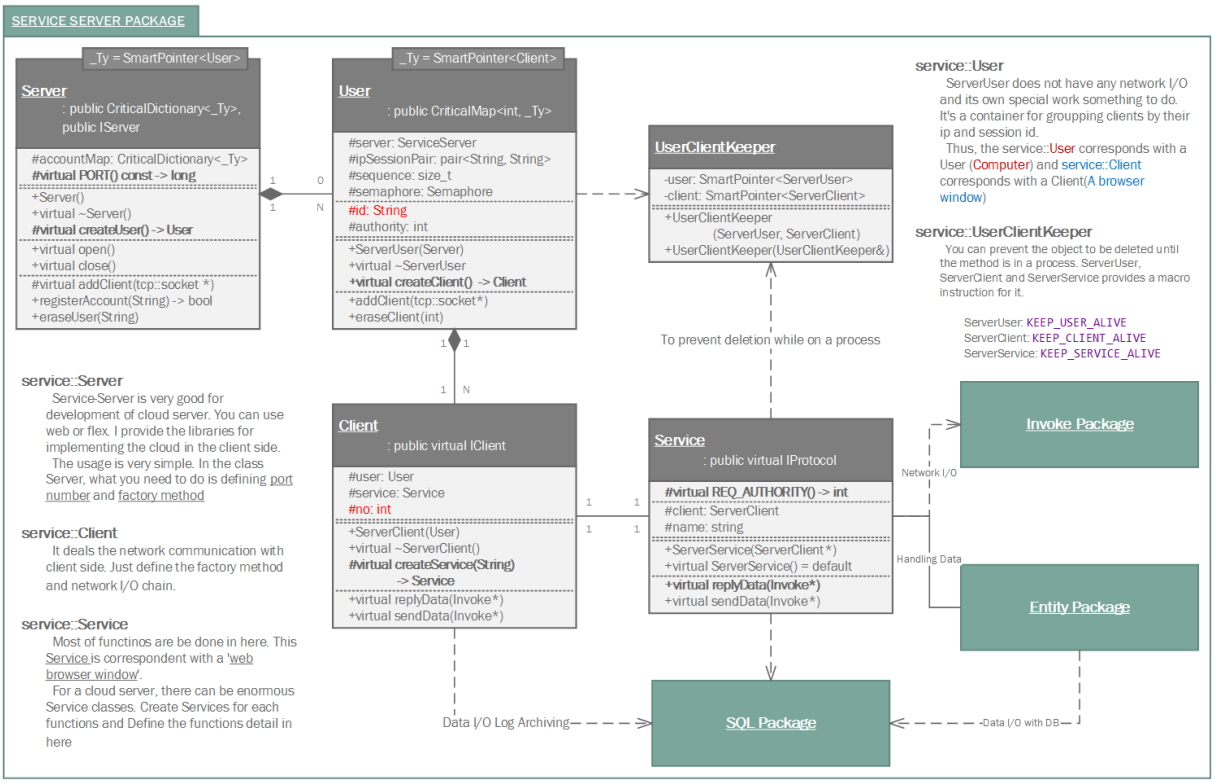
Package of cloud service as a server.

In Samchon Framework, realizing cloud service, there're two classes representing physical server and client as Server and Client. Between the two classes representing boundaries, there is a class representing logical conception user as User. In the terminal node under the Client, Service class represents a control.

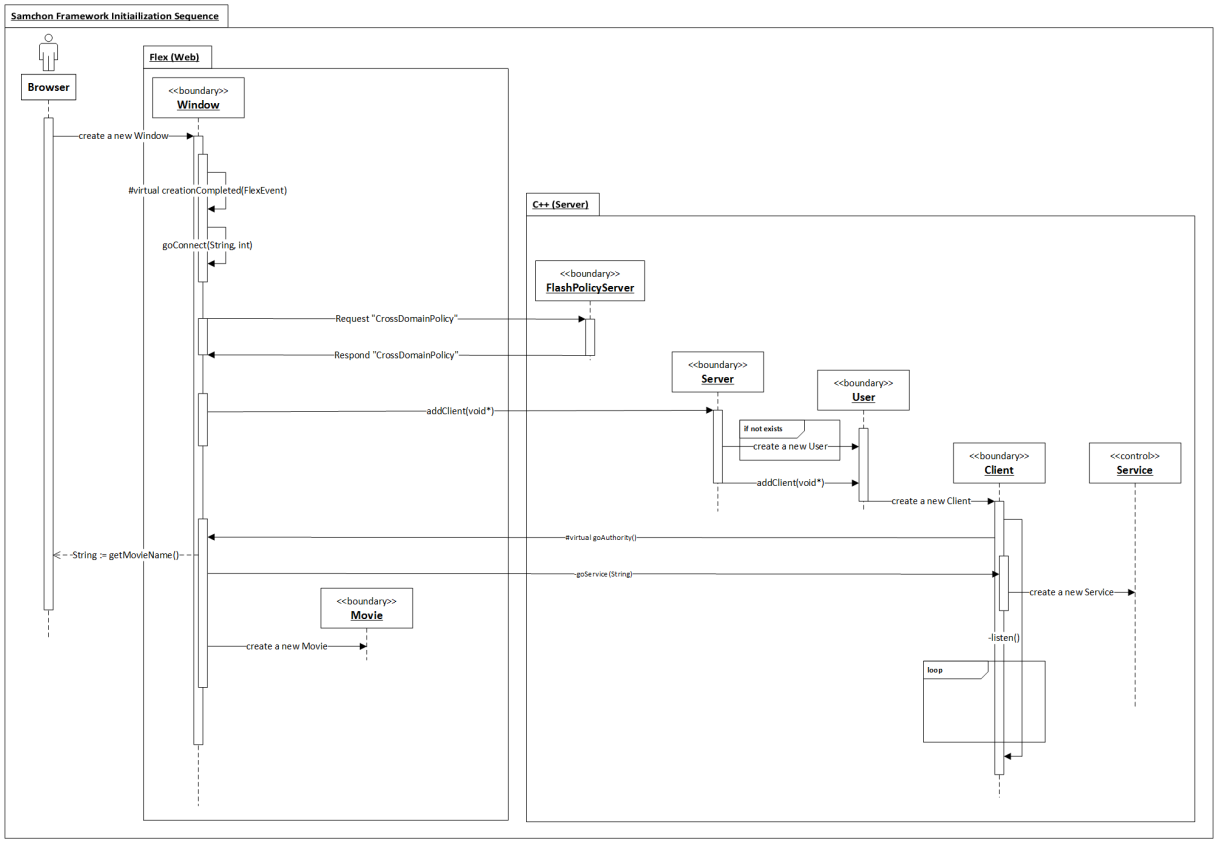
The default protocol of the service package is not 'web socket'. The default only has message protocol of Invoke, does not have handshake or another pre-defined protocol. By the default protocol, you can connect to the cloud server (built from c++, by service package) as a client by a program or plug-in like Flex or C#. However you can't connect by web-browser only allowing socket protocol as web-socket.

To build a cloud service to follow web-socket protocol, inherits derviced classes not from Server, Client, but from WebServer, WebClient or implements IWebServer, IWebClient which are derived from Server and Client.

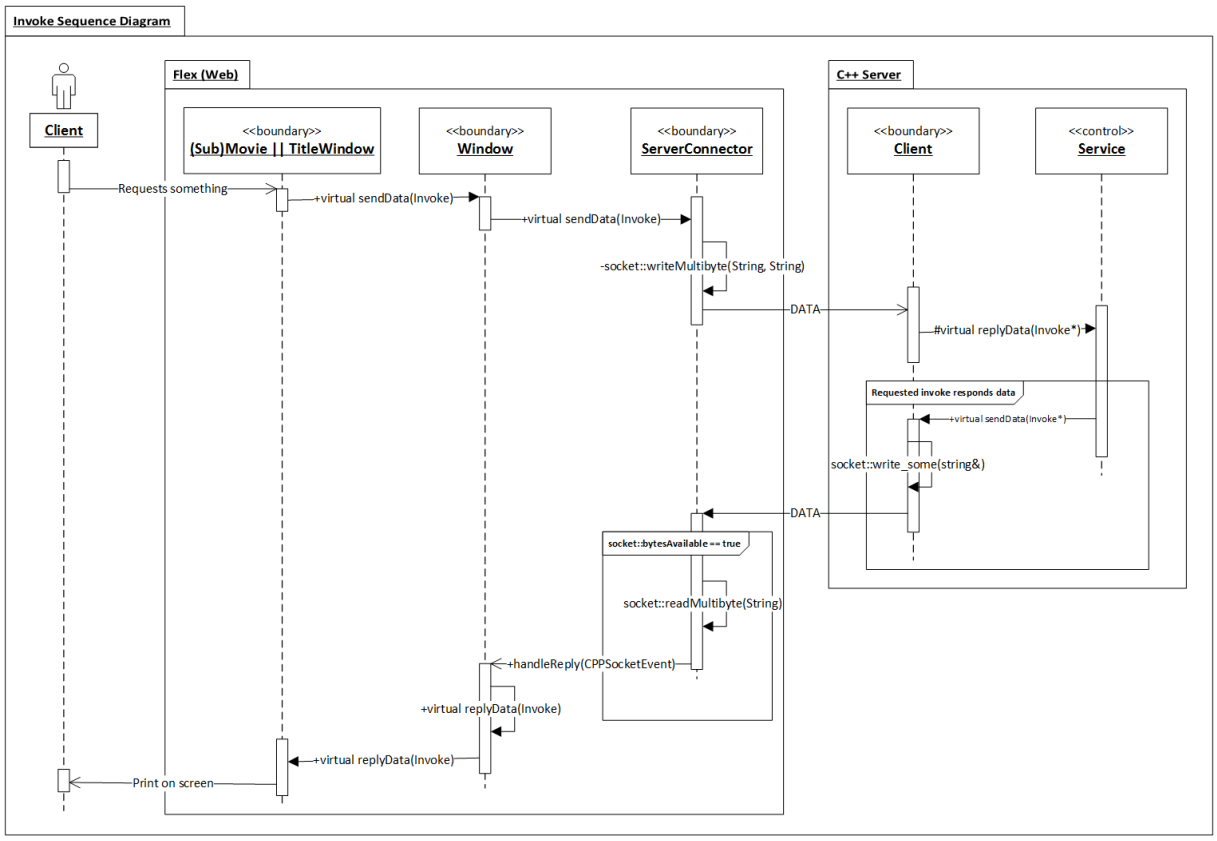
Class Diagram



Sequence Diagram - A client's connection



Sequence Diagram - Network communication



Simple example - Chat service

examples/chat_service/ChatServer.hpp

```
1 #pragma once
2 #include <samchon/protocol/service/Server.hpp>
3
4 namespace samchon
5 {
6     namespace example
7     {
8         namespace chat_service
9         {
10             class ChatRoomArray;
11             class ChatRoom;
12
22             class ChatServer
23                 : public protocol::service::Server
24             {
25             protected:
26                 typedef protocol::service::Server super;
27
28                 virtual auto PORT() const -> int override
29                 {
30                     return 37749;
31                 };
32
33             private:
37                 ChatRoomArray *roomArray;
38
39             public:
43                 ChatServer();
44                 virtual ~ChatServer();
45
46             protected:
47                 virtual auto createUser() -> protocol::service::User* override;
48
49             public:
53                 auto getChatRoomArray() const -> ChatRoomArray*;
54             };
55         };
56     };
57 };
```

examples/chat_service/ChatUser.hpp

```
1 #pragma once
2 #include <samchon/protocol/service/User.hpp>
3
4 namespace samchon
5 {
6     namespace example
7     {
8         namespace chat_service
9         {
19             class ChatUser
20                 : public protocol::service::User
21             {
22             public:
26                 ChatUser(protocol::service::Server*);
27                 virtual ~ChatUser() = default;
28
29             protected:
30                 virtual auto createClient() -> protocol::service::Client* override;
31             };
32         };
33     };
34 };
```

examples/chat_service/ChatClient.hpp

```
1 #pragma once
2 #include <samchon/protocol/service/Client.hpp>
3
4 namespace samchon
5 {
6     namespace example
7     {
8         namespace chat_service
9         {
19             class ChatClient
20                 : public protocol::service::Client
21             {
22             public:
```

```
26                 ChatClient(protocol::service::User*);
27                 virtual ~ChatClient() = default;
28
29             protected:
30                 auto createService(const std::string &) -> protocol::service::Service* override;
31             };
32         };
33     };
34 };
```

examples/chat_service/ChatService.hpp

```
1 #pragma once
2 #include <samchon/protocol/service/Service.hpp>
3
4 namespace samchon
5 {
6     namespace example
7     {
8         namespace chat_service
9         {
10             class ChatRoom;
11
22             class ChatService
23                 : public protocol::service::Service
24             {
25             private:
26                 typedef protocol::service::Service super;
27
32                 ChatRoom *room;
33
34             public:
38                 ChatService(protocol::service::Client*);
39                 virtual ~ChatService();
40
41                 virtual void replyData(std::shared_ptr<protocol::Invoke>) override;
42             };
43         };
44     };
45 };
```

examples/chat_service/ListService.hpp

```
1 #pragma once
2 #include <samchon/protocol/service/Service.hpp>
3
4 namespace samchon
5 {
6     namespace example
7     {
8         namespace chat_service
9         {
10             class ListService;
11
21             class ListService
22                 : public protocol::service::Service
23             {
24             private:
25                 typedef protocol::service::Service super;
26
27             public:
31                 ListService(protocol::service::Client*);
32                 virtual ~ListService() = default;
33
34                 virtual void replyData(std::shared_ptr<protocol::Invoke> invoke) override;
35
36             private:
42                 void createRoom(const std::string &);
43
44                 void handleRoomArray(std::shared_ptr<library::XML>);
45                 void handleMakeRoom(bool);
46                 void handleRoomArray(bool);
47             };
48         };
49     };
50 };
```

examples/chat_service/ChatRoomArray.hpp

```
1 #pragma once
2 #include <samchon/library/CriticalDictionary.hpp>
3 #include <samchon/protocol/IEntityGroup.hpp>
4
```

```
namespace samchon
{
    namespace library { class XML; };
    namespace protocol { class Invoke; };

     namespace example
     {
         namespace chat_service
         {
             class ChatRoom;

             class ChatServer;
             class ChatUser;

             class ChatRoomArray
                 : public CriticalDictionary<std::shared_ptr<ChatRoom>>,
                 public protocol::IEntityGroup
             {
             protected:
                 typedef CriticalDictionary<std::shared_ptr<ChatRoom>> super;

                 virtual auto TAG() const -> std::string { return "roomArray"; };
                 virtual auto CHILD_TAG() const -> std::string { return "room"; };

             private:
                 ChatServer *server;

             public:
                 /* ----------------------------------------------------------------
                     CONSTRUCTORS
                 ---------------------------------------------------------------- */
                 ChatRoomArray(ChatServer*);
                 virtual ~ChatRoomArray() = default;

                 /* ----------------------------------------------------------------
                     NOTIFIER
                 ---------------------------------------------------------------- */
                 void notify();

             private:
                 /* ----------------------------------------------------------------
                     EXPORTERS
                 ---------------------------------------------------------------- */
                 auto toXML() const -> std::shared_ptr<library::XML>;

                 auto toInvoke() const -> std::shared_ptr<protocol::Invoke>;
             };
         };
     };
};
```

examples/chat_service/ChatRoom.hpp

```
#pragma once
#include <samchon/protocol/Entity.hpp>
#include <samchon/protocol/IProtocol.hpp>

#include <string>
#include <samchon/library/CriticalSet.hpp>


namespace samchon
{
    namespace example
    {
        namespace chat_service
        {
            class ChatUser;
            class ChatService;

            class ChatRoomArray;

            class ChatRoom
                : public protocol::Entity,
                public protocol::IProtocol
            {
            protected:
                typedef protocol::Entity super;

                virtual auto TAG() const -> std::string { return "room"; };

            private:
                ChatRoomArray *roomArray;

                std::string name;

```

```
57                  ChatUser *host;
58
62                  library::CriticalSet<ChatService*> participants;
63
64              public:
65                  /* ------------------------------------------------------------
66                      CONSTRUCTORS
67                  ------------------------------------------------------------ */
75                  ChatRoom(ChatRoomArray*, const std::string &, ChatUser*);
76                  virtual ~ChatRoom() = default;
77
82                  void registerClient(ChatService*);
83
88                  void eraseClient(ChatService*);
89
90                  /* ------------------------------------------------------------
91                      CHAIN OF RESPONSIBILITY
92                  ------------------------------------------------------------ */
93                  virtual void replyData(std::shared_ptr<protocol::Invoke>) override;
94                  virtual void sendData(std::shared_ptr<protocol::Invoke>) override;
95
96              public:
97                  /* ------------------------------------------------------------
98                      GETTERS
99                  ------------------------------------------------------------ */
100                  virtual auto toXML() const -> std::shared_ptr<library::XML> override;
101              };
102          };
103      };
104 };
```

examples/chat_service/ChatMessage.hpp

```
1 #pragma once
2 #include <set>
3 #include <string>
4
5 #include <samchon/protocol/Entity.hpp>
6
7 namespace samchon
8 {
9     namespace example
10     {
11         namespace chat_service
12         {
24             class ChatMessage
25                 : public protocol::Entity
26             {
27             protected:
28                 typedef protocol::Entity super;
29
30                 virtual auto TAG() const -> std::string
31                 {
32                     return "message";
33                 };
34
35             private:
39                 std::string orator;
40
45                 std::string listener;
46
50                 std::string message;
51
52             public:
53                 /* ------------------------------------------------------------
54                     CONSTRUCTORS
55                 ------------------------------------------------------------ */
59                 ChatMessage();
60                 virtual ~ChatMessage() = default;
61
62                 virtual void construct(std::shared_ptr<library::XML>);
63
64                 /* ------------------------------------------------------------
65                     GETTERS
66                 ------------------------------------------------------------ */
71                 auto getListener() const -> std::string;
72
73                 virtual auto toXML() const -> std::shared_ptr<library::XML>;
74             };
75         };
76     };
77 };
```

examples/chat_service/ChatServer.cpp

examples/chat_service/ChatUser.cpp

examples/chat_service/ChatClient.cpp

examples/chat_service/ChatService.cpp

```
#include "ChatService.hpp"
#include "ChatRoom.hpp"

#include <samchon/protocol/Invoke.hpp>

using namespace std;
using namespace samchon::protocol;
using namespace samchon::protocol::service;
using namespace samchon::example::chat_service;

ChatService::ChatService(Client *client)
    : super(client)
{

}
ChatService::~ChatService()
{
    if(room == nullptr)
        return;

    room->eraseClient(this);
}

void ChatService::replyData(shared_ptr<Invoke> invoke)
{
    if(invoke->getListener() == "sendMessage")
        room->replyData(invoke);
}
```

examples/chat_service/ListService.cpp

examples/chat_service/ChatRoomArray.cpp

```
#include "ChatRoomArray.hpp"
#include "ChatRoom.hpp"

#include "ChatServer.hpp"
#include "ChatUser.hpp"
#include "ChatClient.hpp"
#include "ListService.hpp"

#include <thread>
#include <samchon/library/XML.hpp>
#include <samchon/protocol/Invoke.hpp>

using namespace std;
using namespace samchon::library;
using namespace samchon::protocol;
using namespace samchon::example::chat_service;

ChatRoomArray::ChatRoomArray(ChatServer *server)
    : super(),
    IEntityGroup()
{
    this->server = server;
}

void ChatRoomArray::notify()
{
    shared_ptr<Invoke> &invoke = this->toInvoke();

    UniqueReadLock uk(get_allocator().getMutex());
    for (auto it = server->begin(); it != server->end(); it++)
    {
        auto user = it->second;
```

```

        for (auto u_it = user->begin(); u_it != user->end(); u_it++)
        {
            auto client = u_it->second;
            auto service = client->getService();

            if (dynamic_cast<ListService>(service) != nullptr)
                thread(&IProtocol::sendData, service, invoke).detach();
        }
    }
}

auto ChatRoomArray::toXML() const -> shared_ptr<XML>
{
    shared_ptr<XML> xml(new XML());
    xml->setTag(TAG());

    UniqueReadLock uk(get_allocator().getMutex());
    for(auto it = begin(); it != end(); it++)
        xml->push_back(it->second->toXML());

    return xml;
}
auto ChatRoomArray::toInvoke() const -> shared_ptr<Invoke>
{
    Invoke *invoke = new Invoke("handleRoomArray", toXML());
    return shared_ptr<Invoke>(invoke);
}
```

examples/chat_service/ChatRoom.cpp

```
#include "ChatRoom.hpp"
#include "ChatRoomArray.hpp"
#include "ChatMessage.hpp"

#include "ChatUser.hpp"
#include "ChatClient.hpp"
#include "ChatService.hpp"

#include <samchon/library/XML.hpp>
#include <samchon/protocol/Invoke.hpp>

using namespace std;
using namespace samchon::library;
using namespace samchon::protocol;
using namespace samchon::example::chat_service;

/* ----------------------------------------------------------
    CONSTRUCTORS
---------------------------------------------------------- */
ChatRoom::ChatRoom(ChatRoomArray *roomArray, const string &name,
    ChatUser *host)
    : super(),
    IProtocol()
{
    this->roomArray = roomArray;

    this->name = name;
    this->host = host;
}

void ChatRoom::registerClient(ChatService *service)
{
    participants.insert(service);

    sendData(shared_ptr<Invoke>(new Invoke("handleRoom", toXML())));
    roomArray->notify();
}
void ChatRoom::eraseClient(ChatService *service)
{
    if (participants.erase(service) == 0)
        roomArray->erase(this->name);
    else
        sendData(shared_ptr<Invoke>(new Invoke("handleRoom", toXML())));

    roomArray->notify();
}

/* ----------------------------------------------------------
    CHAIN OF RESPONSIBILITY
---------------------------------------------------------- */
void ChatRoom::replyData(shared_ptr<Invoke> invoke)
{
    if(invoke->getListener() == "sendMessage")
    {
        shared_ptr<Invoke> ivk(new Invoke("handleMessage"));
```

```
        ivk->at(0) = invoke->at(0);

        sendData(ivk);
    }
}
void ChatRoom::sendData(shared_ptr<Invoke> invoke)
{
    UniqueReadLock uk(participants.get_allocator().getMutex());

    if(invoke->getListener() == "handleMessage")
    {
        ChatMessage message;
        message.construct(invoke->at(0)->getValueAsXML());

        if(message.getListener().empty() == true)
            for (auto it = participants.begin(); it != participants.end(); it++)
                (*it)->sendData(invoke);
        else
            for (auto it = participants.begin(); it != participants.end(); it++)
                if((*it)->getClient()->getUser()->getID() == message.getListener())
                    (*it)->sendData(invoke);
    }
    else
        for (auto it = participants.begin(); it != participants.end(); it++)
            (*it)->sendData(invoke);
}

/* ----------------------------------------------------------
    EXPORTS
---------------------------------------------------------- */
auto ChatRoom::toXML() const -> shared_ptr<XML>
{
    shared_ptr<XML> &xml = super::toXML();
    xml->setProperty("name", name);
    xml->setProperty("host", host->getID());

    for (auto it = participants.begin(); it != participants.end(); it++)
    {
        shared_ptr<XML> participant(new XML());
        participant->setTag("participant");
        participant->setValue((*it)->getClient()->getUser()->getID());

        xml->push_back(participant);
    }
    return xml;
}
```

examples/chat_service/ChatMessage.cpp

```
#include "ChatMessage.hpp"

#include <samchon/library/XML.hpp>

using namespace std;
using namespace samchon::library;
using namespace samchon::protocol;
using namespace samchon::example::chat_service;

/* ----------------------------------------------------------
    CONSTRUCTORS
---------------------------------------------------------- */
ChatMessage::ChatMessage()
    : super()
{
}
void ChatMessage::construct(shared_ptr<XML> xml)
{
    this->orator = xml->getProperty("orator");
    this->message = xml->getProperty("message");

    if(xml->hasProperty("listener") == true)
        this->listener = xml->getProperty("listener");
    else
        this->listener.clear();
}

/* ----------------------------------------------------------
    GETTERS
---------------------------------------------------------- */
auto ChatMessage::getListener() const -> string
{
    return listener;
}
auto ChatMessage::toXML() const -> shared_ptr<XML>
{
    shared_ptr<XML> &xml = super::toXML();
```

```
    xml->setProperty("orator", orator);
    xml->setProperty("message", message);

    if(listener.empty() == false)
        xml->setProperty("listener", listener);
    return xml;
}

```

Author

Jeongho Nam

9.10 samchon::protocol::slave Namespace Reference

Package of external system, within the framework of slave.

Classes

- class ParallelClient

  A slave parallel client.

- class ParallelServer

  A slave parallel server.

- class ParallelSystem

  A slave parallel system.

- class SlaveClient

  A slave client.

- class SlaveServer

  A slave server.

- class SlaveSystem

  A slave system.

9.10.1 Detailed Description

Package of external system, within the framework of slave.

A package for composing distributed or parallel processing system in slave side.

Slave system get orders from master, processes the orders and report the result with its elapsed time for estimating performance of the slave system.

cpp/protocol_slave.png

Author

Jeongho Nam

Chapter 10

Class Documentation

10.1 samchon::library::Base64 Class Reference

Utility class for base64 format's en-decoding.

```
#include <Base64.hpp>
```

Static Public Member Functions

- static auto encode (const ByteArray &) -> std::string

  Encode from binary data to base64-string.

- static auto decode (const std::string &) -> ByteArray

  Decode from base64-string to binary data.

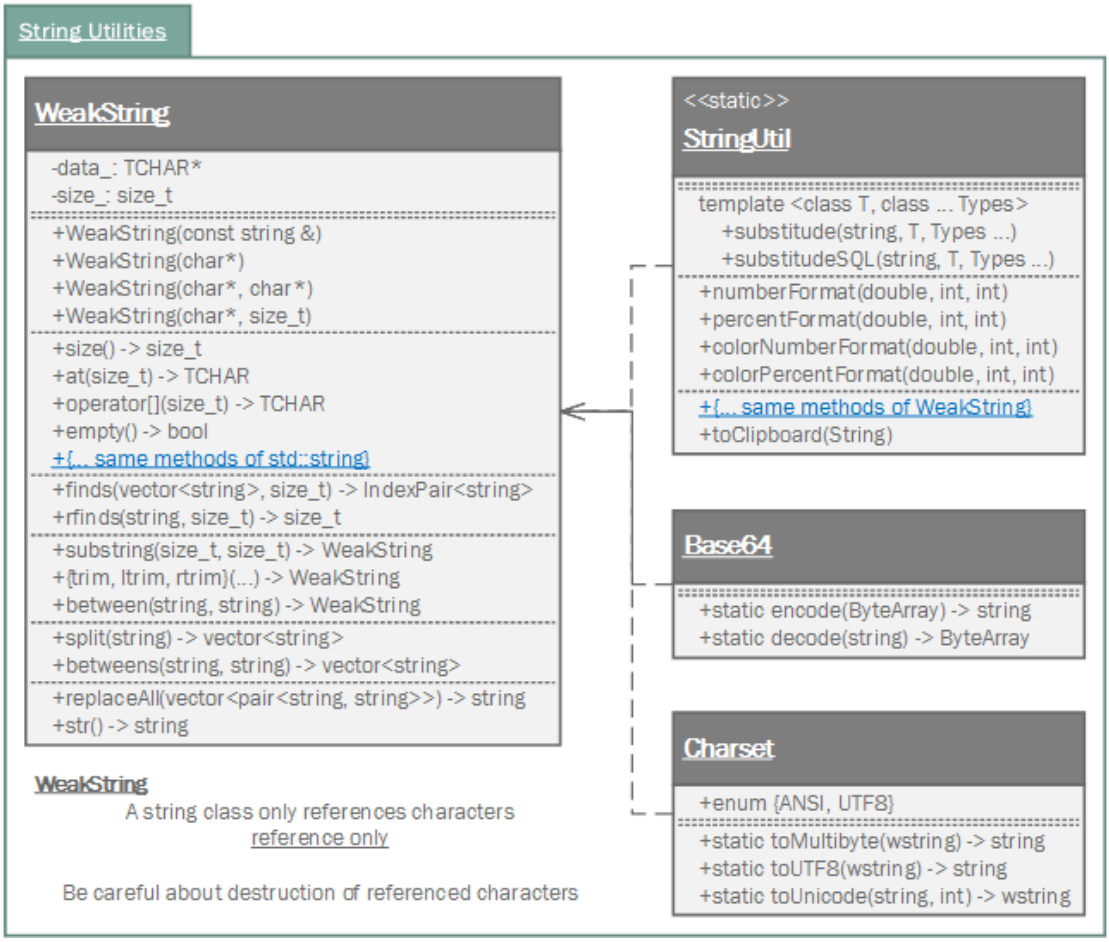
10.1.1 Detailed Description

Utility class for base64 format's en-decoding.

Base64 supports

- Encoding from binary data to base64-string

- Decoding from base64-string to binary data



See also

samchon::library

Author

Jeongho Nam

Definition at line 25 of file Base64.hpp.

10.1.2 Member Function Documentation

10.1.2.1 auto Base64::encode (const ByteArray & *byteArray*) -> std::string [static]

Encode from binary data to base64-string.

Encodes a binary data to base64-string

Parameters

byteArray	binary data you want to encode

Returns

A base64-string encoded from binary data

Definition at line 12 of file Base64.cpp.

10.1.2.2 auto Base64::decode (const std::string & *str*) -> ByteArray [static]

Decode from base64-string to binary data.

Decodes a base64 encoded string to binary data

Note

If the encoded string is not whole but a part of the binary data, Base64Decoder will be better

Parameters

str	A encoded string representing binary data as base64 format

Returns

A binary data decoded from string

Definition at line 76 of file Base64.cpp.

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/Base64.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/Base64.cpp

10.2 samchon::ByteArray Class Reference

Binary data class.

```
#include <ByteArray.hpp>
```

Public Member Functions

- ByteArray ()

  Default Constructor.
- ByteArray (const ByteArray &)

  Copy Constructor.
- ByteArray (ByteArray &&)

  Move Constructor.
- auto getPosition () const -> size_t

  Get position.
- void setPosition (size_t)

  Set poisition.
- template<typename _Ty >
  auto read () const -> _Ty

  Read data.
- template<typename _Ty >
  auto readReversely () const -> _Ty

  Read a reversed data.
- template<typename _Ty >
  void write (const _Ty &val)

  Write a data.
- template<typename _Ty >
  void writeReversely (const _Ty &val)

  Write a data.

- auto compress () const -> ByteArray

  Compress the binary data

  .

- auto decompress () const -> ByteArray

  Decompress the binary data.

Static Public Member Functions

- template<typename _Ty >
  static auto reverse (const _Ty &val) -> _Ty

  Reverse byte ordering.

Private Attributes

- size_t position

  Current position of the ByteArray.

  .

10.2.1 Detailed Description

Binary data class.

ByteArray is the standard class for handling binary data in Samchon Framework.

ByteArray provides methods for reading and writing binary data.

In addition, zlib compression and decompression are supported.

Note

ByteArray is a class only for advanced developer who can access data on byte level as the reason of:

- Handling File Data
- Custom protocol for network I/O
- Optimizing the size of data for performance

If you're not trying to use ByteArray as the reason of what I've mentioned but only for representing entity data, I recomment you to use Entity instead.

See also

samchon::library

Author

Jeongho Nam

Definition at line 30 of file ByteArray.hpp.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 ByteArray::ByteArray ( )

Default Constructor.

Definition at line 9 of file ByteArray.cpp.

References position.

10.2.2.2 ByteArray::ByteArray (const ByteArray & *byteArray*)

Copy Constructor.

Definition at line 14 of file ByteArray.cpp.

References position.

10.2.2.3 ByteArray::ByteArray (ByteArray && *byteArray*)

Move Constructor.

Definition at line 19 of file ByteArray.cpp.

References position.

10.2.3 Member Function Documentation

10.2.3.1 auto ByteArray::getPosition ( ) const -> size_t

Get position.

Get a position represents starting point of bytes to read

Definition at line 29 of file ByteArray.cpp.

References position.

10.2.3.2 void ByteArray::setPosition (size_t *val*)

Set poisition.

Set a position represents starting point of bytes to read

Definition at line 33 of file ByteArray.cpp.

References position.

10.2.3.3 template<typename _Ty > static auto samchon::ByteArray::reverse (const _Ty & *val*) -> _Ty [inline], [static]

Reverse byte ordering.

Creates a copy of data which of byte ordering is reversed.

- BIG_ENDIAN to SMALL_ENDIAN li SMALL_ENDIAN to BIG_ENDIAN

Definition at line 86 of file ByteArray.hpp.

10.2.3.4 template<typename _Ty > auto samchon::ByteArray::read ( ) const -> _Ty [inline]

Read data.

Reads a data(_Ty) from (starting) position of the ByteArray and adds the position following the size.

Template Parameters

_Ty	A type of data to be returned

Exceptions

exception	out of range

Returns

Read data from the byte stream

Definition at line 111 of file ByteArray.hpp.

10.2.3.5 template<typename _Ty > auto samchon::ByteArray::readReversely ( ) const -> _Ty [inline]

Read a reversed data.

Reads data(_Ty) from (starting) position of the ByteArray, which byte ordering is reversed, from the ByteArray and adds the position following the size.

- BIG_ENDIAN to SMALL_ENDIAN li SMALL_ENDIAN to BIG_ENDIAN

Template Parameters

_Ty	A type of data to be returned

Exceptions

exception	out of range

Returns

Read data from the byte stream

Definition at line 133 of file ByteArray.hpp.

10.2.3.6 template<typename _Ty > void samchon::ByteArray::write (const _Ty & *val*) [inline]

Write a data.

Writes a data(_Ty) to tail of the ByteArray

Template Parameters

_Ty	A type of data to write

Parameters

val	Something to write on ByteArray

Definition at line 149 of file ByteArray.hpp.

10.2.3.7 template<typename _Ty > void samchon::ByteArray::writeReversely (const _Ty & *val*) [inline]

Write a data.

Writes a data(_Ty), which byte order is reversed, to tail of the ByteArray

- BIG_ENDIAN to SMALL_ENDIAN li SMALL_ENDIAN to BIG_ENDIAN

Template Parameters

_Ty	A type of data to write

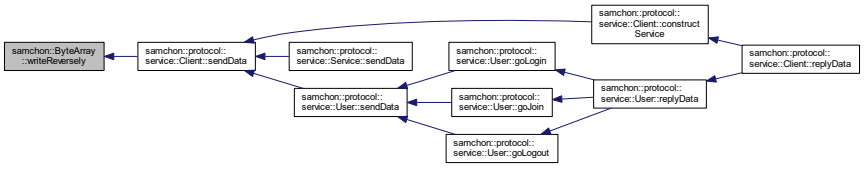
Parameters

val	Something to write on ByteArray

Definition at line 167 of file ByteArray.hpp.

Referenced by samchon::protocol::service::Client::sendData().

Here is the caller graph for this function:



10.2.3.8 auto ByteArray::compress ( ) const -> ByteArray

Compress the binary data

.

Generates a binary data compressed from the ByteArray.

The binary data will be compressed by zlib library.

Returns

ByteArray which is compressed

Definition at line 66 of file ByteArray.cpp.

10.2.3.9 auto ByteArray::decompress ( ) const -> ByteArray

Decompress the binary data.

Generates a decompressed binary data form the ByteArray.

The binary data will be decompressed by zlib library.

Returns

ByteArray that is decompressed

Definition at line 70 of file ByteArray.cpp.

10.2.4 Member Data Documentation

10.2.4.1 size_t samchon::ByteArray::position [private]

Current position of the ByteArray.

.

If you call read, starting pointer will be the position

- (_Ty *ptr* = *(_Ty)*data() + position)

Definition at line 43 of file ByteArray.hpp.

Referenced by ByteArray(), getPosition(), and setPosition().

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/ByteArray.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/ByteArray.cpp

10.3 samchon::library::CaseGenerator Class Reference

Case generator.

```
#include <CaseGenerator.hpp>
```

Public Member Functions

- CaseGenerator (size_t n, size_t r)

  Construct from size of N and R.
- auto size () const -> size_t

  Get size of all cases.
- virtual auto operator[ ] (size_t) const -> std::vector< size_t >=0

  Get x'th case.
- auto at (size_t) const -> std::vector< size_t >
- auto n () const -> size_t

  Get size of the N.
- auto r () const -> size_t

  Get size of the R.
- auto toMatrix () const -> std::vector< std::vector< size_t >>

  Create a matrix containing all cases.

Protected Attributes

- size_t n_

  N, size of the candidates.
- size_t r_

  R, size of elements of each case.
- size_t size_

  Size, the number of all cases.

10.3.1 Detailed Description

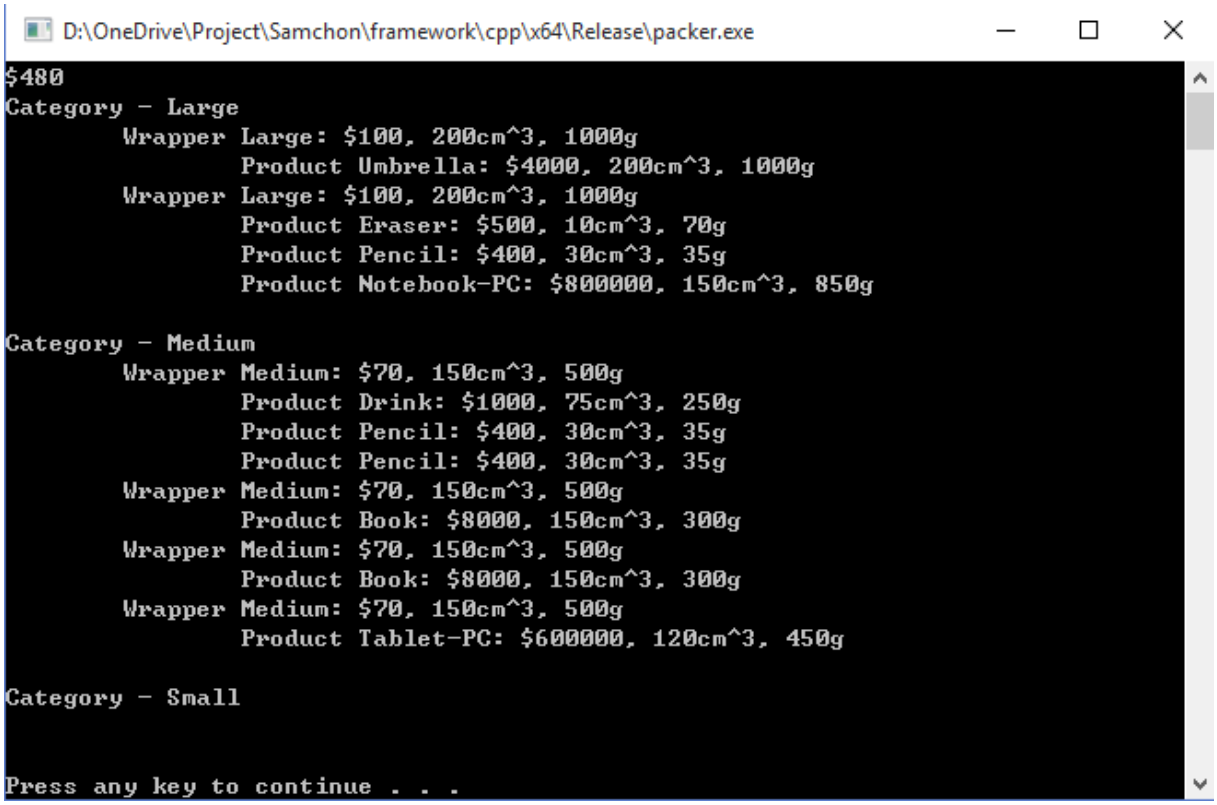
Case generator.

An abstract case generator using like a matrix

- nTTr(n^r) -> CombinedPermutationGenerator
- nPr -> PermutationGenerator
- n! -> FactorialGenerator

Example source

Packer - Find the best packaging solution



examples/packer/Packer.hpp

```
1 #pragma once
2 #include <vector>
3 #include <memory>
4 #include <string>
5
6 namespace samchon
7 {
8     namespace example
9     {
10         namespace packer
11         {
12             class WrapperArray;
13             class Wrapper;
14             class Product;
15
32             class Packer
33                 : public std::vector<std::shared_ptr<WrapperArray>>
34             {
35             private:
36                 typedef std::vector<std::shared_ptr<WrapperArray>> super;
37
41                 std::vector<Product> *productArray;
```

```
42
46                  std::vector<Wrapper> *wrapperArray;
47
48              public:
49                  /* -------------------------------------------------------
50                      CONSTRUCTORS
51                  ------------------------------------------------------- */
58                  Packer(std::vector<Product> *, std::vector<Wrapper> *);
59
67                  Packer(const Packer &);
68
69                  /* -------------------------------------------------------
70                      CALCULATE AND OPTIMIZE
71                  ------------------------------------------------------- */
75                  void optimize();
76
80                  auto calcPrice() const -> int;
81
82                  /* -------------------------------------------------------
83                      EXPORT
84                  ------------------------------------------------------- */
88                  auto toString() const -> std::string;
89              };
90          };
91      };
92 };
```

examples/packer/WrapperArray.hpp

```
1 #pragma once
2 #include <vector>
3 #include <memory>
4
5 namespace samchon
6 {
7     namespace example
8     {
9         namespace packer
10         {
11             class Wrapper;
12             class Product;
13
26             class WrapperArray
27                 : private std::vector<std::shared_ptr<Wrapper>>
28             {
29             private:
30                 typedef std::vector<std::shared_ptr<Wrapper>> super;
31
35                 std::vector<Product*> reserved;
36
40                 Wrapper *sample;
41
42             public:
43                 /* -------------------------------------------------------
44                     CONSTRUCTORS
45                 ------------------------------------------------------- */
51                 WrapperArray(Wrapper *);
52
63                 auto tryInsert(Product*) -> bool;
64
65                 /* -------------------------------------------------------
66                     CALCULATE AND OPTIMIZE
67                 ------------------------------------------------------- */
83                 void optimize();
84
89                 auto calcPrice() const -> int;
90
91                 /* -------------------------------------------------------
92                     EXPORT
93                 ------------------------------------------------------- */
97                 auto toString() const -> std::string;
98             };
99         };
100     };
101 };
```

examples/packer/Wrapper.hpp

```
1 #pragma once
2 #include <vector>
3 #include "Instance.hpp"
4
5 namespace samchon
6 {
7     namespace example
```

```
    {
        namespace packer
        {
            class Product;

            class Wrapper
                : public Instance,
                private std::vector<Product*>
            {
            private:
                typedef Instance super;

            public:
                /* ---------------------------------------------------------
                    CONSTRUCTORS
                --------------------------------------------------------- */
                Wrapper(const std::string &name, int price, int volume, int weight);

                Wrapper(const Wrapper &wrapper);
                virtual ~Wrapper() = default;

                auto tryInsert(Product *) -> bool;

                /* ---------------------------------------------------------
                    EXPORT
                --------------------------------------------------------- */
                virtual auto toString() const -> std::string override;
            };
        };
    };
};
```

examples/packer/Product.hpp

```
#pragma once
#include "Instance.hpp"

namespace samchon
{
    namespace example
    {
        namespace packer
        {
            class Product
                : public Instance
            {
            private:
                typedef Instance super;

            public:
                /* ---------------------------------------------------------
                    CONSTRUCTOR
                --------------------------------------------------------- */
                Product(const std::string &name, int price, int volume, int weight);
                virtual ~Product() = default;

                /* ---------------------------------------------------------
                    EXPORT
                --------------------------------------------------------- */
                virtual auto toString() const -> std::string override;
            };
        };
    };
};
```

examples/packer/Instance.hpp

```
#pragma once
#include <string>

namespace samchon
{
    namespace example
    {
        namespace packer
        {
            class Instance
            {
            protected:
                std::string name;

                int price;

                int volume;

```

```
                int weight;

            public:
                /* ---------------------------------------------------------
                    CONSTRUCTORS
                --------------------------------------------------------- */
                Instance(const std::string &name, int price, int volume, int weight);
                virtual ~Instance() = default;

                /* ---------------------------------------------------------
                    GETTERS
                --------------------------------------------------------- */
                auto getName() const -> std::string;

                auto getPrice() const -> int;

                auto getVolume() const -> int;

                auto getWeight() const -> int;

                /* ---------------------------------------------------------
                    EXPORT
                --------------------------------------------------------- */
                virtual auto toString() const -> std::string;
            };
        };
    };
};
```

examples/packer/main.cpp

```
#include <iostream>
#include <vector>

#include "Packer.hpp"
#   include "Product.hpp"
#   include "Wrapper.hpp"

#ifdef _WIN64
#   ifdef _DEBUG
#       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "x64/Release/SamchonFramework.lib")
#   endif
#else
#   ifdef _DEBUG
#       pragma comment(lib, "Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "Release/SamchonFramework.lib")
#   endif
#endif

using namespace std;
using namespace samchon::example::packer;

void main()
{
    vector<Product> productArray =
    {

        Product("Eraser", 500, 10, 70),
        Product("Pencil", 400, 30, 35),
        Product("Pencil", 400, 30, 35),
        Product("Pencil", 400, 30, 35),
        Product("Book", 8000, 150, 300),
        Product("Book", 8000, 150, 300),
        Product("Drink", 1000, 75, 250),
        Product("Umbrella", 4000, 200, 1000),
        Product("Notebook-PC", 800000, 150, 850),
        Product("Tablet-PC", 600000, 120, 450)
    };
    vector<Wrapper> wrapperArray =
    {
        Wrapper("Large", 100, 200, 1000),
        Wrapper("Medium", 70, 150, 500),
        Wrapper("Small", 50, 100, 250)
    };

    Packer packer(&productArray, &wrapperArray);
    packer.optimize();

    cout << packer.toString() << endl;
    system("pause");
}
```

examples/packer/Packer.cpp

```
#include "Packer.hpp"

#include "WrapperArray.hpp"
#include "Wrapper.hpp"
#include "Product.hpp"

#include <samchon/library/CombinedPermutationGenerator.hpp>

using namespace std;
using namespace samchon::library;
using namespace samchon::example::packer;

Packer::Packer(vector<Product> *productArray, vector<Wrapper> *wrapperArray)
    : super()
{
    this->productArray = productArray;
    this->wrapperArray = wrapperArray;

    for(size_t i = 0; i < wrapperArray->size(); i++)
        emplace_back(new WrapperArray(&wrapperArray->at(i)));
}
Packer::Packer(const Packer &packer)
    : Packer(packer.productArray, packer.wrapperArray)
{
}

/* ---------------------------------------------------------
    CALCULATE AND OPTIMIZE
--------------------------------------------------------- */
void Packer::optimize()
{
    if(empty() == true || productArray->empty() == true)
        return;

    CombinedPermutationGenerator caseGenerator(size(), productArray->size());
    shared_ptr<Packer> minPacker = nullptr;

    for (size_t i = 0; i < caseGenerator.size(); i++)
    {
        vector<size_t> &row = caseGenerator[i];
        shared_ptr<Packer> packer(new Packer(*this));
        bool validity = true;

        for (size_t j = 0; j < row.size(); j++)
        {
            Product *product = &productArray->at(j);
            shared_ptr<WrapperArray> &wrapperArray = packer->at( row[j] );

            if (wrapperArray->tryInsert(product) == false)
            {
                validity = false;
                break;
            }
        }

        if(validity == false)
            continue;

        //OPTIMIZE ALL WRAPPERS IN A PACKER
        for (size_t j = 0; j < packer->size(); j++)
            packer->at(j)->optimize();

        if (minPacker == nullptr ||
            packer->calcPrice() < minPacker->calcPrice())
        {
            minPacker = packer;
        }
    }

    //COPY
    assign(minPacker->begin(), minPacker->end());
}
auto Packer::calcPrice() const -> int
{
    int price = 0;
    for (size_t i = 0; i < size(); i++)
        price += at(i)->calcPrice();

    return price;
}

/* ---------------------------------------------------------
    EXPORT
--------------------------------------------------------- */
auto Packer::toString() const -> string
```

```
{
    string str = "$" + to_string(calcPrice()) + "\n";
    for (int i = 0; i < size(); i++)
        str += at(i)->toString() + "\n";

    return move(str);
}
```

examples/packer/WrapperArray.cpp

```
#include "WrapperArray.hpp"
#   include "Wrapper.hpp"
#   include "Product.hpp"

#include <samchon/library/FactorialGenerator.hpp>
#include <mutex>

using namespace std;
using namespace samchon::library;
using namespace samchon::example::packer;

/* ---------------------------------------------------------
    CONSTRUCTOR
--------------------------------------------------------- */
WrapperArray::WrapperArray(Wrapper *sample)
    : super()
{
    this->sample = sample;
}

/* ---------------------------------------------------------
    CALCULATE AND OPTIMIZE
--------------------------------------------------------- */
auto WrapperArray::tryInsert(Product *product) -> bool
{
    if (product->getVolume() > sample->getVolume() ||
        product->getWeight() > sample->getWeight())
    {
        return false;
    }

    reserved.push_back(product);
    return true;
}
void WrapperArray::optimize()
{
    if(reserved.empty() == true)
        return;

    FactorialGenerator factorial(reserved.size());
    shared_ptr<WrapperArray> minWrapperArray = nullptr;

    mutex mtx;

    //#pragma omp parallel for
    for (int i = 0; i < factorial.size(); i++)
    {
        shared_ptr<WrapperArray> wrapperArray(new WrapperArray(this->sample));
        vector<size_t> &row = factorial[i];

        for (size_t j = 0; j < row.size(); j++)
        {
            Product *product = this->reserved[row[j]];

            if (wrapperArray->empty() == true ||
                wrapperArray->at(wrapperArray->size() - 1)->tryInsert(product) == false)
            {
                Wrapper *wrapper = new Wrapper(*this->sample);
                wrapper->tryInsert(product);

                wrapperArray->emplace_back(wrapper);
            }
        }

        //unique_lock<mutex> uk(mtx);
        if (minWrapperArray == nullptr ||
            wrapperArray->size() < minWrapperArray->size())
        {
            minWrapperArray = wrapperArray;
        }
    }

    assign(minWrapperArray->begin(), minWrapperArray->end());
}
auto WrapperArray::calcPrice() const -> int
{
```

```
    return sample->getPrice() * (int)size();
}

/* ---------------------------------------------------------
    EXPORT
--------------------------------------------------------- */
auto WrapperArray::toString() const -> string
{
    string str = "Category - " + sample->getName() + "\n";
    for (size_t i = 0; i < size(); i++)
        str += at(i)->toString() + "\n";

    return move(str);
}
```

examples/packer/Wrapper.cpp

```
#include "Wrapper.hpp"
#   include "Product.hpp"

using namespace std;
using namespace samchon::example::packer;

/* ---------------------------------------------------------
    CONSTRUCTORS
--------------------------------------------------------- */
Wrapper::Wrapper(const string &name, int price, int volume, int weight)
    : super(name, price, volume, weight),
    vector<Product*>()
{
}
Wrapper::Wrapper(const Wrapper &wrapper)
    : super(wrapper),
    vector<Product*>()
{
}

auto Wrapper::tryInsert(Product *product) -> bool
{
    int volume = 0;
    int weight = 0;

    for (size_t i = 0; i < size(); i++)
    {
        volume += at(i)->getVolume();
        weight += at(i)->getWeight();
    }

    if (product->getVolume() + volume > this->volume ||
        product->getWeight() + weight > this->weight)
    {
        return false;
    }

    push_back(product);
    return true;
}

/* ---------------------------------------------------------
    EXPORT
--------------------------------------------------------- */
auto Wrapper::toString() const -> string
{
    string str = "\tWrapper " + super::toString() + "\n";
    for (size_t i = 0; i < size(); i++)
        str += "\t\t" + at(i)->toString() + ((i == size() - 1) ? "" : "\n");

    return move(str);
};
```

examples/packer/Product.cpp

```
#pragma once
#include "Product.hpp"

using namespace std;
using namespace samchon::example::packer;

Product::Product(const string &name, int price, int volume, int weight)
    : super(name, price, volume, weight)
{
}
auto Product::toString() const -> string
{
    return "Product " + super::toString();
```

```
}
```

examples/packer/Instance.cpp

```
#include "Instance.hpp"

#include <samchon/library/StringUtil.hpp>

using namespace std;
using namespace samchon::library;
using namespace samchon::example::packer;

/* ---------------------------------------------------------
    CONSTRUCTORS
--------------------------------------------------------- */
Instance::Instance(const string &name, int price, int volume, int weight)
{
    this->name = name;
    this->price = price;
    this->volume = volume;
    this->weight = weight;
}

/* ---------------------------------------------------------
    GETTERS
--------------------------------------------------------- */
auto Instance::getName() const -> string
{
    return name;
}
auto Instance::getPrice() const -> int
{
    return price;
}
auto Instance::getVolume() const -> int
{
    return volume;
}
auto Instance::getWeight() const -> int
{
    return weight;
}

/* ---------------------------------------------------------
    EXPORT
--------------------------------------------------------- */
auto Instance::toString() const -> string
{
    return StringUtil::substitute
        (
            "{1}: ${2}, {3}cm^3, {4}g",
            name, price, volume, weight
        );
}
```

Result of the example

```
D:\OneDrive\Project\Samchon\framework\cpp\x64\Release\packer.exe
$480
Category - Large
        Wrapper Large: $100, 200cm^3, 1000g
                Product Umbrella: $4000, 200cm^3, 1000g
        Wrapper Large: $100, 200cm^3, 1000g
                Product Eraser: $500, 10cm^3, 70g
                Product Pencil: $400, 30cm^3, 35g
                Product Notebook-PC: $800000, 150cm^3, 850g

Category - Medium
        Wrapper Medium: $70, 150cm^3, 500g
                Product Drink: $1000, 75cm^3, 250g
                Product Pencil: $400, 30cm^3, 35g
                Product Pencil: $400, 30cm^3, 35g
        Wrapper Medium: $70, 150cm^3, 500g
                Product Book: $8000, 150cm^3, 300g
        Wrapper Medium: $70, 150cm^3, 500g
                Product Book: $8000, 150cm^3, 300g
        Wrapper Medium: $70, 150cm^3, 500g
                Product Tablet-PC: $600000, 120cm^3, 450g

Category - Small


Press any key to continue . . .
```

See also

samchon::library

Author

Jeongho Nam

Definition at line 56 of file CaseGenerator.hpp.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 CaseGenerator::CaseGenerator (size_t *n*, size_t *r*)

Construct from size of N and R.

Parameters

n	Size of candidates
r	Size of elements of each case

Definition at line 9 of file CaseGenerator.cpp.

10.3.3 Member Function Documentation

10.3.3.1 auto CaseGenerator::size ( ) const -> size_t

Get size of all cases.

Returns

Get a number of the all cases

Definition at line 18 of file CaseGenerator.cpp.

10.3.3.2 virtual auto samchon::library::CaseGenerator::operator[ ] (size_t) const -> std::vector< size_t > [pure virtual]

Get x'th case.

Returns

The row of the x'th in combined permuation case

Implemented in samchon::library::CombinedPermutationGenerator, and samchon::library::PermutationGenerator.

10.3.3.3 auto CaseGenerator::at (size_t *index*) const -> std::vector<size_t>

CaseGenerator::operator[ ]()

Definition at line 22 of file CaseGenerator.cpp.

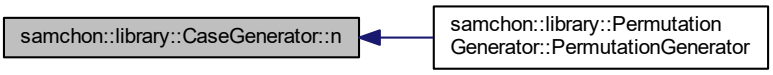
10.3.3.4 auto CaseGenerator::n ( ) const -> size_t

Get size of the N.

Definition at line 30 of file CaseGenerator.cpp.

Referenced by samchon::library::PermutationGenerator::PermutationGenerator().

Here is the caller graph for this function:



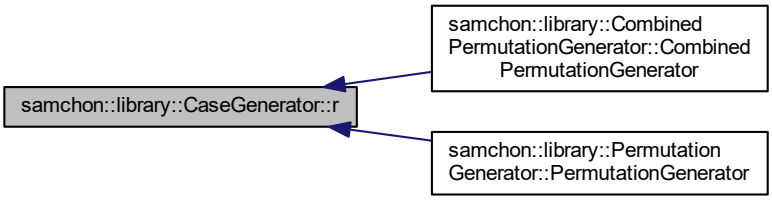
10.3.3.5 auto CaseGenerator::r ( ) const -> size_t

Get size of the R.

Definition at line 34 of file CaseGenerator.cpp.

Referenced by samchon::library::CombinedPermutationGenerator::CombinedPermutationGenerator(), and samchon::library::PermutationGenerator::PermutationGenerator().

Here is the caller graph for this function:



10.3.3.6 auto CaseGenerator::toMatrix ( ) const -> std::vector<std::vector<size_t>>

Create a matrix containing all cases.

Warning

Size of n or r is enourmouse, then stack overflow can be occured.

Returns

A matrix containing all cases.

Definition at line 42 of file CaseGenerator.cpp.

10.3.4 Member Data Documentation

10.3.4.1 size_t samchon::library::CaseGenerator::n_ [protected]

N, size of the candidates.

Definition at line 62 of file CaseGenerator.hpp.

10.3.4.2 size_t samchon::library::CaseGenerator::r_ [protected]

R, size of elements of each case.

Definition at line 67 of file CaseGenerator.hpp.

10.3.4.3 size_t samchon::library::CaseGenerator::size_ [protected]

Size, the number of all cases.

Definition at line 72 of file CaseGenerator.hpp.

Referenced by samchon::library::CombinedPermutationGenerator::CombinedPermutationGenerator(), and samchon::library::PermutationGenerator::PermutationGenerator().

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/CaseGenerator.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/CaseGenerator.cpp

10.4 samchon::library::Charset Class Reference

A utility class supporting conversion between multiple character-sets.

`#include <Charset.hpp>`

Static Public Member Functions

- static auto toMultibyte (const std::string &) -> std::string

  Convert utf-8 to multibyte.

- static auto toMultibyte (const std::wstring &) -> std::string

  Convert unicode to multibyte.

- static auto toUTF8 (const std::string &) -> std::string

  Convert multibyte to utf-8.

- static auto toUTF8 (const std::wstring &) -> std::string

  Convert unicode to utf-8.

- static auto toUnicode (const std::string &, int) -> std::wstring

  Convert multibyte or utf-8 to unicode.

10.4.1 Detailed Description

A utility class supporting conversion between multiple character-sets.

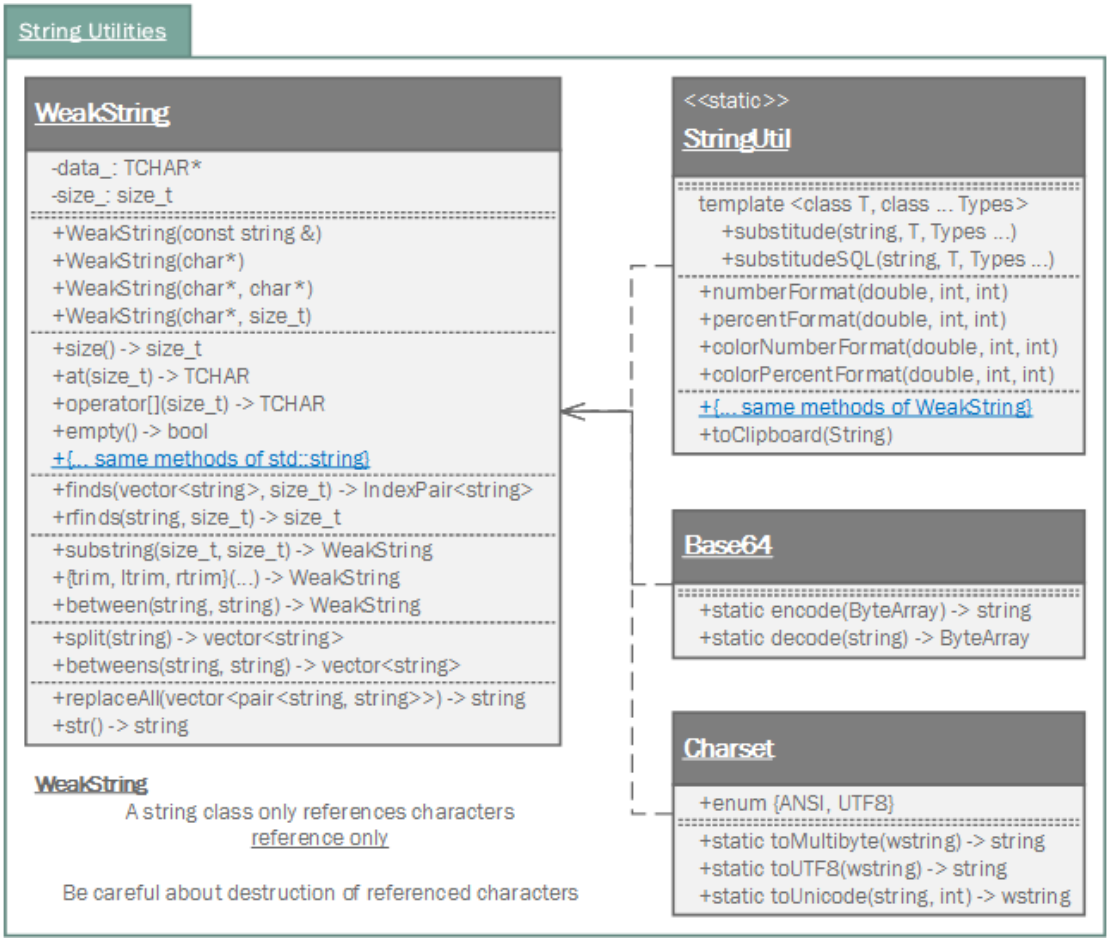
Charset class, it's easier to think iconv in linux

Supported character-sets

- Multibyte (ANSI)

- UTF-8

- Unicode (UTF-16)



Warning

In some system, std::wstring is not for utf-8 but for unicode (utf-16).

See also

samchon::library

Author

Jeongho Nam

Definition at line 30 of file Charset.hpp.

10.4.2 Member Function Documentation

10.4.2.1 static auto samchon::library::Charset::toMultibyte (const std::string &) -> std::string [static]

Convert utf-8 to multibyte.

Converts utf-8 string to multibyte string.

If the utf-8 string has header int the front, it will be erased.

Warning

If utf-8 header is not at the front of string, it can't be erased and may cause breakage on letters

Parameters

str	A utf-8 string would be converted to multibyte

Returns

A multibyte string

10.4.2.2 static auto samchon::library::Charset::toMultibyte (const std::wstring &) -> std::string [static]

Convert unicode to multibyte.

Converts unicode string to multibyte string
If the unicode string has header in the front, it will be erased

Warning

If unicode header is not at the front of string, it can't be erased and may cause breakage on letters

Parameters

str	A unicode string to be converted

Returns

A multibyte string

10.4.2.3 static auto samchon::library::Charset::toUTF8 (const std::string &) -> std::string [static]

Convert multibyte to utf-8.

Converts multibyte string to utf-8 string
If the inputted string is not multibyte, then may cause breakage on letters

Warning

Converted utf-8 string does not contain the header

Parameters

str	A multibyte string would be converted

Returns

A utf-8 string

10.4.2.4 static auto samchon::library::Charset::toUTF8 (const std::wstring &) -> std::string [static]

Convert unicode to utf-8.

Converts unicode string to utf-8 string
Generated utf-8 string follows unicode string wheter to have header or not

- If the unicode string has header on first, utf-8 will also have the header, too
- If not, generated utf-8 string doesn't have the header, either

Parameters

str	A unicode string would be converted

Returns

A utf-8 string

10.4.2.5 static auto samchon::library::Charset::toUnicode (const std::string & , int) -> std::wstring [static]

Convert multibyte or utf-8 to unicode.

Parameters

str	A multibyte or utf-8 string would be converted
Designate	str is multibyte or utf-8 string

Returns

A unicode string

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

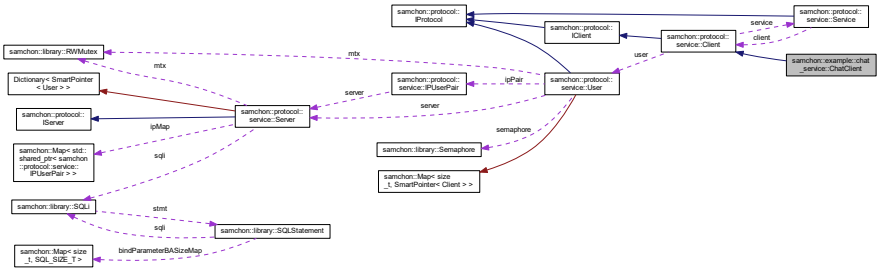
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/Charset.hpp

10.5 samchon::example::chat_service::ChatClient Class Reference

A client.

`#include <ChatClient.hpp>`

Collaboration diagram for samchon::example::chat_service::ChatClient:



Public Member Functions

- ChatClient (protocol::service::User *)

  Construct from an user.

Protected Member Functions

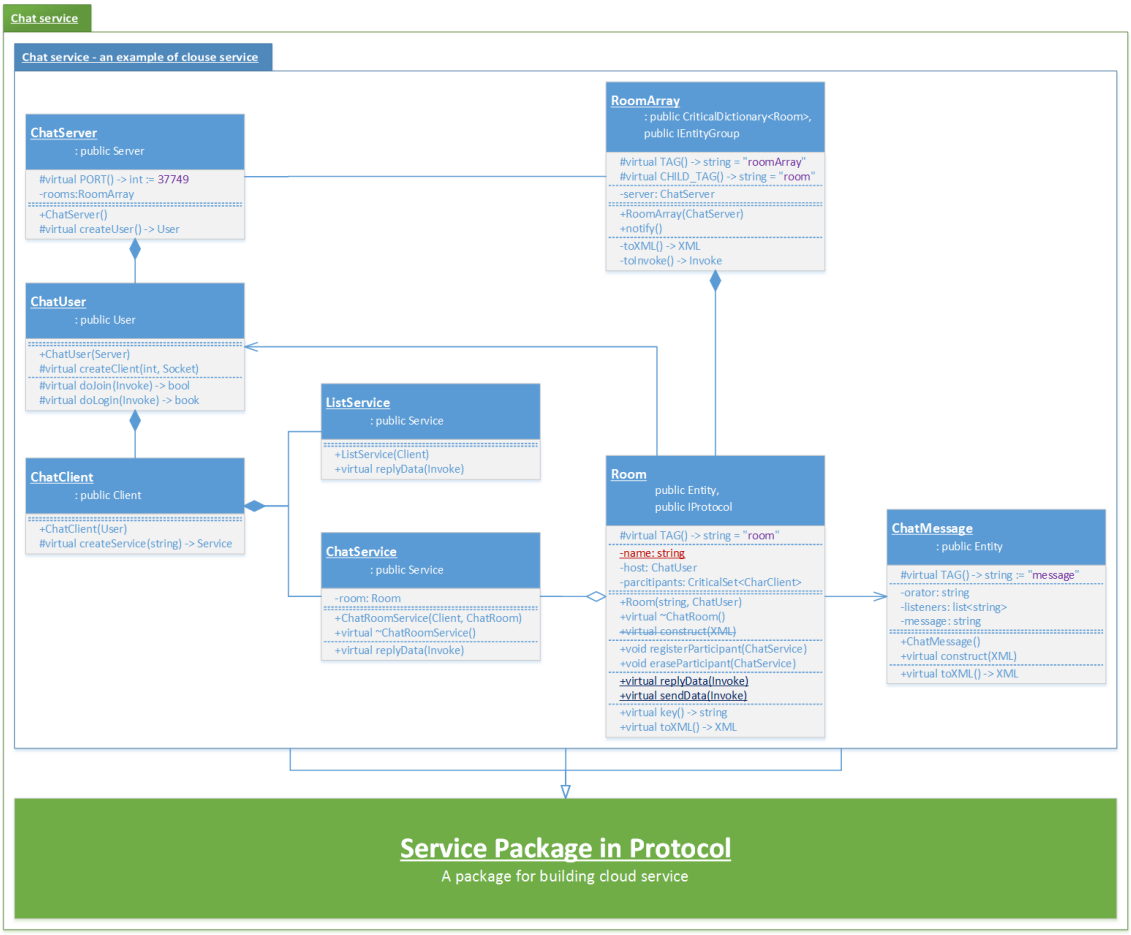
- auto createService (const std::string &) -> protocol::service::Service *override

  Factory method of Service.

Additional Inherited Members

10.5.1 Detailed Description

A client.



Author

Jeongho Nam

Definition at line 19 of file ChatClient.hpp.

10.5.2 Constructor & Destructor Documentation

10.5.2.1 samchon::example::chat_service::ChatClient::ChatClient (protocol::service::User *)

Construct from an user.

10.5.3 Member Function Documentation

10.5.3.1 auto samchon::example::chat_service::ChatClient::createService (const std::string &) -> protocol::service::Service *override [override], [protected], [virtual]

Factory method of Service.

Parameters

name	Requested name representing a Service from the (physical) client

Returns

A new Service belongs the the Client

Implements samchon::protocol::service::Client.

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

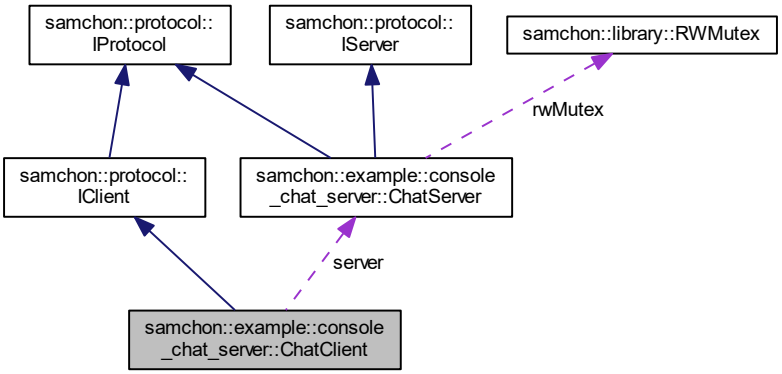
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/chat_service/ChatClient.hpp

10.6 samchon::example::console_chat_server::ChatClient Class Reference

A chat client.

```
#include <ChatClient.hpp>
```

Collaboration diagram for samchon::example::console_chat_server::ChatClient:



Public Member Functions

- ChatClient (ChatServer ∗, protocol::Socket ∗)

  Construct from a server and socket.
- virtual ∼ChatClient ()

  Default Destructor.
- virtual void replyData (std::shared_ptr< protocol::Invoke >) override

  Shift responsibility to related chain.

Private Attributes

- ChatServer ∗ server

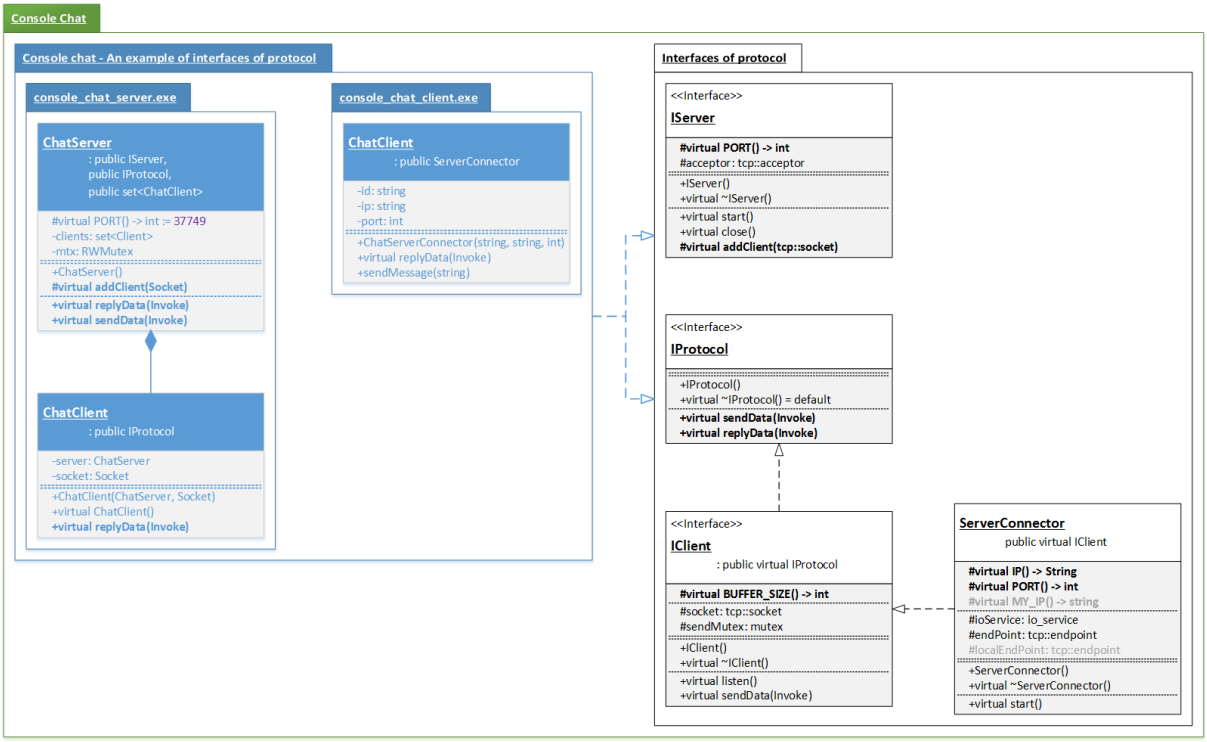
  A ChatServer that the ChatClient is belonged to.

Additional Inherited Members

10.6.1 Detailed Description

A chat client.

A simple chat client belonged to a chat server



Author

Jeongho Nam

Definition at line 21 of file ChatClient.hpp.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 ChatClient::ChatClient (ChatServer * *server*, protocol::Socket * *socket*)

Construct from a server and socket.

Parameters

server	A server the client is belonged to.
socket	A socket interacting with physical client.

Definition at line 11 of file ChatClient.cpp.

References server, and samchon::protocol::IClient::socket.

10.6.2.2 ChatClient::~ChatClient ( ) [virtual]

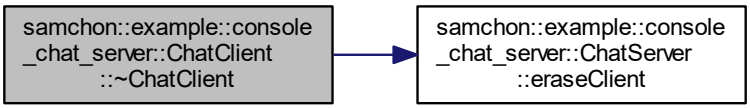
Default Destructor.

Erase from ChatServer::clientSet.

Definition at line 17 of file ChatClient.cpp.

References samchon::example::console_chat_server::ChatServer::eraseClient(), and server.

Here is the call graph for this function:



10.6.3 Member Function Documentation

10.6.3.1 void ChatClient::replyData (std::shared_ptr< protocol::Invoke >) [override], [virtual]

Shift responsibility to related chain.

Call ChatServer::replyData().

Definition at line 22 of file ChatClient.cpp.

References samchon::example::console_chat_server::ChatServer::replyData(), and server.

Here is the call graph for this function:

10.6.4 Member Data Documentation

10.6.4.1 ChatServer* samchon::example::console_chat_server::ChatClient::server [private]

A ChatServer that the ChatClient is belonged to.

Definition at line 30 of file ChatClient.hpp.

Referenced by ChatClient(), replyData(), and ~ChatClient().

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

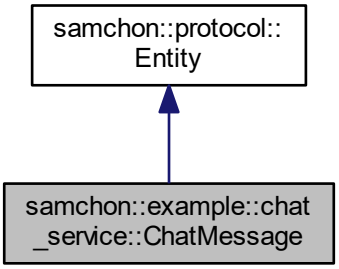
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/console_chat_server/ChatClient.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/console_chat_server/ChatClient.cpp

10.7 samchon::example::chat_service::ChatMessage Class Reference

A chat message.

```
#include <ChatMessage.hpp>
```

Collaboration diagram for samchon::example::chat_service::ChatMessage:



Public Member Functions

- ChatMessage ()

  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >)

  Construct data of the Entity from an XML object.
- auto getListener () const -> std::string

  Get listener.
- virtual auto toXML () const -> std::shared_ptr< library::XML >

  Get an XML object represents the Entity.

Protected Member Functions

- virtual auto TAG () const -> std::string

  A tag name when represented by XML.

Private Attributes

- std::string orator

  An orator, who told.
- std::string listener

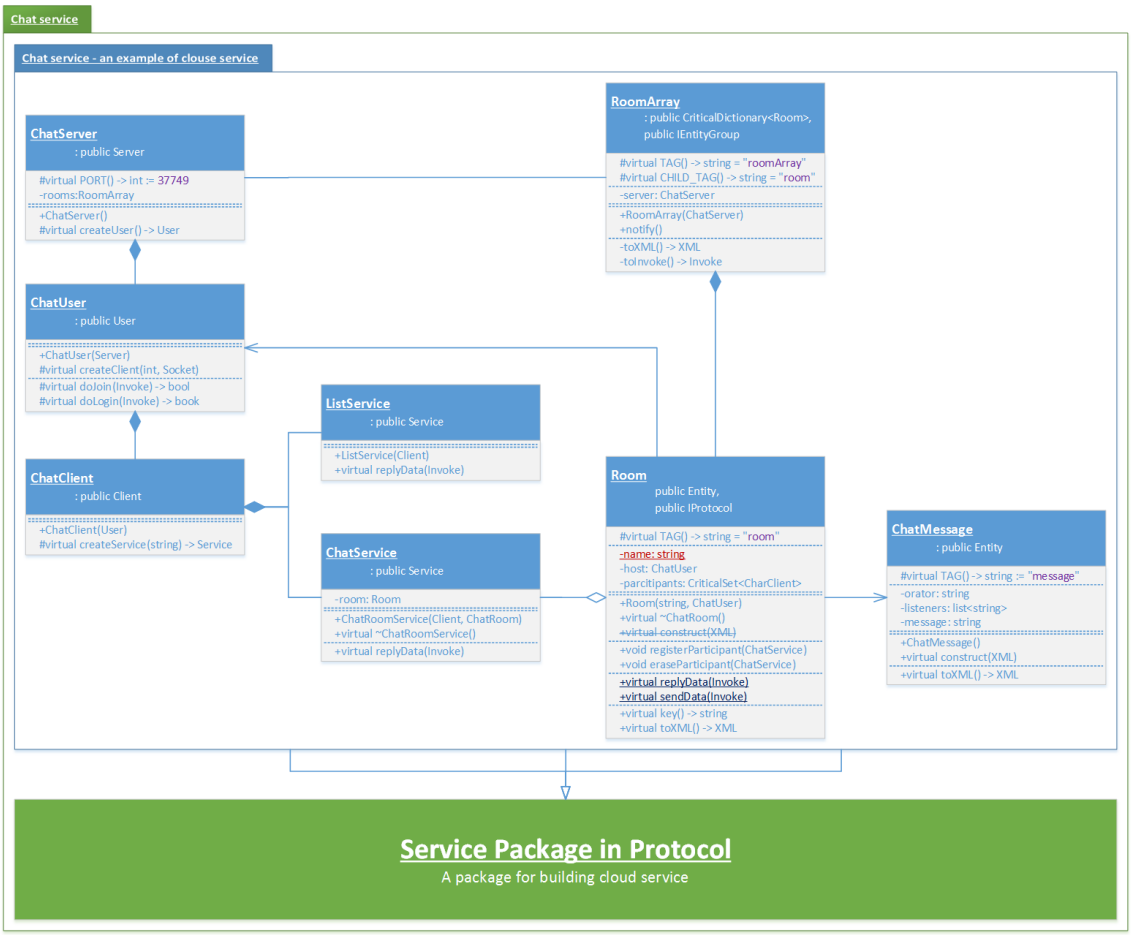
  A listener, target of whispering.
- std::string message

  A message, what to say.

10.7.1 Detailed Description

A chat message.

An entity class representing a chat message.



Author

Jeongho Nam

Definition at line 24 of file ChatMessage.hpp.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 ChatMessage::ChatMessage ( )

Default Constructor.

Definition at line 13 of file ChatMessage.cpp.

10.7.3 Member Function Documentation

10.7.3.1 virtual auto samchon::example::chat_service::ChatMessage::TAG ( ) const -> std::string [inline], [protected], [virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Definition at line 30 of file ChatMessage.hpp.

10.7.3.2 void ChatMessage::construct (std::shared_ptr< library::XML >) [virtual]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Implements samchon::protocol::Entity.

Definition at line 17 of file ChatMessage.cpp.

References listener, message, and orator.

10.7.3.3 auto ChatMessage::getListener ( ) const -> std::string

Get listener.

Used to determine which (physical) client will get the message.

Definition at line 31 of file ChatMessage.cpp.

References listener.

10.7.3.4 auto ChatMessage::toXML ( ) const -> std::shared_ptr<library::XML> [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

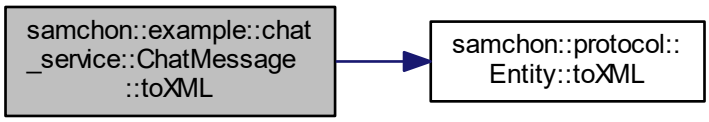
An XML object representing the Entity.

Reimplemented from samchon::protocol::Entity.

Definition at line 35 of file ChatMessage.cpp.

References listener, message, orator, and samchon::protocol::Entity::toXML().

Here is the call graph for this function:



10.7.4 Member Data Documentation

10.7.4.1 std::string samchon::example::chat_service::ChatMessage::orator [private]

An orator, who told.

Definition at line 33 of file ChatMessage.hpp.

Referenced by construct(), and toXML().

10.7.4.2 std::string samchon::example::chat_service::ChatMessage::listener [private]

A listener, target of whispering.

If there's not a listener to whisper, the value is "ALL".

Definition at line 45 of file ChatMessage.hpp.

Referenced by construct(), getListener(), and toXML().

10.7.4.3 std::string samchon::example::chat_service::ChatMessage::message [private]

A message, what to say.

Definition at line 50 of file ChatMessage.hpp.

Referenced by construct(), and toXML().

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

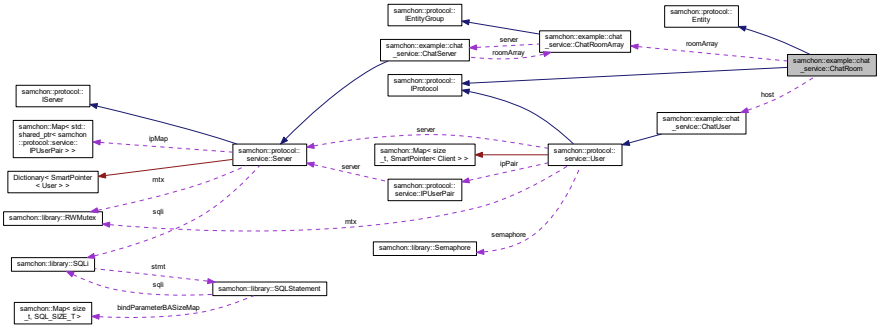
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/chat_service/ChatMessage.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/chat_service/ChatMessage.cpp

10.8 samchon::example::chat_service::ChatRoom Class Reference

A chat room.

`#include <ChatRoom.hpp>`

Collaboration diagram for samchon::example::chat_service::ChatRoom:



Public Member Functions

- ChatRoom (ChatRoomArray ∗, const std::string &, ChatUser ∗)

  Construct from rooms, name and host.
- void registerClient (ChatService ∗)

  Register a participant client.
- void eraseClient (ChatService ∗)

  Erase a participant client.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the Entity.

Protected Member Functions

- virtual auto TAG () const -> std::string

  A tag name when represented by XML.

Private Attributes

- ChatRoomArray ∗ roomArray

  All rooms that the room is belonged to.
- std::string name

  Name of the room.
- ChatUser ∗ host

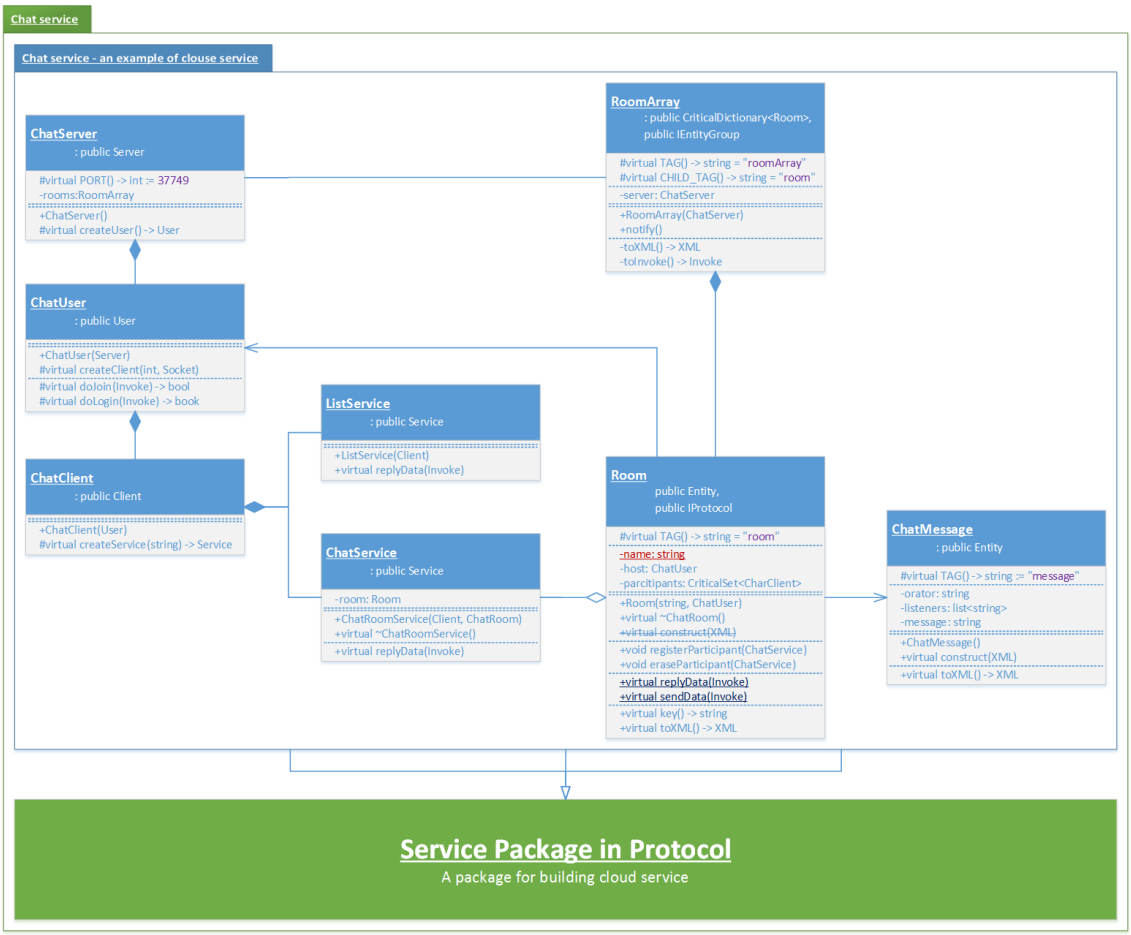
  A host, who established the room.
- library::CriticalSet< ChatService ∗ > participants

  Participants, ChatUser(s) in the room.

10.8.1 Detailed Description

A chat room.

An entity class representing a chat room containing participants. The ChatRoom class takes a responsibility of Invoke message chain between clients (participating ChatUser(s)) objects.



Author

Jeongho Nam

Definition at line 33 of file ChatRoom.hpp.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 ChatRoom::ChatRoom (ChatRoomArray *, const std::string &, ChatUser *)

Construct from rooms, name and host.

Parameters

rooms	Rooms managing ChatRoom(s).
name	Name of the room, the identifier.
host	A host who established the room.

Definition at line 20 of file ChatRoom.cpp.

References host, name, and roomArray.

10.8.3 Member Function Documentation

10.8.3.1 virtual auto samchon::example::chat_service::ChatRoom::TAG ( ) const -> std::string [inline], [protected], [virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Definition at line 40 of file ChatRoom.hpp.

10.8.3.2 void ChatRoom::registerClient (ChatService * *service*)

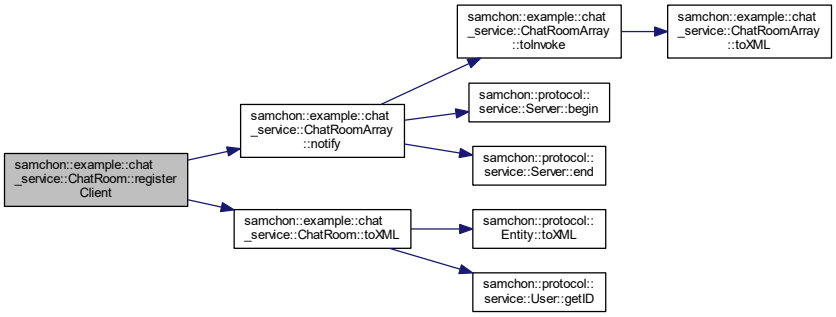
Register a participant client.

Registers a participant and notify the participation to another clients.

Definition at line 30 of file ChatRoom.cpp.

References samchon::example::chat_service::ChatRoomArray::notify(), participants, roomArray, and toXML().

Here is the call graph for this function:



10.8.3.3 void ChatRoom::eraseClient (ChatService * *service*)

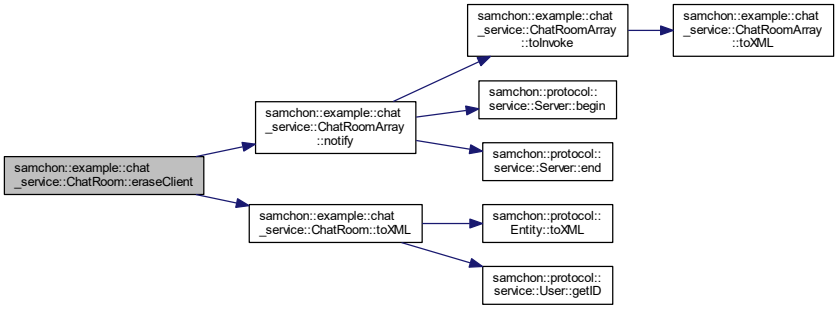
Erase a participant client.

Erases a participant and notify others about the exiting.

Definition at line 37 of file ChatRoom.cpp.

References samchon::example::chat_service::ChatRoomArray::notify(), participants, roomArray, and toXML().

Here is the call graph for this function:



10.8.3.4 auto ChatRoom::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

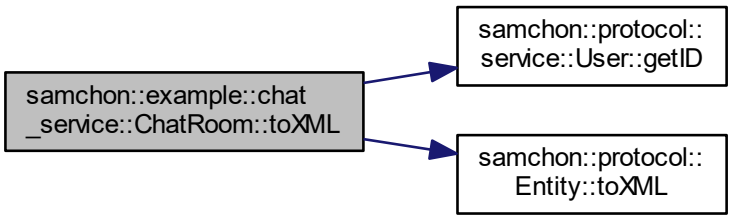
Reimplemented from samchon::protocol::Entity.

Definition at line 85 of file ChatRoom.cpp.

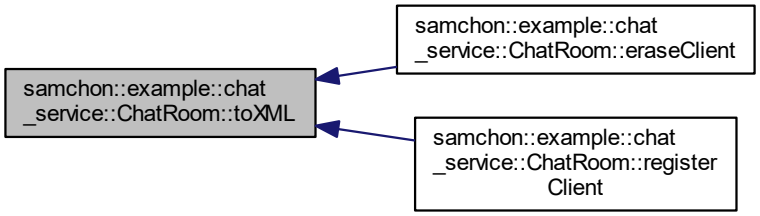
References samchon::protocol::service::User::getID(), host, name, participants, and samchon::protocol::Entity::to↩XML().

Referenced by eraseClient(), and registerClient().

Here is the call graph for this function:



Here is the caller graph for this function:



10.8.4 Member Data Documentation

10.8.4.1 ChatRoomArray* samchon::example::chat_service::ChatRoom::roomArray [private]

All rooms that the room is belonged to.

Definition at line 40 of file ChatRoom.hpp.

Referenced by ChatRoom(), eraseClient(), and registerClient().

10.8.4.2 std::string samchon::example::chat_service::ChatRoom::name [private]

Name of the room.

The name is the identifier of the ChatRoom.

Definition at line 52 of file ChatRoom.hpp.

Referenced by ChatRoom(), and toXML().

10.8.4.3 ChatUser* samchon::example::chat_service::ChatRoom::host [private]

A host, who established the room.

Definition at line 57 of file ChatRoom.hpp.

Referenced by ChatRoom(), and toXML().

10.8.4.4 library::CriticalSet<ChatService*> samchon::example::chat_service::ChatRoom::participants [private]

Participants, ChatUser(s) in the room.

Definition at line 62 of file ChatRoom.hpp.

Referenced by eraseClient(), registerClient(), and toXML().

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

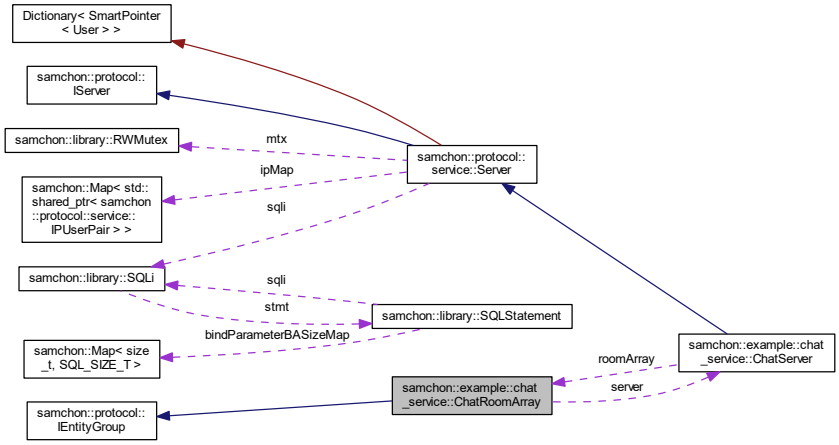
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/chat_service/ChatRoom.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/chat_service/ChatRoom.cpp

10.9 samchon::example::chat_service::ChatRoomArray Class Reference

A manager of chat rooms.

`#include <ChatRoomArray.hpp>`

Collaboration diagram for samchon::example::chat_service::ChatRoomArray:



Public Member Functions

- ChatRoomArray (ChatServer *)

  Construct from a chat server.
- void notify ()

  Notify changes of chat rooms.

Protected Member Functions

- virtual auto CHILD_TAG () const -> std::string

  A tag name of children.

Private Member Functions

- auto toXML () const -> std::shared_ptr< library::XML >

  Get an XML object representing the ChatRoomArray.
- auto toInvoke () const -> std::shared_ptr< protocol::Invoke >

  Get an Invoke message used to notifying change.

Private Attributes

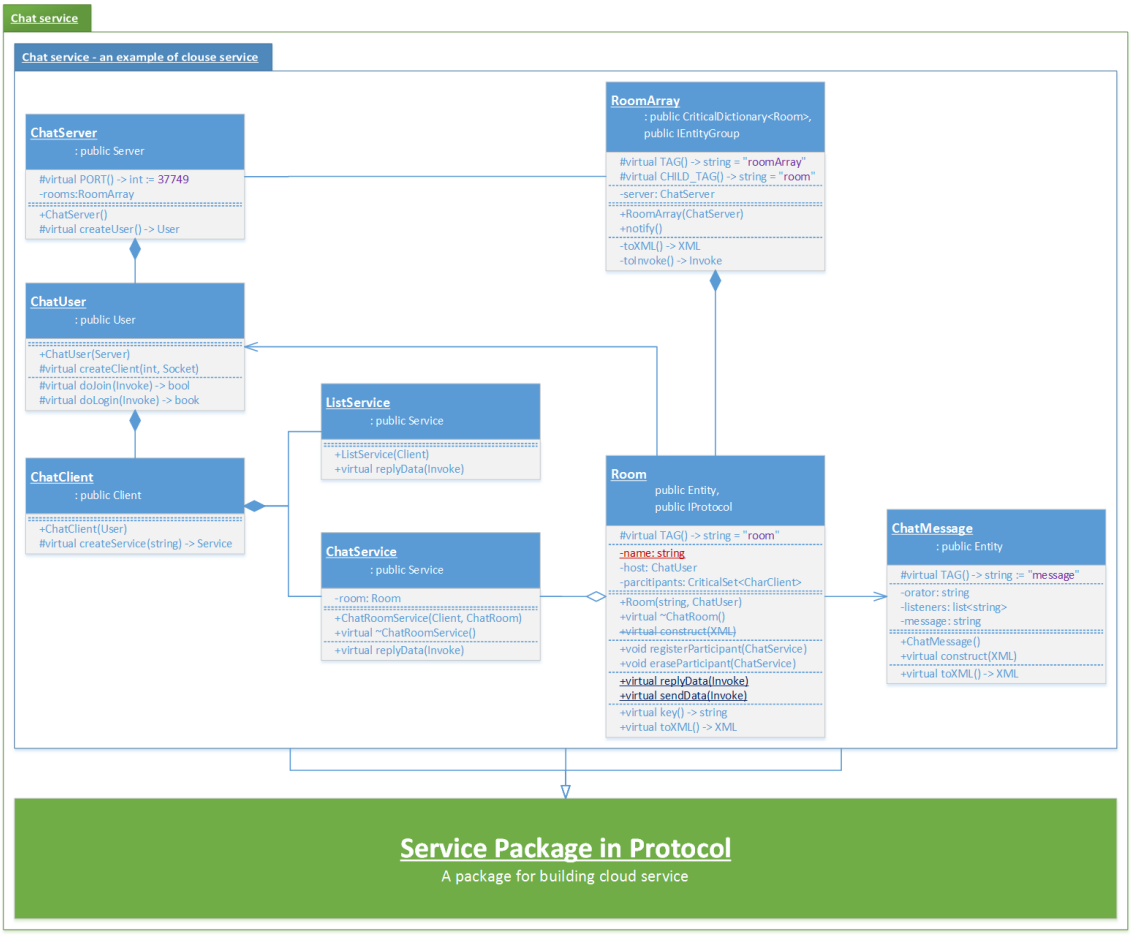
- ChatServer * server

  A chat server that rooms are belonged to.

10.9.1 Detailed Description

A manager of chat rooms.

An entity list of chat rooms and a manager of chat rooms taking a responsibility of notifying changes of children chat rooms to a related chat server.



Author

Jeongho Nam

Definition at line 31 of file ChatRoomArray.hpp.

10.9.2 Constructor & Destructor Documentation

10.9.2.1 ChatRoomArray::ChatRoomArray (ChatServer * *server*)

Construct from a chat server.

Definition at line 18 of file ChatRoomArray.cpp.

References server.

10.9.3 Member Function Documentation

10.9.3.1 virtual auto samchon::example::chat_service::ChatRoomArray::CHILD_TAG ( ) const -> std::string [inline], [protected], [virtual]

A tag name of children.

< TAG>
<CHILD_TAG />
<CHILD_TAG />
</TAG>

Implements samchon::protocol::IEntityGroup.

Definition at line 39 of file ChatRoomArray.hpp.

10.9.3.2 void ChatRoomArray::notify ( )

Notify changes of chat rooms.

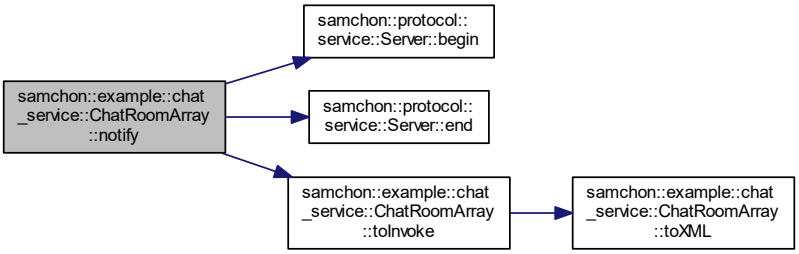
Notify changes of chat rooms to all clients in the chat server.

Definition at line 25 of file ChatRoomArray.cpp.

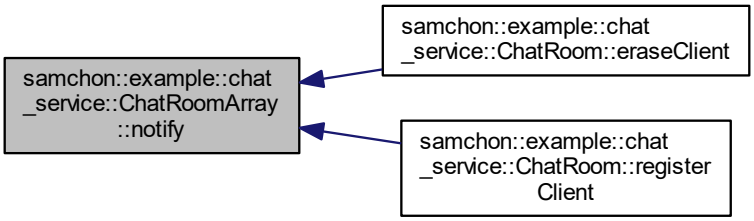
References samchon::protocol::service::Server::begin(), samchon::protocol::service::Server::end(), server, and toInvoke().

Referenced by samchon::example::chat_service::ChatRoom::eraseClient(), and samchon::example::chat_↩service::ChatRoom::registerClient().

Here is the call graph for this function:



Here is the caller graph for this function:



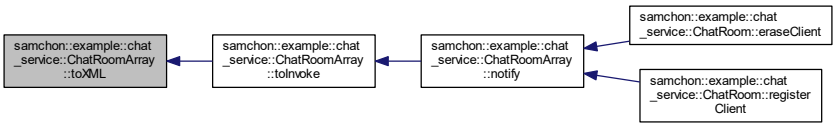
10.9.3.3 auto ChatRoomArray::toXML ( ) const -> std::shared_ptr<library::XML> [private]

Get an XML object representing the ChatRoomArray.

Definition at line 45 of file ChatRoomArray.cpp.

Referenced by toInvoke().

Here is the caller graph for this function:



10.9.3.4 auto ChatRoomArray::toInvoke ( ) const -> std::shared_ptr<protocol::Invoke> [private]

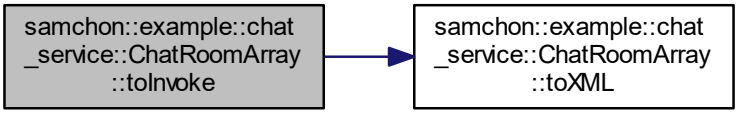
Get an Invoke message used to notifying change.

Definition at line 56 of file ChatRoomArray.cpp.

References toXML().

Referenced by notify().

Here is the call graph for this function:



Here is the caller graph for this function:

10.9.4 Member Data Documentation

10.9.4.1 ChatServer* samchon::example::chat_service::ChatRoomArray::server [private]

A chat server that rooms are belonged to.

Definition at line 39 of file ChatRoomArray.hpp.

Referenced by ChatRoomArray(), and notify().

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

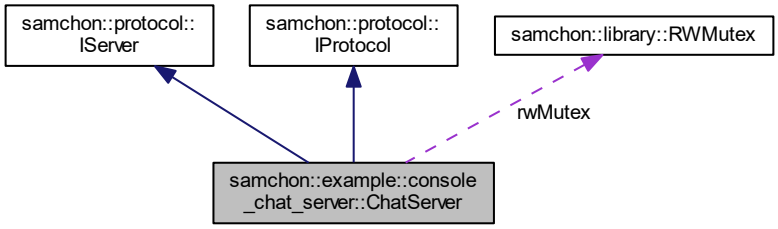
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/chat_service/ChatRoomArray.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/chat_service/ChatRoomArray.cpp

10.10 samchon::example::console_chat_server::ChatServer Class Reference

A chat server.

`#include <ChatServer.hpp>`

Collaboration diagram for samchon::example::console_chat_server::ChatServer:



Public Member Functions

- ChatServer ()

  Default Constructor.
- void eraseClient (ChatClient *client)

  Erase client when disconnected.
- virtual void replyData (std::shared_ptr< protocol::Invoke >) override

  Handling replied message.
- virtual void sendData (std::shared_ptr< protocol::Invoke >) override

  Sending a message.

Protected Member Functions

- virtual auto PORT () const -> int override

  Port number: #33749.
- virtual void addClient (protocol::Socket *socket) override

  Add a client.

Private Attributes

- std::set< ChatClient * > clientSet

  List of clients.
- library::RWMutex rwMutex

  A rw_mutex managing clientSet.

Additional Inherited Members

10.10.1 Detailed Description

A chat server.

Author

Jeongho Nam

Definition at line 30 of file ChatServer.hpp.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 ChatServer::ChatServer ( )

Default Constructor.

Definition at line 12 of file ChatServer.cpp.

10.10.3 Member Function Documentation

10.10.3.1 void ChatServer::eraseClient (ChatClient * *client*)

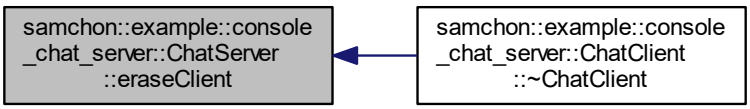
Erase client when disconnected.

Definition at line 25 of file ChatServer.cpp.

References clientSet, and rwMutex.

Referenced by samchon::example::console_chat_server::ChatClient::~ChatClient().

Here is the caller graph for this function:



10.10.3.2 virtual auto samchon::example::console_chat_server::ChatServer::PORT ( ) const -> int [inline], [override], [protected], [virtual]

Port number: #33749.

Reimplemented from samchon::protocol::IServer.

Definition at line 63 of file ChatServer.hpp.

10.10.3.3 void ChatServer::addClient (protocol::Socket * *socket*) [override], [protected], [virtual]

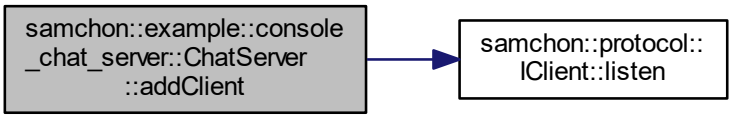
Add a client.

Implements samchon::protocol::IServer.

Definition at line 16 of file ChatServer.cpp.

References clientSet, samchon::protocol::IClient::listen(), and rwMutex.

Here is the call graph for this function:



10.10.3.4 void ChatServer::replyData (std::shared_ptr< protocol::Invoke >) [override], [virtual]

Handling replied message.

When got a message from a client, sends the message to all clients.

Parameters

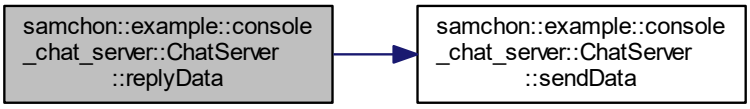
invoke	An invoke message represents a chat.

Definition at line 31 of file ChatServer.cpp.

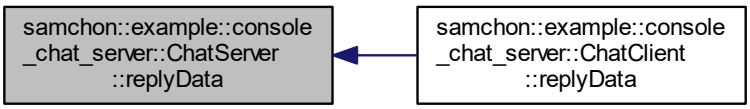
References sendData().

Referenced by samchon::example::console_chat_server::ChatClient::replyData().

Here is the call graph for this function:



Here is the caller graph for this function:



10.10.3.5 void ChatServer::sendData (std::shared_ptr< protocol::Invoke >) [override], [virtual]

Sending a message.

Send a message to all clients. Called by replyData().

Parameters

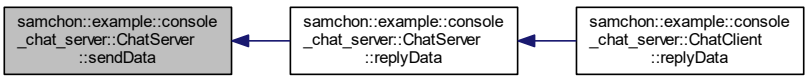
invoke	An invoke message represents a chat.

Definition at line 35 of file ChatServer.cpp.

References clientSet, and rwMutex.

Referenced by replyData().

Here is the caller graph for this function:



10.10.4 Member Data Documentation

10.10.4.1 std::set<ChatClient*> samchon::example::console_chat_server::ChatServer::clientSet [private]

List of clients.

Definition at line 40 of file ChatServer.hpp.

Referenced by addClient(), eraseClient(), and sendData().

10.10.4.2 library::RWMutex samchon::example::console_chat_server::ChatServer::rwMutex [private]

A rw_mutex managing clientSet.

Definition at line 45 of file ChatServer.hpp.

Referenced by addClient(), eraseClient(), and sendData().

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

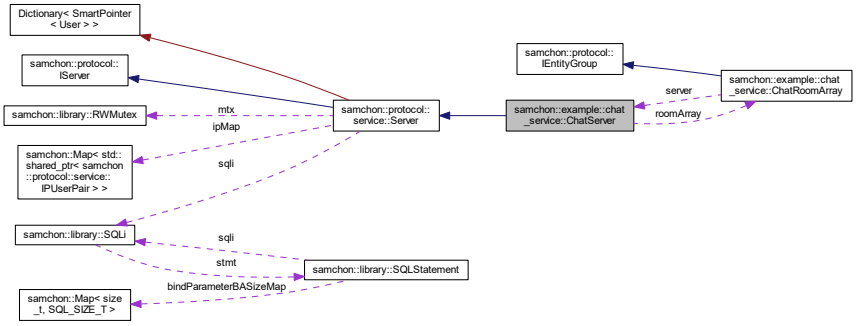
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/console_chat_server/ChatServer.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/console_chat_server/ChatServer.cpp

10.11 samchon::example::chat_service::ChatServer Class Reference

A chat server.

`#include <ChatServer.hpp>`

Collaboration diagram for samchon::example::chat_service::ChatServer:



Public Member Functions

- ChatServer ()

  Default Constructor.

- auto getChatRoomArray () const -> ChatRoomArray *

  Get chat room list.

Protected Member Functions

- virtual auto PORT () const -> int override

  Port number of the server.

- virtual auto createUser () -> protocol::service::User *override

  Factory method of User.

Private Attributes

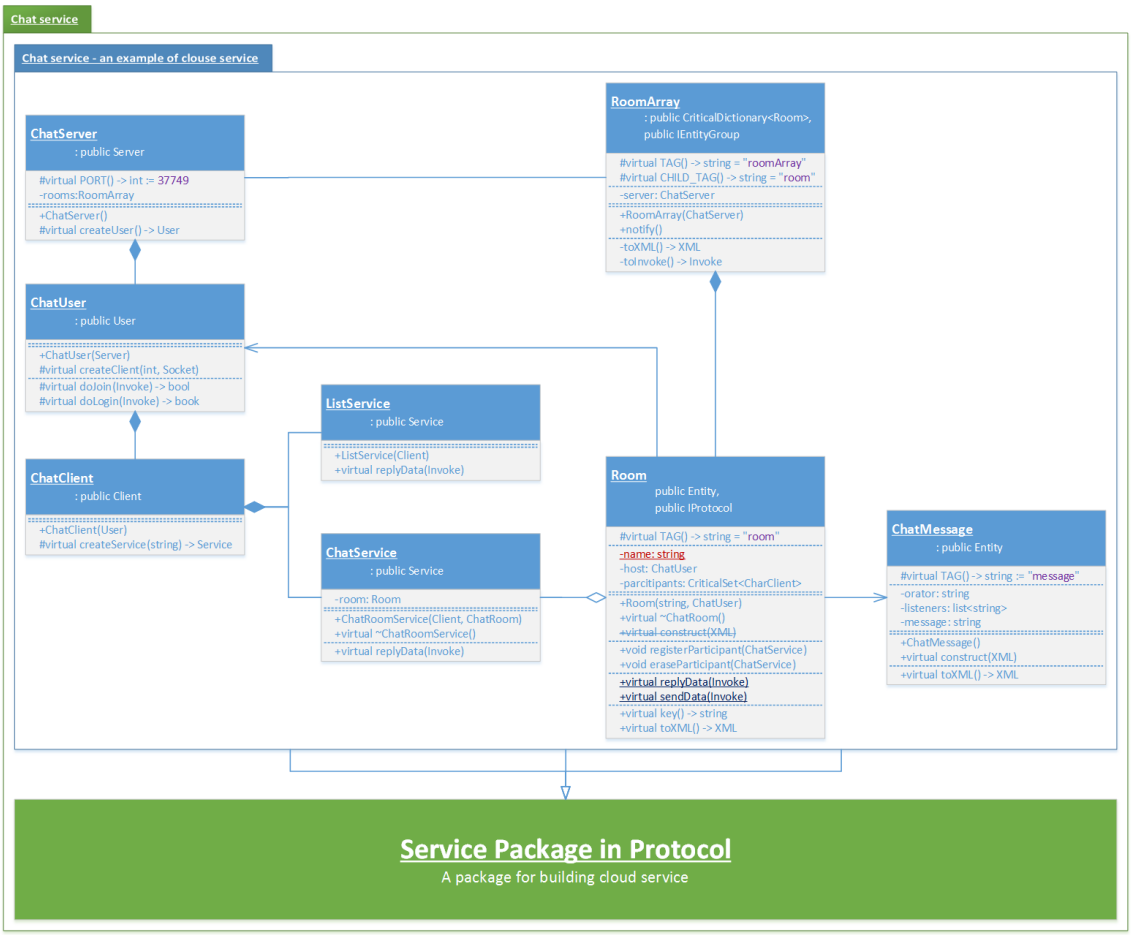
- ChatRoomArray * roomArray

  A manager of chat rooms.

Additional Inherited Members

10.11.1 Detailed Description

A chat server.



Author

Jeongho Nam

Definition at line 22 of file ChatServer.hpp.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 samchon::example::chat_service::ChatServer::ChatServer ( )

Default Constructor.

10.11.3 Member Function Documentation

10.11.3.1 virtual auto samchon::example::chat_service::ChatServer::PORT ( ) const -> int [inline], [override], [protected], [virtual]

Port number of the server.

Reimplemented from samchon::protocol::IServer.

Definition at line 28 of file ChatServer.hpp.

10.11.3.2 virtual auto samchon::example::chat_service::ChatServer::createUser ( ) -> protocol::service::User *override [override], [protected], [virtual]

Factory method of User.

Implements samchon::protocol::service::Server.

10.11.3.3 auto samchon::example::chat_service::ChatServer::getChatRoomArray ( ) const -> ChatRoomArray *

Get chat room list.

10.11.4 Member Data Documentation

10.11.4.1 ChatRoomArray* samchon::example::chat_service::ChatServer::roomArray [private]

A manager of chat rooms.

Definition at line 31 of file ChatServer.hpp.

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

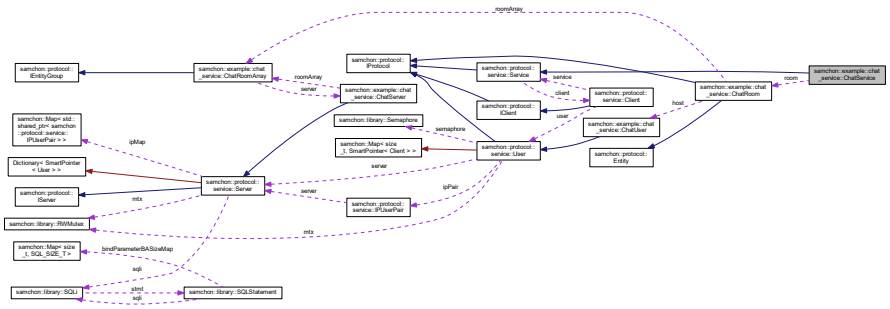
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/chat_service/ChatServer.hpp

10.12 samchon::example::chat_service::ChatService Class Reference

A chat service.

```
#include <ChatService.hpp>
```

Collaboration diagram for samchon::example::chat_service::ChatService:



Public Member Functions

- ChatService (protocol::service::Client *)

  Construct from a client.

Private Attributes

- ChatRoom * room

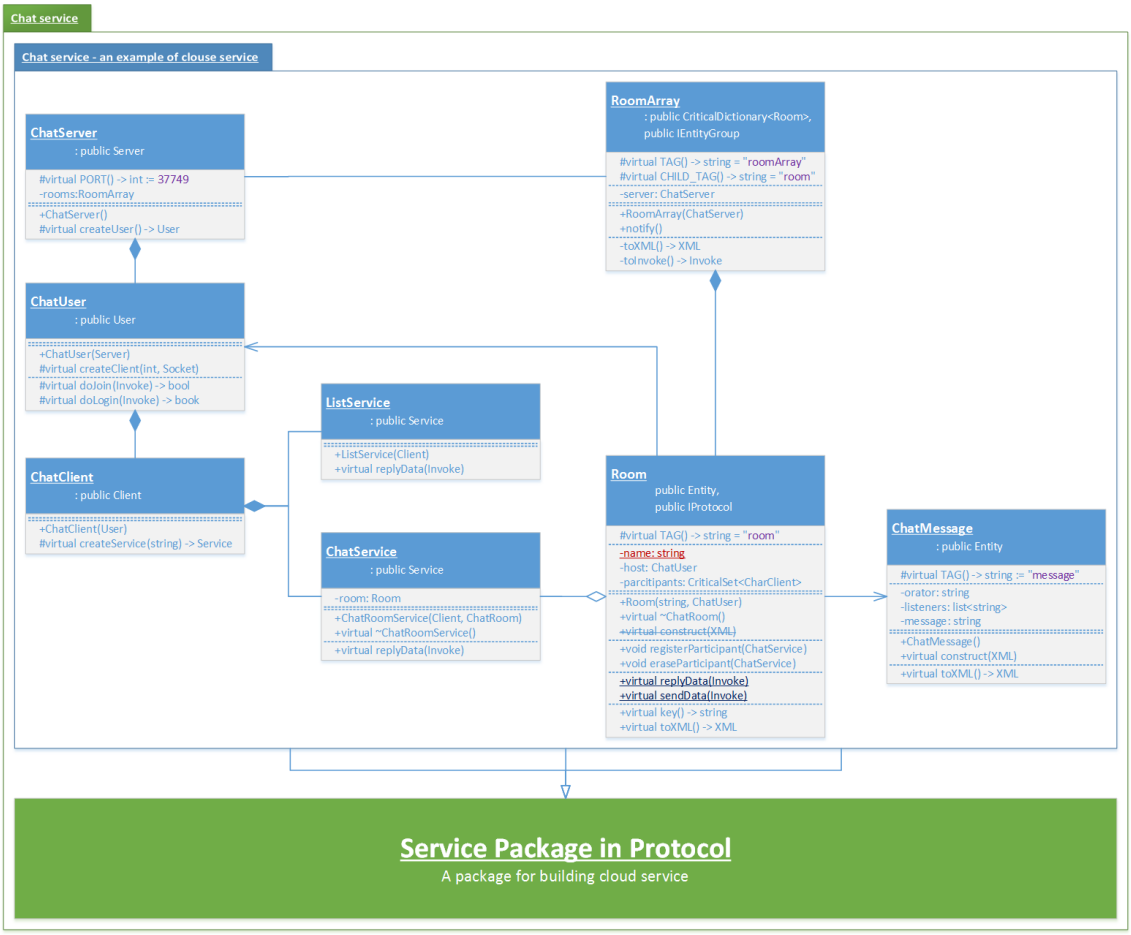
  A chat room.

Additional Inherited Members

10.12.1 Detailed Description

A chat service.

A chat service about a chat room.



Author

Jeongho Nam

Definition at line 22 of file ChatService.hpp.

10.12.2 Constructor & Destructor Documentation

10.12.2.1 ChatService::ChatService (protocol::service::Client * *client*)

Construct from a client.

Definition at line 11 of file ChatService.cpp.

10.12.3 Member Data Documentation

10.12.3.1 ChatRoom* samchon::example::chat_service::ChatService::room [private]

A chat room.

A chat room that related client is participating in.

Definition at line 32 of file ChatService.hpp.

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

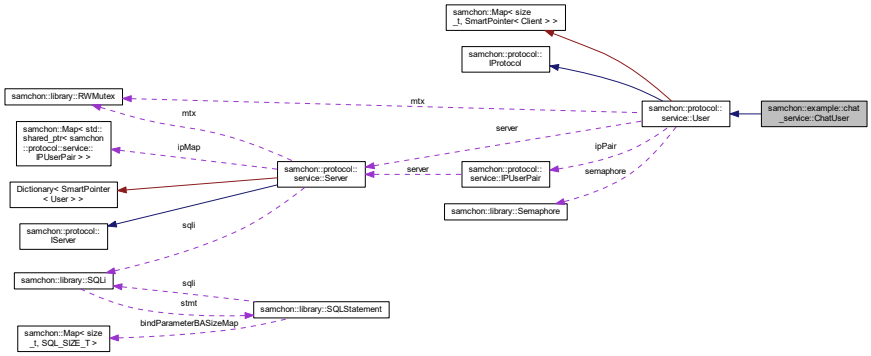
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/chat_service/ChatService.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/chat_service/ChatService.cpp

10.13 samchon::example::chat_service::ChatUser Class Reference

An user in chat server.

`#include <ChatUser.hpp>`

Collaboration diagram for samchon::example::chat_service::ChatUser:



Public Member Functions

- ChatUser (protocol::service::Server ∗)

  Construct from server.

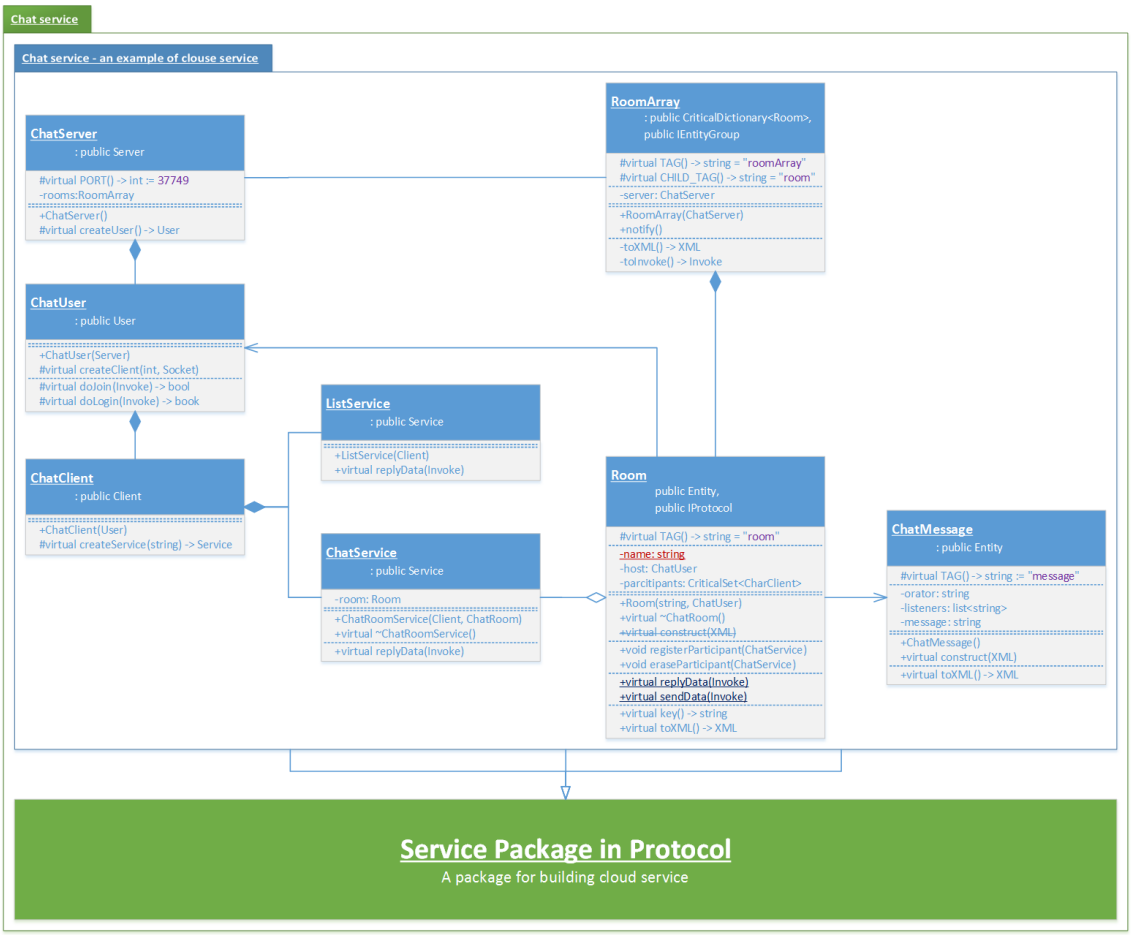
Protected Member Functions

- virtual auto createClient () -> protocol::service::Client ∗override

  Factory method of a Client.

10.13.1 Detailed Description

An user in chat server.



Author

Jeongho Nam

Definition at line 19 of file ChatUser.hpp.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 samchon::example::chat_service::ChatUser::ChatUser (protocol::service::Server *)

Construct from server.

10.13.3 Member Function Documentation

10.13.3.1 virtual auto samchon::example::chat_service::ChatUser::createClient ( ) -> protocol::service::Client *override [override], [protected], [virtual]

Factory method of a Client.

Implements samchon::protocol::service::User.

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

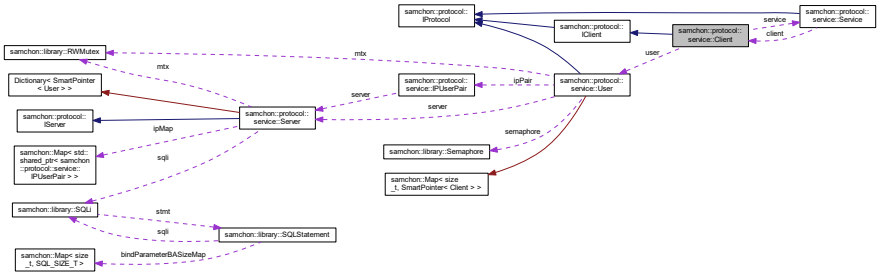
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/chat_service/ChatUser.hpp

10.14 samchon::protocol::service::Client Class Reference

A network boundary with the client in an User.

`#include <Client.hpp>`

Collaboration diagram for samchon::protocol::service::Client:



Public Member Functions

- Client (User ∗)

  Construct from User, No and socket.
- auto getUser () const -> User ∗

  Get User.
- auto getService () const -> Service ∗

  Get Service.
- auto getNo () const -> size_t

  Get no.
- void sendData (std::shared_ptr< Invoke >)

  Send Invoke message to (physical) client.
- void replyData (std::shared_ptr< Invoke >)

  Reply Invoke message from (physical) client.

Protected Member Functions

- virtual auto createService (const std::string &) -> Service ∗=0

  Factory method of Service.

Protected Attributes

- User ∗ user

  An User of the Client.
- size_t no

  A sequence number of the Client in an User.
- Service ∗ service

  A Service belongs to the Client.

Private Member Functions

- void constructService (const std::string &)

  Construct Service.

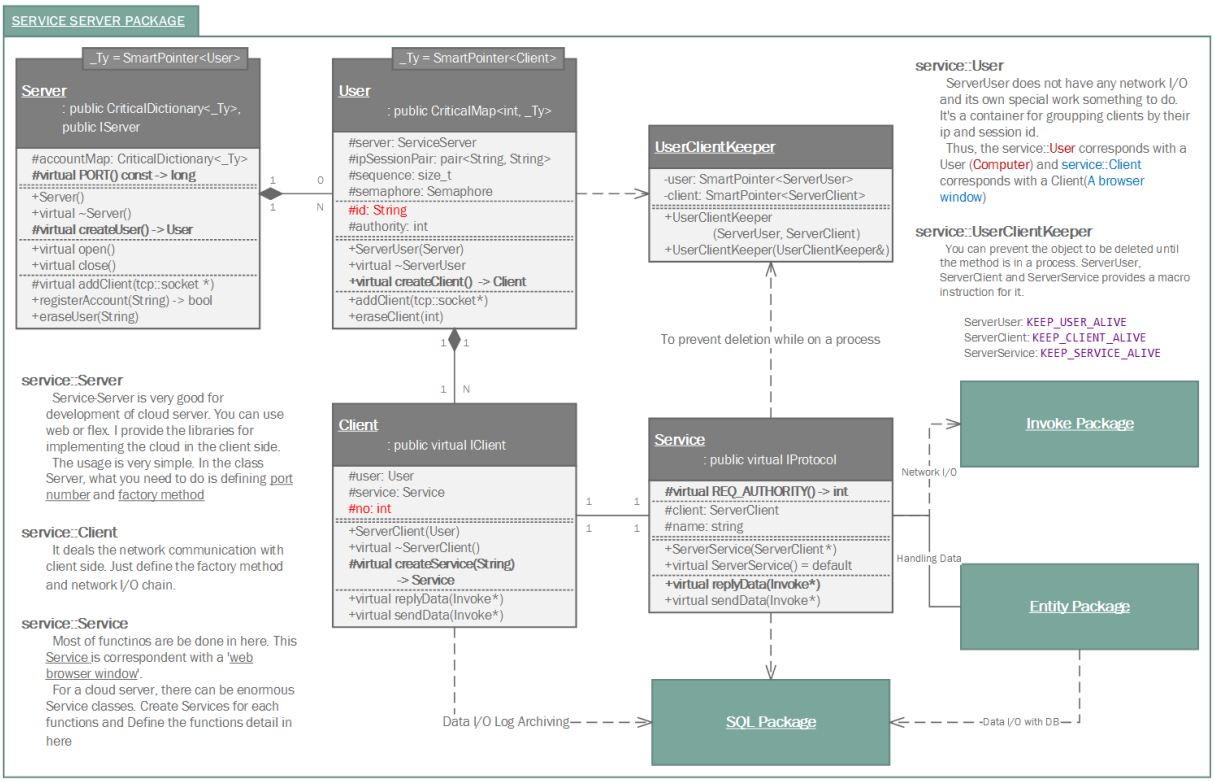
10.14.1 Detailed Description

A network boundary with the client in an User.

Client is an object interacting with physical client by socket.

The Client is correspond with a Window in JS (UI) 1:1

- A Window, the object in UI
- A Window, matching with window of a internet browser



Note

Method to override

- createUser()

See also

samchon::protocol
samchon::protocol::service

Author

Jeongho Nam

Definition at line 45 of file Client.hpp.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 Client::Client (User * *user*)

Construct from User, No and socket.

Parameters

user	An User containing the Client

Definition at line 30 of file Client.cpp.

References user.

10.14.3 Member Function Documentation

10.14.3.1 virtual auto samchon::protocol::service::Client::createService (const std::string &) -> Service * [protected],[pure virtual]

Factory method of Service.

Parameters

name	Requested name representing a Service from the (physical) client

Returns

A new Service belongs the the Client

Implemented in samchon::example::chat_service::ChatClient.

Referenced by constructService().

Here is the caller graph for this function:

10.14.3.2 auto Client::getUser ( ) const -> User*

Get User.

Definition at line 48 of file Client.cpp.

References user.

10.14.3.3 auto Client::getService ( ) const -> Service*

Get Service.

Definition at line 52 of file Client.cpp.

References service.

10.14.3.4 auto Client::getNo ( ) const -> size_t

Get no.

Definition at line 56 of file Client.cpp.

References no.

10.14.3.5 void Client::sendData (std::shared_ptr< Invoke >) [virtual]

Send Invoke message to (physical) client.

Method sendData of Client does not only send Invoke message to client system, but also archives the Invoke message to database system as a historical-log.

Note

Method sendData monopolies a critical section.

How many threads have called that sendData, the Invoke messages are delivered sequentially.

- Using sendData with a new thread is not recommended

Warning

You can't override this method, sendData.

Parameters

invoke	Invoke message to send

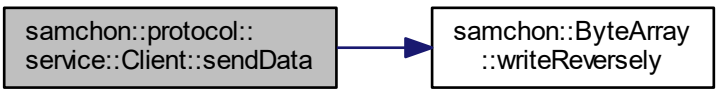
Reimplemented from samchon::protocol::IClient.

Definition at line 64 of file Client.cpp.

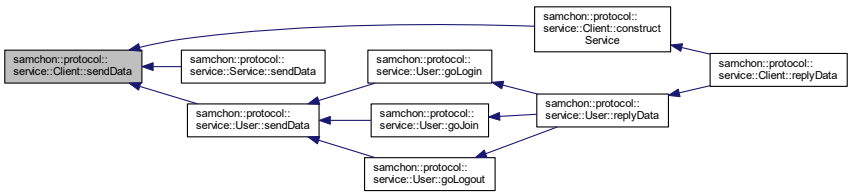
References samchon::protocol::IClient::sendMtx, samchon::protocol::IClient::socket, and samchon::ByteArray↩ ::writeReversely().

Referenced by constructService(), samchon::protocol::service::Service::sendData(), and samchon::protocol↩ ::service::User::sendData().

Here is the call graph for this function:



Here is the caller graph for this function:



10.14.3.6 void Client::replyData (std::shared_ptr< Invoke >) [virtual]

Reply Invoke message from (physical) client.

Handles replied Invoke message from client system.

1. Constructs Service
   - Constructs Service by requested service name.
   - Notifies client system whether the user satisfies the authority.
2. Shifts responsibility to related chain.
   - Invoke message is about member (join, login, etc.), shift to User.
   - Except that all, shifts to Service.
3. Archives the Invoke message to Database system as a historical-log.
   - Invoke to HISTORY_INVOKE table, with User's session and Client's sequence id.
   - InvokeParameters(s) to HISTORY_INVOKE_PARAMETER with key of HISTORY_INVOKE as foriegn key.

Note

Historical-log

HISTORY_INVOKE and HISTORY_INVOKE_PARAMETER has a 1:N relationship

If you want to modify archiving method, override method of Server::archiveReplyData()

Shifting responsibility to Service

Service::replyData will be called by a new thread. The thread will acquire an admission from semaphore in User to avoid exhausted allocation of threads.

Exception handling

If an std::exception has thrown from Service::replyData, the exception will archived in Database (HISTORY_INV↩OKE_EXCEPTION)

The most famous thrown parameters like std::exception, std::invalid_argument and std::runtime_error, those are all derived from std::exception.
-> http://www.cplusplus.com/reference/stdexcept/

- std::exception
  - std::logic_error
    - std::domain_error
    - std::invalid_argument
    - std::length_error
    - std::out_of_range
    - std::future_error

- std::runtime_error
  - std::range_error
  - std::overflow_error
  - std::underflow_error
  - std::system_error

Warning

You can't override this method, replyData.

If you want to modify replyData's rule, override related objects' replyData

Parameters

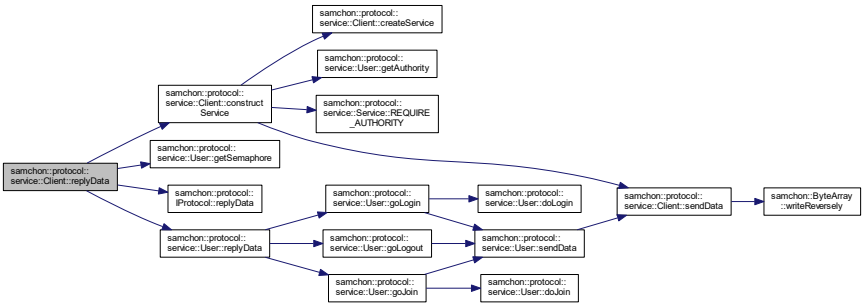
invoke	Invoke message replied from client system

Reimplemented from samchon::protocol::IProtocol.

Definition at line 94 of file Client.cpp.

References constructService(), samchon::protocol::service::User::getSemaphore(), samchon::protocol::IProtocol↩ ::replyData(), samchon::protocol::service::User::replyData(), service, and user.

Here is the call graph for this function:



10.14.3.7 void Client::constructService (const std::string & *name*) [private]

Construct Service.

Constructs Service by requested name from client and notify whether the User is satisfying authority.

Parameters

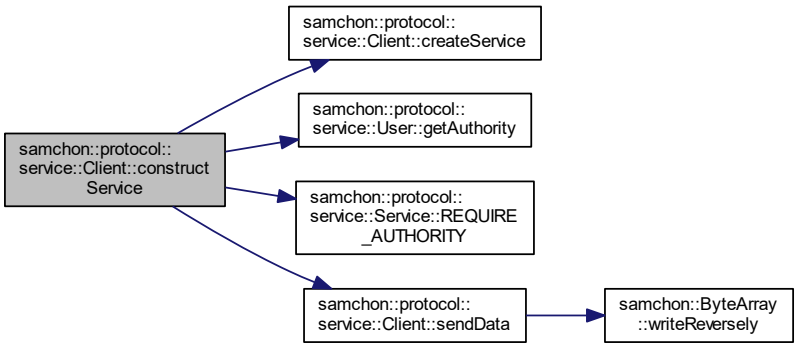
name	A name representing type of Service

Definition at line 123 of file Client.cpp.

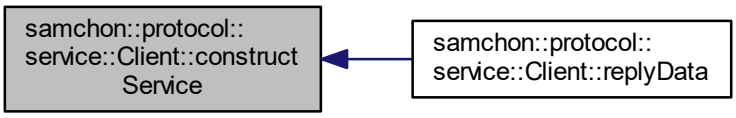
References createService(), samchon::protocol::service::User::getAuthority(), samchon::protocol::service::↩ Service::name, samchon::protocol::service::Service::REQUIRE_AUTHORITY(), sendData(), service, and user.

Referenced by replyData().

Here is the call graph for this function:



Here is the caller graph for this function:



10.14.4 Member Data Documentation

10.14.4.1 User* samchon::protocol::service::Client::user [protected]

An User of the Client.

Definition at line 58 of file Client.hpp.

Referenced by Client(), constructService(), getUser(), and replyData().

10.14.4.2 size_t samchon::protocol::service::Client::no [protected]

A sequence number of the Client in an User.

Definition at line 63 of file Client.hpp.

Referenced by getNo().

10.14.4.3 Service* samchon::protocol::service::Client::service [protected]

A Service belongs to the Client.

Definition at line 68 of file Client.hpp.

Referenced by constructService(), getService(), and replyData().

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

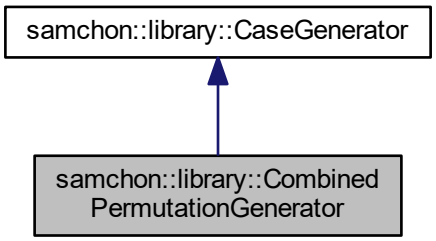
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/service/Client.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/service/Client.cpp

10.15 samchon::library::CombinedPermutationGenerator Class Reference

A combined-permutation case generator.

`#include <CombinedPermutationGenerator.hpp>`

Collaboration diagram for samchon::library::CombinedPermutationGenerator:



Public Member Functions

- CombinedPermutationGenerator (size_t n, size_t r)

  Construct from size of N and R.

- virtual auto operator[ ] (size_t) const -> std::vector< size_t > override

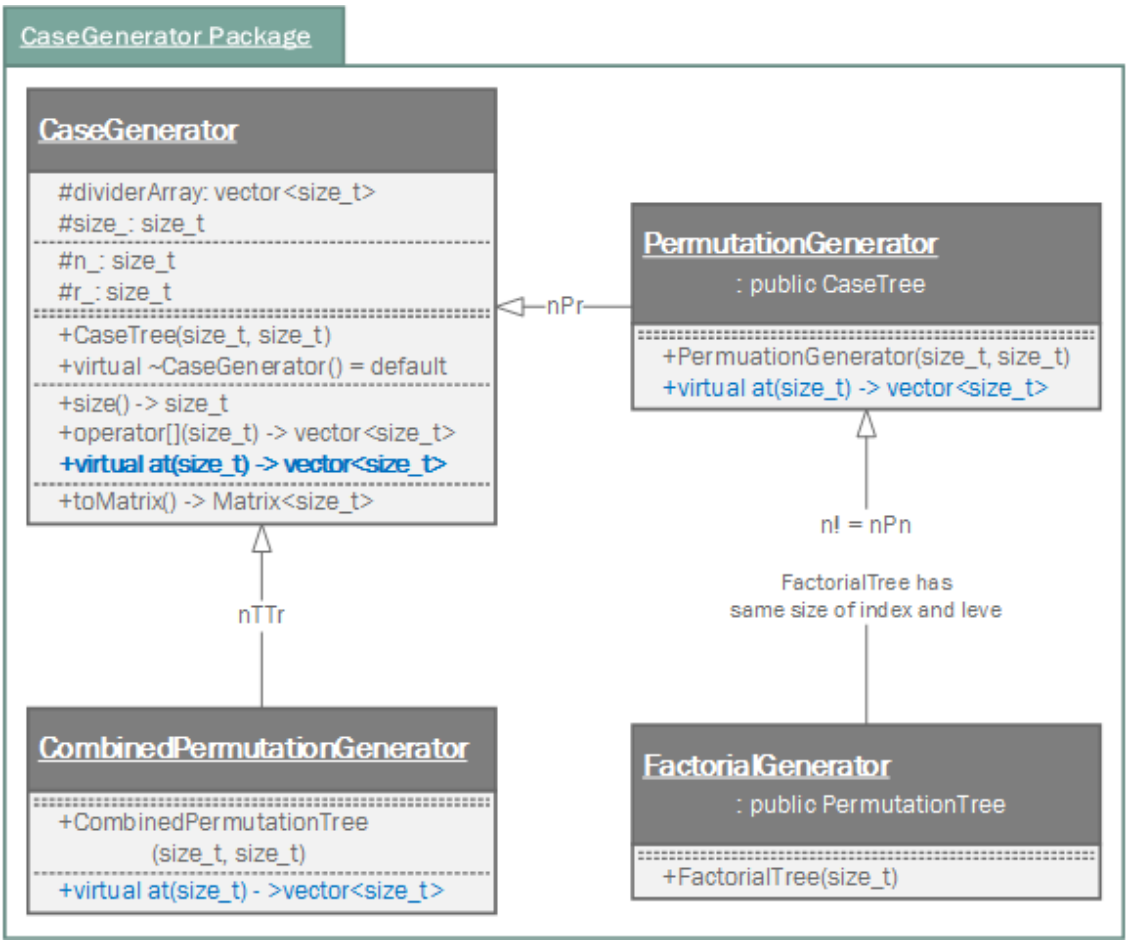
  Get x'th case.

Additional Inherited Members

10.15.1 Detailed Description

A combined-permutation case generator.

${}_{n}TT_{r}$



See also

samchon::library

Author

Jeongho Nam

Definition at line 21 of file CombinedPermutationGenerator.hpp.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 CombinedPermutationGenerator::CombinedPermutationGenerator (size_t *n*, size_t *r*)

Construct from size of N and R.

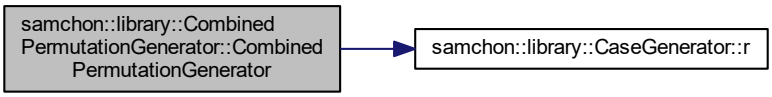
Parameters

n	Size of candidates
r	Size of elements of each case

Definition at line 6 of file CombinedPermutationGenerator.cpp.

References samchon::library::CaseGenerator::r(), and samchon::library::CaseGenerator::size_.

Here is the call graph for this function:



10.15.3 Member Function Documentation

10.15.3.1 auto CombinedPermutationGenerator::operator[ ] (size_t) const -> std::vector< size_t > [override], [virtual]

Get x'th case.

Returns

The row of the x'th in combined permuation case

Implements samchon::library::CaseGenerator.

Definition at line 18 of file CombinedPermutationGenerator.cpp.

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

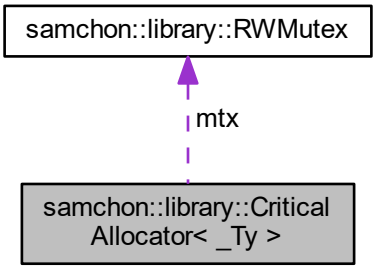
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/CombinedPermutationGenerator.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/CombinedPermutationGenerator.cpp

10.16 samchon::library::CriticalAllocator< _Ty > Class Template Reference

An allocator ensuring concurrency.

`#include <CriticalAllocator.hpp>`

Collaboration diagram for samchon::library::CriticalAllocator< _Ty >:



Public Member Functions

- auto getMutex () -> RWMutex &

  Get rw_mutex.
- template<class _U , class... _Args>
  void construct (_U *ptr, _Args &&...args)

  Lock when constructed.
  [Inherited] Construct an object.
  .
- template<class _U >
  void destroy (_U *ptr)

  Lock when destroyed
  [Inherited] Destory an object.
- auto allocate (size_type n, std::allocator< void >::const_pointer hint=NULL) -> pointer

  Lock when allocated
  [Inherited] Allocate block of storage.
- void deallocate (pointer ptr, size_type size)

  Lock when deallocated
  [Inherited] Release block of storage.

Private Attributes

- RWMutex mtx

  A rw_mutex for handling concurrency.

10.16.1 Detailed Description

template<class _Ty>class samchon::library::CriticalAllocator< _Ty >

An allocator ensuring concurrency.

CriticalAllocator is a std::allocator ensuring concurrency.

CriticalAllocator keeps safety in multi-threading environment automatically. But it ensures only safety. If a logic needs a mutual exclusion not only level of the container, it'd better to avoid using the CriticalAllocator and use RWMutex by yourself.

[Inherited]]

Allocators are classes that define memory models to be used by some parts of the Standard Library, and most specifically, by STL containers.

This section describes the default allocator template allocator (lowercase). This is the allocator that all standard containers will use if their last (and optional) template parameter is not specified, and is the only predefined allocator in the standard library. </p<

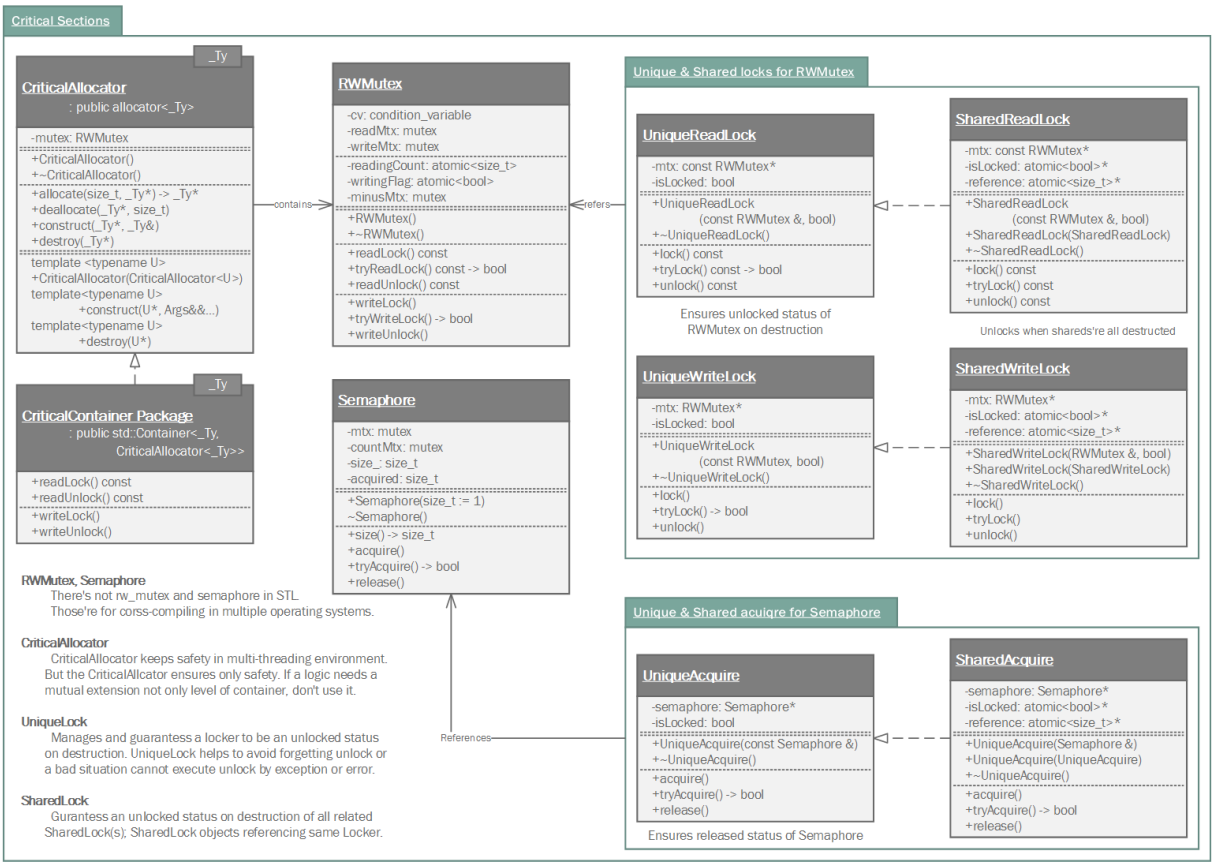
Other allocators may be defined. Any class Alloc for which allocator_traits<Alloc> produces a valid instantiation with the appropriate members defined can be used as an allocator on standard containers (Alloc may or may not implement the functionality through member functions). </p<

Except for its destructor, no member of the standard default allocator class template shall introduce data races. Calls to member functions that allocate or deallocate storage shall occur in a single total order, and each such deallocation shall happen before the next allocation (if any) in this order.

Technically, a memory model described by allocators might be specialized for each type of object to be allocated and even may store local data for each container they work with. Although this does not happen with the default allocator.

Referenced comments of std::allocator

- http://www.cplusplus.com/reference/memory/allocator/



Template Parameters

_Ty	Type of the elements allocated by the object (aliased as member type value_type).

See also

samchon::library

Author

Jeongho Nam

Definition at line 50 of file CriticalAllocator.hpp.

10.16.2 Member Function Documentation

10.16.2.1 template<class _Ty > auto samchon::library::CriticalAllocator< _Ty >::getMutex ( ) -> RWMutex& [inline]

Get rw_mutex.

Definition at line 70 of file CriticalAllocator.hpp.

References samchon::library::CriticalAllocator< _Ty >::mtx.

10.16.2.2 template<class _Ty > template<class _U , class... _Args> void samchon::library::CriticalAllocator< _Ty >::construct (_U * *ptr,* _Args &&... *args*) [inline]

Lock when constructed.
[Inherited] Construct an object.
.

Locks a mutex for concurrency when an elmented is constructed.

[Inherited]

Constructs an element object on the location pointed by p.

Notice that this does not allocate space for the element. It should already be available at p (see member allocate to allocate space).

Template Parameters

_U	
_Args	

Parameters

ptr	Pointer to a location with enough storage space to contain an element of type _U.
args	Arguments forwarded to the constructor. Args is a list of zero or more types.

Definition at line 104 of file CriticalAllocator.hpp.

10.16.2.3 template<class _Ty > template<class _U > void samchon::library::CriticalAllocator< _Ty >::destroy (_U * *ptr*) [inline]

Lock when destroyed
[Inherited] Destory an object.

Locks a mutex for concurrency when an child is destroyed

[Inherited]

Destroys in-place the object pointed by p.

Notice that this does not deallocate the storage for the element (see member deallocate to release storage space).

Template Parameters

_U	

Parameters

ptr	Pointer to the object to be destroyed.

Definition at line 129 of file CriticalAllocator.hpp.

10.16.2.4 template<class _Ty > auto samchon::library::CriticalAllocator< _Ty >::allocate (size_type *n*, std::allocator< void >::const_pointer *hint* = NULL) -> pointer [inline]

Lock when allocated
[Inherited] Allocate block of storage.

Locks a mutex for concurrency when children elements are deallocated.

[Inherited]

Attempts to allocate a block of storage with a size large enough to contain n elements of member type value_type (an alias of the allocator's template parameter), and returns a pointer to the first element.

The storage is aligned appropriately for objects of type value_type, but they are not constructed.

In the standard default allocator, the block of storage is allocated using ::operator new one or more times, and throws bad_alloc if it cannot allocate the total amount of storage requested.

Parameters

n	Number of elements (each of size sizeof(value_type)) to be allocated. The member type size_type is an alias of size_t (in the standard default allocator) size_t is an unsigned integral type.
hint	Either 0 or a value previously obtained by another call to allocate and not yet freed with deallocate. When it is not 0, this value may be used as a hint to improve performance by allocating the new block near the one specified. The address of an adjacent element is often a good choice.

Returns

A pointer to the initial element in the block of storage.

pointer and const_pointer are member types (defined as aliases of T∗ and const T∗ respectively in std::allocator).

The standard default allocator throws bad_alloc if it cannot allocate the requested amount of storage.

Definition at line 168 of file CriticalAllocator.hpp.

10.16.2.5 template<class _Ty > void samchon::library::CriticalAllocator< _Ty >::deallocate (pointer *ptr*, size_type *size*) [inline]

Lock when deallocated
[Inherited] Release block of storage.

Locks a mutex for concurrency when children elements are deallocated.

[Inherited]

Releases a block of storage previously allocated with member allocate and not yet released.

The elements in the array are not destroyed by a call to this member function.

In the default allocator, the block of storage is at some point deallocated using ::operator delete (either during the function call, or later).

Parameters

ptr	Pointer to a block of storage previously allocated with allocator::allocate. pointer is a member type (defined as an alias of T∗ in std::allocator<_Ty>).
size	Number of elements allocated on the call to allocator::allocate for this block of storage. The member type size_type is an alias of size_t (in the standard default allocator). size_t is an unsigned integral type.

Definition at line 196 of file CriticalAllocator.hpp.

10.16.3 Member Data Documentation

10.16.3.1 template<class _Ty > RWMutex samchon::library::CriticalAllocator< _Ty >::mtx [private]

A rw_mutex for handling concurrency.

Definition at line 59 of file CriticalAllocator.hpp.

Referenced by samchon::library::CriticalAllocator< _Ty >::getMutex().

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/CriticalAllocator.hpp

10.17 samchon::library::Date Class Reference

Date (year, month, day in month)

```
#include <Date.hpp>
```

Public Member Functions

- Date ()

  Default Constructor.
- Date (const super &)

  Copy Constructor.
- Date (super &&)
- Date (int year, int month, int date)

  Construct from date elements.
- Date (const WeakString &)

  Constructor by std::string.
- Date (long long linuxTime)

  Construct from linux_time.
- void set (const std::string &)

  Setter by string.
- virtual void set (const WeakString &)

  Setter by weak_string.
- void set (int year, int month, int date)

  Setter by elements of date.
- void set (long long linuxTime)

  Setter by linux_time.
- virtual void setYear (int)

  Set year of the Date.
- virtual void setMonth (int)

Set month of the Date.

- virtual void setDate (int)
- virtual void addYear (int)

  Add years to the Date.
- virtual void addMonth (int)

  Add months to the Date.
- virtual void addWeek (int)

  Add weeks to the Date.
- virtual void addDate (int)

  Add days to the Date.
- auto getYear () const -> int

  Get year of the Date.
- auto getMonth () const -> int

  Get month of the Date.
- auto getDate () const -> int

  Get the day in month of the Date.
- auto getDay () const -> int

  Get the day in week of the Date.
- auto toLinuxTime () const -> long long

  Converts the Date to linux_time.
- virtual auto toString () const -> std::string

  Converts the Date to std::string.

Protected Member Functions

- auto toTM () const -> struct::tm

  Converts the Date to struct tm.

Static Protected Member Functions

- static auto calcSeconds (int year, int month, int date) -> long long

  Calculates how many seconds have flowen since 0000-01-01 00:00:00.
- static auto calcLastDates (int year) -> std::array< int, 12 >

  Calculates and gets an array of final date of each month for that year.

Static Protected Attributes

- static std::chrono::system_clock::time_point TP_1970 = chrono::system_clock::from_time_t(0)

  time_point for 1970-01-01 09:00:00

10.17.1 Detailed Description

Date (year, month, day in month)

Date class is a chrono::time_point<std::chrono::system_clock>.

Date represents the date with year, month and day in month. The basic date (local time) is from your operating system.

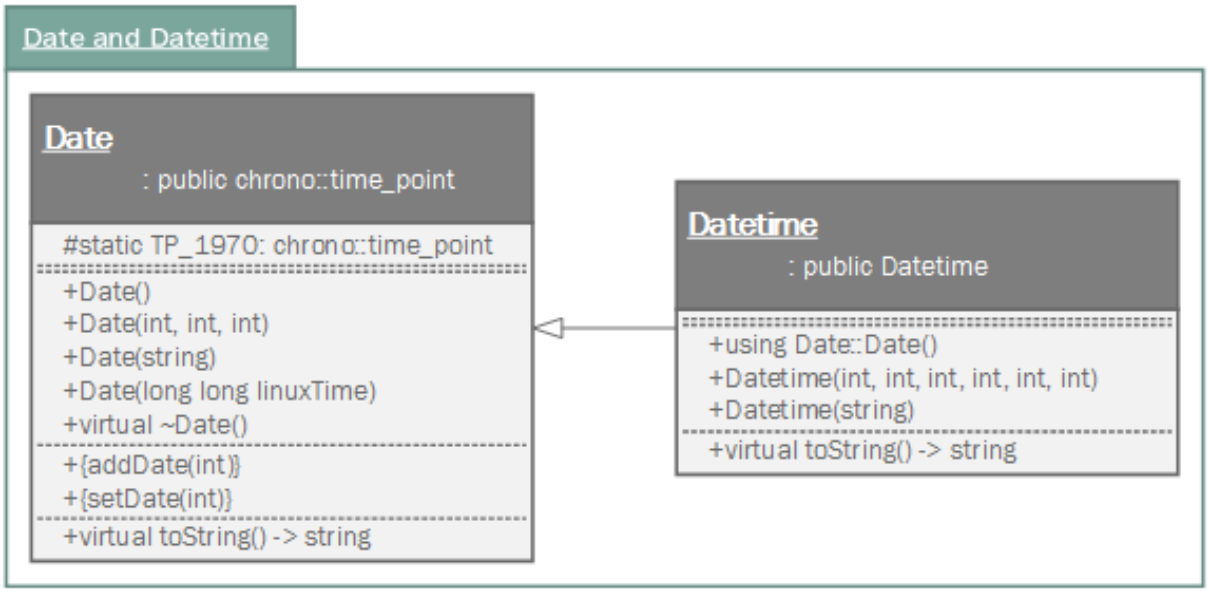
[Inherited]

A time_point object expresses a point in time relative to a clock's epoch.

Internally, the object stores an object of a duration type, and uses the Clock type as a reference for its epoch.

Referenced comments of std::chrono::time_point

- http://www.cplusplus.com/reference/chrono/time_point/



See also

samchon::library

Author

Jeongho Nam

Definition at line 37 of file Date.hpp.

10.17.2 Constructor & Destructor Documentation

10.17.2.1 Date::Date ( )

Default Constructor.

The date will be now (Now's date from operating system)

Definition at line 18 of file Date.cpp.

10.17.2.2 Date::Date (const super & *date*)

Copy Constructor.

Copy the date from another Date

Parameters

date	Target to be copied

Definition at line 23 of file Date.cpp.

10.17.2.3 Date::Date (super && *date*)

Move Constructor

Gets the date from other and truncate the other

Parameters

date	Target to be moved

Definition at line 27 of file Date.cpp.

10.17.2.4 Date::Date (int *year*, int *month*, int *date*)

Construct from date elements.

Parameters

year	ex) 2015
month	January to December. 1 $\sim$ 12
date	day in month. 1 $\sim$ 31

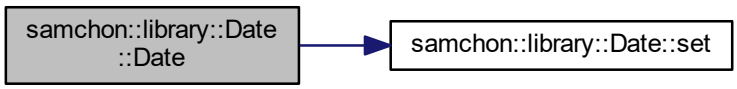
Exceptions

invalid_argument	month or date is invalid.

Definition at line 32 of file Date.cpp.

References set().

Here is the call graph for this function:



10.17.2.5 Date::Date (const WeakString & *wStr*)

Constructor by std::string.

Parameters

str	A weak_string expressing the date. (1991-01-01)

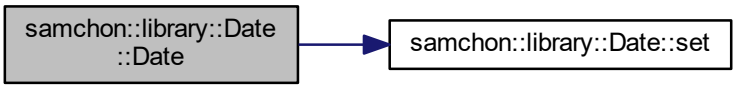
Exceptions

invalid_argument	month or date is invalid.

Definition at line 36 of file Date.cpp.

References set().

Here is the call graph for this function:



10.17.2.6 Date::Date (long long *linuxTime*)

Construct from linux_time.

Parameters

linuxTime	linux_time to be converted

Definition at line 40 of file Date.cpp.

10.17.3 Member Function Documentation

10.17.3.1 long long Date::calcSeconds (int *year*, int *month*, int *date*) -> long long [static],[protected]

Calculates how many seconds have flowen since 0000-01-01 00:00:00.

This static method is used for calculate the linux_time (since 1970-01-01 09:00:00)

Returns

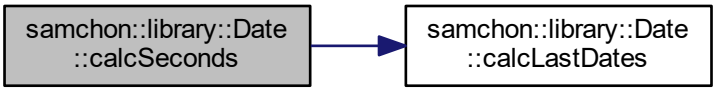
linux_time of the Date

Definition at line 97 of file Date.cpp.

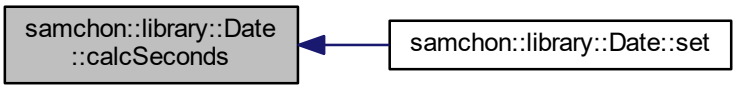
References calcLastDates().

Referenced by set().

Here is the call graph for this function:



Here is the caller graph for this function:



10.17.3.2 auto Date::calcLastDates (int *year*) -> std::array<int, 12> [static], [protected]

Calculates and gets an array of final date of each month for that year.

Parameters

year	The target year

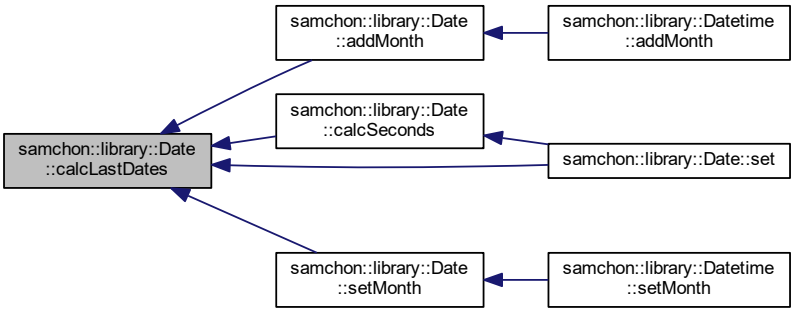
Returns

Array of fianl date of each month for that year

Definition at line 85 of file Date.cpp.

Referenced by addMonth(), calcSeconds(), set(), and setMonth().

Here is the caller graph for this function:



10.17.3.3 void samchon::library::Date::set (const std::string &)

Setter by string.

Parameters

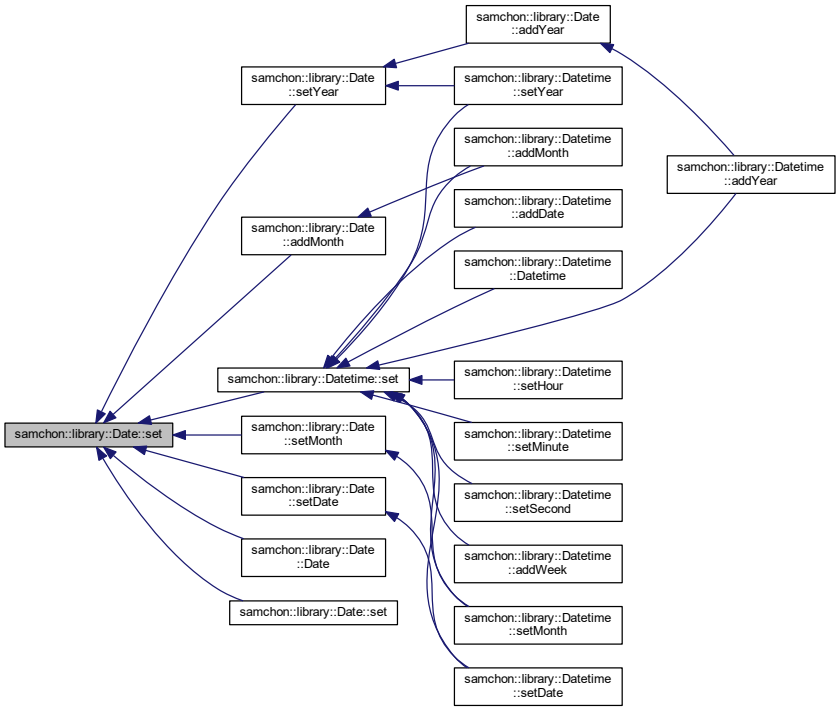
str	Y-m-d

Exceptions

invalid_argument	month or date is invalid.

Referenced by addMonth(), Date(), samchon::library::Datetime::set(), set(), setDate(), setMonth(), and setYear().

Here is the caller graph for this function:



10.17.3.4 void Date::set (const WeakString & *wStr*) [virtual]

Setter by weak_string.

Parameters

wstr	Y-m-d

Exceptions

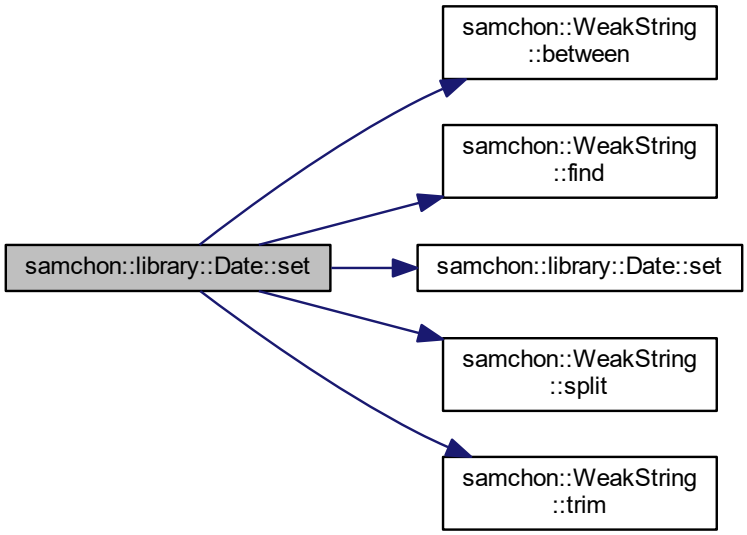
invalid_argument	month or date is invalid.

Reimplemented in samchon::library::Datetime.

Definition at line 46 of file Date.cpp.

References samchon::WeakString::between(), samchon::WeakString::find(), set(), samchon::WeakString::split(), and samchon::WeakString::trim().

Here is the call graph for this function:



10.17.3.5 void Date::set (int *year*, int *month*, int *date*)

Setter by elements of date.

Parameters

year	ex) 2015
month	January to December. 1 ~ 12
date	day in month. 1 ~ 31

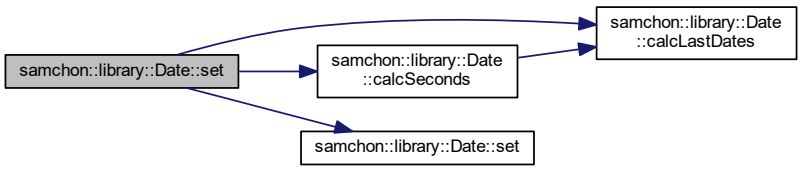
Exceptions

invalid_argument	month or date is invalid.

Definition at line 64 of file Date.cpp.

References calcLastDates(), calcSeconds(), and set().

Here is the call graph for this function:



10.17.3.6 void Date::set (long long *linuxTime*)

Setter by linux_time.

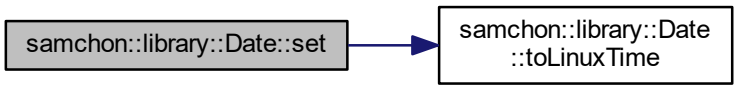
Parameters

linuxTime	linux_time to be converted

Definition at line 76 of file Date.cpp.

References toLinuxTime().

Here is the call graph for this function:



10.17.3.7 void Date::setYear (int *year*) [virtual]

Set year of the Date.

If previous date is leaf month's expiration date and target year what you want is not leaf, the date will be 28

- 2000-02-29 -> setYear(2001) -> 2001-02-28
- 2001-02-28 -> setYear(2000) -> 2000-02-08

Parameters

val	Target year

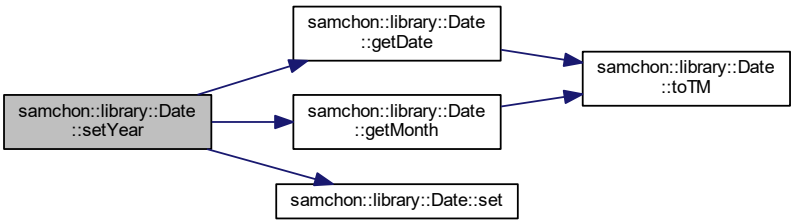
Reimplemented in samchon::library::Datetime.

Definition at line 119 of file Date.cpp.

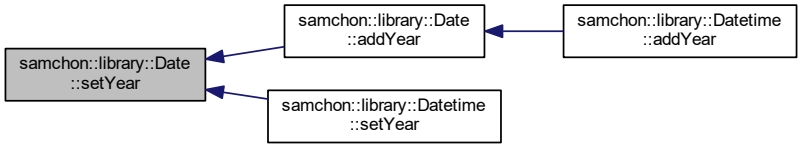
References getDate(), getMonth(), and set().

Referenced by addYear(), and samchon::library::Datetime::setYear().

Here is the call graph for this function:



Here is the caller graph for this function:



10.17.3.8 void Date::setMonth (int *month*) [virtual]

Set month of the Date.

If the expiration date of the month will be shrinked, the date will be changed to the expiration date

- 2000-03-31 -> setMonth(4) -> 2000-04-30
- 2007-08-31 -> setMonth(9) -> 2007-09-30

Parameters

val	Target month (1 - 12)

Exceptions

invalid_argument	month is out of range

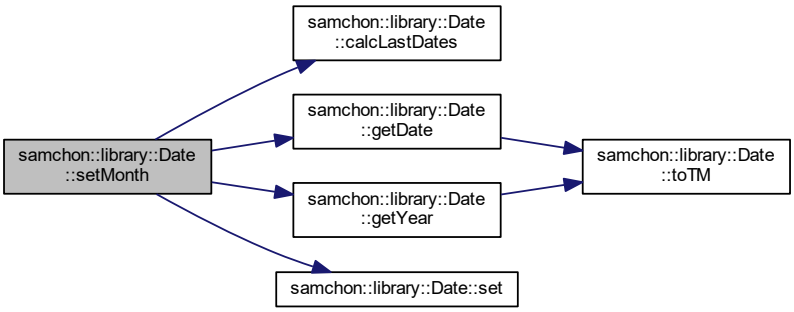
Reimplemented in samchon::library::Datetime.

Definition at line 138 of file Date.cpp.

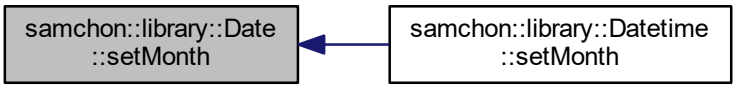
References calcLastDates(), getDate(), getYear(), and set().

Referenced by samchon::library::Datetime::setMonth().

Here is the call graph for this function:



Here is the caller graph for this function:



10.17.3.9 void Date::setDate (int *val*) [virtual]

Set date of the Date

Parameters

val	Target day in month

Exceptions

invalid_argument	When date is over expiration date in month

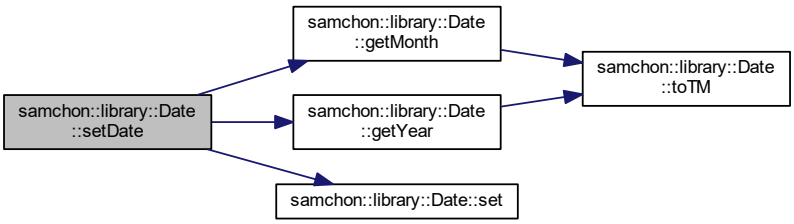
Reimplemented in samchon::library::Datetime.

Definition at line 156 of file Date.cpp.

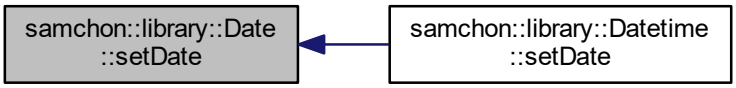
References getMonth(), getYear(), and set().

Referenced by samchon::library::Datetime::setDate().

Here is the call graph for this function:



Here is the caller graph for this function:



10.17.3.10 void Date::addYear (int *val*) [virtual]

Add years to the Date.

Parameters

val	Years to add

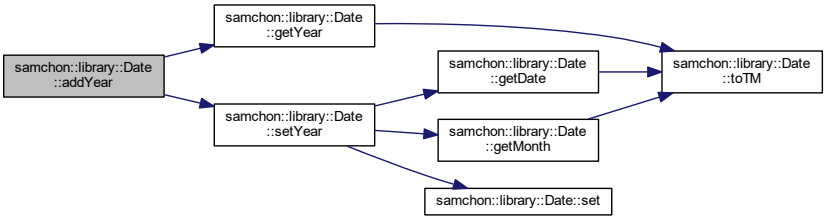
Reimplemented in samchon::library::Datetime.

Definition at line 162 of file Date.cpp.

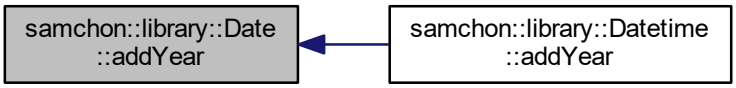
References getYear(), and setYear().

Referenced by samchon::library::Datetime::addYear().

Here is the call graph for this function:



Here is the caller graph for this function:



10.17.3.11 void Date::addMonth (int *val*) [virtual]

Add months to the Date.

Not a matter to val is over 12. If the month is over 12, then years will be added.

Parameters

val	Months to add

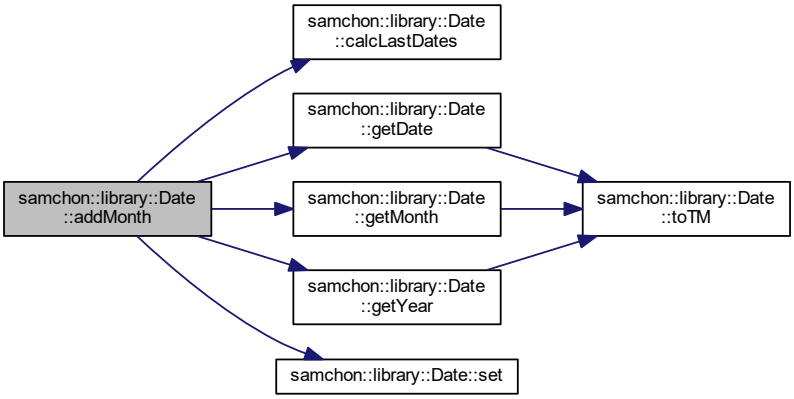
Reimplemented in samchon::library::Datetime.

Definition at line 167 of file Date.cpp.

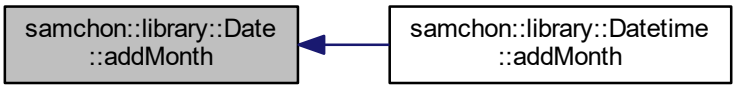
References calcLastDates(), getDate(), getMonth(), getYear(), and set().

Referenced by samchon::library::Datetime::addMonth().

Here is the call graph for this function:



Here is the caller graph for this function:



10.17.3.12 void Date::addWeek (int *val*) [virtual]

Add weeks to the Date.

Not a matter that val is too huge. If the adding weeks derives modification in month or year, then it will be.

Parameters

val	Weeks to add

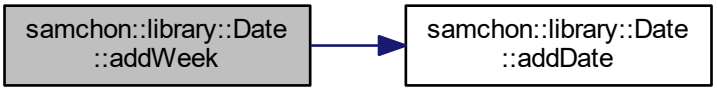
Reimplemented in samchon::library::Datetime.

Definition at line 190 of file Date.cpp.

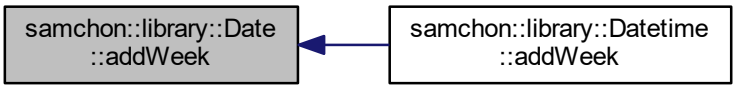
References addDate().

Referenced by samchon::library::Datetime::addWeek().

Here is the call graph for this function:



Here is the caller graph for this function:



10.17.3.13 void Date::addDate (int *val*) [virtual]

Add days to the Date.

Not a matter that val is too huge. If the adding dates derives modification in month or year, then it will be.

Parameters

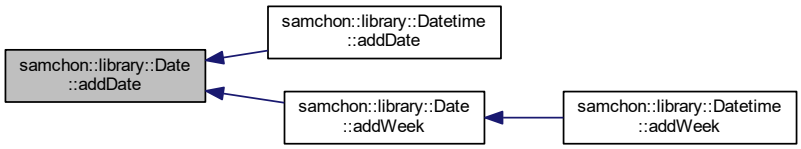
val	Days to add

Reimplemented in samchon::library::Datetime.

Definition at line 194 of file Date.cpp.

Referenced by samchon::library::Datetime::addDate(), and addWeek().

Here is the caller graph for this function:



10.17.3.14 int Date::getYear ( ) const -> int

Get year of the Date.

Returns

Year

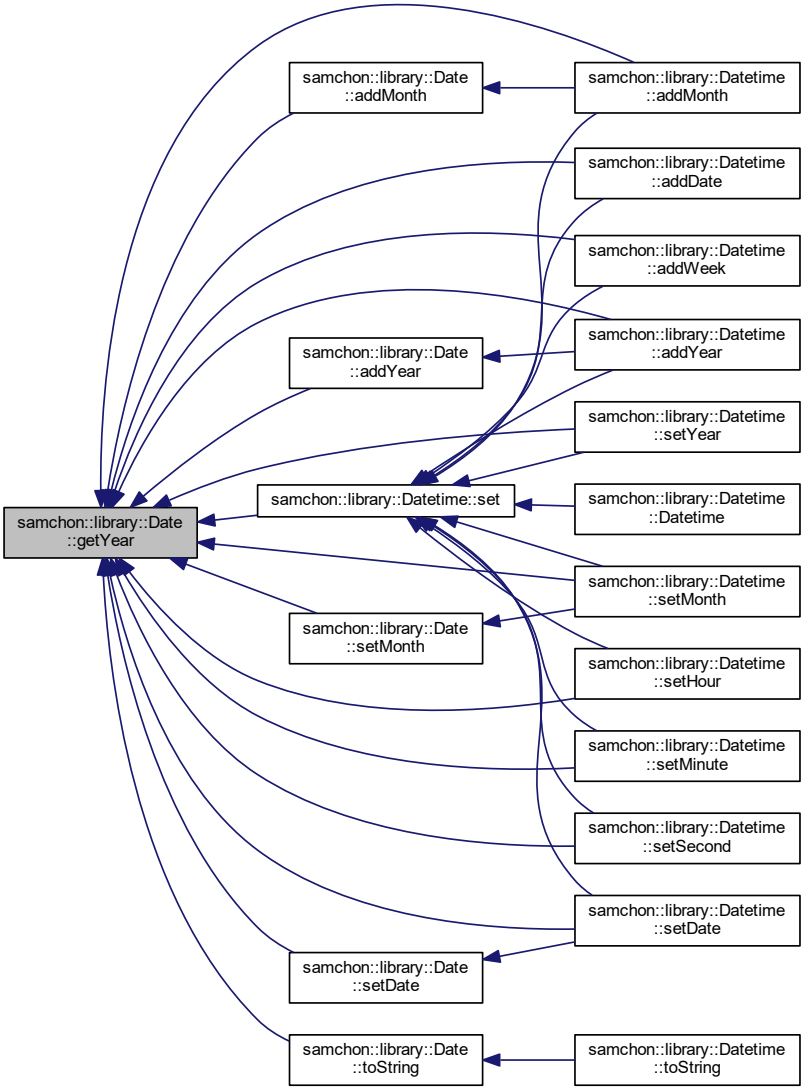
Definition at line 209 of file Date.cpp.

References toTM().

Referenced by samchon::library::Datetime::addDate(), samchon::library::Datetime::addMonth(), addMonth(), samchon::library::Datetime::addWeek(), samchon::library::Datetime::addYear(), addYear(), samchon::library↩ ::Datetime::set(), samchon::library::Datetime::setDate(), setDate(), samchon::library::Datetime::setHour(), samchon::library::Datetime::setMinute(), samchon::library::Datetime::setMonth(), setMonth(), samchon::library::↩ Datetime::setSecond(), samchon::library::Datetime::setYear(), and toString().

Here is the call graph for this function:

Here is the caller graph for this function:



10.17.3.15 int Date::getMonth ( ) const -> int

Get month of the Date.

Returns

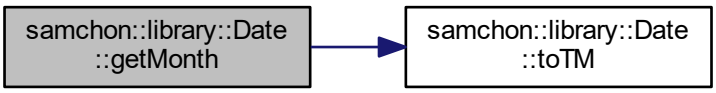
Month as integer (1: January, 12: December)

Definition at line 214 of file Date.cpp.

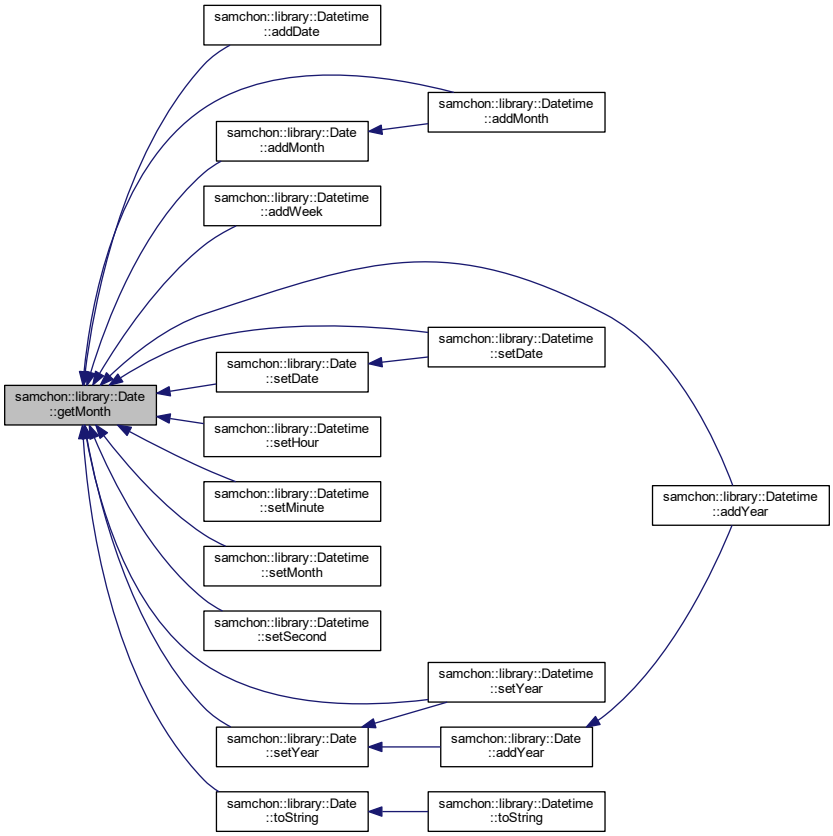
References toTM().

Referenced by samchon::library::Datetime::addDate(), samchon::library::Datetime::addMonth(), addMonth(), samchon::library::Datetime::addWeek(), samchon::library::Datetime::addYear(), samchon::library::Datetime::set↩Date(), setDate(), samchon::library::Datetime::setHour(), samchon::library::Datetime::setMinute(), samchon↩::library::Datetime::setMonth(), samchon::library::Datetime::setSecond(), samchon::library::Datetime::setYear(), setYear(), and toString().

Here is the call graph for this function:



Here is the caller graph for this function:



10.17.3.16 int Date::getDate ( ) const -> int

Get the day in month of the Date.

Returns

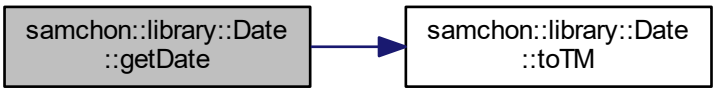
Day in month (1 to 31)

Definition at line 224 of file Date.cpp.

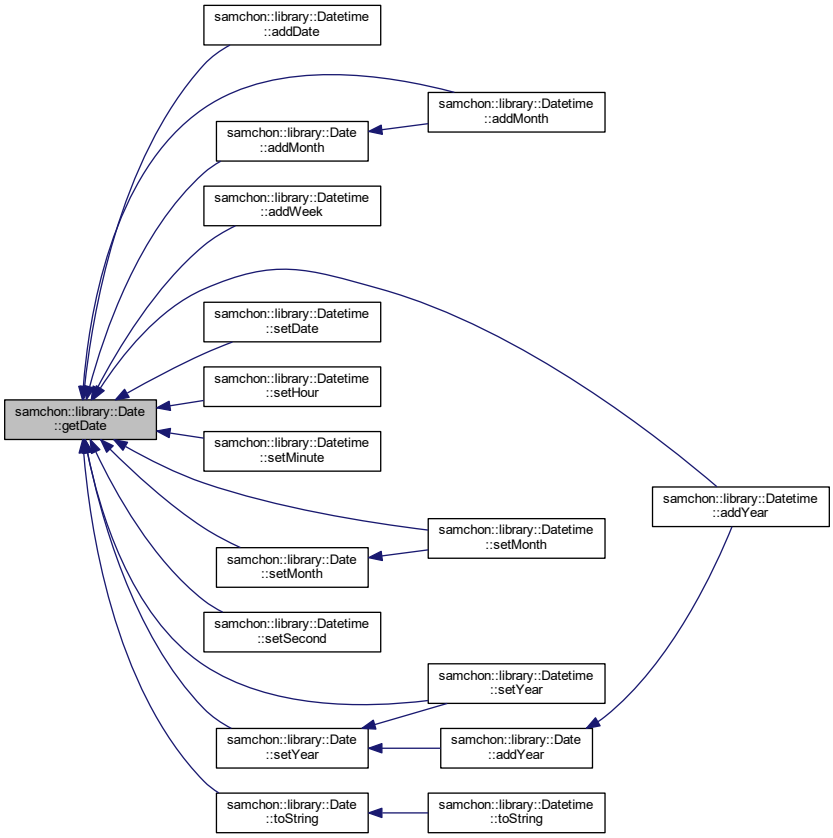
References toTM().

Referenced by samchon::library::Datetime::addDate(), samchon::library::Datetime::addMonth(), addMonth(), samchon::library::Datetime::addWeek(), samchon::library::Datetime::addYear(), samchon::library::Datetime↩ ::setDate(), samchon::library::Datetime::setHour(), samchon::library::Datetime::setMinute(), samchon::library::↩ Datetime::setMonth(), setMonth(), samchon::library::Datetime::setSecond(), samchon::library::Datetime::setYear(), setYear(), and toString().

Here is the call graph for this function:



Here is the caller graph for this function:



10.17.3.17 int Date::getDay ( ) const -> int

Get the day in week of the Date.

Returns

Day in week to integer (0: Sunday, 1: Monday, ..., 6: Saturday)

Definition at line 219 of file Date.cpp.

References toTM().

Here is the call graph for this function:

10.17.3.18 auto Date::toTM ( ) const -> struct ::tm [protected]

Converts the Date to struct tm.

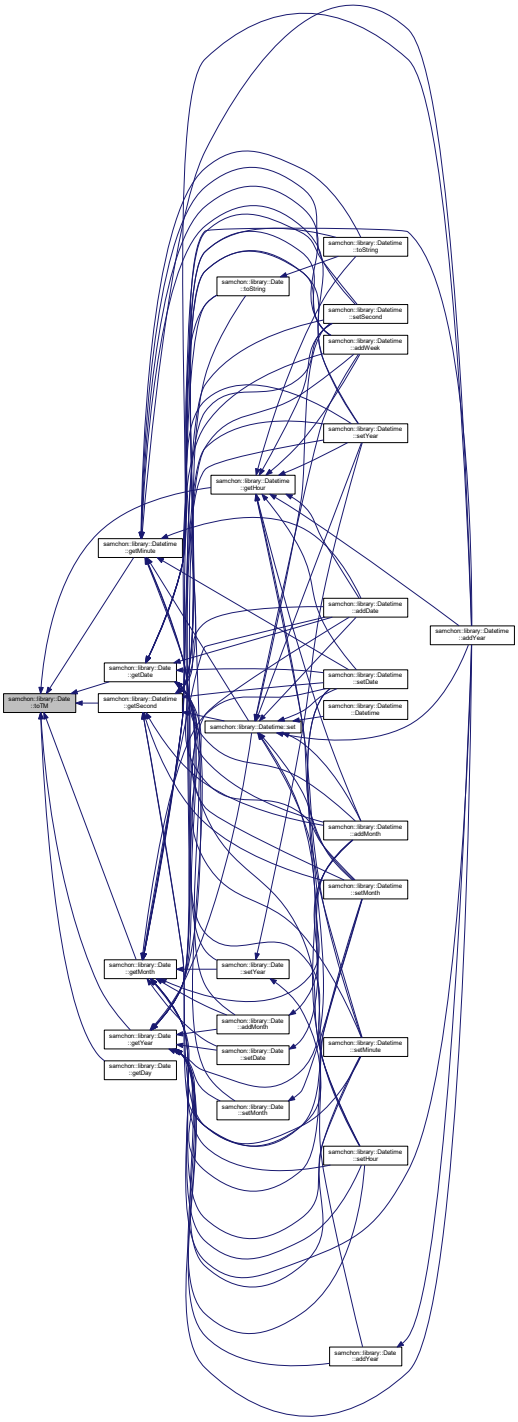
Returns

struct tm representing the Date

Definition at line 230 of file Date.cpp.

Referenced by getDate(), getDay(), samchon::library::Datetime::getHour(), samchon::library::Datetime::getMinute(), getMonth(), samchon::library::Datetime::getSecond(), and getYear().

Here is the caller graph for this function:



10.17.3.19 auto Date::toLinuxTime ( ) const -> long long

Converts the Date to linux_time.

Returns

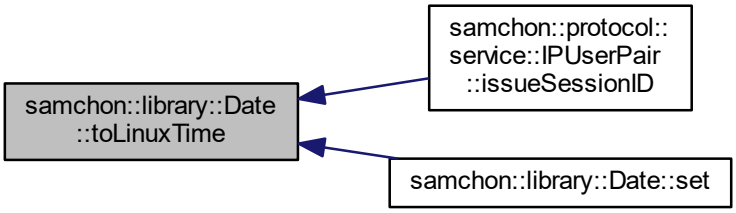
linux_time of the Date

Definition at line 202 of file Date.cpp.

References TP_1970.

Referenced by samchon::protocol::service::IPUserPair::issueSessionID(), and set().

Here is the caller graph for this function:



10.17.3.20 auto Date::toString ( ) const -> std::string [virtual]

Converts the Date to std::string.

Returns

std::string expressing the Date

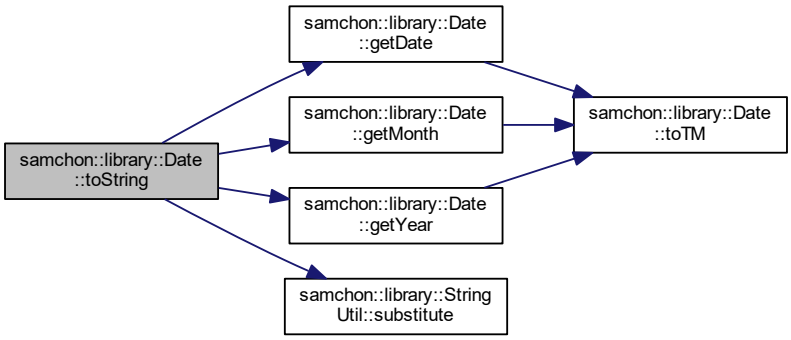
Reimplemented in samchon::library::Datetime.

Definition at line 238 of file Date.cpp.

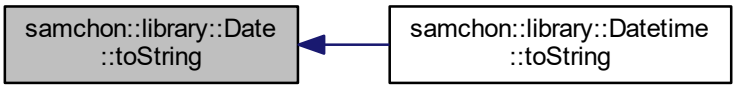
References getDate(), getMonth(), getYear(), and samchon::library::StringUtil::substitute().

Referenced by samchon::library::Datetime::toString().

Here is the call graph for this function:



Here is the caller graph for this function:



10.17.4 Member Data Documentation

10.17.4.1 chrono::system_clock::time_point Date::TP_1970 = chrono::system_clock::from_time_t(0) [static], [protected]

time_point for 1970-01-01 09:00:00

It's the zero(0) in linux_time

Definition at line 48 of file Date.hpp.

Referenced by toLinuxTime().

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

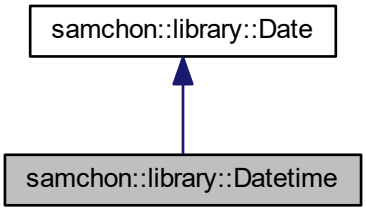
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/Date.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/Date.cpp

10.18 samchon::library::Datetime Class Reference

Date and time.

```
#include <Datetime.hpp>
```

Collaboration diagram for samchon::library::Datetime:



Public Member Functions

- Datetime ()
- Datetime (const std::chrono::system_clock::time_point &)

Copy Constructor.

- Datetime (std::chrono::system_clock::time_point &&)
- Datetime (int year, int month, int date, int hour=0, int min=0, int sec=0)

  Construct from datetime elements.
- Datetime (const WeakString &)

  Construct from WeakString.
- Datetime (long long linuxTime)

  Construct from linux_time.
- virtual void set (const WeakString &) override

  Set by weak_string.
- void set (int year, int month, int date, int hours=0, int minutes=0, int seconds=0)
- void set (long long linuxTime)
- virtual void setYear (int)

  Set year of the Date.
- virtual void setMonth (int)

  Set month of the Date.
- virtual void setDate (int)
- void setHour (int)

  Set hour.
- void setMinute (int)

  Set minute.
- void setSecond (int)

  Set hour.
- virtual void addYear (int) override

  Add years to the Date.
- virtual void addMonth (int) override

  Add months to the Date.
- virtual void addWeek (int) override

  Add weeks to the Date.
- virtual void addDate (int) override

  Add days to the Date.
- void addHour (int)

  Add hours.
- void addMinute (int)

  Add minutes.
- void addSecond (int)

  Add seconds.
- auto getHour () const -> int

  Get hour.
- auto getMinute () const -> int

  Get minute.
- auto getSecond () const -> int

  Get second.
- virtual auto toString () const -> std::string override

  Converts the Datetime to std::string.

Additional Inherited Members

10.18.1 Detailed Description

Date and time.

Datetime is a Date having hours, minutes and seconds addictionally.

[Inherited]

Date class is a chrono::time_point<std::chrono::system_clock>.

Date represents the date with year, month and day in month. The basic date (local time) is from your operating system.

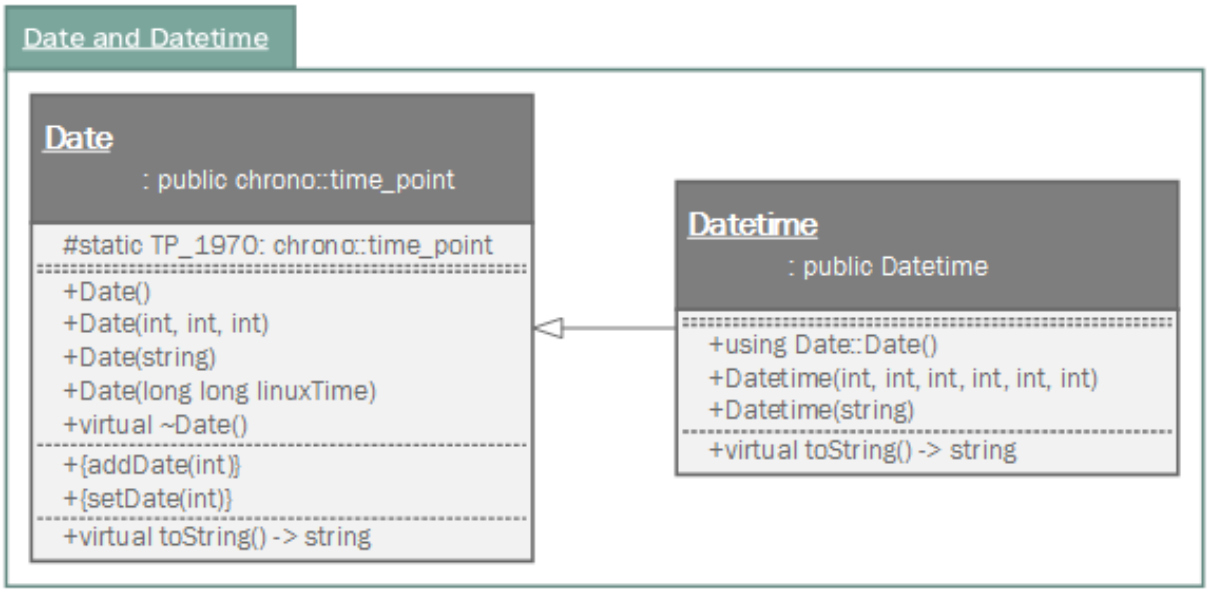
[Inherited]

A time_point object expresses a point in time relative to a clock's epoch.

Internally, the object stores an object of a duration type, and uses the Clock type as a reference for its epoch.

Referenced comments of std::chrono::time_point

- http://www.cplusplus.com/reference/chrono/time_point/



See also

samchon::library

Author

Jeongho Nam

Definition at line 19 of file Datetime.hpp.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 Datetime::Datetime ( )

Default Constructor

The datetime will be now (Now from operating system)

Definition at line 15 of file Datetime.cpp.

10.18.2.2 samchon::library::Datetime::Datetime (const std::chrono::system_clock::time_point &)

Copy Constructor.

Copy the date from another Date

Parameters

date	Target to be copied

10.18.2.3 samchon::library::Datetime::Datetime (std::chrono::system_clock::time_point &&)

Move Constructor

Gets the date from other and truncate the other

Parameters

date	Target to be moved

10.18.2.4 Datetime::Datetime (int *year*, int *month*, int *date*, int *hour* = 0, int *min* = 0, int *sec* = 0)

Construct from datetime elements.

Parameters

year	ex) 2015
month	January to December. 1 $\sim$ 12
date	Day in month. 1 $\sim$ 31
hour	Hours (0-23)
min	Minutes (0-59)
sec	Seconds (0-59)

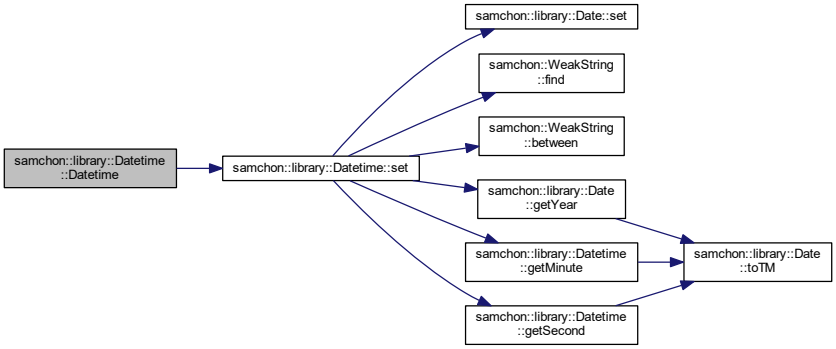
Exceptions

invalid_argument	parameter is out of its own range.

Definition at line 27 of file Datetime.cpp.

References set().

Here is the call graph for this function:



10.18.2.5 Datetime::Datetime (const WeakString & *wStr*)

Construct from WeakString.

Parameters

wstr	A weak_string expressing the date. (1991-01-01 09:27:03)

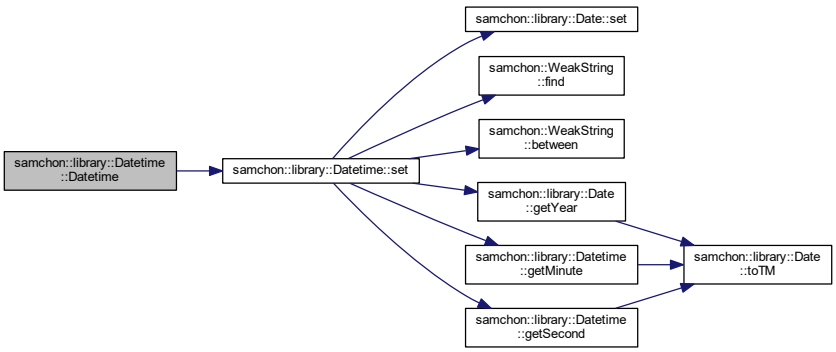
Exceptions

invalid_argument	parameter is out of its own range.

Definition at line 32 of file Datetime.cpp.

References set().

Here is the call graph for this function:



10.18.2.6 Datetime::Datetime (long long *linuxTime*)

Construct from linux_time.

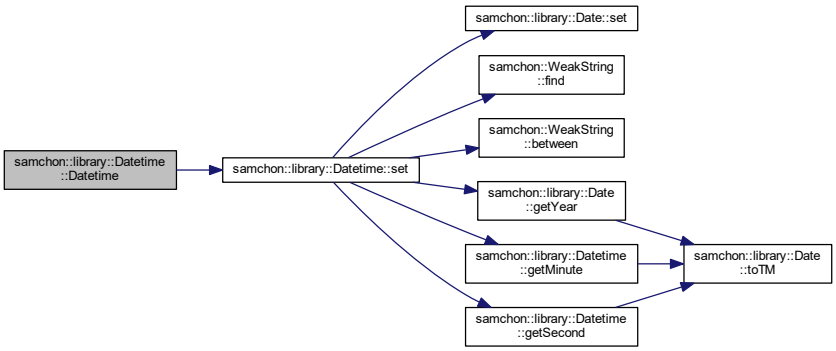
Parameters

linuxTime	linux_time to be converted

Definition at line 37 of file Datetime.cpp.

References set().

Here is the call graph for this function:



10.18.3 Member Function Documentation

10.18.3.1 void Datetime::set (const WeakString & *wStr*) [override], [virtual]

Set by weak_string.

Parameters

wstr	A weak_string expressing the date. (1991-01-01 09:27:03)

Exceptions

invalid_argument	parameter is out of its own range.

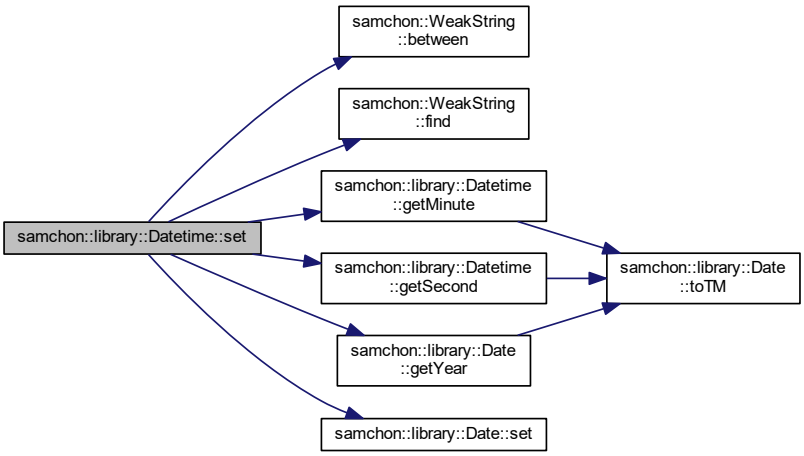
Reimplemented from samchon::library::Date.

Definition at line 44 of file Datetime.cpp.

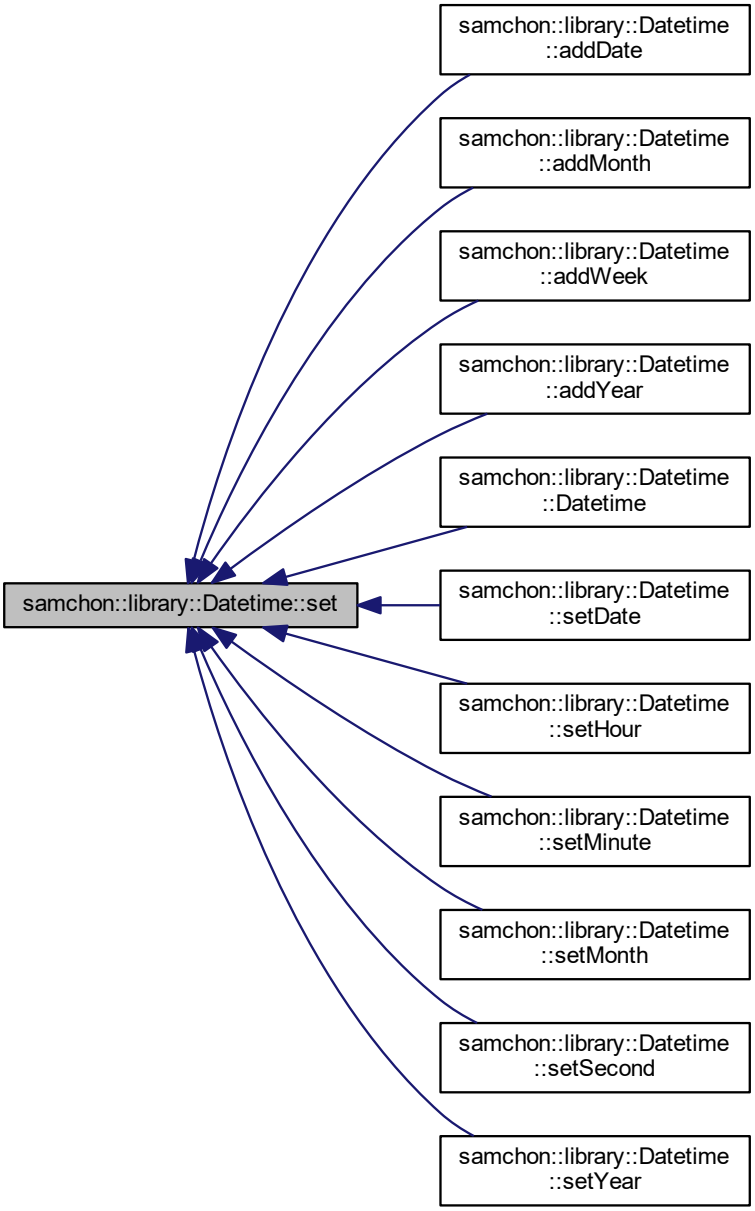
References samchon::WeakString::between(), samchon::WeakString::find(), getMinute(), getSecond(), samchon↩ ::library::Date::getYear(), and samchon::library::Date::set().

Referenced by addDate(), addMonth(), addWeek(), addYear(), Datetime(), setDate(), setHour(), setMinute(), set↩ Month(), setSecond(), and setYear().

Here is the call graph for this function:



Here is the caller graph for this function:



10.18.3.2 void Datetime::set (int *year*, int *month*, int *date*, int *hours* = 0, int *minutes* = 0, int *seconds* = 0)

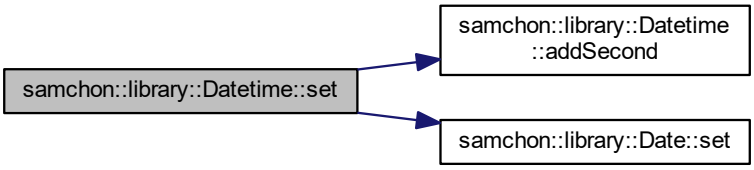
Exceptions

invalid_argument	parameter is out of range (ex: month -> 13)

Definition at line 57 of file Datetime.cpp.

References addSecond(), and samchon::library::Date::set().

Here is the call graph for this function:

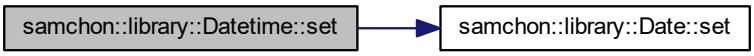


10.18.3.3 void Datetime::set (long long *linuxTime*)

Definition at line 62 of file Datetime.cpp.

References samchon::library::Date::set().

Here is the call graph for this function:



10.18.3.4 void Datetime::setYear (int *val*) [virtual]

Set year of the Date.

If previous date is leaf month's expiration date and target year what you want is not leaf, the date will be 28

- 2000-02-29 -> setYear(2001) -> 2001-02-28
- 2001-02-28 -> setYear(2000) -> 2000-02-08

Parameters

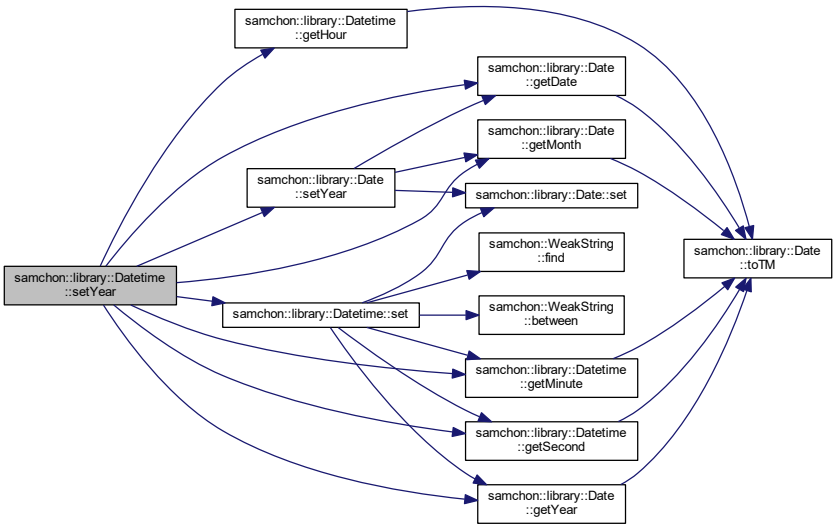
val	Target year

Reimplemented from samchon::library::Date.

Definition at line 71 of file Datetime.cpp.

References samchon::library::Date::getDate(), getHour(), getMinute(), samchon::library::Date::getMonth(), get↩Second(), samchon::library::Date::getYear(), set(), and samchon::library::Date::setYear().

Here is the call graph for this function:



10.18.3.5 void Datetime::setMonth (int *val*) [virtual]

Set month of the Date.

If the expiration date of the month will be shrinked, the date will be changed to the expiration date

- 2000-03-31 -> setMonth(4) -> 2000-04-30

- 2007-08-31 -> setMonth(9) -> 2007-09-30

Parameters

val	Target month (1 - 12)

Exceptions

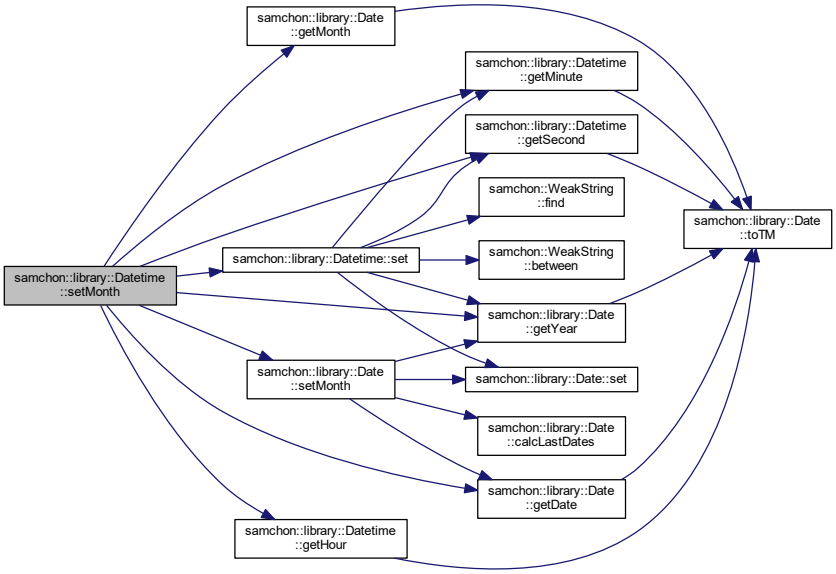
invalid_argument	month is out of range

Reimplemented from samchon::library::Date.

Definition at line 80 of file Datetime.cpp.

References samchon::library::Date::getDate(), getHour(), getMinute(), samchon::library::Date::getMonth(), get↩Second(), samchon::library::Date::getYear(), set(), and samchon::library::Date::setMonth().

Here is the call graph for this function:



10.18.3.6 void Datetime::setDate (int *val*) [virtual]

Set date of the Date

Parameters

val	Target day in month

Exceptions

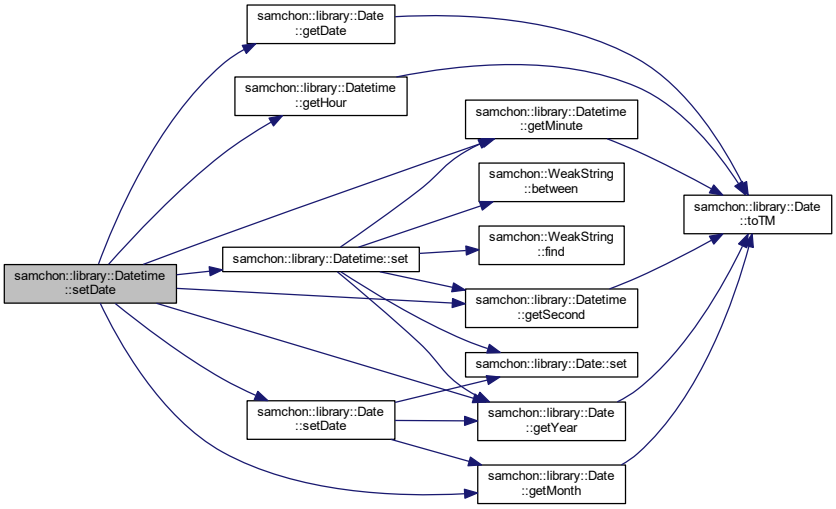
invalid_argument	When date is over expiration date in month

Reimplemented from samchon::library::Date.

Definition at line 89 of file Datetime.cpp.

References samchon::library::Date::getDate(), getHour(), getMinute(), samchon::library::Date::getMonth(), get↩Second(), samchon::library::Date::getYear(), set(), and samchon::library::Date::setDate().

Here is the call graph for this function:



10.18.3.7 void Datetime::setHour (int *val*)

Set hour.

Parameters

val	Target hour (0 - 23)

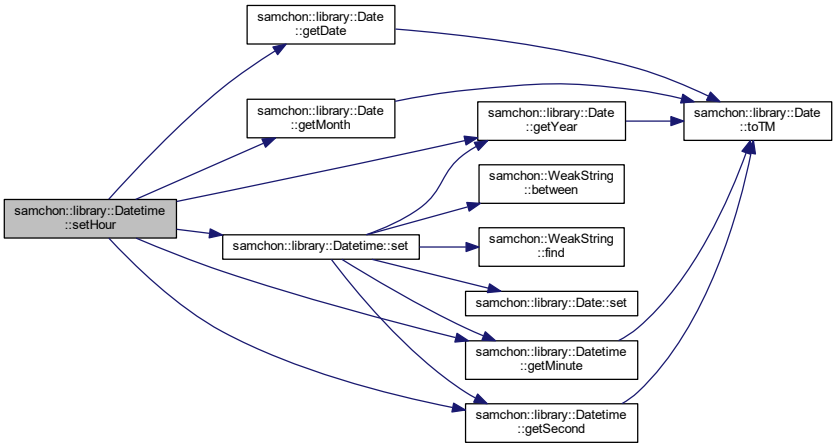
Exceptions

invalid_argument	When hour is out of range

Definition at line 98 of file Datetime.cpp.

References samchon::library::Date::getDate(), getMinute(), samchon::library::Date::getMonth(), getSecond(), samchon::library::Date::getYear(), and set().

Here is the call graph for this function:



10.18.3.8 void Datetime::setMinute (int *val*)

Set minute.

Parameters

val	Target minute (0 - 59)

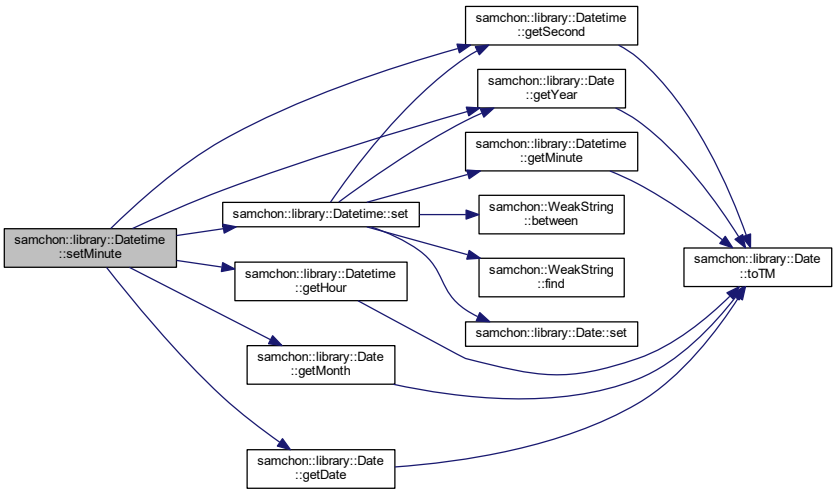
Exceptions

invalid_argument	When minute is out of range

Definition at line 102 of file Datetime.cpp.

References samchon::library::Date::getDate(), getHour(), samchon::library::Date::getMonth(), getSecond(), samchon::library::Date::getYear(), and set().

Here is the call graph for this function:



10.18.3.9 void Datetime::setSecond (int *val*)

Set hour.

Parameters

val	Target second (0 - 59)

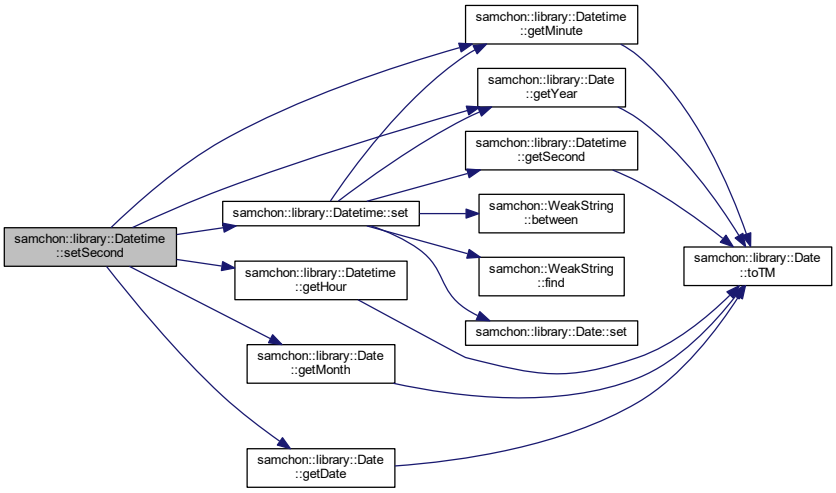
Exceptions

invalid_argument	When second is out of range

Definition at line 106 of file Datetime.cpp.

References samchon::library::Date::getDate(), getHour(), getMinute(), samchon::library::Date::getMonth(), samchon::library::Date::getYear(), and set().

Here is the call graph for this function:



10.18.3.10 void Datetime::addYear (int *val*) [override], [virtual]

Add years to the Date.

Parameters

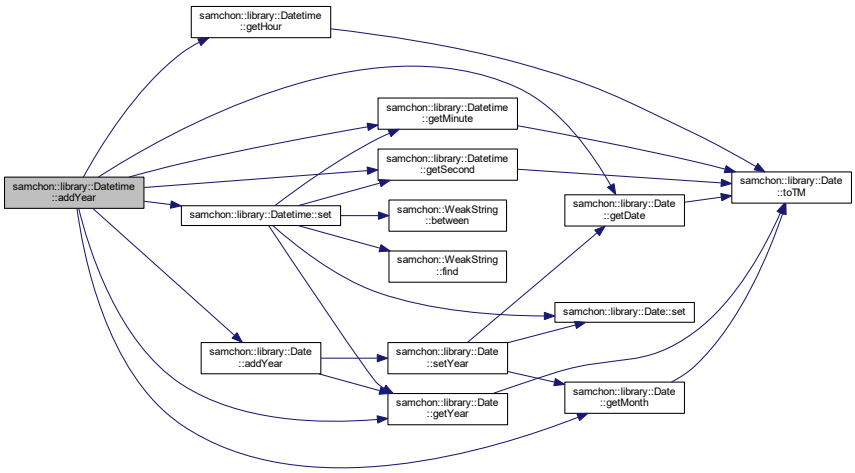
val	Years to add

Reimplemented from samchon::library::Date.

Definition at line 112 of file Datetime.cpp.

References samchon::library::Date::addYear(), samchon::library::Date::getDate(), getHour(), getMinute(), samchon::library::Date::getMonth(), getSecond(), samchon::library::Date::getYear(), and set().

Here is the call graph for this function:



10.18.3.11 void Datetime::addMonth (int *val*) [override], [virtual]

Add months to the Date.

Not a matter to val is over 12. If the month is over 12, then years will be added.

Parameters

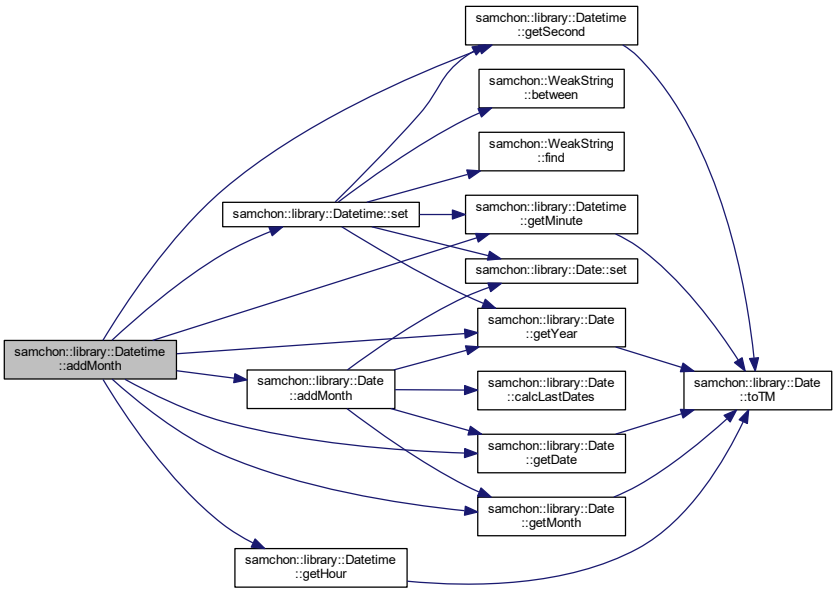
val	Months to add

Reimplemented from samchon::library::Date.

Definition at line 121 of file Datetime.cpp.

References samchon::library::Date::addMonth(), samchon::library::Date::getDate(), getHour(), getMinute(), samchon::library::Date::getMonth(), getSecond(), samchon::library::Date::getYear(), and set().

Here is the call graph for this function:



10.18.3.12 void Datetime::addWeek (int *val*) [override], [virtual]

Add weeks to the Date.

Not a matter that val is too huge. If the adding weeks derives modification in month or year, then it will be.

Parameters

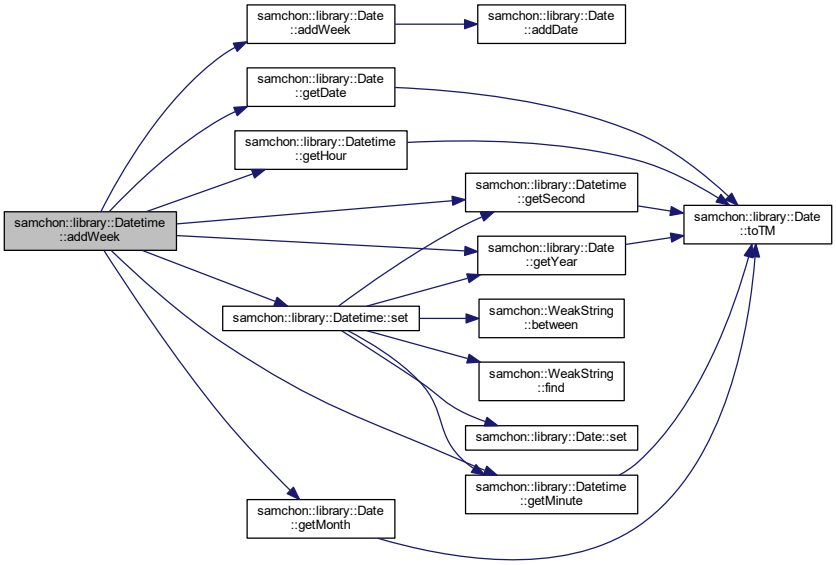
val	Weeks to add

Reimplemented from samchon::library::Date.

Definition at line 130 of file Datetime.cpp.

References samchon::library::Date::addWeek(), samchon::library::Date::getDate(), getHour(), getMinute(), samchon::library::Date::getMonth(), getSecond(), samchon::library::Date::getYear(), and set().

Here is the call graph for this function:



10.18.3.13 void Datetime::addDate (int *val*) [override], [virtual]

Add days to the Date.

Not a matter that val is too huge. If the adding dates derives modification in month or year, then it will be.

Parameters

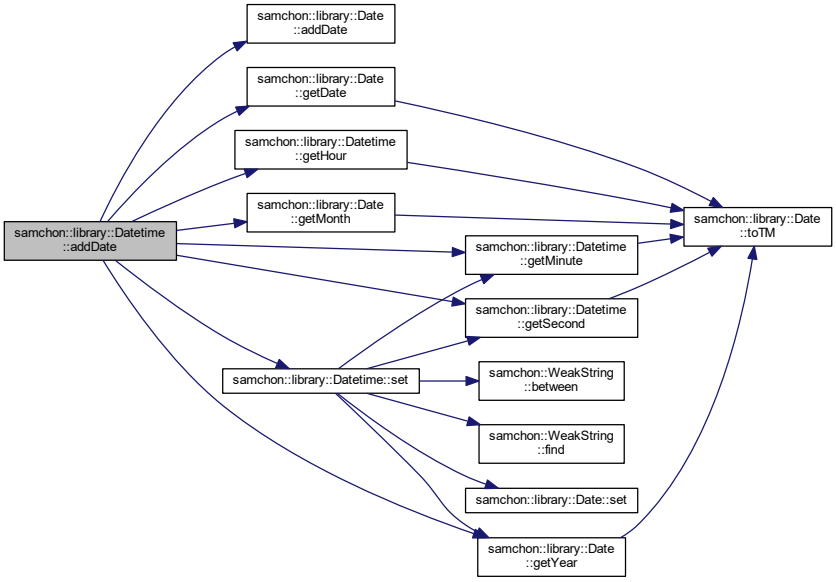
val	Days to add

Reimplemented from samchon::library::Date.

Definition at line 139 of file Datetime.cpp.

References samchon::library::Date::addDate(), samchon::library::Date::getDate(), getHour(), getMinute(), samchon::library::Date::getMonth(), getSecond(), samchon::library::Date::getYear(), and set().

Here is the call graph for this function:



10.18.3.14 void Datetime::addHour (int *val*)

Add hours.

It's same with operator+=(chrono::hours(val))

Do not worry about the out of range of the hour(val) Parent items like date, month and years will be adjusted automatically.

Parameters

val	hours to add

Definition at line 148 of file Datetime.cpp.

10.18.3.15 void Datetime::addMinute (int *val*)

Add minutes.

It's same with operator+=(chrono::minutes(val))

Do not worry about the out of range of the minutes(val) Parent items like date, month and years will be adjusted automatically.

Parameters

val	minutes to add

Definition at line 152 of file Datetime.cpp.

10.18.3.16 void Datetime::addSecond (int *val*)

Add seconds.

It's same with operator+=(chrono::seconds(val))

Do not worry about the out of range of the seconds(val) Parent items like date, month and years will be adjusted automatically.

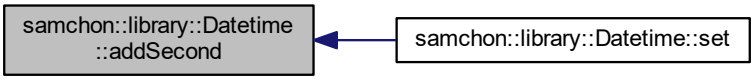
Parameters

val	seconds to add

Definition at line 156 of file Datetime.cpp.

Referenced by set().

Here is the caller graph for this function:



10.18.3.17 int Datetime::getHour ( ) const -> int

Get hour.

Returns

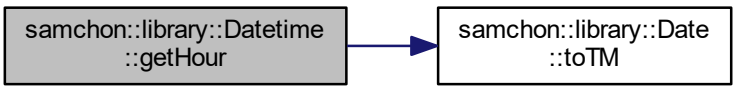
hour of the Datetime to integer

Definition at line 162 of file Datetime.cpp.

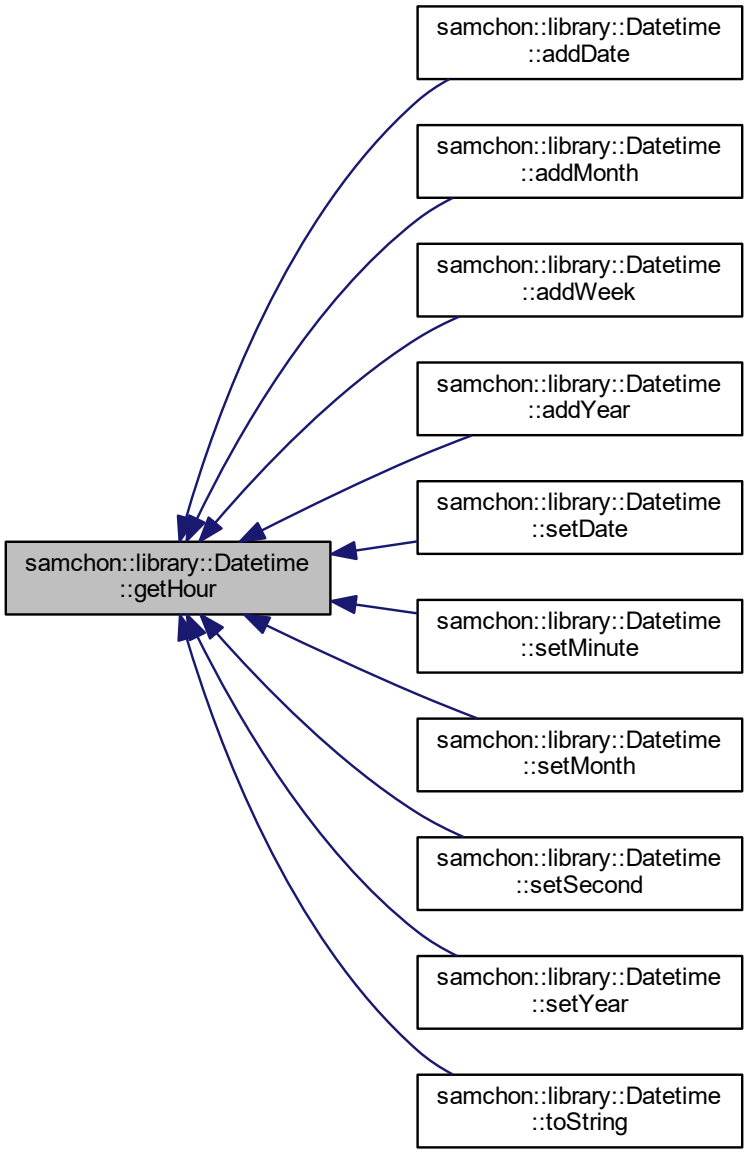
References samchon::library::Date::toTM().

Referenced by addDate(), addMonth(), addWeek(), addYear(), setDate(), setMinute(), setMonth(), setSecond(), setYear(), and toString().

Here is the call graph for this function:



Here is the caller graph for this function:



10.18.3.18 int Datetime::getMinute ( ) const -> int

Get minute.

Returns

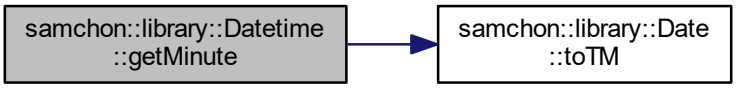
minute of the Datetime to integer

Definition at line 167 of file Datetime.cpp.

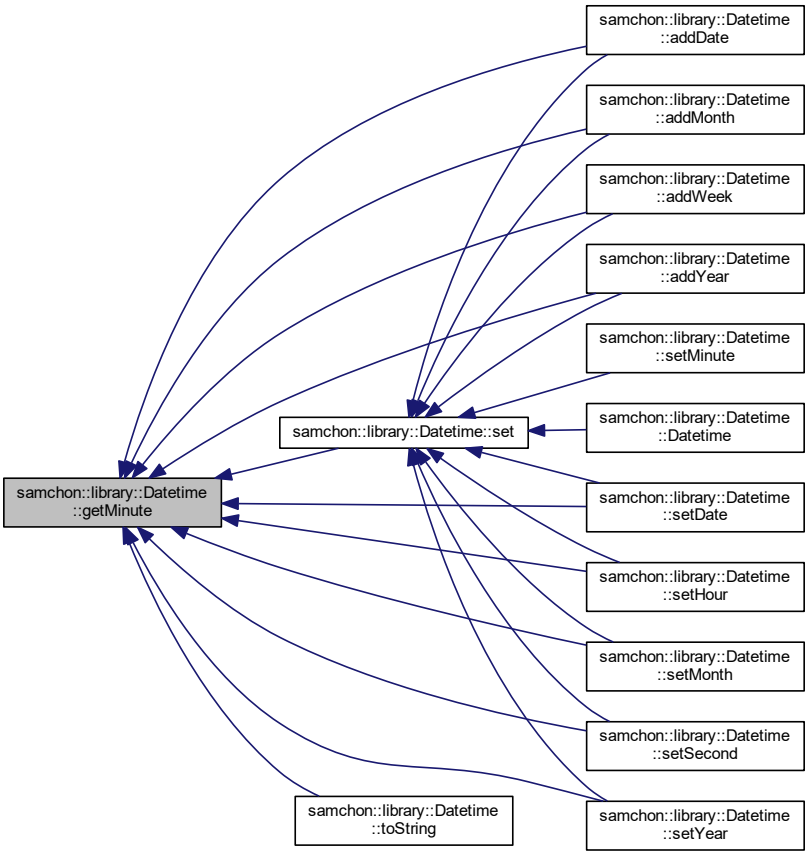
References samchon::library::Date::toTM().

Referenced by addDate(), addMonth(), addWeek(), addYear(), set(), setDate(), setHour(), setMonth(), setSecond(), setYear(), and toString().

Here is the call graph for this function:



Here is the caller graph for this function:



10.18.3.19 int Datetime::getSecond ( ) const -> int

Get second.

Returns

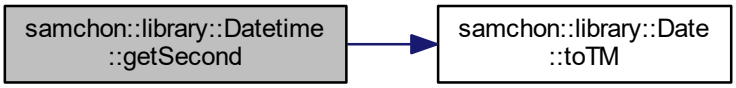
second of the Datetime to integer

Definition at line 172 of file Datetime.cpp.

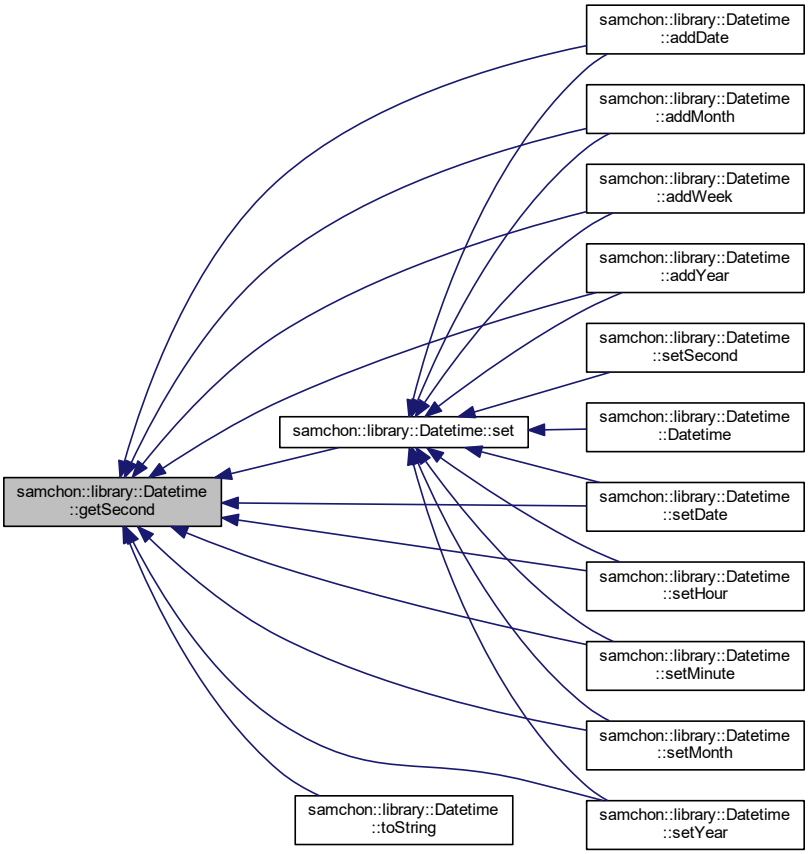
References samchon::library::Date::toTM().

Referenced by addDate(), addMonth(), addWeek(), addYear(), set(), setDate(), setHour(), setMinute(), setMonth(), setYear(), and toString().

Here is the call graph for this function:



Here is the caller graph for this function:



10.18.3.20 auto Datetime::toString ( ) const -> std::string [override], [virtual]

Converts the Datetime to std::string.

Returns

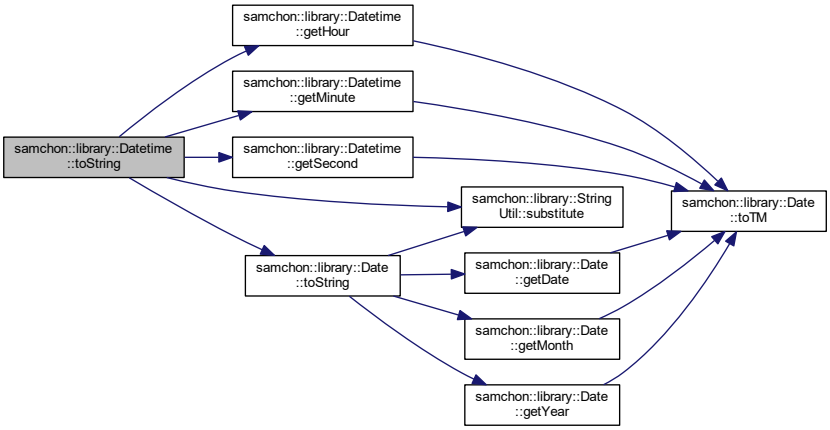
std::string expressing the Datetime

Reimplemented from samchon::library::Date.

Definition at line 179 of file Datetime.cpp.

References getHour(), getMinute(), getSecond(), samchon::library::StringUtil::substitute(), and samchon::library::↩ Date::toString().

Here is the call graph for this function:



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

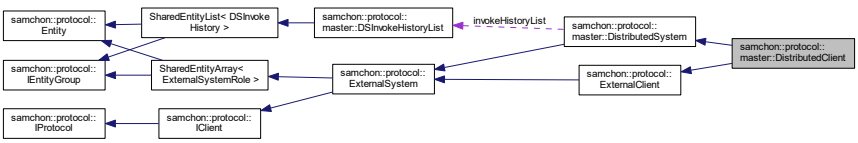
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/Datetime.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/Datetime.cpp

10.19 samchon::protocol::master::DistributedClient Class Reference

A network driver for a distributed client.

`#include <DistributedClient.hpp>`

Collaboration diagram for samchon::protocol::master::DistributedClient:



Public Member Functions

- DistributedClient ()

  Default Constructor.

Additional Inherited Members

10.19.1 Detailed Description

A network driver for a distributed client.

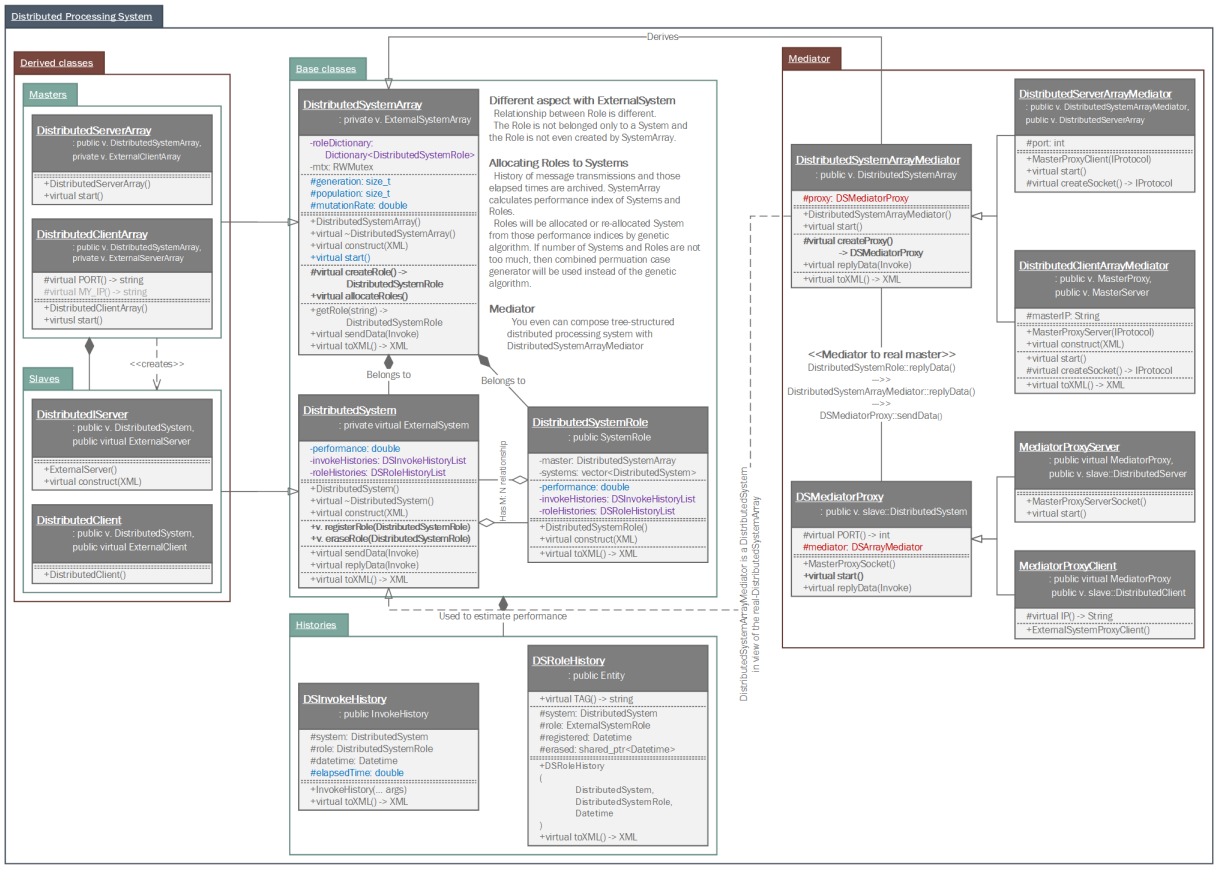
DistributedClient is an DistributedSystem specialized in client driver.

[Inherited]

DistributedSystem class is an ExternalSystem having role as DistributedSystemRole objects with performance index and history log allocation of DistributedSystemRole objects and have requsted Invoke message to related external system.

The greatest difference between DistributedSystem and ExternalSystem is, unlike the ExternalSystem class which having is ExternalSystemRole objects exclusively, DistributedSystem does not monopoly a DistributedSystemRole. The DistributedSystemRole can allocatedto multiple DistributedSystem.

DistributedSystem and DistributedSystemRole classes have performance index and history log archiving what they've requested to the related distributed system. Allocating DistributedSystemRole objects to DistributedSyste will be determined those performance index and history log for optimize distribution of elasped time about each Invoke message processing.

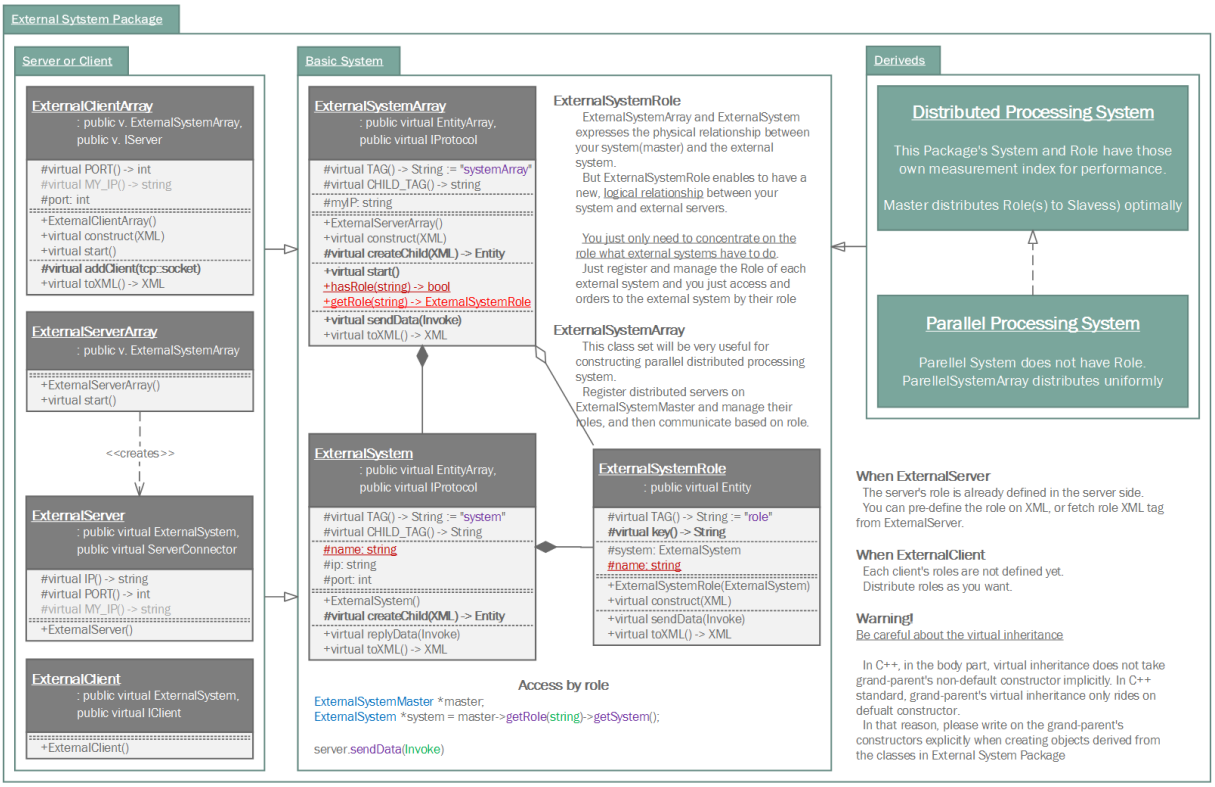


Todo

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, ExternalSystem is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystemRole.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an ExternalSystem. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 22 of file DistributedClient.hpp.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 samchon::protocol::master::DistributedClient::DistributedClient ( )

Default Constructor.

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

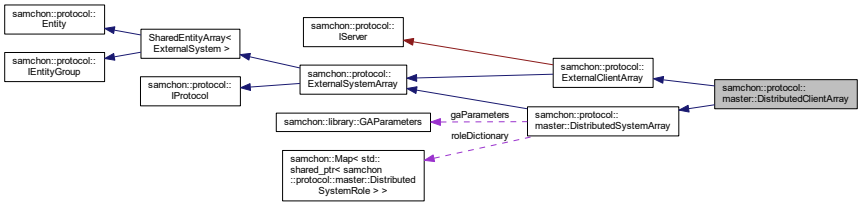
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DistributedClient.hpp

10.20 samchon::protocol::master::DistributedClientArray Class Reference

An array of DistributedClient(s).

`#include <DistributedClientArray.hpp>`

Collaboration diagram for samchon::protocol::master::DistributedClientArray:



Public Member Functions

- DistributedClientArray ()

  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override

  Construct data of the Entity from an XML object.
- virtual void start () override

  Start interaction.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the EntityGroup.

Additional Inherited Members

10.20.1 Detailed Description

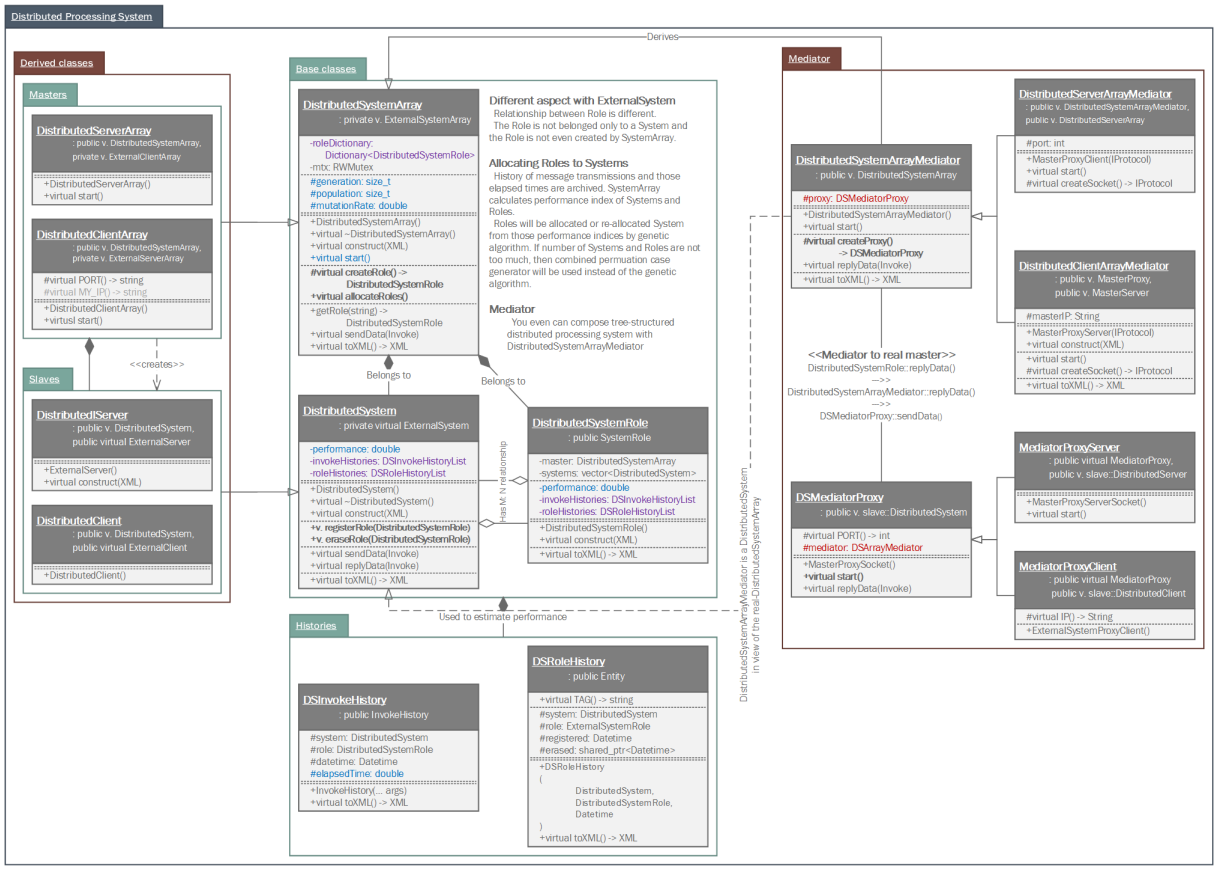
An array of DistributedClient(s).

DistributedClientArray is an DistributedSystemArray specialized in clients' driver.

[Inherited]

DistributedSystemArray class is an ExternalSystemArray and an abstract class containing and managing distributed system drivers within framework of master. The DistributedSystemArray class allocates DistributedSystemRole objects to DistributedSystem objects for optimization.

Unlike the ExternalSystemRole class which is dependent on an ExternalSystem, DistributedSystemRole is not dependent on a DistributedSystem and even can be allocated to multiple DistributedSystem objects. Allocation of DistributedSystemRole(s) to DistributedSystem(s) is determined by estimation of performance index pursuing best distribution. If number of DistributedSystem and DistributedSystemRole objects is not a lot, optimize by combined permuation algorithm. Else the objects are a lot, the allocation will be optimized by genetic algorithm.



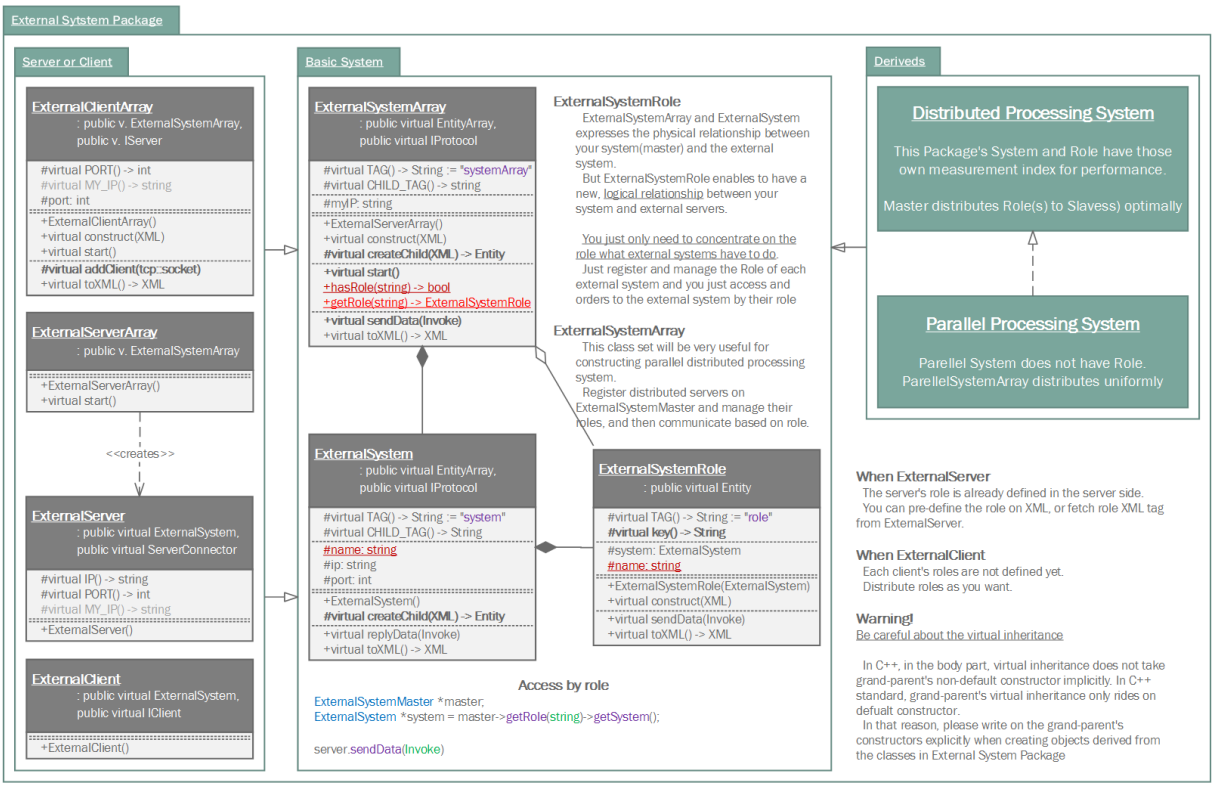
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

- ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 22 of file DistributedClientArray.hpp.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 samchon::protocol::master::DistributedClientArray::DistributedClientArray ( )

Default Constructor.

10.20.3 Member Function Documentation

10.20.3.1 virtual void samchon::protocol::master::DistributedClientArray::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Reimplemented from samchon::protocol::ExternalClientArray.

Reimplemented in samchon::protocol::master::DistributedClientArrayMediator.

10.20.3.2 virtual void samchon::protocol::master::DistributedClientArray::start ( ) [override],[virtual]

Start interaction.

An abstract method starting interaction with external systems.

If external systems are servers, starts connection to them, else clients, opens a server and accepts the external systems. You can addict your own procudures of starting drivers, but if you directly override method of abstract ExternalSystemArray, be careful about virtual inheritance.

- DistributedClientArray::start() calls DistributedSystemArray::start() and ExternalClientArray:start()

Reimplemented from samchon::protocol::ExternalClientArray.

Reimplemented in samchon::protocol::master::DistributedClientArrayMediator.

10.20.3.3 virtual auto samchon::protocol::master::DistributedClientArray::toXML ( ) const -> std::shared_ptr<library::XML> [override],[virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::ExternalClientArray.

Reimplemented in samchon::protocol::master::DistributedClientArrayMediator.

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

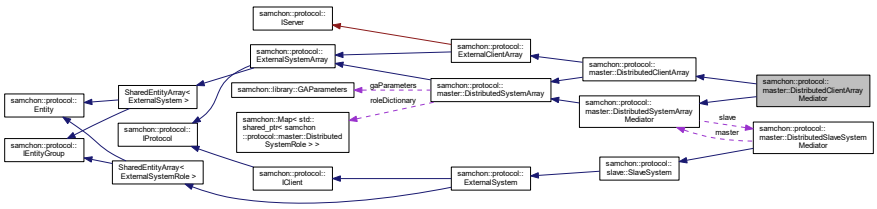
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DistributedClientArray.hpp

10.21 samchon::protocol::master::DistributedClientArrayMediator Class Reference

A mediator of distributed system between master and slaves.

`#include <DistributedClientArrayMediator.hpp>`

Collaboration diagram for samchon::protocol::master::DistributedClientArrayMediator:



Public Member Functions

- DistributedClientArrayMediator ()

  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override

  Construct data of the Entity from an XML object.
- virtual void start () override

  Start interaction.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the EntityGroup.

Additional Inherited Members

10.21.1 Detailed Description

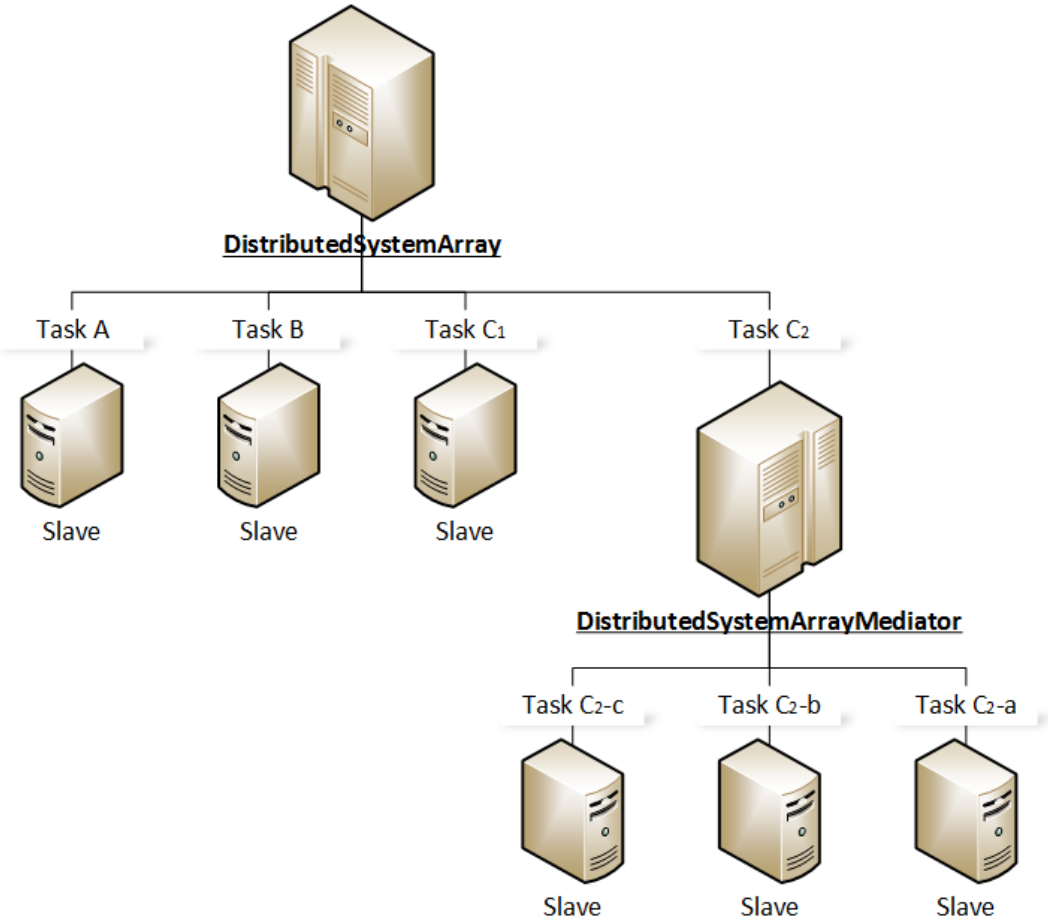
A mediator of distributed system between master and slaves.

DistributedClientArrayMediator is a DistributedSystemArrayMediator specialized in clients' driver. Distributed↩ ClientArrayMediator is a client for its master and also manages children clients' slaves.

[Inherited]

DistributedSystemArrayMediator class is used to realizing a mediator system of tree-structured distributed processing system. DistributedSystemArrayMediator intermediates between origin-master(DistributedSystemArray) and its slaves(slave::DistributedSystem).

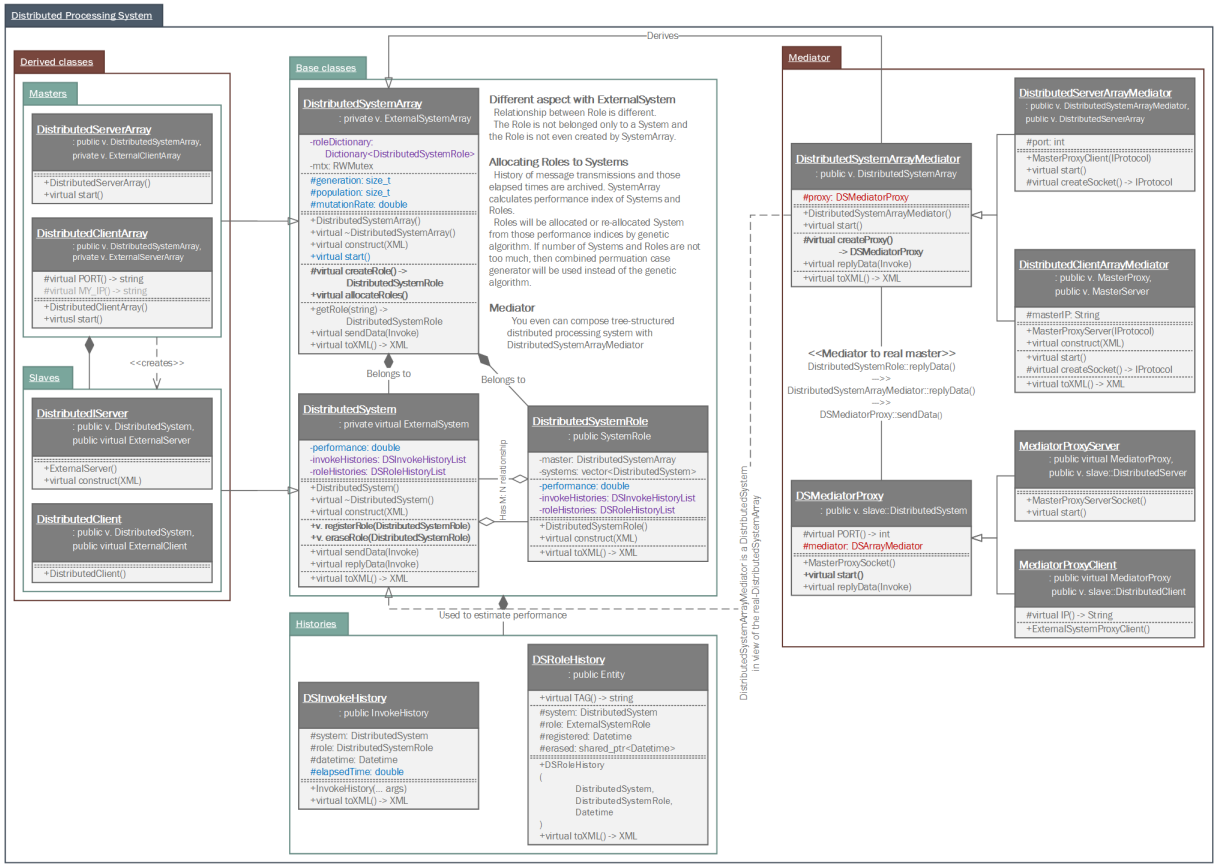
DistributedSystemArrayMediator is a master(DistributedSystemArray) within the framework of slaves systems (slave::DistributedSystem objects) and also can be a slave system (slave::DistributedSystem) within framework of its master(DistributedSystemArray) by having DistributedSlaveSystemMediator.



[Inherited]

DistributedSystemArray class is an ExternalSystemArray and an abstract class containing and managing distributed system drivers within framework of master. The DistributedSystemArray class allocates DistributedSystemRole objects to DistributedSystem objects for optimization.

Unlike the ExternalSystemRole class which is dependent on an ExternalSystem, DistributedSystemRole is not dependent on a DistributedSystem and even can be allocated to multiple DistributedSystem objects. Allocation of DistributedSystemRole(s) to DistributedSystem(s) is determined by estimation of performance index pursuing best distribution. If number of DistributedSystem and DistributedSystemRole objects is not a lot, optimize by combined permuation algorithm. Else the objects are a lot, the allocation will be optimized by genetic algorithm.



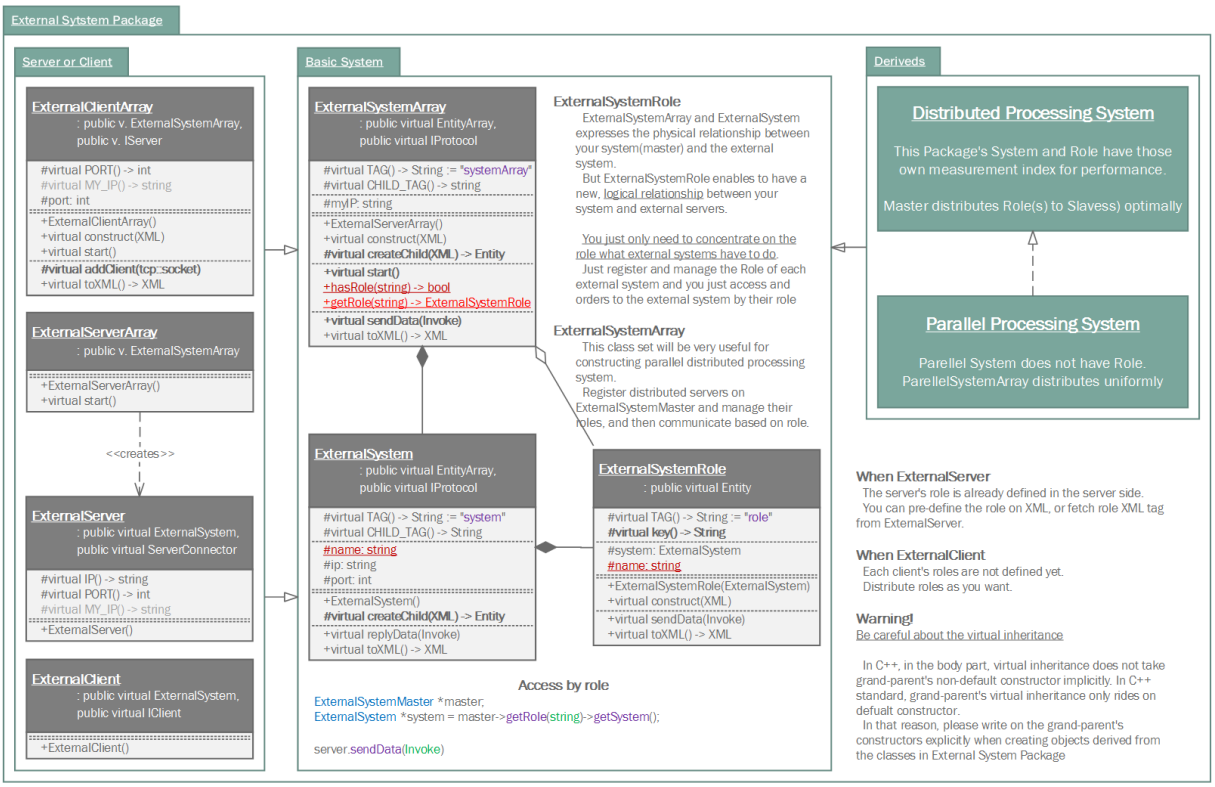
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

- ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 24 of file DistributedClientArrayMediator.hpp.

10.21.2 Constructor & Destructor Documentation

10.21.2.1 samchon::protocol::master::DistributedClientArrayMediator::DistributedClientArrayMediator ( )

Default Constructor.

10.21.3 Member Function Documentation

10.21.3.1 virtual void samchon::protocol::master::DistributedClientArrayMediator::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Reimplemented from samchon::protocol::master::DistributedClientArray.

10.21.3.2 virtual void samchon::protocol::master::DistributedClientArrayMediator::start ( ) [override], [virtual]

Start interaction.

An abstract method starting interaction with external systems.

If external systems are servers, starts connection to them, else clients, opens a server and accepts the external systems. You can addict your own procudures of starting drivers, but if you directly override method of abstract ExternalSystemArray, be careful about virtual inheritance.

- DistributedClientArray::start() calls DistributedSystemArray::start() and ExternalClientArray:start()

Reimplemented from samchon::protocol::master::DistributedClientArray.

10.21.3.3 virtual auto samchon::protocol::master::DistributedClientArrayMediator::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::master::DistributedClientArray.

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

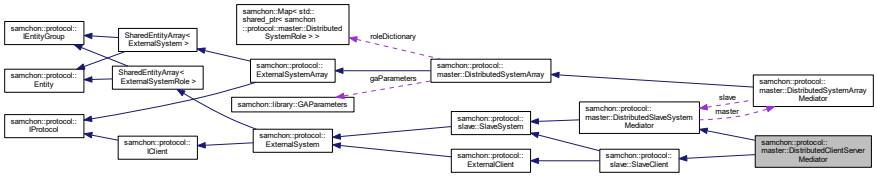
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DistributedClientArrayMediator.hpp

10.22 samchon::protocol::master::DistributedClientServerMediator Class Reference

A distributed slave client for mediation.

`#include <DistributedSlaveClientMediator.hpp>`

Collaboration diagram for samchon::protocol::master::DistributedClientServerMediator:



Public Member Functions

- DistributedClientServerMediator ()

  Default Constructor.

Additional Inherited Members

10.22.1 Detailed Description

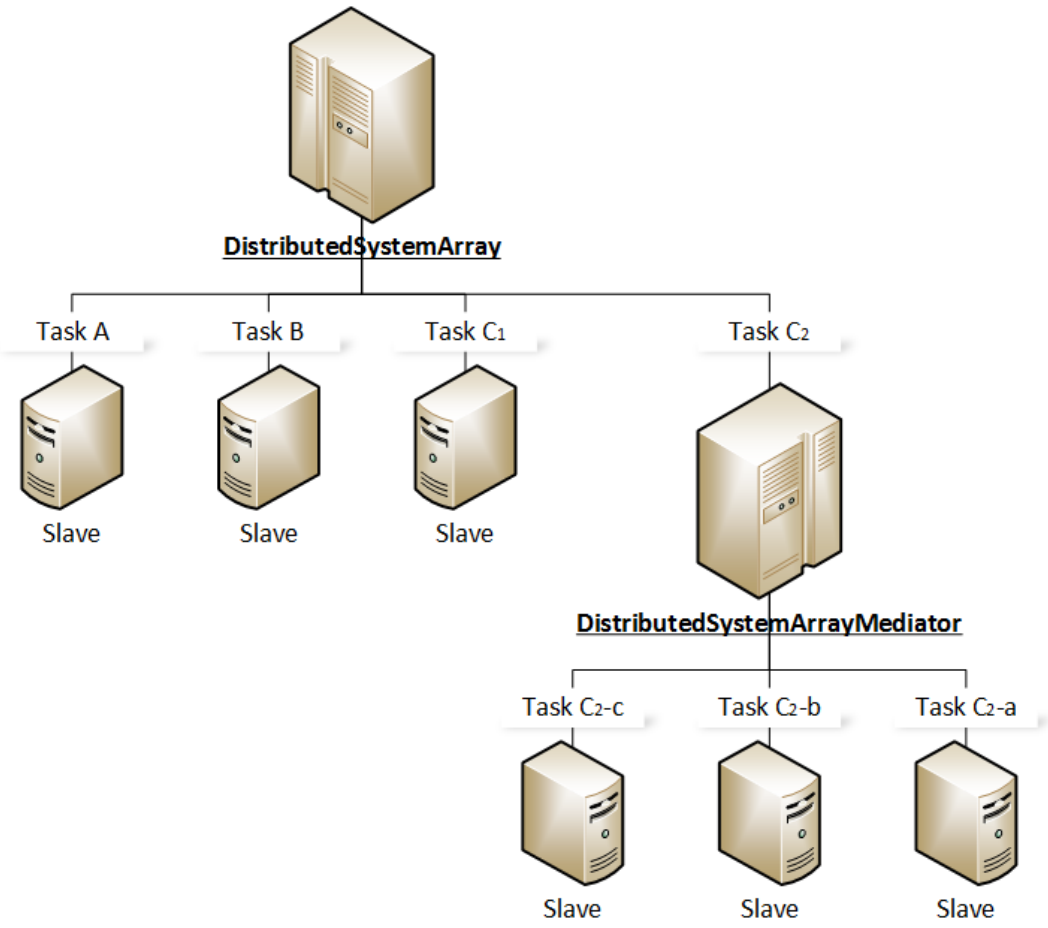
A distributed slave client for mediation.

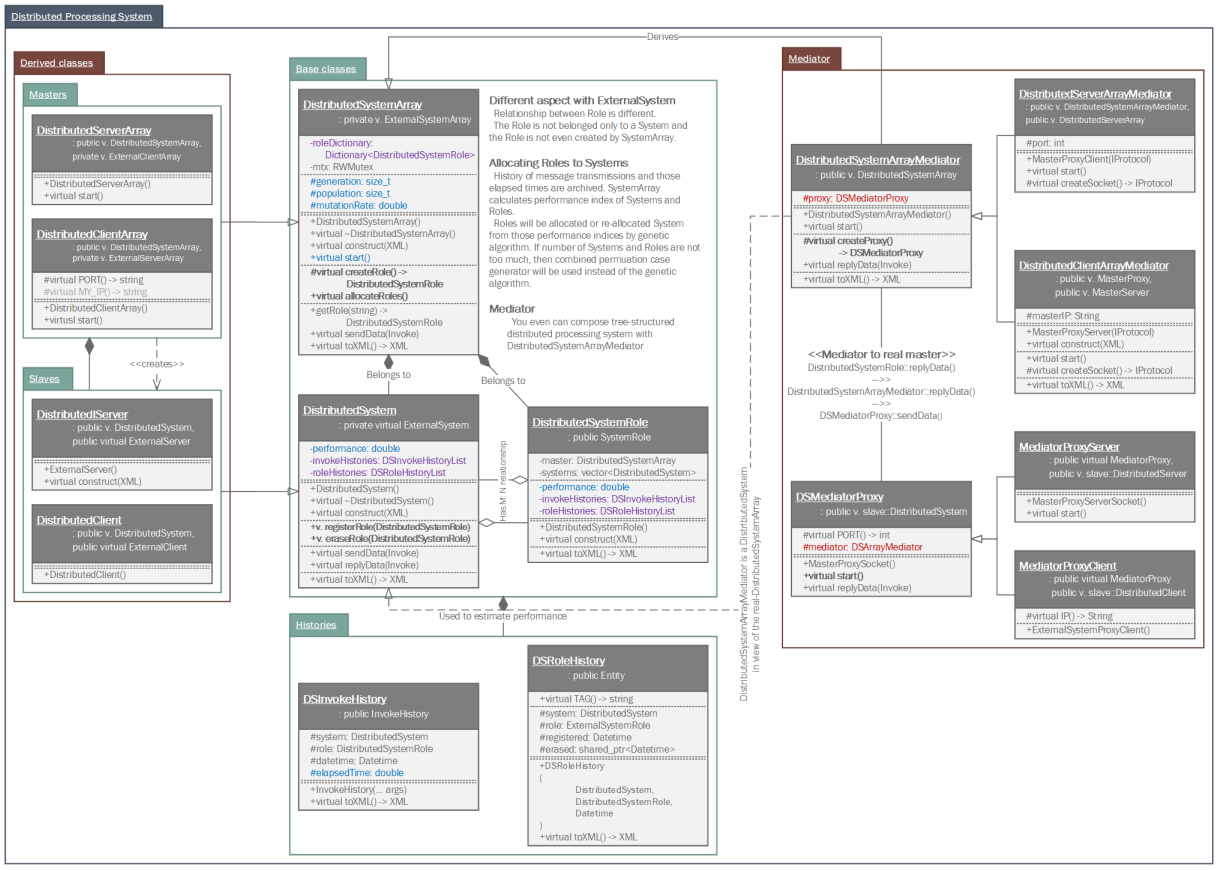
DistributedClientServerMediator is a DistributedSlaveSystemMediator class specialized in client driver.

[Inherited]

DistributedSlaveSystemMediator is an abstract class derived from slave::SlaveSystem for realizing tree-structured distriuted processins system. DistributedSlaveSystemMediator acts a slave system's role for its master, on the above level.

DistributedSystemArrayMediator and DistributedSlaveSystemMediator classes are paired to act a mediator system role by acting their own role, master system role for below slaves and slave system role for above master system.


DistributedSystemArray
Task A
Task B
Task C1
Task C2
Slave
Slave
Slave
DistributedSystemArrayMediator
Task C2-c
Task C2-b
Task C2-a
Slave
Slave
Slave


Distributed Processing System
Derived classes
Masters
DistributedServerArray
: public v. DistributedSystemArray,
private v. ExternalClientArray
+DistributedServerArray()
+virtual start()
DistributedClientArray
: public v. DistributedSystemArray,
private v. ExternalServerArray
#virtual PORT() -> string
#virtual MY_IP() -> string
+DistributedClientArray()
+virtual start()
<<creates>>
Slaves
DistributedIServer
: public v. DistributedSystem,
public virtual ExternalServer
+ExternalServer()
+virtual construct(XML)
DistributedClient
: public v. DistributedSystem,
public virtual ExternalClient
+DistributedClient()
Derives
Base classes
DistributedSystemArray
: private v. ExternalSystemArray
-roleDictionary:
Dictionary<DistributedSystemRole>
-mtx: RWMutex
#generation: size_t
#population: size_t
#mutationRate: double
+DistributedSystemArray()
+virtual ~DistributedSystemArray()
+virtual construct(XML)
+virtual start()
#virtual createRole() ->
DistributedSystemRole
+virtual allocateRoles()
+getRole(string) ->
DistributedSystemRole
+virtual sendData(Invoke)
+virtual toXML() -> XML
Different aspect with ExternalSystem
Relationship between Role is different.
The Role is not belonged only to a System and the Role is not even created by SystemArray.
Allocating Roles to Systems
History of message transmissions and those elapsed times are archived. SystemArray calculates performance index of Systems and Roles.
Roles will be allocated or re-allocated System from those performance indices by genetic algorithm. If number of Systems and Roles are not too much, then combined permuation case generator will be used instead of the genetic algorithm.
Mediator
You even can compose tree-structured distributed processing system with DistributedSystemArrayMediator
Belongs to
Belongs to
DistributedSystem
: private virtual ExternalSystem
-performance: double
-invokeHistories: DSInvokeHistoryList
-roleHistories: DSRoleHistoryList
+DistributedSystem()
+virtual ~DistributedSystem()
+virtual construct(XML)
+v. registerRole(DistributedSystemRole)
+v. eraseRole(DistributedSystemRole)
+virtual sendData(Invoke)
+virtual replyData(Invoke)
+virtual toXML() -> XML
Has N:N relationship
DistributedSystemRole
: public SystemRole
-master: DistributedSystemArray
-systems: vector<DistributedSystem>
-performance: double
-invokeHistories: DSInvokeHistoryList
-roleHistories: DSRoleHistoryList
+DistributedSystemRole()
+virtual construct(XML)
+virtual toXML() -> XML
Used to estimate performance
Histories
DSInvokeHistory
: public InvokeHistory
#system: DistributedSystem
#role: DistributedSystemRole
#datetime: Datetime
#elapsedTime: double
+InvokeHistory(... args)
+virtual toXML() -> XML
DSRoleHistory
: public Entity
+virtual TAG() -> string
#system: DistributedSystem
#role: ExternalSystemRole
#registered: Datetime
#erased: shared_ptr<Datetime>
+DSRoleHistory
(
DistributedSystem,
DistributedSystemRole,
Datetime
)
+virtual toXML() -> XML
DistributedSystemArrayMediator is a DistributedSystem in view of the real-DistributedSystemArray
Mediator
DistributedSystemArrayMediator
: public v. DistributedSystemArray
#proxy: DSMediatorProxy
+DistributedSystemArrayMediator()
+virtual start()
#virtual createProxy()
-> DSMediatorProxy
+virtual replyData(Invoke)
+virtual toXML() -> XML
<<Mediator to real master>>
DistributedSystemRole::replyData()
-->>
DistributedSystemArrayMediator::replyData()
-->>
DSMediatorProxy::sendData()
DSMediatorProxy
: public v. slave::DistributedSystem
#virtual PORT() -> int
#mediator: DSArrayMediator
+MasterProxySocket()
+virtual start()
+virtual replyData(Invoke)
DistributedServerArrayMediator
: public v. DistributedSystemArrayMediator,
public v. DistributedServerArray
#port: int
+MasterProxyClient(IProtocol)
+virtual start()
#virtual createSocket() -> IProtocol
DistributedClientArrayMediator
: public v. MasterProxy,
public v. MasterServer
#masterIP: String
+MasterProxyServer(IProtocol)
+virtual construct(XML)
+virtual start()
#virtual createSocket() -> IProtocol
+virtual toXML() -> XML
MediatorProxyServer
: public virtual MediatorProxy,
public v. slave::DistributedServer
+MasterProxyServerSocket()
+virtual start()
MediatorProxyClient
: public virtual MediatorProxy
public v. slave::DistributedClient
#virtual IP() -> String
+ExternalSystemProxyClient()

[Inherited]

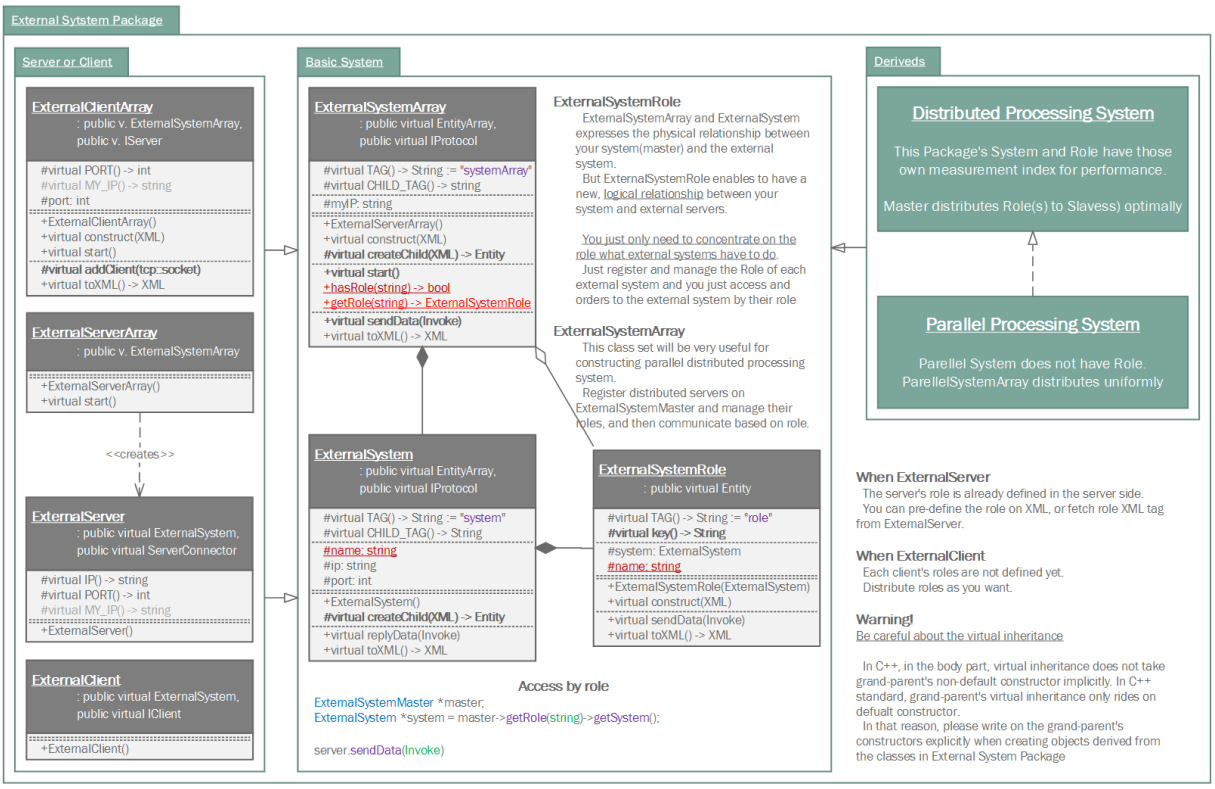
SlaveSystem, literally, means a slave system belongs to a maste system.

The SlaveSystem class is used in opposite side system of master::DistributedSystem and master::ParallelSystem and reports elapsed time of each commmand (by Invoke message) for estimation of its performance.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, External↩System is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystemRole.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an External↩SystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an ExternalSystem. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 23 of file DistributedSlaveClientMediator.hpp.

10.22.2 Constructor & Destructor Documentation

10.22.2.1 samchon::protocol::master::DistributedClientServerMediator::DistributedClientServerMediator ( )

Default Constructor.

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

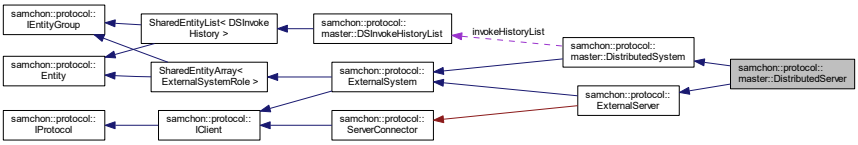
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DistributedSlaveClientMediator.hpp

10.23 samchon::protocol::master::DistributedServer Class Reference

A network driver for a distributed server.

`#include <DistributedServer.hpp>`

Collaboration diagram for samchon::protocol::master::DistributedServer:



Public Member Functions

- DistributedServer ()

  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override

  Construct data of the Entity from an XML object.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the EntityGroup.

Additional Inherited Members

10.23.1 Detailed Description

A network driver for a distributed server.

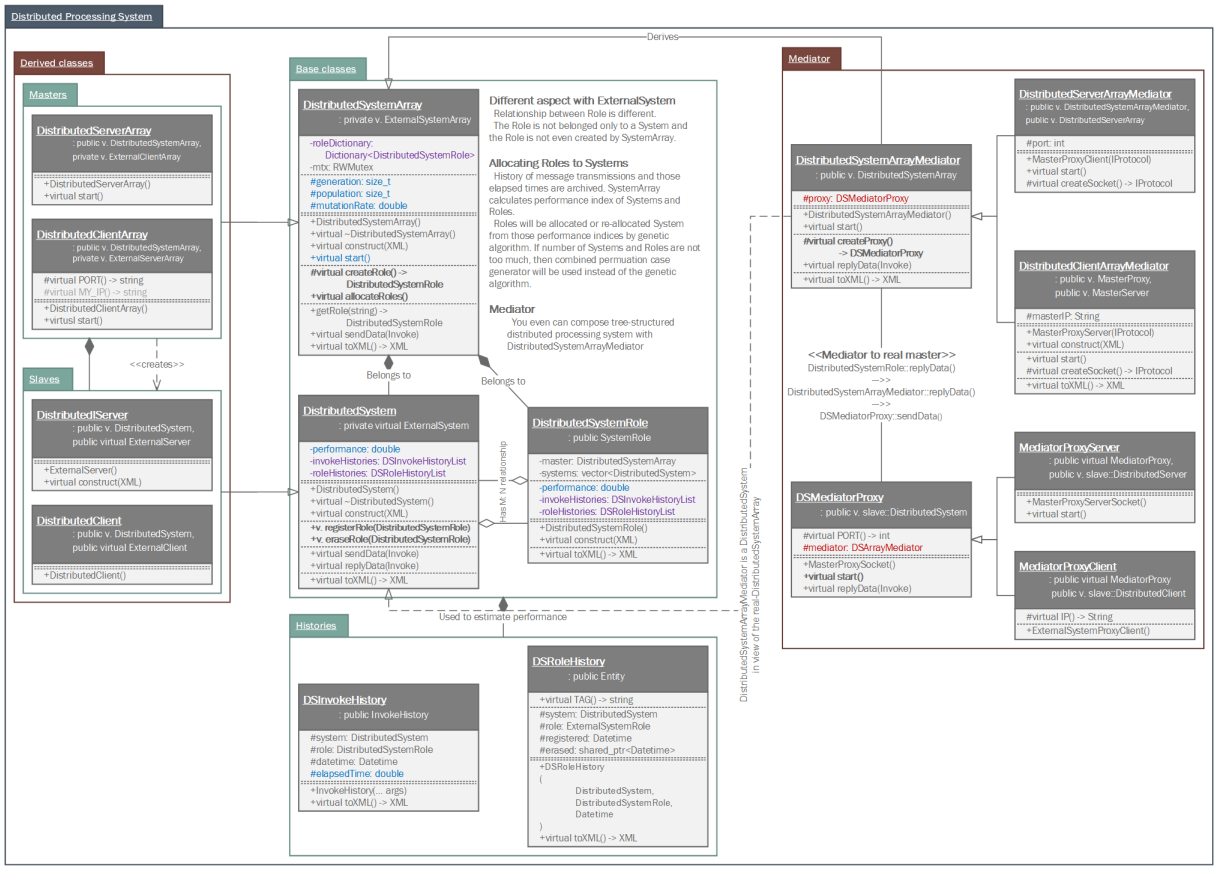
DistributedServer is an DistributedSystem specialized in server driver.

[Inherited]

DistributedSystem class is an ExternalSystem having role as DistributedSystemRole objects with performance index and history log allocation of DistributedSystemRole objects and have requsted Invoke message to related external system.

The greatest difference between DistributedSystem and ExternalSystem is, unlike the ExternalSystem class which having is ExternalSystemRole objects exclusively, DistributedSystem does not monopoly a DistributedSystemRole. The DistributedSystemRole can allocatedto multiple DistributedSystem.

DistributedSystem and DistributedSystemRole classes have performance index and history log archiving what they've requested to the related distributed system. Allocating DistributedSystemRole objects to DistributedSyste will be determined those performance index and history log for optimize distribution of elasped time about each Invoke message processing.

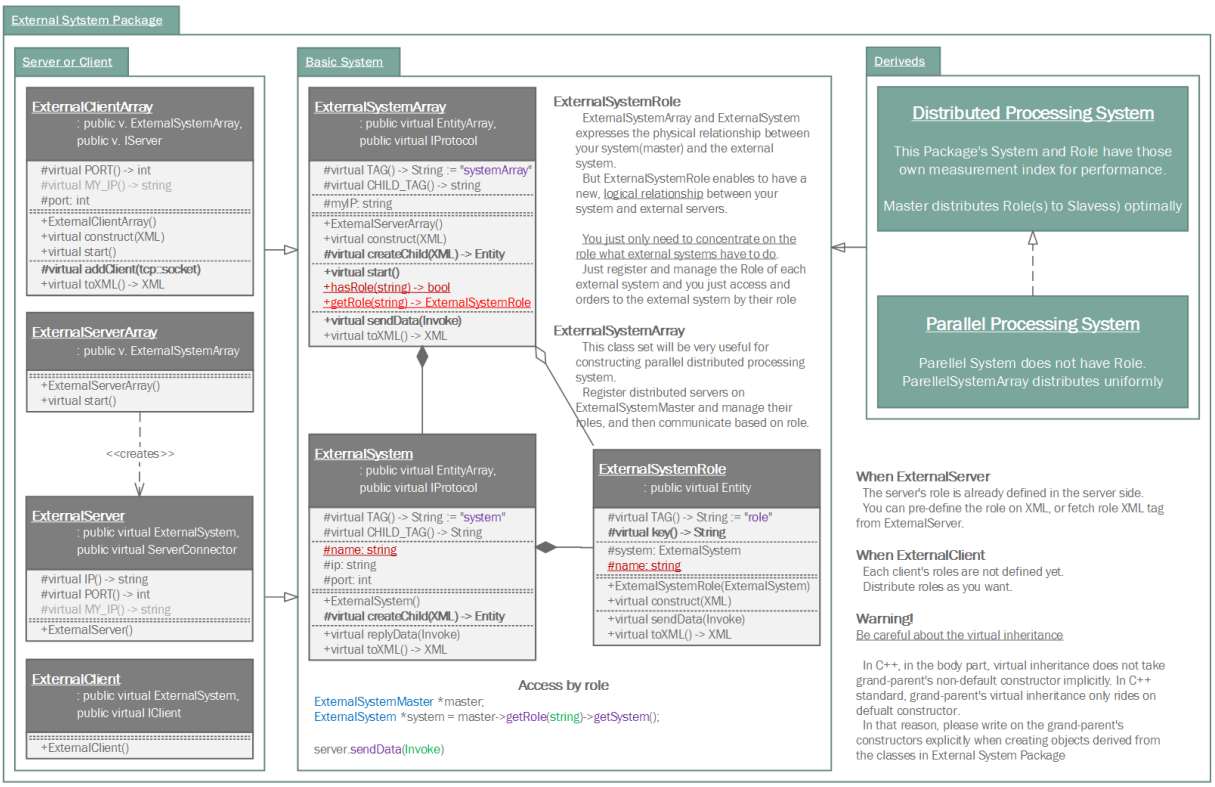


Todo

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, External↩System is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystemRole.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an External↩ SystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an ExternalSystem. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 22 of file DistributedServer.hpp.

10.23.2 Constructor & Destructor Documentation

10.23.2.1 samchon::protocol::master::DistributedServer::DistributedServer ( )

Default Constructor.

10.23.3 Member Function Documentation

10.23.3.1 virtual void samchon::protocol::master::DistributedServer::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Reimplemented from samchon::protocol::ExternalServer.

10.23.3.2 virtual auto samchon::protocol::master::DistributedServer::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::ExternalServer.

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

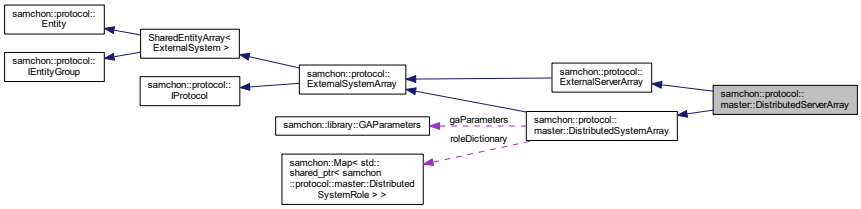
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DistributedServer.hpp

10.24 samchon::protocol::master::DistributedServerArray Class Reference

An array of DistributedServer(s).

```
#include <DistributedServerArray.hpp>
```

Collaboration diagram for samchon::protocol::master::DistributedServerArray:



Public Member Functions

- DistributedServerArray ()

  Default Constructor.
- virtual void start () override

  Start interaction.

Additional Inherited Members

10.24.1 Detailed Description

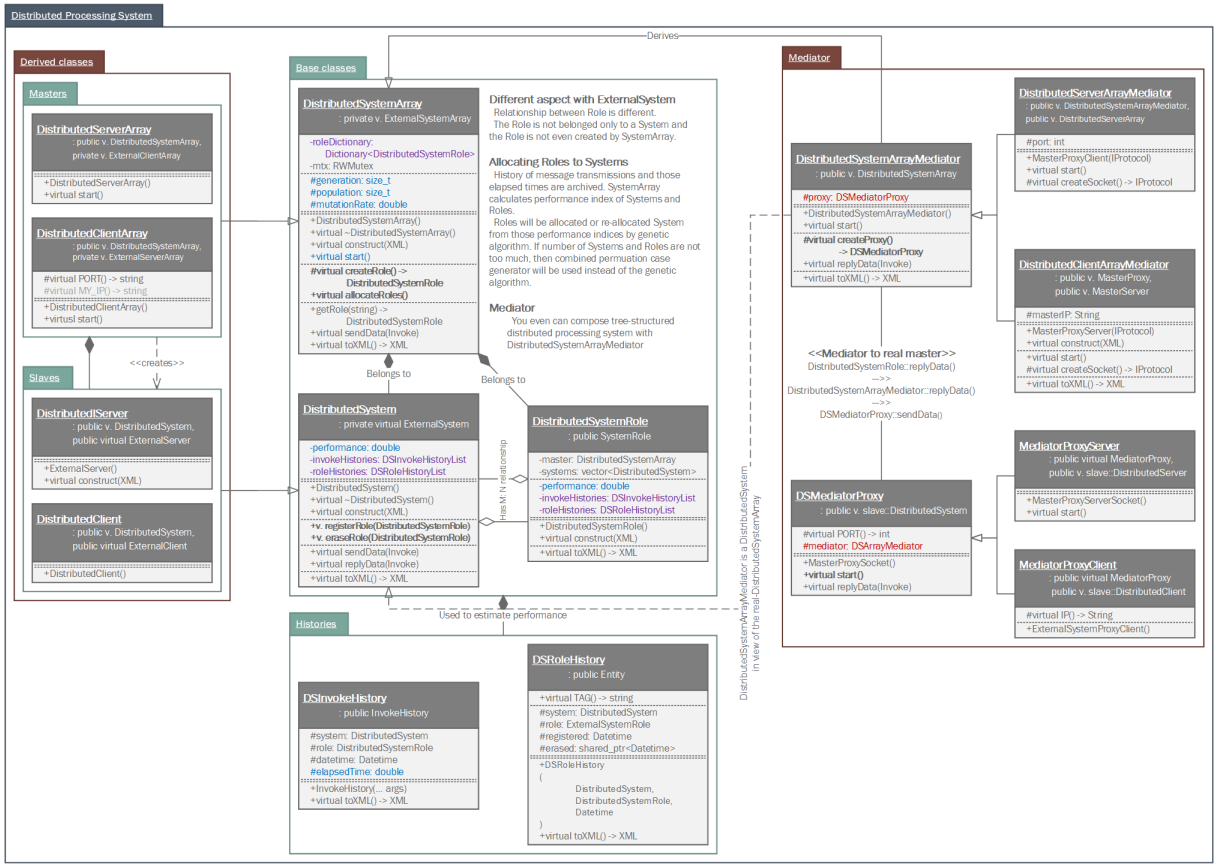
An array of DistributedServer(s).

DistributedServerArray is an DistributedSystemArray specialized in servers' driver.

[Inherited]

DistributedSystemArray class is an ExternalSystemArray and an abstract class containing and managing distributed system drivers within framework of master. The DistributedSystemArray class allocates DistributedSystemRole objects to DistributedSystem objects for optimization.

Unlike the ExternalSystemRole class which is dependent on an ExternalSystem, DistributedSystemRole is not dependent on a DistributedSystem and even can be allocated to multiple DistributedSystem objects. Allocation of DistributedSystemRole(s) to DistributedSystem(s) is determined by estimation of performance index pursuing best distribution. If number of DistributedSystem and DistributedSystemRole objects is not a lot, optimize by combined permuation algorithm. Else the objects are a lot, the allocation will be optimized by genetic algorithm.



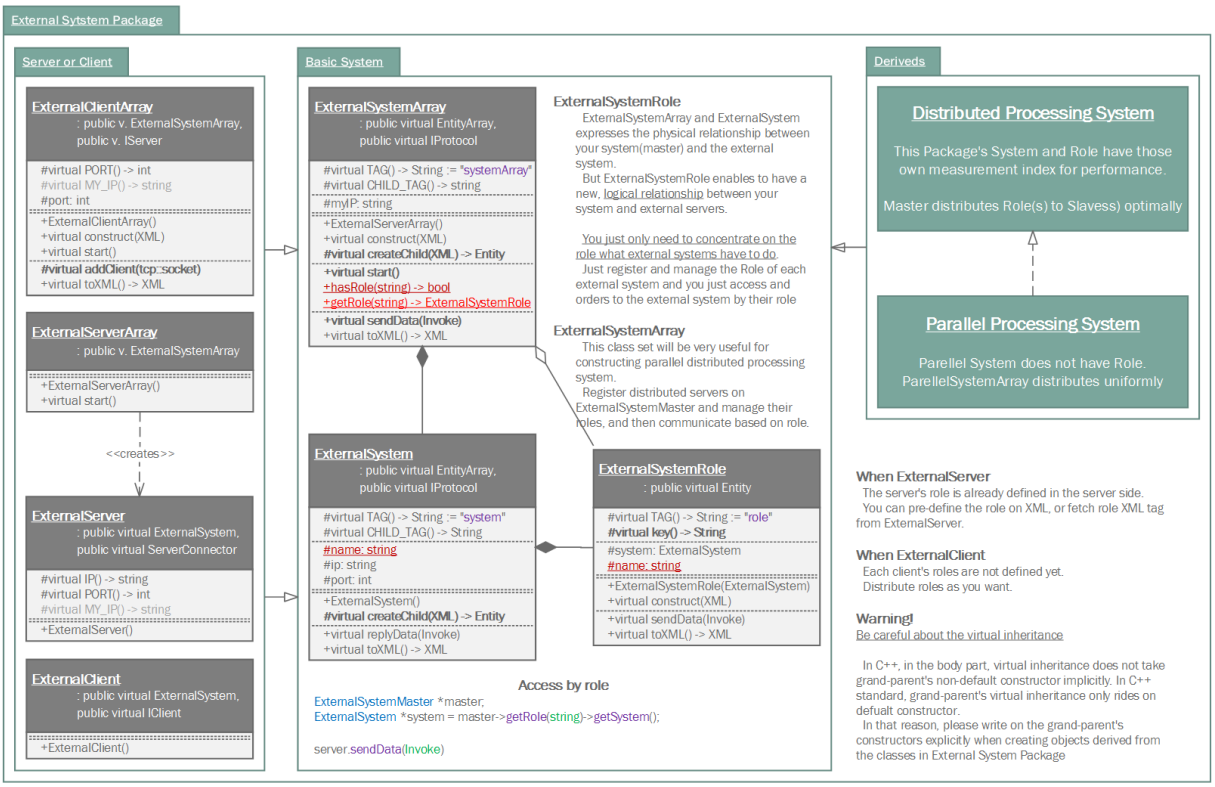
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

- ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 22 of file DistributedServerArray.hpp.

10.24.2 Constructor & Destructor Documentation

10.24.2.1 samchon::protocol::master::DistributedServerArray::DistributedServerArray ( )

Default Constructor.

10.24.3 Member Function Documentation

10.24.3.1 virtual void samchon::protocol::master::DistributedServerArray::start ( ) [override], [virtual]

Start interaction.

An abstract method starting interaction with external systems.

If external systems are servers, starts connection to them, else clients, opens a server and accepts the external systems. You can addict your own procudures of starting drivers, but if you directly override method of abstract ExternalSystemArray, be careful about virtual inheritance.

- DistributedClientArray::start() calls DistributedSystemArray::start() and ExternalClientArray:start()

Reimplemented from samchon::protocol::ExternalServerArray.

Reimplemented in samchon::protocol::master::DistributedServerArrayMediator.

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

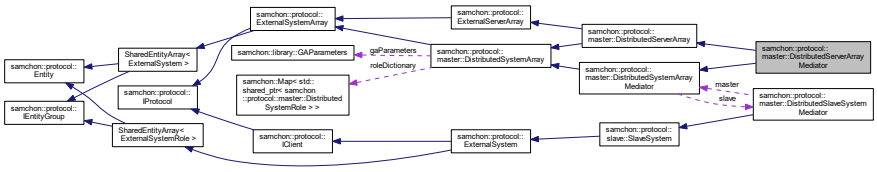
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DistributedServerArray.hpp

10.25 samchon::protocol::master::DistributedServerArrayMediator Class Reference

A mediator of distributed system between master and slaves.

`#include <DistributedServerArrayMediator.hpp>`

Collaboration diagram for samchon::protocol::master::DistributedServerArrayMediator:



Public Member Functions

- DistributedServerArrayMediator ()

  Default Constructor.

- virtual void start () override

  Start interaction.

Additional Inherited Members

10.25.1 Detailed Description

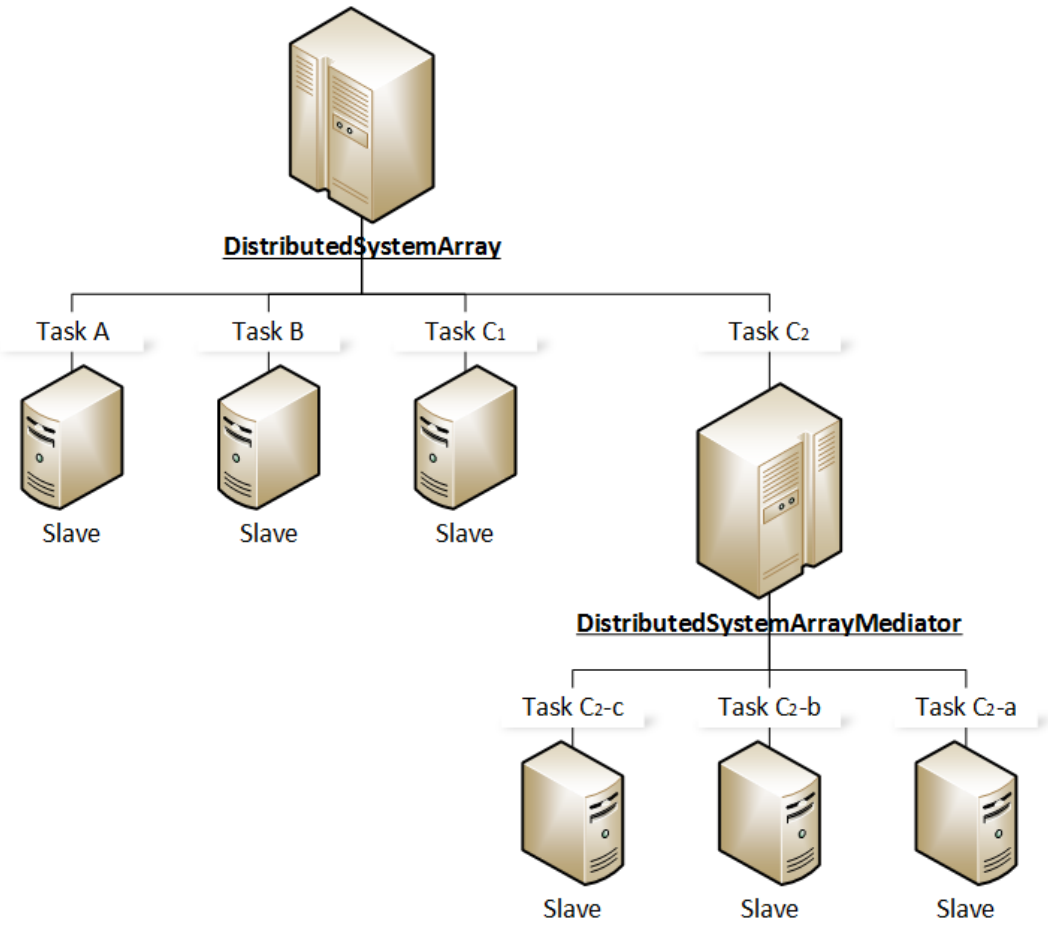
A mediator of distributed system between master and slaves.

DistributedServerArrayMediator is a DistributedSystemArrayMediator specialized in servers' driver. Distributed↩ ServerArrayMediator is a server for its master and also manages children servers' slaves.

[Inherited]

DistributedSystemArrayMediator class is used to realizing a mediator system of tree-structured distributed processing system. DistributedSystemArrayMediator intermediates between origin-master(DistributedSystemArray) and its slaves(slave::DistributedSystem).

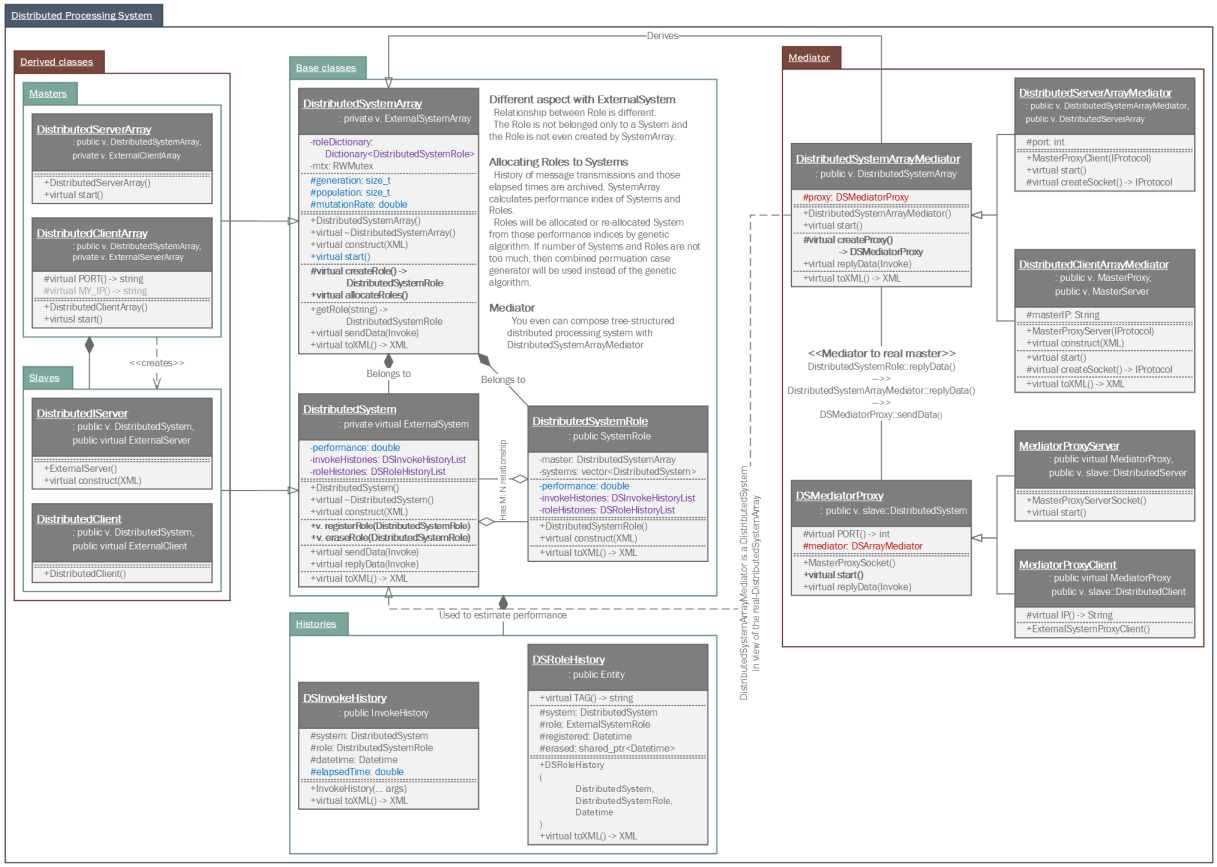
DistributedSystemArrayMediator is a master(DistributedSystemArray) within the framework of slaves systems (slave::DistributedSystem objects) and also can be a slave system (slave::DistributedSystem) within framework of its master(DistributedSystemArray) by having DistributedSlaveSystemMediator.



[Inherited]

DistributedSystemArray class is an ExternalSystemArray and an abstract class containing and managing distributed system drivers within framework of master. The DistributedSystemArray class allocates DistributedSystemRole objects to DistributedSystem objects for optimization.

Unlike the ExternalSystemRole class which is dependent on an ExternalSystem, DistributedSystemRole is not dependent on a DistributedSystem and even can be allocated to multiple DistributedSystem objects. Allocation of DistributedSystemRole(s) to DistributedSystem(s) is determined by estimation of performance index pursuing best distribution. If number of DistributedSystem and DistributedSystemRole objects is not a lot, optimize by combined permuation algorithm. Else the objects are a lot, the allocation will be optimized by genetic algorithm.



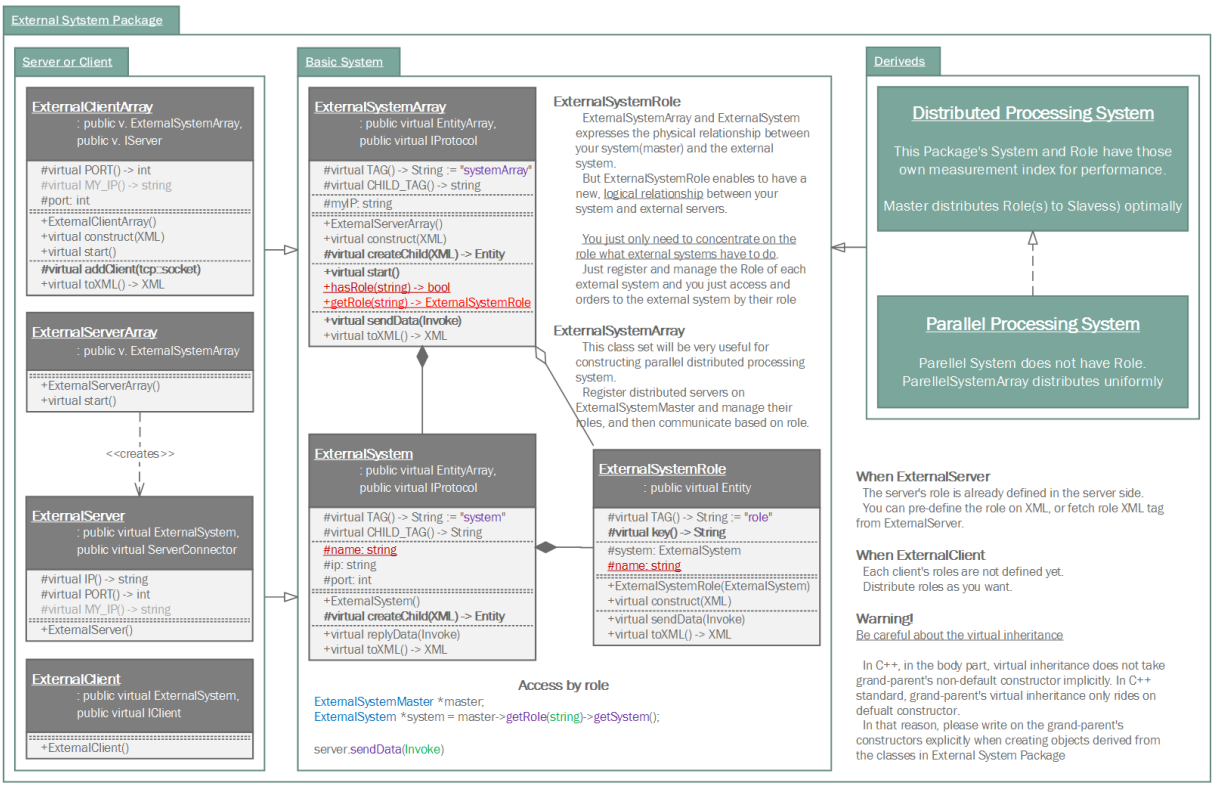
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

- ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 24 of file DistributedServerArrayMediator.hpp.

10.25.2 Constructor & Destructor Documentation

10.25.2.1 samchon::protocol::master::DistributedServerArrayMediator::DistributedServerArrayMediator ( )

Default Constructor.

10.25.3 Member Function Documentation

10.25.3.1 virtual void samchon::protocol::master::DistributedServerArrayMediator::start ( ) [override], [virtual]

Start interaction.

An abstract method starting interaction with external systems.

If external systems are servers, starts connection to them, else clients, opens a server and accepts the external systems. You can addict your own procudures of starting drivers, but if you directly override method of abstract ExternalSystemArray, be careful about virtual inheritance.

- DistributedClientArray::start() calls DistributedSystemArray::start() and ExternalClientArray:start()

Reimplemented from samchon::protocol::master::DistributedServerArray.

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

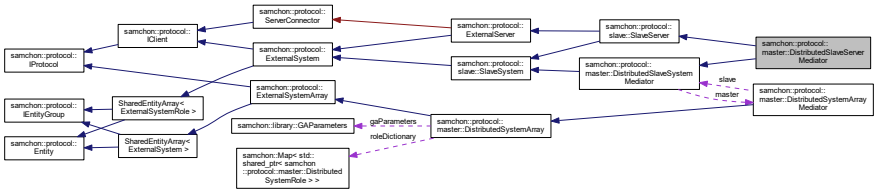
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DistributedServerArrayMediator.↩ hpp

10.26 samchon::protocol::master::DistributedSlaveServerMediator Class Reference

A distributed slave server for mediation.

`#include <DistributedSlaveServerMediator.hpp>`

Collaboration diagram for samchon::protocol::master::DistributedSlaveServerMediator:



Public Member Functions

- DistributedSlaveServerMediator ()

  Default Constructor.

Additional Inherited Members

10.26.1 Detailed Description

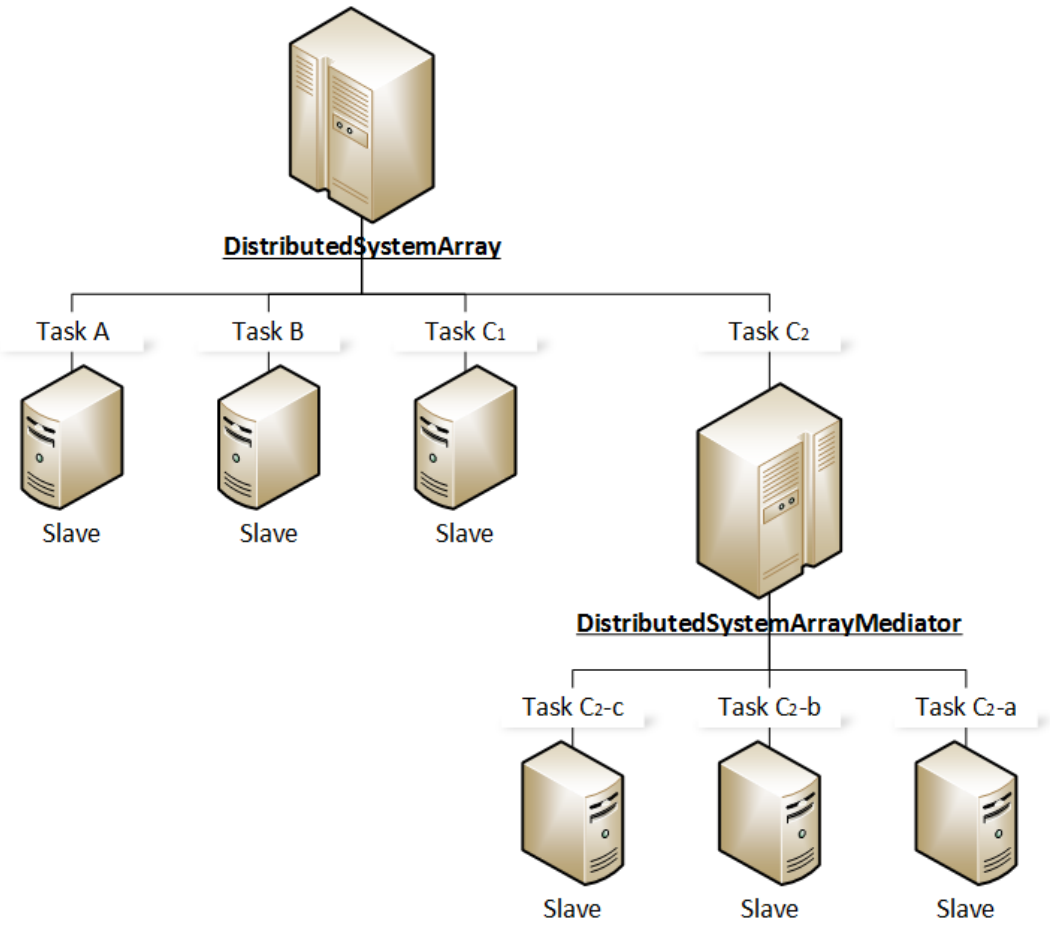
A distributed slave server for mediation.

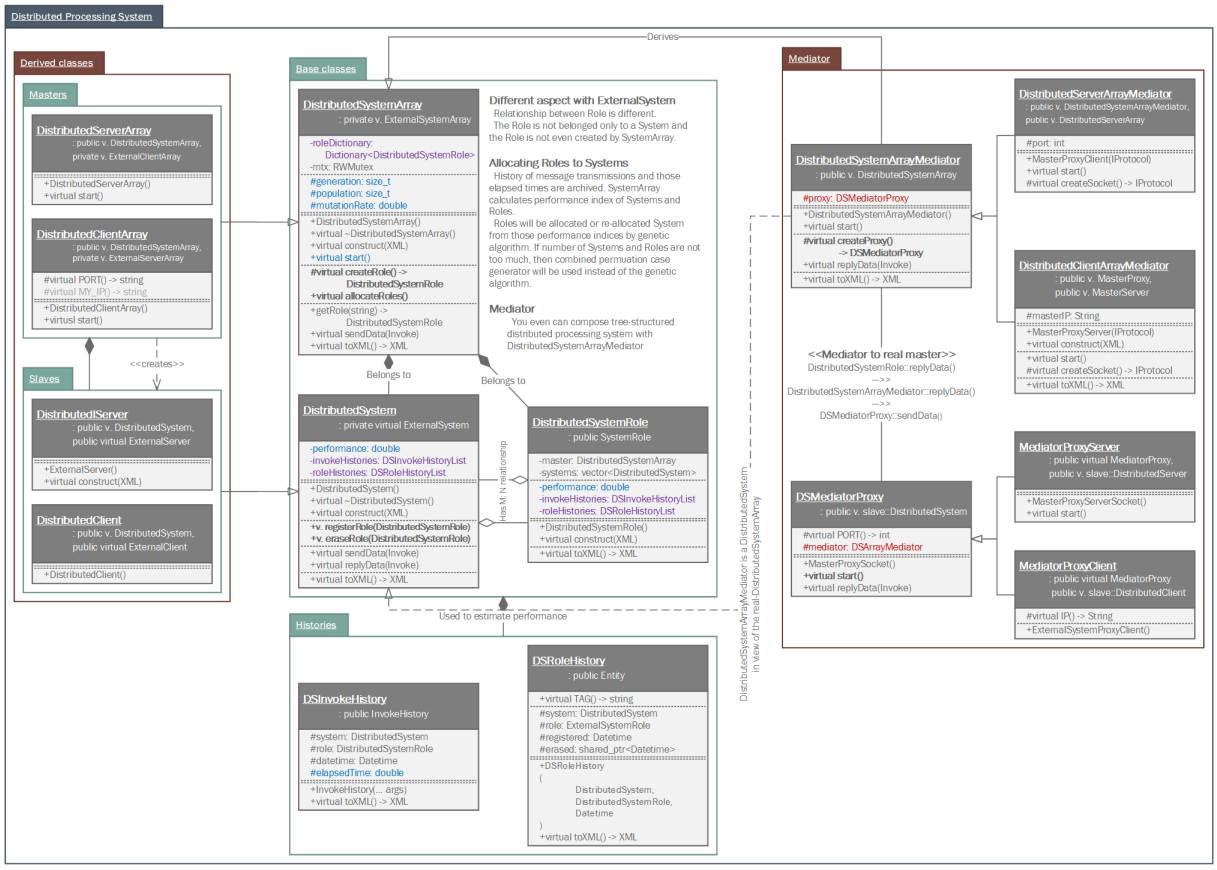
DistributedSlaveServerMediator is a DistributedSlaveSystemMediator class specialized in server driver.

[Inherited]

DistributedSlaveSystemMediator is an abstract class derived from slave::SlaveSystem for realizing tree-structured distriuted processins system. DistributedSlaveSystemMediator acts a slave system's role for its master, on the above level.

DistributedSystemArrayMediator and DistributedSlaveSystemMediator classes are paired to act a mediator system role by acting their own role, master system role for below slaves and slave system role for above master system.


DistributedSystemArray
Task A
Task B
Task C1
Task C2
Slave
Slave
Slave
DistributedSystemArrayMediator
Task C2-c
Task C2-b
Task C2-a
Slave
Slave
Slave


Distributed Processing System
Derived classes
Masters
DistributedServerArray
: public v. DistributedSystemArray,
private v. ExternalClientArray
+DistributedServerArray()
+virtual start()
DistributedClientArray
: public v. DistributedSystemArray,
private v. ExternalServerArray
#virtual PORT() -> string
#virtual MY_IP() -> string
+DistributedClientArray()
+virtual start()
<<creates>>
Slaves
DistributedIServer
: public v. DistributedSystem,
public virtual ExternalServer
+ExternalServer()
+virtual construct(XML)
DistributedClient
: public v. DistributedSystem,
public virtual ExternalClient
+DistributedClient()
Derives
Base classes
DistributedSystemArray
: private v. ExternalSystemArray
-roleDictionary:
Dictionary<DistributedSystemRole>
-mtx: RWMutex
#generation: size_t
#population: size_t
#mutationRate: double
+DistributedSystemArray()
+virtual ~DistributedSystemArray()
+virtual construct(XML)
+virtual start()
#virtual createRole() ->
DistributedSystemRole
+virtual allocateRoles()
+getRole(string) ->
DistributedSystemRole
+virtual sendData(Invoke)
+virtual toXML() -> XML
Different aspect with ExternalSystem
Relationship between Role is different.
The Role is not belonged only to a System and the Role is not even created by SystemArray.
Allocating Roles to Systems
History of message transmissions and those elapsed times are archived. SystemArray calculates performance index of Systems and Roles.
Roles will be allocated or re-allocated System from those performance indices by genetic algorithm. If number of Systems and Roles are not too much, then combined permuation case generator will be used instead of the genetic algorithm.
Mediator
You even can compose tree-structured distributed processing system with DistributedSystemArrayMediator
Belongs to
Belongs to
DistributedSystem
: private virtual ExternalSystem
-performance: double
-invokeHistories: DSInvokeHistoryList
-roleHistories: DSRoleHistoryList
+DistributedSystem()
+virtual ~DistributedSystem()
+virtual construct(XML)
+v. registerRole(DistributedSystemRole)
+v. eraseRole(DistributedSystemRole)
+virtual sendData(Invoke)
+virtual replyData(Invoke)
+virtual toXML() -> XML
Has N:N relationship
DistributedSystemRole
: public SystemRole
-master: DistributedSystemArray
-systems: vector<DistributedSystem>
-performance: double
-invokeHistories: DSInvokeHistoryList
-roleHistories: DSRoleHistoryList
+DistributedSystemRole()
+virtual construct(XML)
+virtual toXML() -> XML
Used to estimate performance
Histories
DSInvokeHistory
: public InvokeHistory
#system: DistributedSystem
#role: DistributedSystemRole
#datetime: Datetime
#elapsedTime: double
+InvokeHistory(... args)
+virtual toXML() -> XML
DSRoleHistory
: public Entity
+virtual TAG() -> string
#system: DistributedSystem
#role: ExternalSystemRole
#registered: Datetime
#erased: shared_ptr<Datetime>
+DSRoleHistory
(
DistributedSystem,
DistributedSystemRole,
Datetime
)
+virtual toXML() -> XML
DistributedSystemArrayMediator is a DistributedSystem in view of the real-DistributedSystemArray
Mediator
DistributedSystemArrayMediator
: public v. DistributedSystemArray
#proxy: DSMediatorProxy
+DistributedSystemArrayMediator()
+virtual start()
#virtual createProxy()
-> DSMediatorProxy
+virtual replyData(Invoke)
+virtual toXML() -> XML
<<Mediator to real master>>
DistributedSystemRole::replyData()
-->>
DistributedSystemArrayMediator::replyData()
-->>
DSMediatorProxy::sendData()
DSMediatorProxy
: public v. slave::DistributedSystem
#virtual PORT() -> int
#mediator: DSArrayMediator
+MasterProxySocket()
+virtual start()
+virtual replyData(Invoke)
DistributedServerArrayMediator
: public v. DistributedSystemArrayMediator,
public v. DistributedServerArray
#port: int
+MasterProxyClient(IProtocol)
+virtual start()
#virtual createSocket() -> IProtocol
DistributedClientArrayMediator
: public v. MasterProxy,
public v. MasterServer
#masterIP: String
+MasterProxyServer(IProtocol)
+virtual construct(XML)
+virtual start()
#virtual createSocket() -> IProtocol
+virtual toXML() -> XML
MediatorProxyServer
: public virtual MediatorProxy,
public v. slave::DistributedServer
+MasterProxyServerSocket()
+virtual start()
MediatorProxyClient
: public virtual MediatorProxy
public v. slave::DistributedClient
#virtual IP() -> String
+ExternalSystemProxyClient()

[Inherited]

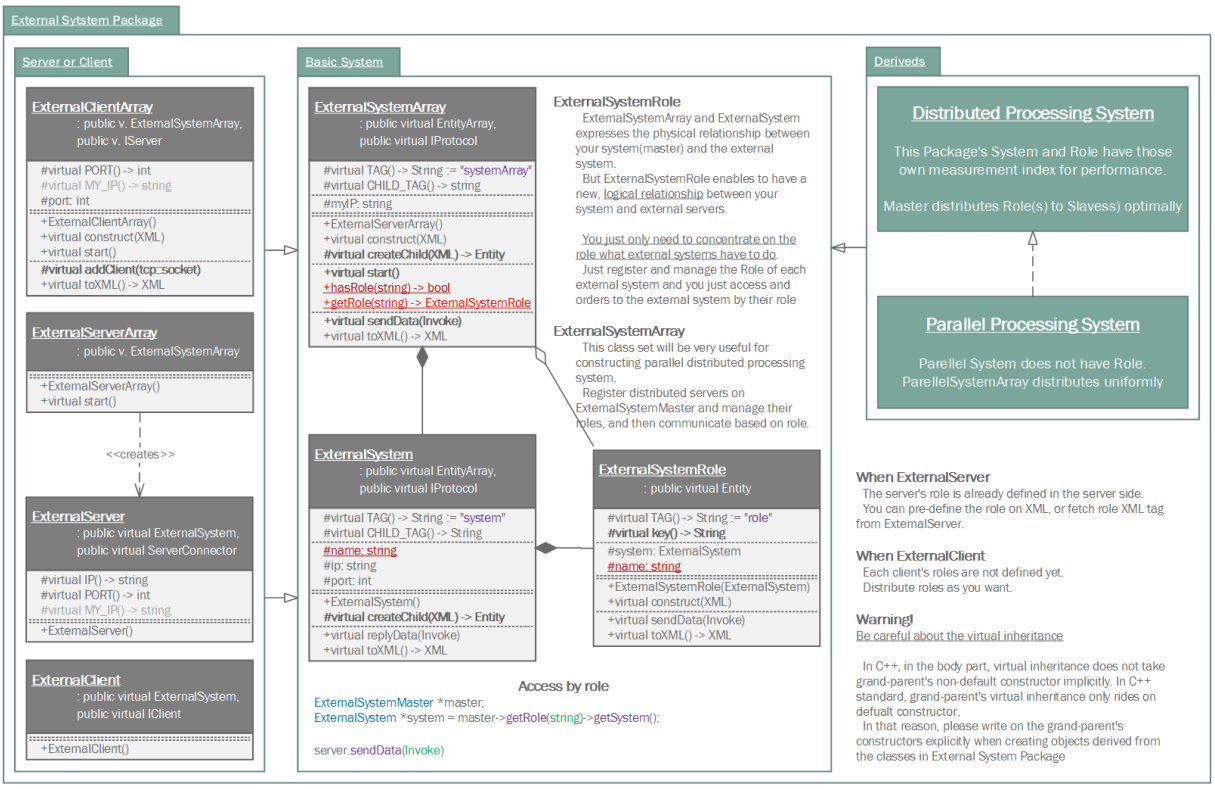
SlaveSystem, literally, means a slave system belongs to a maste system.

The SlaveSystem class is used in opposite side system of master::DistributedSystem and master::ParallelSystem and reports elapsed time of each commmand (by Invoke message) for estimation of its performance.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, External↩System is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystemRole.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an External↩SystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an ExternalSystem. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 23 of file DistributedSlaveServerMediator.hpp.

10.26.2 Constructor & Destructor Documentation

10.26.2.1 samchon::protocol::master::DistributedSlaveServerMediator::DistributedSlaveServerMediator ( )

Default Constructor.

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

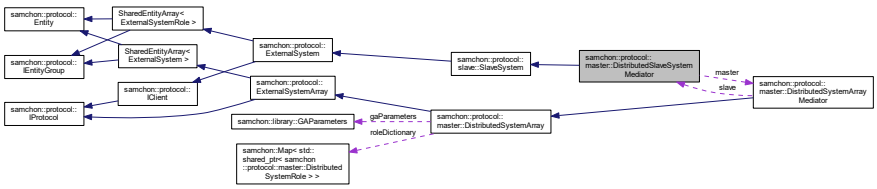
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DistributedSlaveServerMediator.↩hpp

10.27 samchon::protocol::master::DistributedSlaveSystemMediator Class Reference

A distributed slave system for mediation.

`#include <DistributedSlaveSystemMediator.hpp>`

Collaboration diagram for samchon::protocol::master::DistributedSlaveSystemMediator:



Public Member Functions

- DistributedSlaveSystemMediator ()

  Default Constructor.
- virtual void replyData (std::shared_ptr< Invoke >) override

  Handling replied message from an external system.
- virtual auto TAG () const -> std::string override

  A tag name when represented by XML.

Protected Attributes

- DistributedSystemArrayMediator * master

  A master system for mediation.

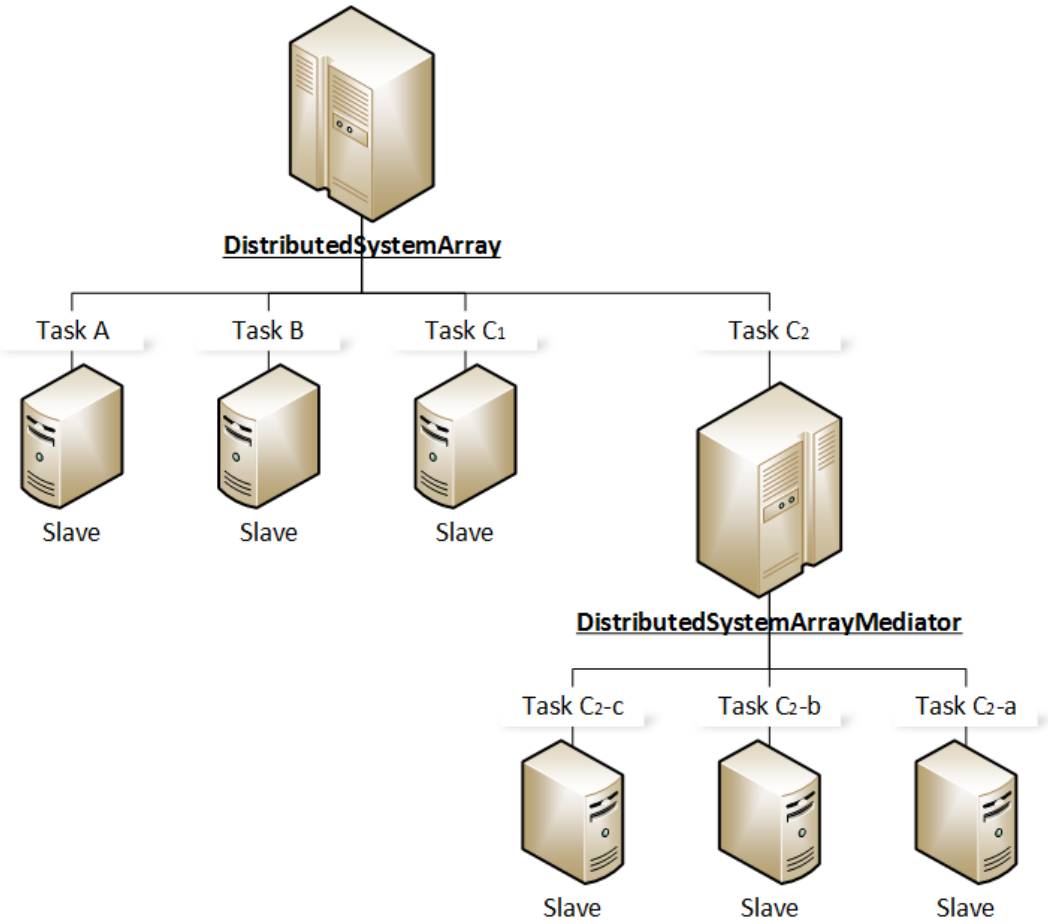
Additional Inherited Members

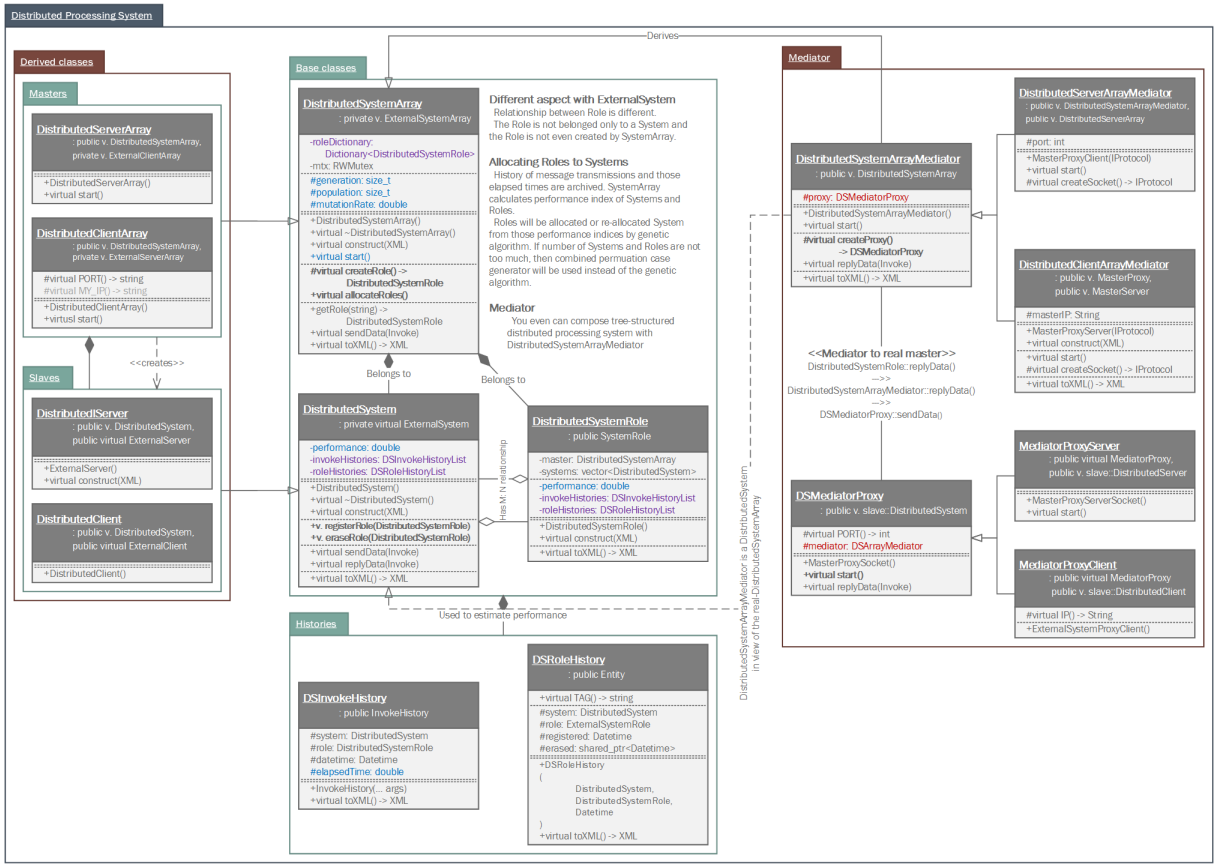
10.27.1 Detailed Description

A distributed slave system for mediation.

DistributedSlaveSystemMediator is an abstract class derived from slave::SlaveSystem for realizing tree-structured distriuted processins system. DistributedSlaveSystemMediator acts a slave system's role for its master, on the above level.

DistributedSystemArrayMediator and DistributedSlaveSystemMediator classes are paired to act a mediator system role by acting their own role, master system role for below slaves and slave system role for above master system.





[Inherited]

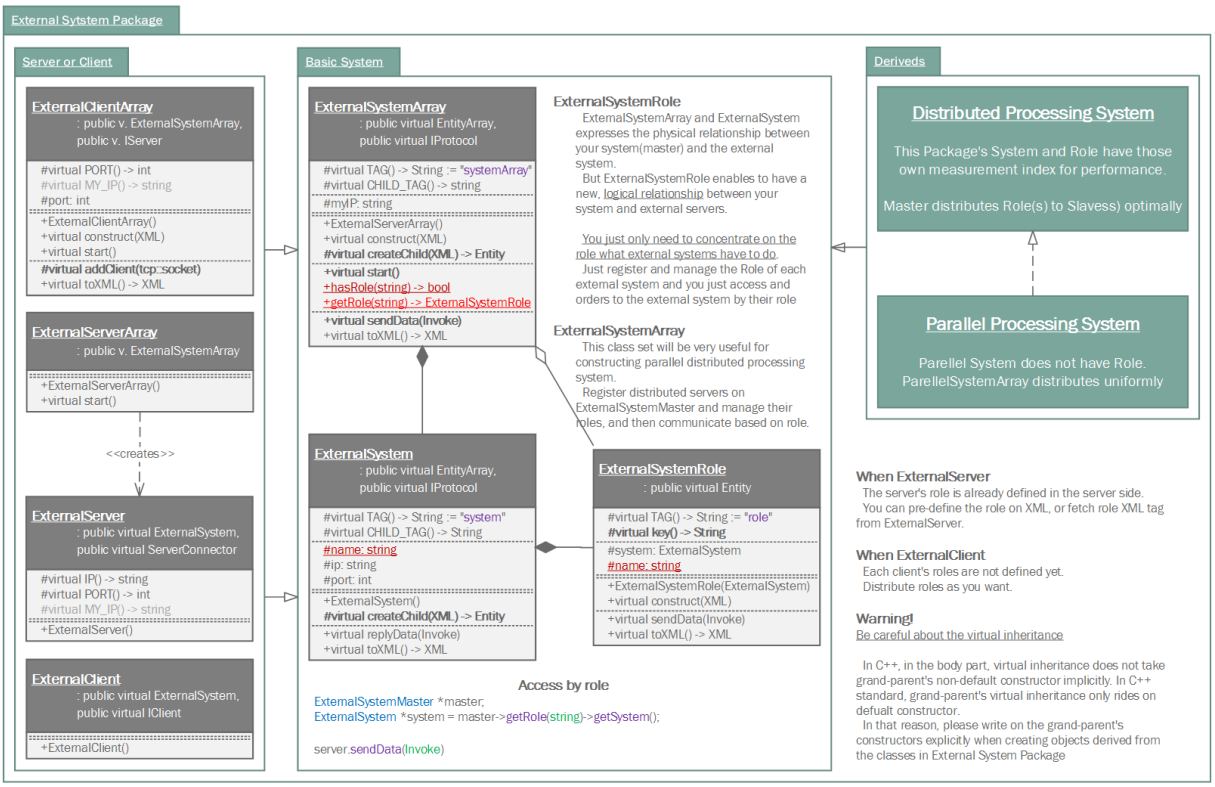
SlaveSystem, literally, means a slave system belongs to a maste system.

The SlaveSystem class is used in opposite side system of master::DistributedSystem and master::ParallelSystem and reports elapsed time of each commmand (by Invoke message) for estimation of its performance.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, External↩System is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystemRole.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an External↩ SystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an ExternalSystem. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 35 of file DistributedSlaveSystemMediator.hpp.

10.27.2 Constructor & Destructor Documentation

10.27.2.1 samchon::protocol::master::DistributedSlaveSystemMediator::DistributedSlaveSystemMediator ( )

Default Constructor.

10.27.3 Member Function Documentation

10.27.3.1 virtual void samchon::protocol::master::DistributedSlaveSystemMediator::replyData (std::shared_ptr< Invoke >) [override], [virtual]

Handling replied message from an external system.

If there's a related ExternalSystemRole, shifts responsibility to the ExternalSystemRole.

Parameters

invoke	Replied Invoke message

Reimplemented from samchon::protocol::ExternalSystem.

10.27.3.2 virtual auto samchon::protocol::master::DistributedSlaveSystemMediator::TAG ( ) const -> std::string [override], [virtual]

A tag name when represented by XML.

Returns

A tag name

Reimplemented from samchon::protocol::ExternalSystem.

10.27.4 Member Data Documentation

10.27.4.1 DistributedSystemArrayMediator* samchon::protocol::master::DistributedSlaveSystemMediator::master [protected]

A master system for mediation.

Definition at line 45 of file DistributedSlaveSystemMediator.hpp.

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

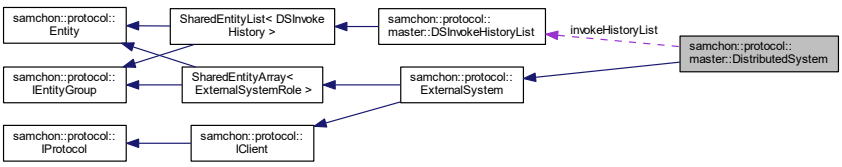
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DistributedSlaveSystemMediator.↩hpp

10.28 samchon::protocol::master::DistributedSystem Class Reference

A network driver for a distributed system.

```
#include <DistributedSystem.hpp>
```

Collaboration diagram for samchon::protocol::master::DistributedSystem:



Public Member Functions

- DistributedSystem ()

  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override

  Construct data of the Entity from an XML object.
- virtual void sendData (std::shared_ptr< Invoke >) override

  Sends message to a related system.
- virtual void replyData (std::shared_ptr< Invoke >) override

  Handling replied message from an external system.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the EntityGroup.

Protected Attributes

- size_t inProgress

  A number of processes running on the distributed system.
- size_t processed

  A number of processed have runned.
- double avgElapsedTime

  Average of elapsed time.
- double performance

  A performance index.
- DSRoleHistoryList * roleHistoryList

  A list of history log for role allocations.
- DSInvokeHistoryList * invokeHistoryList

  A list of history log for reported Invoke messages.

Additional Inherited Members

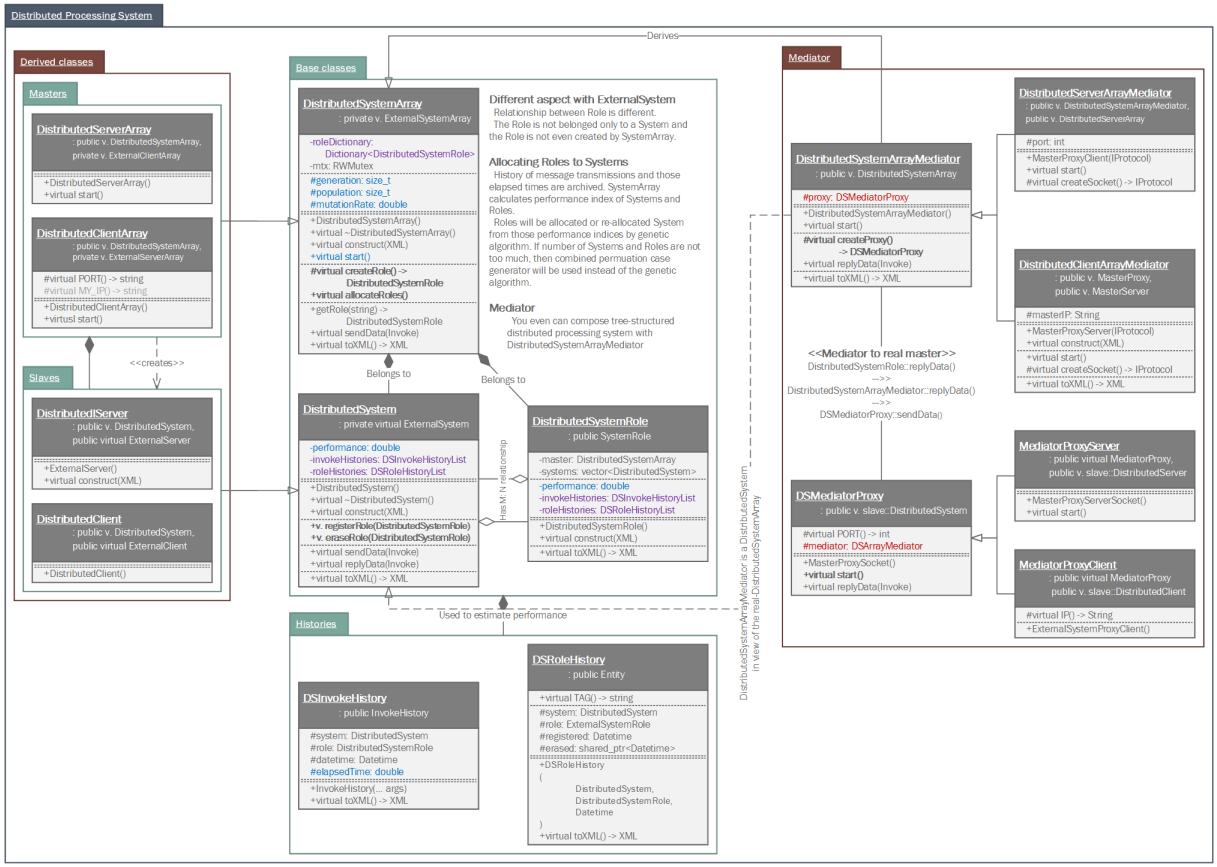
10.28.1 Detailed Description

A network driver for a distributed system.

DistributedSystem class is an ExternalSystem having role as DistributedSystemRole objects with performance index and history log allocation of DistributedSystemRole objects and have requsted Invoke message to related external system.

The greatest difference between DistributedSystem and ExternalSystem is, unlike the ExternalSystem class which having is ExternalSystemRole objects exclusively, DistributedSystem does not monopoly a DistributedSystemRole. The DistributedSystemRole can allocatedto multiple DistributedSystem.

DistributedSystem and DistributedSystemRole classes have performance index and history log archiving what they've requested to the related distributed system. Allocating DistributedSystemRole objects to DistributedSyste will be determined those performance index and history log for optimize distribution of elasped time about each Invoke message processing.

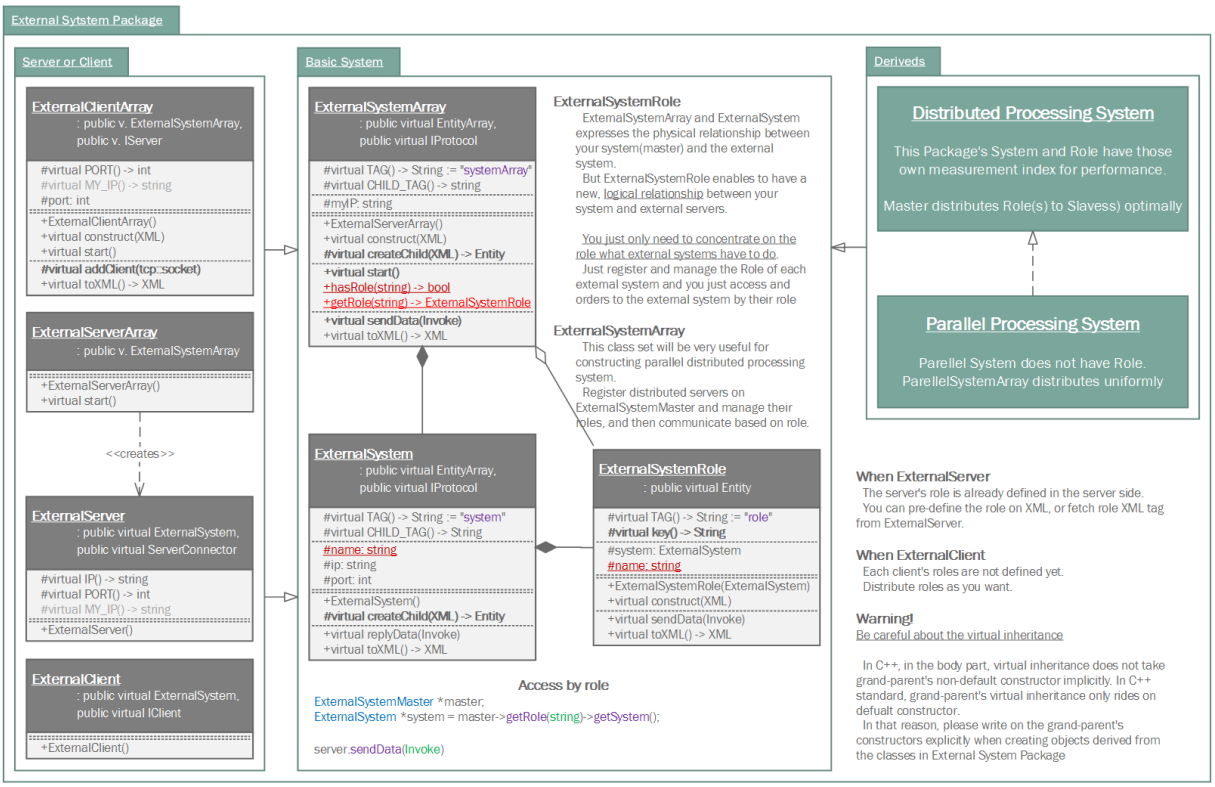


Todo

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, External↩System is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystemRole.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an External↩ SystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an ExternalSystem. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 43 of file DistributedSystem.hpp.

10.28.2 Constructor & Destructor Documentation

10.28.2.1 samchon::protocol::master::DistributedSystem::DistributedSystem ( )

Default Constructor.

10.28.3 Member Function Documentation

10.28.3.1 virtual void samchon::protocol::master::DistributedSystem::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Reimplemented from samchon::protocol::ExternalSystem.

Reimplemented in samchon::protocol::master::DistributedServer.

10.28.3.2 virtual void samchon::protocol::master::DistributedSystem::sendData (std::shared_ptr< Invoke >) [override], [virtual]

Sends message to a related system.

Parameters

invoke	Invoke message to send

Reimplemented from samchon::protocol::IClient.

10.28.3.3 virtual void samchon::protocol::master::DistributedSystem::replyData (std::shared_ptr< Invoke >) [override], [virtual]

Handling replied message from an external system.

If there's a related ExternalSystemRole, shifts responsibility to the ExternalSystemRole.

Parameters

invoke	Replied Invoke message

Reimplemented from samchon::protocol::ExternalSystem.

10.28.3.4 virtual auto samchon::protocol::master::DistributedSystem::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::ExternalSystem.

Reimplemented in samchon::protocol::master::DistributedServer.

10.28.4 Member Data Documentation

10.28.4.1 size_t samchon::protocol::master::DistributedSystem::inProgress [protected]

A number of processes running on the distributed system.

Adds one count when sendData() is called and minus one count if a DSInvokeHistory, which means reported Invoke history, was reported by replyData().

If an DistributedSystemRole is allocated to multiple DistributedSystem objects, which DistributedSystem will be used is determined by the *process size* and performance index of the ExternalSystemRole.

Definition at line 66 of file DistributedSystem.hpp.

10.28.4.2 size_t samchon::protocol::master::DistributedSystem::processed [protected]

A number of processed have runned.

Definition at line 71 of file DistributedSystem.hpp.

10.28.4.3 double samchon::protocol::master::DistributedSystem::avgElapsedTime [protected]

Average of elapsed time.

Average elapsed time of handling Invoke message (request) in frame work of tbe related distributed system. Not in frame work of distributed system's driver, DistributeDSystem.

The ultimate optimization target of distributed processing system is to minimizing standard deviation of the average elapsed time of the DistributedSystem.

Definition at line 83 of file DistributedSystem.hpp.

10.28.4.4 double samchon::protocol::master::DistributedSystem::performance [protected]

A performance index.

A performance index z is calculated by normalization calculating reverse number of whole distributed system's average elapsed times and its z value between the normal distribution.

- X = (

  If a distributed system has no history of handling Invoke message, then set the performance index to 0 as default. The performance index will be re-calcuated whenever re-allocation of roles is done (Distributed↩ System::allocateRoles()).

  Definition at line 102 of file DistributedSystem.hpp.

10.28.4.5 DSRoleHistoryList* samchon::protocol::master::DistributedSystem::roleHistoryList [protected]

A list of history log for role allocations.

Definition at line 107 of file DistributedSystem.hpp.

10.28.4.6 DSInvokeHistoryList* samchon::protocol::master::DistributedSystem::invokeHistoryList [protected]

A list of history log for reported Invoke messages.

Definition at line 112 of file DistributedSystem.hpp.

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

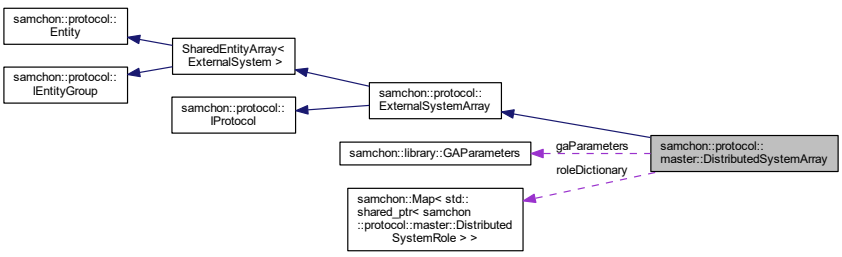
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DistributedSystem.hpp

10.29 samchon::protocol::master::DistributedSystemArray Class Reference

An array of distributed system drivers.

`#include <DistributedSystemArray.hpp>`

Collaboration diagram for samchon::protocol::master::DistributedSystemArray:



Public Member Functions

- DistributedSystemArray ()

  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override

  Construct data of the Entity from an XML object.
- virtual void start () override

  Start interaction.
- virtual void allocateRoles ()

  Allocate roles to each distributed processing system.
- auto getRole (const std::string &) const -> std::shared_ptr< DistributedSystemRole >

  Get a role.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the EntityGroup.

Protected Member Functions

- virtual auto createRole (std::shared_ptr< library::XML >) -> DistributedSystemRole *=0

  Factory method of a DistributedSystemRole.

Protected Attributes

- Dictionary< std::shared_ptr< DistributedSystemRole > > roleDictionary

  A dictionary of role objects.
- struct library::GAParameters gaParameters

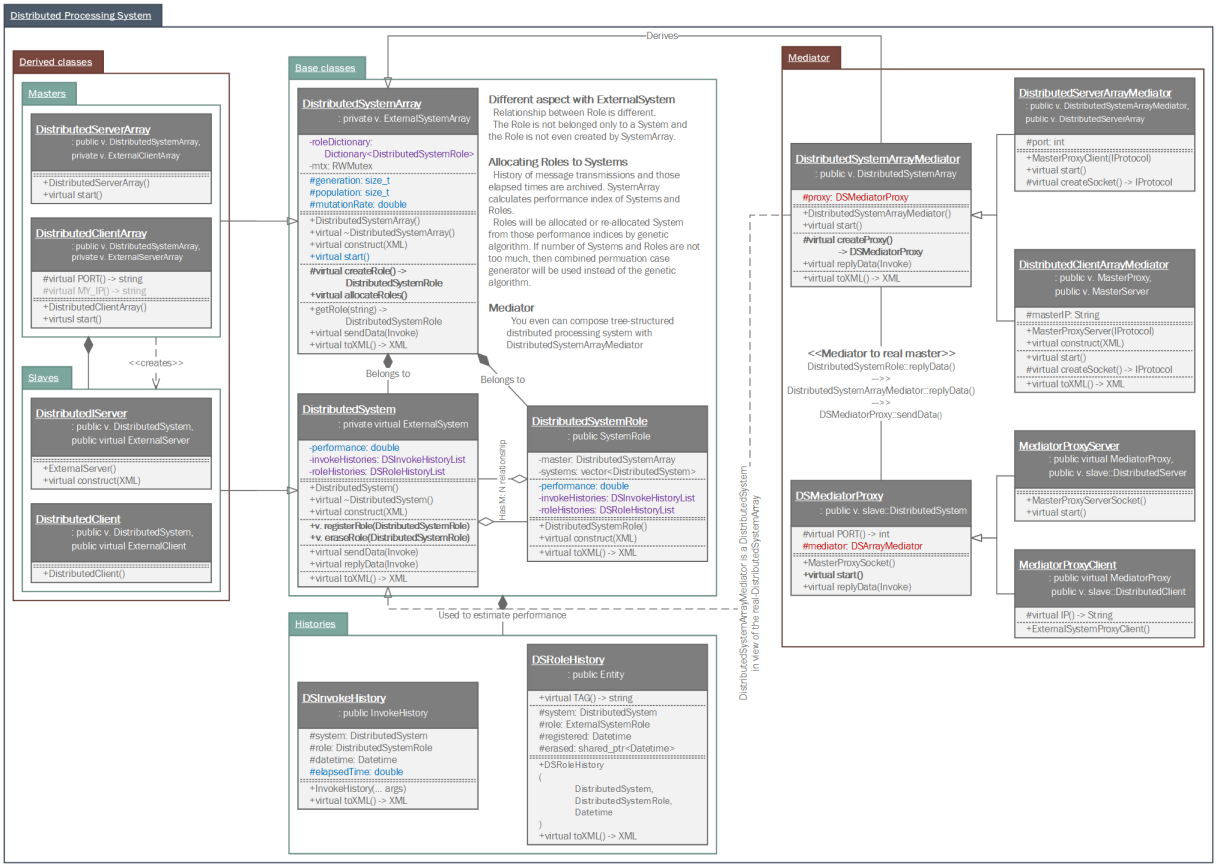
  A structure of parameters for genetic algorithm.

10.29.1 Detailed Description

An array of distributed system drivers.

DistributedSystemArray class is an ExternalSystemArray and an abstract class containing and managing distributed system drivers within framework of master. The DistributedSystemArray class allocates DistributedSystemRole objects to DistributedSystem objects for optimization.

Unlike the ExternalSystemRole class which is dependent on an ExternalSystem, DistributedSystemRole is not dependent on a DistributedSystem and even can be allocated to multiple DistributedSystem objects. Allocation of DistributedSystemRole(s) to DistributedSystem(s) is determined by estimation of performance index pursuing best distribution. If number of DistributedSystem and DistributedSystemRole objects is not a lot, optimize by combined permuation algorithm. Else the objects are a lot, the allocation will be optimized by genetic algorithm.



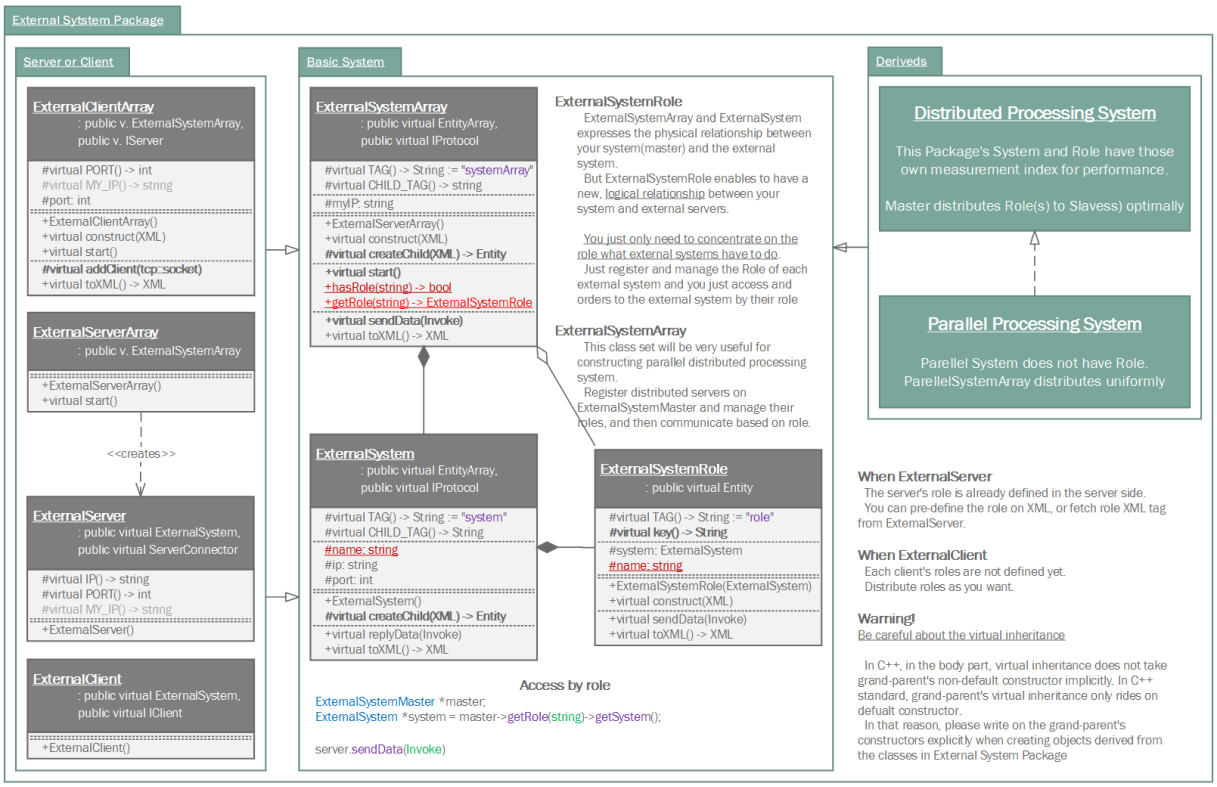
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

- ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 39 of file DistributedSystemArray.hpp.

10.29.2 Constructor & Destructor Documentation

10.29.2.1 samchon::protocol::master::DistributedSystemArray::DistributedSystemArray ( )

Default Constructor.

10.29.3 Member Function Documentation

10.29.3.1 virtual void samchon::protocol::master::DistributedSystemArray::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Reimplemented from samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

Reimplemented in samchon::protocol::master::DistributedSystemArrayMediator, samchon::protocol::master::DistributedClientArrayMediator, and samchon::protocol::master::DistributedClientArray.

10.29.3.2 virtual auto samchon::protocol::master::DistributedSystemArray::createRole (std::shared_ptr< library::XML >) -> DistributedSystemRole * [protected], [pure virtual]

Factory method of a DistributedSystemRole.

Parameters

An	XML object represents the role.

Returns

A new DistributedSystemRole.

10.29.3.3 virtual void samchon::protocol::master::DistributedSystemArray::start ( ) [override], [virtual]

Start interaction.

An abstract method starting interaction with external systems.

If external systems are servers, starts connection to them, else clients, opens a server and accepts the external systems. You can addict your own procudures of starting drivers, but if you directly override method of abstract ExternalSystemArray, be careful about virtual inheritance.

- DistributedClientArray::start() calls DistributedSystemArray::start() and ExternalClientArray:start()

Implements samchon::protocol::ExternalSystemArray.

Reimplemented in samchon::protocol::master::DistributedSystemArrayMediator, samchon::protocol::master::DistributedClientArrayMediator, samchon::protocol::master::DistributedClientArray, samchon::protocol::master::DistributedServerArrayMediator, and samchon::protocol::master::DistributedServerArray.

10.29.3.4 virtual void samchon::protocol::master::DistributedSystemArray::allocateRoles ( ) [virtual]

Allocate roles to each distributed processing system.

Allocates DistributedSystemRole objects to Distributedsystem objects. The allocation will be done when:

- New distributed system (client) has connected.
- Ordinary distributed system is disconnected.
- Deviation of elapsed time of whole DistributedSystem overs limited value.

10.29.3.5 auto samchon::protocol::master::DistributedSystemArray::getRole (const std::string &) const -> std::shared_ptr< DistributedSystemRole >

Get a role.

Parameters

name	An identifier of an DistributedSystemRole

Returns

A shared pointer of specialized role

10.29.3.6 virtual auto samchon::protocol::master::DistributedSystemArray::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

Reimplemented in samchon::protocol::master::DistributedSystemArrayMediator, samchon::protocol::master::DistributedClientArrayMediator, and samchon::protocol::master::DistributedClientArray.

10.29.4 Member Data Documentation

10.29.4.1 Dictionary<std::shared_ptr<DistributedSystemRole> > samchon::protocol::master::DistributedSystemArray::roleDictionary [protected]

A dictionary of role objects.

Definition at line 49 of file DistributedSystemArray.hpp.

10.29.4.2 struct library::GAParameters samchon::protocol::master::DistributedSystemArray::gaParameters [protected]

A structure of parameters for genetic algorithm.

Definition at line 54 of file DistributedSystemArray.hpp.

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

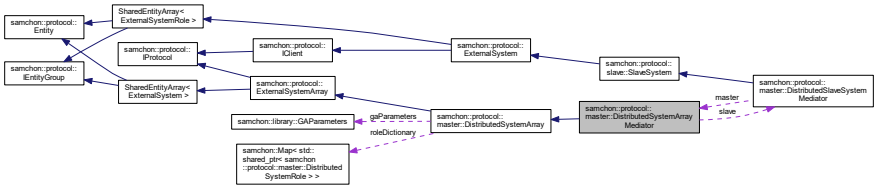
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DistributedSystemArray.hpp

10.30 samchon::protocol::master::DistributedSystemArrayMediator Class Reference

A mediator of distributed system between master and slaves.

```
#include <DistributedSystemArrayMediator.hpp>
```

Collaboration diagram for samchon::protocol::master::DistributedSystemArrayMediator:



Public Member Functions

- DistributedSystemArrayMediator ()

  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override

  Construct data of the Entity from an XML object.
- virtual void start () override

  Start interaction.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the EntityGroup.

Protected Member Functions

- virtual auto createSlave () const -> DistributedSlaveSystemMediator *=0

  Factory method of slave system for mediation.

Protected Attributes

- DistributedSlaveSystemMediator * slave

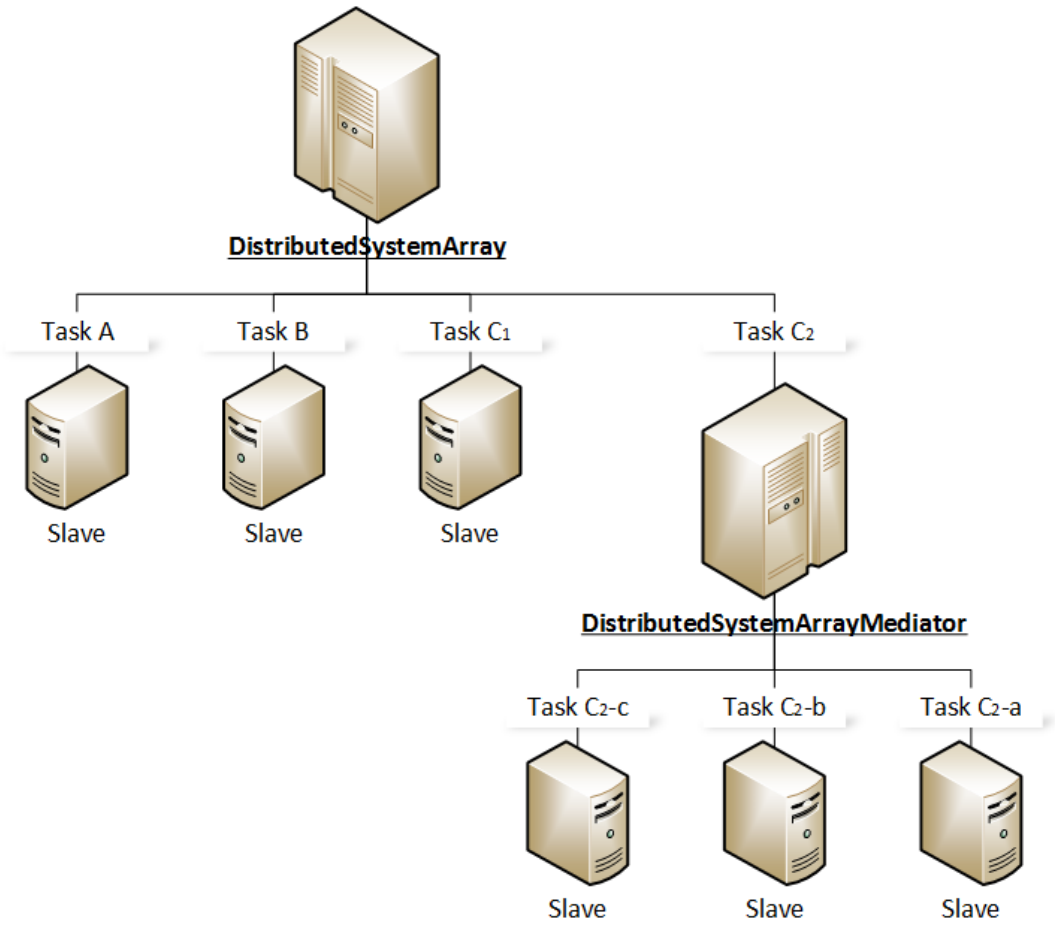
  A slave system for mediation.

10.30.1 Detailed Description

A mediator of distributed system between master and slaves.

DistributedSystemArrayMediator class is used to realizing a mediator system of tree-structured distributed processing system. DistributedSystemArrayMediator intermediates between origin-master(DistributedSystemArray) and its slaves(slave::DistributedSystem).

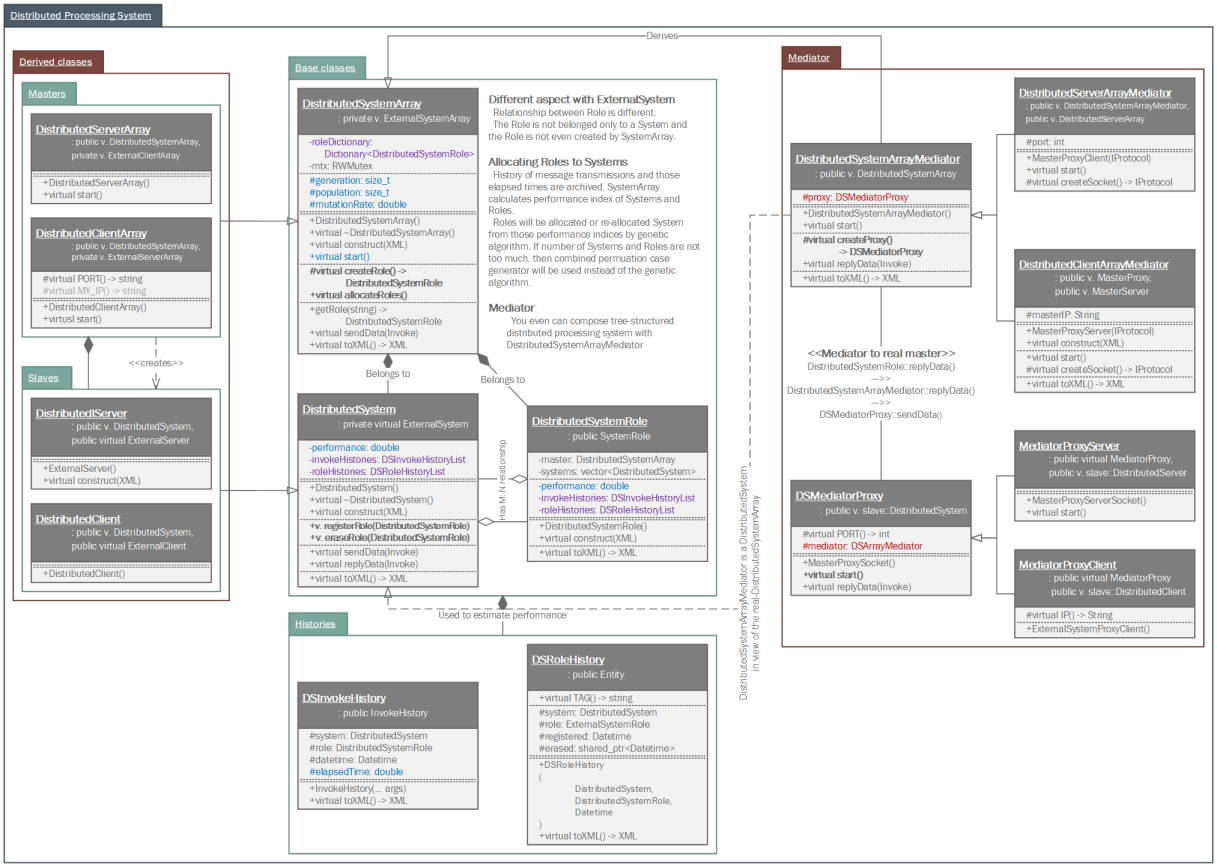
DistributedSystemArrayMediator is a master(DistributedSystemArray) within the framework of slaves systems (slave::DistributedSystem objects) and also can be a slave system (slave::DistributedSystem) within framework of its master(DistributedSystemArray) by having DistributedSlaveSystemMediator.



[Inherited]

DistributedSystemArray class is an ExternalSystemArray and an abstract class containing and managing distributed system drivers within framework of master. The DistributedSystemArray class allocates DistributedSystemRole objects to DistributedSystem objects for optimization.

Unlike the ExternalSystemRole class which is dependent on an ExternalSystem, DistributedSystemRole is not dependent on a DistributedSystem and even can be allocated to multiple DistributedSystem objects. Allocation of DistributedSystemRole(s) to DistributedSystem(s) is determined by estimation of performance index pursuing best distribution. If number of DistributedSystem and DistributedSystemRole objects is not a lot, optimize by combined permuation algorithm. Else the objects are a lot, the allocation will be optimized by genetic algorithm.



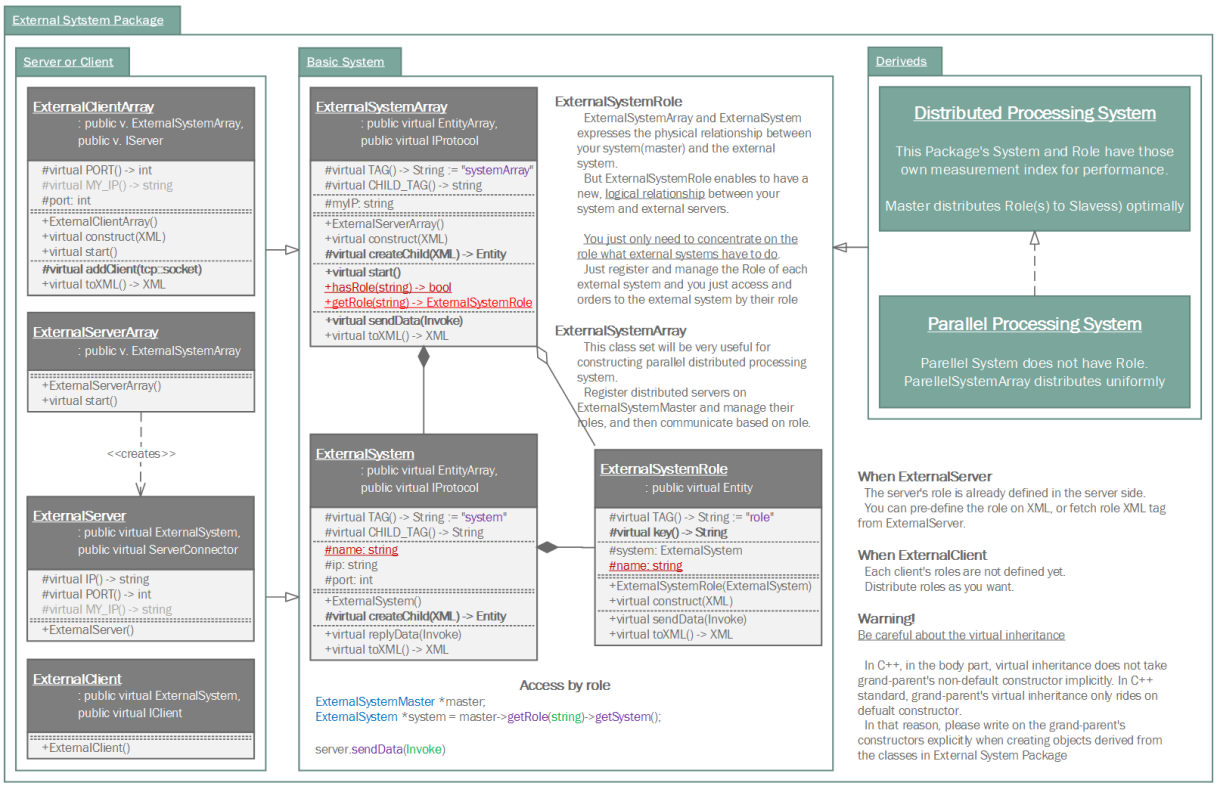
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

- ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 33 of file DistributedSystemArrayMediator.hpp.

10.30.2 Constructor & Destructor Documentation

10.30.2.1 samchon::protocol::master::DistributedSystemArrayMediator::DistributedSystemArrayMediator ( )

Default Constructor.

10.30.3 Member Function Documentation

10.30.3.1 virtual void samchon::protocol::master::DistributedSystemArrayMediator::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Reimplemented from samchon::protocol::master::DistributedSystemArray.

Reimplemented in samchon::protocol::master::DistributedClientArrayMediator.

10.30.3.2 virtual auto samchon::protocol::master::DistributedSystemArrayMediator::createSlave ( ) const -> DistributedSlaveSystemMediator * [protected], [pure virtual]

Factory method of slave system for mediation.

Exceptions

invalid_parameter	If DistributedServerArrayMediator, slave must be a type of DistributedSlave↩ ServerMediator

10.30.3.3 virtual void samchon::protocol::master::DistributedSystemArrayMediator::start ( ) [override], [virtual]

Start interaction.

An abstract method starting interaction with external systems.

If external systems are servers, starts connection to them, else clients, opens a server and accepts the external systems. You can addict your own procudures of starting drivers, but if you directly override method of abstract ExternalSystemArray, be careful about virtual inheritance.

- DistributedClientArray::start() calls DistributedSystemArray::start() and ExternalClientArray:start()

Reimplemented from samchon::protocol::master::DistributedSystemArray.

Reimplemented in samchon::protocol::master::DistributedClientArrayMediator, and samchon::protocol↩ ::master::DistributedServerArrayMediator.

10.30.3.4 virtual auto samchon::protocol::master::DistributedSystemArrayMediator::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value

<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::master::DistributedSystemArray.

Reimplemented in samchon::protocol::master::DistributedClientArrayMediator.

10.30.4 Member Data Documentation

10.30.4.1 DistributedSlaveSystemMediator* samchon::protocol::master::DistributedSystemArrayMediator::slave [protected]

A slave system for mediation.

Definition at line 43 of file DistributedSystemArrayMediator.hpp.

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

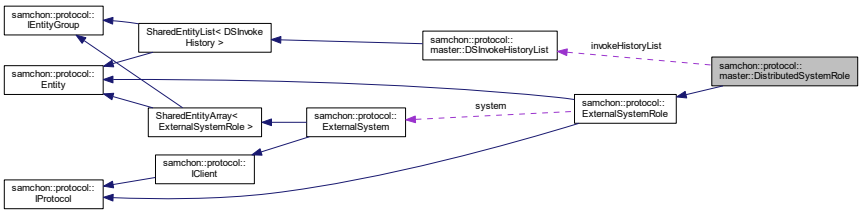
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DistributedSystemArray↩ Mediator.hpp

10.31 samchon::protocol::master::DistributedSystemRole Class Reference

A role of distributed processing system.

`#include <DistributedSystemRole.hpp>`

Collaboration diagram for samchon::protocol::master::DistributedSystemRole:



Public Member Functions

- DistributedSystemRole ()

  Default Constructor.
- auto getSystem () const -> ExternalSystem *=delete

  Get an external system -> deprecated.
- auto getPerformance () const -> double

  Get performance.
- auto getAllocationHistoryList () const -> DSRoleHistoryList *

  Get allocation histories.
- auto getInvokeHistoryList () const -> DSInvokeHistoryList *

  Get invoke histories.
- virtual void sendData (std::shared_ptr< Invoke >) override

Send a message.

– virtual auto toXML () const -> std::shared_ptr< library::XML > override

Get an XML object represents the Entity.

Protected Attributes

– std::set< DistributedSystem * > allocatedSystems

Allocated systems of about the role, at now.

– double performance

A required performance index.

– DSRoleHistoryList * allocationHistoryList

A history log for archiving role allocation to a system.

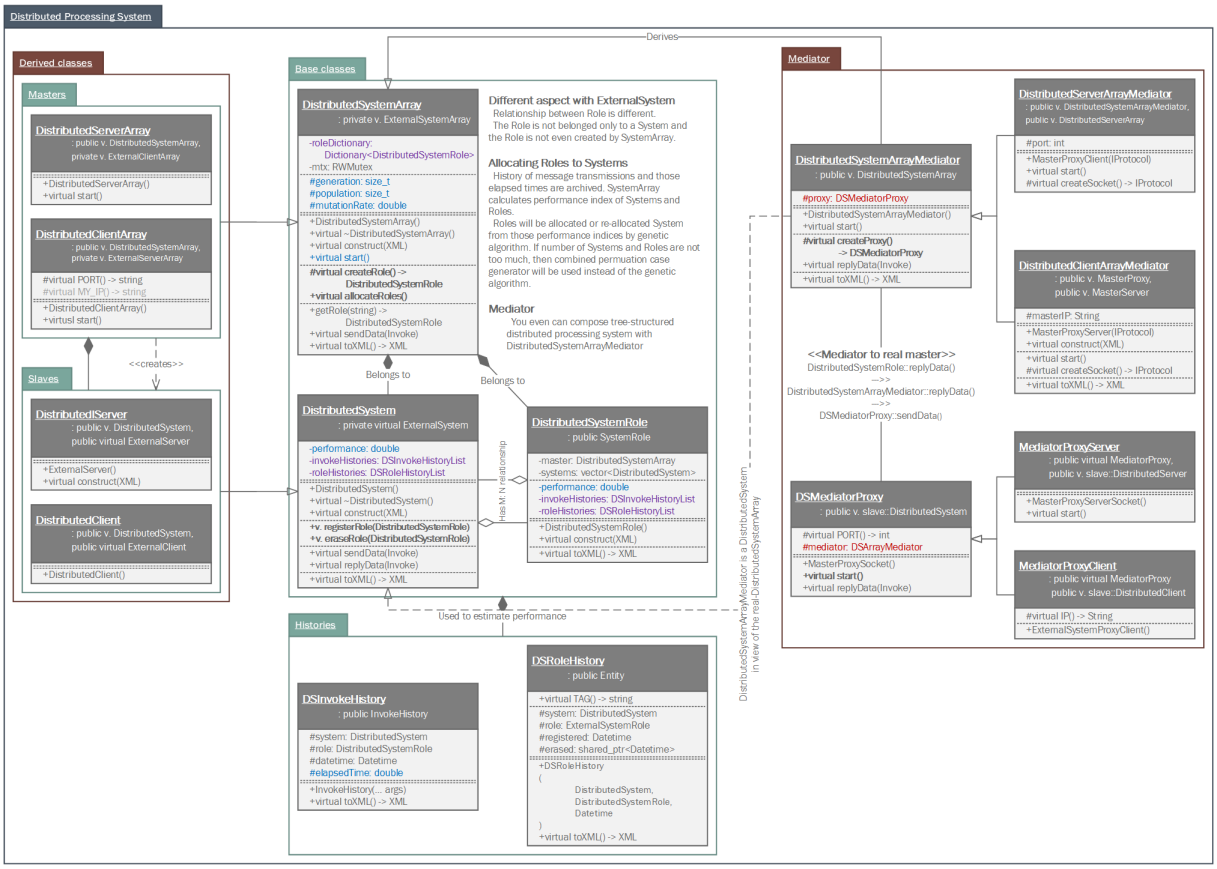
– DSInvokeHistoryList * invokeHistoryList

A history log for archiving elapsed time for each process.

10.31.1 Detailed Description

A role of distributed processing system.

DistributedSystemRole is an ExternalSystemRole allocates for distributed processing system(s). Unlike the ExternalSystemRole, DistributedSystemRole can belonged to multiple DistributedSystem(s).



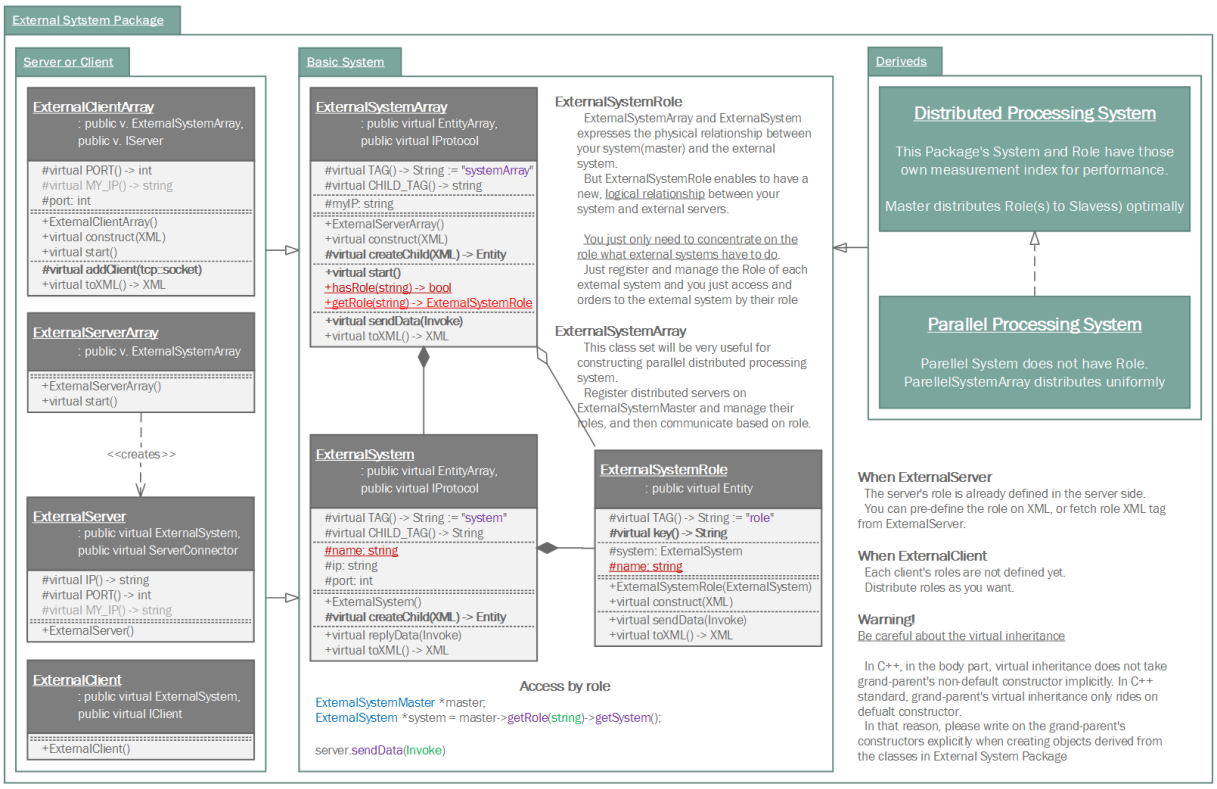
[Inherited]

ExternalSystemRole is a 'control' class groupping methods, handling Invoke messages interacting with an external system that the ExternalSystemRole is belonged to, by a subject or unit of a module.

ExternalSystemRole can be a "logical proxy" for an ExternalSystem which is containing the ExternalSystem↩Role. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an

ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystem↩ Role acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 34 of file DistributedSystemRole.hpp.

10.31.2 Constructor & Destructor Documentation

10.31.2.1 samchon::protocol::master::DistributedSystemRole::DistributedSystemRole ( )

Default Constructor.

10.31.3 Member Function Documentation

10.31.3.1 auto samchon::protocol::master::DistributedSystemRole::getSystem ( ) const -> ExternalSystem *=delete [delete]

Get an external system -> deprecated.

Unlike ExternalSysemRole, DistributedSystemRole can be belonged to multiple DistributedSystem(s). By that reason, the getSystem(), a getter of a single ExternalSystem is deprecated.

10.31.3.2 auto samchon::protocol::master::DistributedSystemRole::getPerformance ( ) const -> double

Get performance.

10.31.3.3 auto samchon::protocol::master::DistributedSystemRole::getAllocationHistoryList ( ) const -> DSRoleHistoryList *

Get allocation histories.

10.31.3.4 auto samchon::protocol::master::DistributedSystemRole::getInvokeHistoryList ( ) const -> DSInvokeHistoryList *

Get invoke histories.

10.31.3.5 virtual void samchon::protocol::master::DistributedSystemRole::sendData (std::shared_ptr< Invoke >) [override], [virtual]

Send a message.

Sends Invoke message to a network system or shifts the responsibility to related chain.

Parameters

invoke	An Invoke message to send to a network system

Reimplemented from samchon::protocol::ExternalSystemRole.

10.31.3.6 virtual auto samchon::protocol::master::DistributedSystemRole::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::ExternalSystemRole.

10.31.4 Member Data Documentation

10.31.4.1 std::set<DistributedSystem*> samchon::protocol::master::DistributedSystemRole::allocatedSystems [protected]

Allocated systems of about the role, at now.

Definition at line 46 of file DistributedSystemRole.hpp.

10.31.4.2 double samchon::protocol::master::DistributedSystemRole::performance [protected]

A required performance index.

Definition at line 51 of file DistributedSystemRole.hpp.

10.31.4.3 DSRoleHistoryList* samchon::protocol::master::DistributedSystemRole::allocationHistoryList [protected]

A history log for archiving role allocation to a system.

Definition at line 56 of file DistributedSystemRole.hpp.

10.31.4.4 DSInvokeHistoryList* samchon::protocol::master::DistributedSystemRole::invokeHistoryList [protected]

A history log for archiving elapsed time for each process.

Definition at line 61 of file DistributedSystemRole.hpp.

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

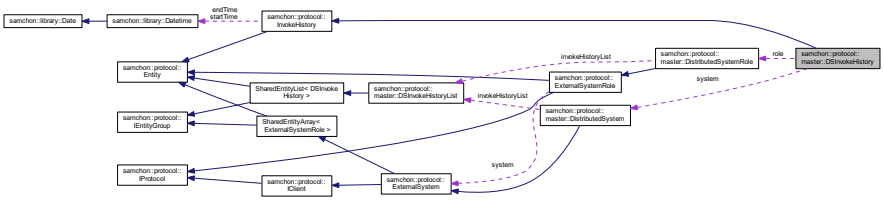
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DistributedSystemRole.hpp

10.32 samchon::protocol::master::DSInvokeHistory Class Reference

A reported history of an Invoke message.

`#include <DSInvokeHistory-DESKTOP-3MP6UH0.hpp>`

Collaboration diagram for samchon::protocol::master::DSInvokeHistory:



Public Member Functions

- DSInvokeHistory (master::DistributedSystem *, master::DistributedSystemRole *)

  Construct from a system and role.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the Entity.
- DSInvokeHistory (DistributedSystem *, DistributedSystemRole *)

  Construct from a system and role.
- auto getSystem () const -> DistributedSystem *

  Get source system.
- auto getRole () const -> DistributedSystemRole *

  Gets source role.

Protected Attributes

- master::DistributedSystem * system

  Source system.
- master::DistributedSystemRole * role

  Source role.
- DistributedSystem * system

  Source system.
- DistributedSystemRole * role

  Source role.

10.32.1 Detailed Description

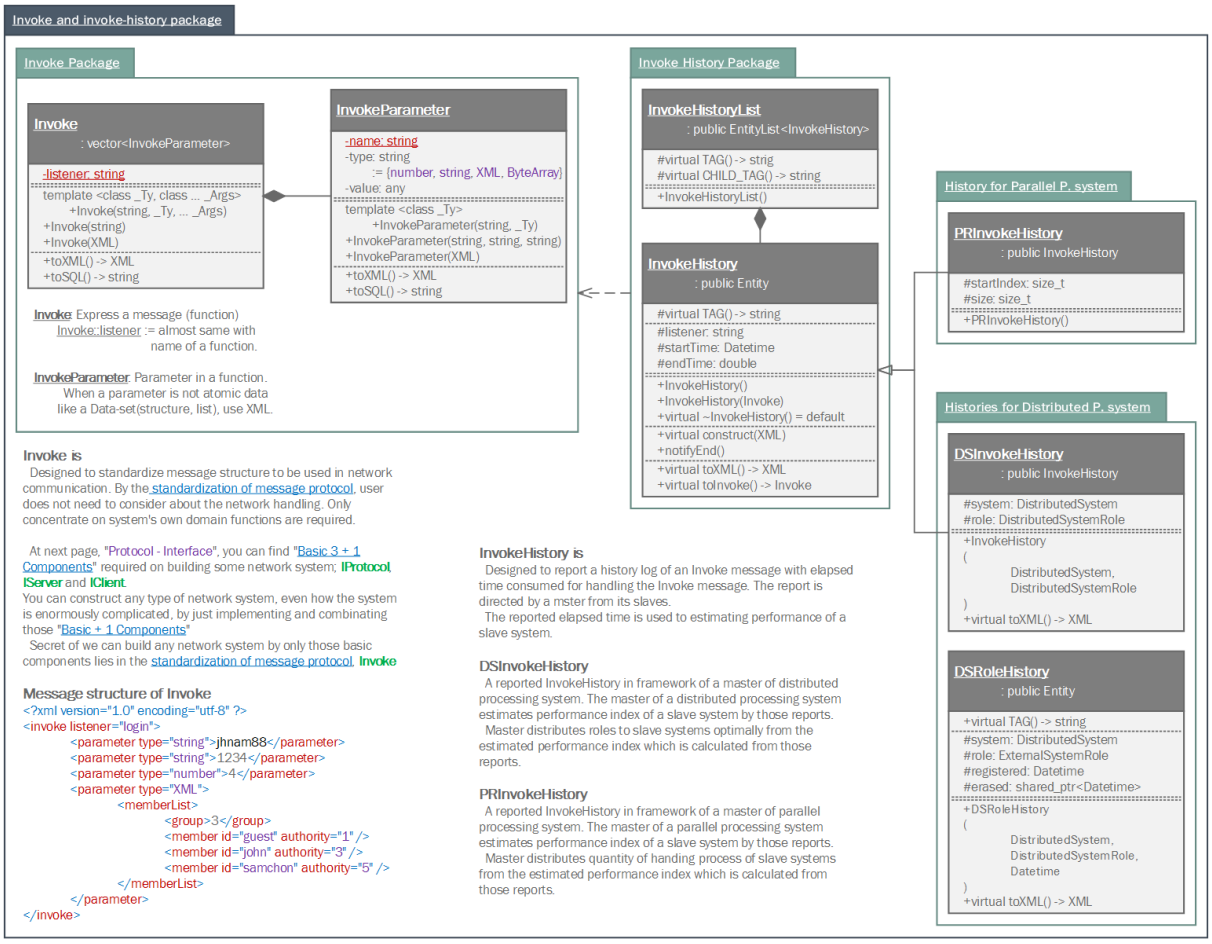
A reported history of an Invoke message.

Inherited

A history of an Invoke message.

InvokeHistory is a class for reporting history log of an Invoke message with elapsed time from a slave to its master.

With the elapsed time, consumed time for a process of handling the Invoke message, InvokeHistory is reported to the master. The master utilizies the elapsed time to estimating performances of each slave system. With the estimated performan index, master retrives the optimal solution of distributing processes.



See also

samchon::protocol
samchon::protocol::master
samchon::protocol::slave

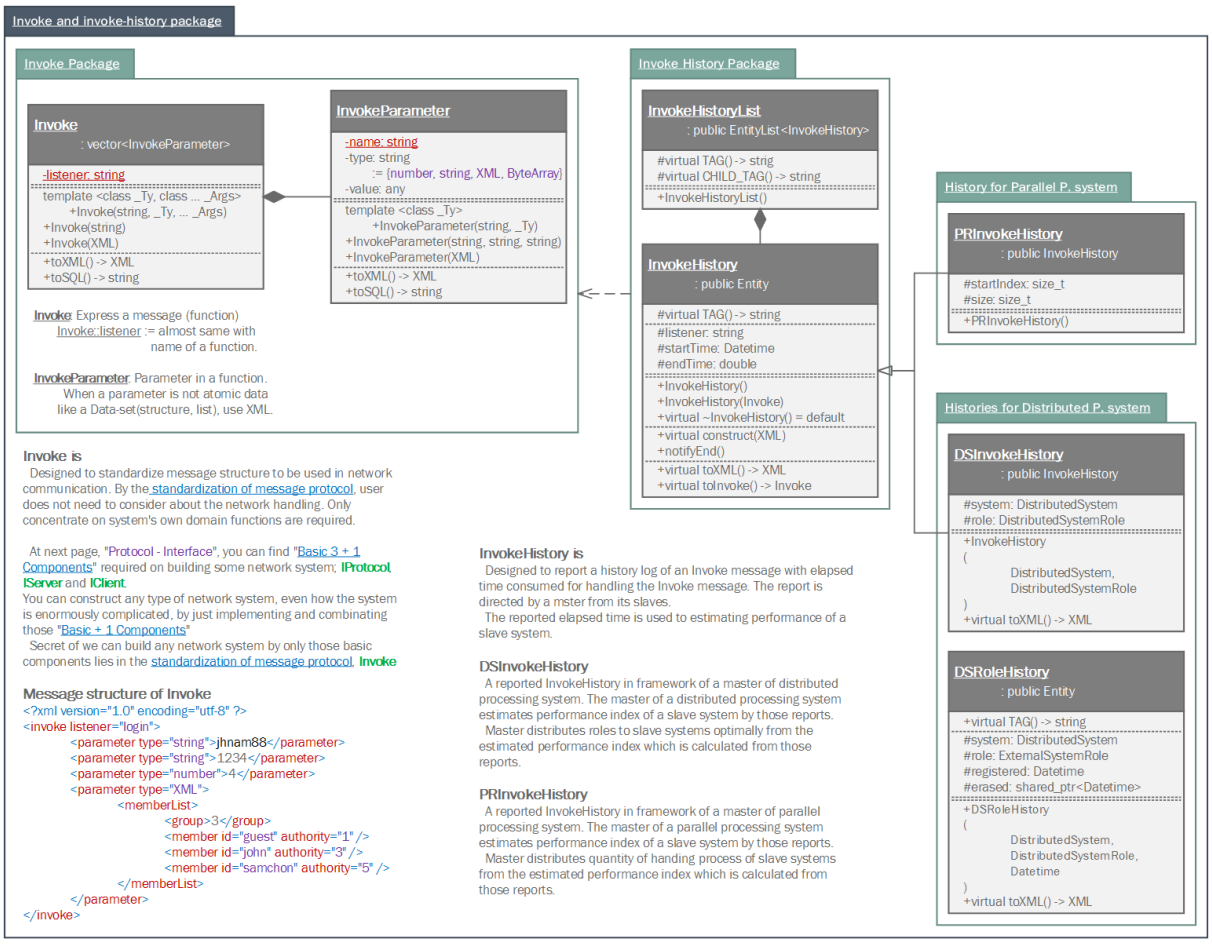
Author

Jeongho Nam

[Inherited]

InvokeHistory is a class for reporting history log of an Invoke message with elapsed time from a slave to its master.

With the elapsed time, consumed time for a process of handling the Invoke message, InvokeHistory is reported to the master. The master utilizies the elapsed time to estimating performances of each slave system. With the estimated performan index, master retrives the optimal solution of distributing processes.



See also

samchon::protocol
samchon::protocol::master
samchon::protocol::slave

Author

Jeongho Nam

Definition at line 22 of file DSInvokeHistory-DESKTOP-3MP6UH0.hpp.

10.32.2 Constructor & Destructor Documentation

10.32.2.1 DSInvokeHistory::DSInvokeHistory (master::DistributedSystem * *system*, master::DistributedSystemRole * *role*)

Construct from a system and role.

Parameters

system	a source system
role	a source role

Definition at line 8 of file DSInvokeHistory.cpp.

References role, and system.

10.32.2.2 samchon::protocol::master::DSInvokeHistory::DSInvokeHistory (DistributedSystem * , DistributedSystemRole *)

Construct from a system and role.

Parameters

system	a source system.
role	a source role.

10.32.3 Member Function Documentation

10.32.3.1 virtual auto samchon::protocol::master::DSInvokeHistory::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::InvokeHistory.

10.32.3.2 auto DSInvokeHistory::getSystem ( ) const -> DistributedSystem*

Get source system.

Definition at line 15 of file DSInvokeHistory.cpp.

References system.

10.32.3.3 auto DSInvokeHistory::getRole ( ) const -> DistributedSystemRole*

Gets source role.

Definition at line 19 of file DSInvokeHistory.cpp.

References role.

10.32.4 Member Data Documentation

10.32.4.1 master::DistributedSystem* samchon::protocol::master::DSInvokeHistory::system [protected]

Source system.

Definition at line 31 of file DSInvokeHistory-DESKTOP-3MP6UH0.hpp.

Referenced by DSInvokeHistory(), and getSystem().

10.32.4.2 master::DistributedSystemRole* samchon::protocol::master::DSInvokeHistory::role [protected]

Source role.

Definition at line 36 of file DSInvokeHistory-DESKTOP-3MP6UH0.hpp.

Referenced by DSInvokeHistory(), and getRole().

10.32.4.3 DistributedSystem* samchon::protocol::master::DSInvokeHistory::system [protected]

Source system.

Definition at line 32 of file DSInvokeHistory.hpp.

10.32.4.4 DistributedSystemRole* samchon::protocol::master::DSInvokeHistory::role [protected]

Source role.

Definition at line 37 of file DSInvokeHistory.hpp.

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

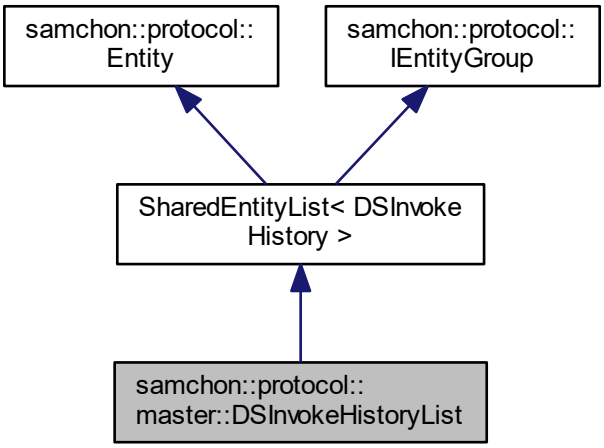
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DSInvokeHistory-DESKTOP-3MP6UH0.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DSInvokeHistory.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DSInvokeHistory.cpp

10.33 samchon::protocol::master::DSInvokeHistoryList Class Reference

A list of DSInvokeHistory.

```
#include <DSInvokeHistoryList.hpp>
```

Collaboration diagram for samchon::protocol::master::DSInvokeHistoryList:



Public Member Functions

- DSInvokeHistoryList ()
  Default Constructor.

Protected Member Functions

- virtual auto createChild (std::shared_ptr< library::XML >) -> DSInvokeHistory *override
  Factory method of a child Entity.

10.33.1 Detailed Description

A list of DSInvokeHistory.

Author

Jeongho Nam

Definition at line 18 of file DSInvokeHistoryList.hpp.

10.33.2 Constructor & Destructor Documentation

10.33.2.1 samchon::protocol::master::DSInvokeHistoryList::DSInvokeHistoryList ( )

Default Constructor.

10.33.3 Member Function Documentation

10.33.3.1 virtual auto samchon::protocol::master::DSInvokeHistoryList::createChild (std::shared_ptr< library::XML >) -> DSInvokeHistory *override [override], [protected], [virtual]

Factory method of a child Entity.

EntityGroup::createChild() is a factory method creating a new child Entity which is belonged to the EntityGroup. This method is called by EntityGroup::construct(). The children construction methods Entity::construct() will be called by abstract method of the EntityGroup::construct().

Returns

A new child Entity belongs to EntityGroup.

Implements samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/DSInvokeHistoryList.hpp

10.34 samchon::protocol::Entity Class Reference

An entity, a standard data class.

```
#include <Entity.hpp>
```

Public Member Functions

- virtual auto TAG () const -> std::string=0

  A tag name when represented by XML.
- Entity ()

  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >)=0

  Construct data of the Entity from an XML object.
- virtual auto key () const -> std::string

  Get a key that can identify the Entity uniquely.
- virtual auto toXML () const -> std::shared_ptr< library::XML >

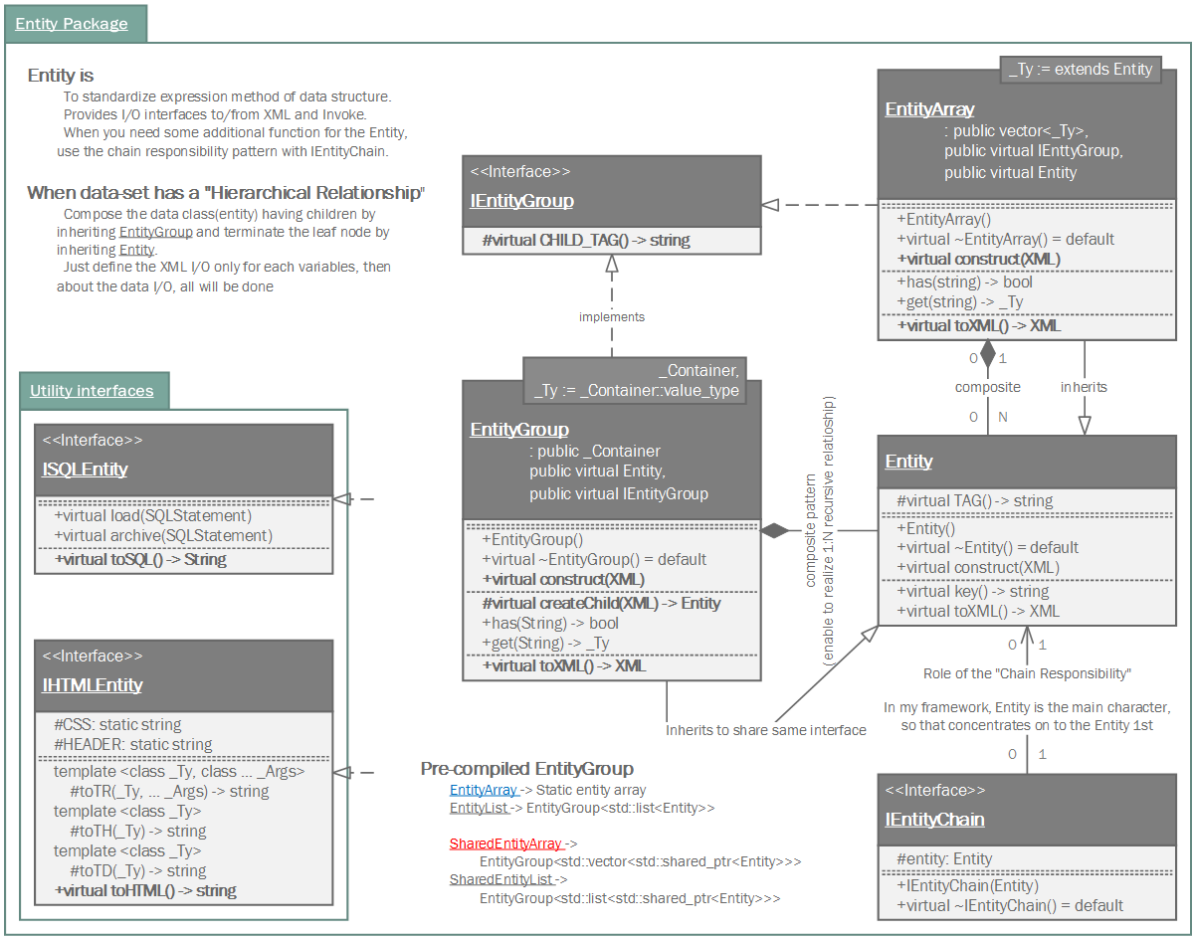
  Get an XML object represents the Entity.

10.34.1 Detailed Description

An entity, a standard data class.

Entity is a class for standardization of expression method using on network I/O by XML. If Invoke is a standard message protocol of Samchon Framework which must be kept, Entity is a recommended semi-protocol of message for expressing a data class. Following the semi-protocol Entity is not imposed but encouraged.

As we could get advantages from standardization of message for network I/O with Invoke, we can get additional advantage from standardizing expression method of data class with Entity. We do not need to know a part of network communication. Thus, with the Entity, we can only concentrate on entity's own logics and relationships between another entities. Entity does not need to how network communications are being done.



Example source

```
#include <iostream>

#include <samchon/protocol/Entity.hpp>
#include <samchon/protocol/IHTMLEntity.hpp>
#include <samchon/protocol/SharedEntityArray.hpp>

#include <samchon/library/XML.hpp>

#ifdef _WIN64
#   ifdef _DEBUG
#       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "x64/Release/SamchonFramework.lib")
#   endif
#else
#   ifdef _DEBUG
#       pragma comment(lib, "Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "Release/SamchonFramework.lib")
#   endif
#endif

using namespace std;
using namespace samchon::library;
using namespace samchon::protocol;

class Member
    : public Entity, public virtual IHTMLEntity
{
protected:
    typedef Entity super;

    string id;
    string name;
    int age;
    int grade;

public:
    /* ----------------------------------------------------------------------
        CONSTRUCTORS
```

```
    ------------------------------------------------------------------- */
    Member()
        : super(), IHTMLEntity()
    {
    };
    Member(const string &id, const string &name, int age, int grade)
        : super(), IHTMLEntity()
    {
        this->id = id;
        this->name = name;
        this->age = age;
        this->grade = grade;
    };
    virtual ~Member() = default;

    virtual void construct(shared_ptr<XML> xml) override
    {
        this->id = xml->getProperty("id");
        this->name = xml->getProperty("name");
        this->age = xml->getProperty<int>("age");
        this->grade = xml->getProperty<int>("grade");
    };

    /* -------------------------------------------------------------------
        GETTERSs
    ------------------------------------------------------------------- */
    virtual auto key() const -> std::string override
    {
        return this->id;
    };

    /* -------------------------------------------------------------------
        XML EXPORTERS
    ------------------------------------------------------------------- */
    virtual auto TAG() const -> string override
    {
        return "member";
    };
    virtual auto toXML() const -> shared_ptr<XML>
    {
        shared_ptr<XML> &xml = super::toXML();
        xml->setProperty("id", id);
        xml->setProperty("name", name);
        xml->setProperty("age", age);
        xml->setProperty("grade", grade);

        return move(xml);
    };
    virtual auto toHTML() const -> string
    {
        return toTR(id, name, age, grade);
    };
};

class MemberArray
    : public SharedEntityArray<Member>,
    public virtual IHTMLEntity
{
protected:
    typedef SharedEntityArray<Member> super;

    string application;
    int department;
    Member *chief;

public:
    /* -------------------------------------------------------------------
        CONSTRUCTORS
    ------------------------------------------------------------------- */
    MemberArray()
        : super(), IHTMLEntity()
    {
        this->chief = nullptr;
    };
    virtual ~MemberArray() = default;

    // You don't need to consider children(Member) objects
    // Just concentrate on constructing MemberArray's own member variables
    virtual void construct(shared_ptr<XML> xml) override
    {
        super::construct(xml);

        this->application = xml->getProperty("application");
        this->department = xml->getProperty<int>("department");

        if(xml->hasProperty("chief") == true && this->has( xml->getProperty("chief") ) == true)
            this->chief = this->get( xml->getProperty("cheif") ).get();
```

```
    };

protected:
    //FACTORY METHOD FOR MEMBER
    virtual auto createChild(shared_ptr<XML> = nullptr) -> Member* override
    {
        return new Member();
    };

    /* ---------------------------------------------------------------------
        XML EXPORTERS
    --------------------------------------------------------------------- */
public:
    virtual auto TAG() const -> string override
    {
        return "memberArray";
    };
    virtual auto CHILD_TAG() const -> string override
    {
        return "member";
    };

    // You don't need to consider children(Member) objects
    // Just concentrate on expressing MemberArray's own member variables
    virtual auto toXML() const -> shared_ptr<XML>
    {
        shared_ptr<XML> &xml = super::toXML();
        xml->setProperty("application", application);
        xml->setProperty("department", department);

        if(chief != nullptr)
            xml->setProperty("cheif", chief->key());

        return move(xml);
    };
    virtual auto toHTML() const -> string
    {
        string html = "<table>\n";
        html += toTH("ID", "Name", "Age", "Grade") + "\n";

        for (size_t i = 0; i < 2; i++)
            html += at(i)->toHTML() + "\n";

        html += "</table>";
        return move(html);
    };
};

void main()
{
    string str = string("") +
        "<memberArray application='framework' department='7' cheif='samchon'>\n" +
        "   <member id='samchon' name='Jeongho Nam' age='27' grade='5' />" +
        "   <member id='submaster' name='No Name' age='100' grade='4' />" +
        "   <member id='john' name='John Doe' age='33' grade='2' />" +
        "   <member id='bad_man' name='Bad Man' age='44' grade='-1' />" +
        "   <member id='guest' name='Guest' age='0' grade='0' />" +
        "</memberArray>";
    shared_ptr<XML> xml(new XML(str));

    MemberArray memberArray;
    memberArray.construct(xml);

    memberArray.emplace_back(new Member("freshman", "A fresh man", 20, 2));
    memberArray.emplace_back(new Member("senior", "A senior", 70, 2));

    cout << memberArray.toXML()->toString() << endl << endl;
    cout << memberArray.toHTML() << endl;
    system("pause");
}
```

Result of the example

```
D:\OneDrive\Project\Samchon\framework\cpp\Debug\entity.exe
<memberArray application="framework" department="7">
        <member age="27" grade="5" id="samchon" name="Jeongho" />
        <member age="100" grade="4" id="submaster" name="No" />
        <member age="33" grade="2" id="john" name="John" />
        <member age="44" grade="-1" id="bad_man" name="Bad" />
        <member age="0" grade="0" id="guest" name="Guest" />
        <member age="20" grade="2" id="freshman" name="A fresh man" />
        <member age="70" grade="2" id="senior" name="A senior" />
</memberArray>

<table>
        <th>
                <td>ID</td>
                <td>Name</td>
                <td>Age</td>
                <td>Grade</td>
        </th>
        <tr>
                <td>samchon</td>
                <td>Jeongho</td>
                <td>27</td>
                <td>5</td>
        </tr>
        <tr>
                <td>submaster</td>
                <td>No</td>
                <td>100</td>
                <td>4</td>
        </tr>
</table>
Press any key to continue . . .
```

Note

I say repeatedly. Expression method of Entity is recommended, but not imposed. It's a semi protocol for network I/O but not a essential protocol must be kept. The expression method of Entity, using on network I/O, is expressed by XML string.

If your own network system has a critical performance issue on communication data class, it would be better to using binary communication (with ByteArray or boost::serialization). Don't worry about the problem! Invoke also provides methods for binary data (ByteArray).

See also

protocol

Author

Jeongho Nam

Definition at line 52 of file Entity.hpp.

10.34.2 Constructor & Destructor Documentation

10.34.2.1 Entity::Entity ( )

Default Constructor.

Definition at line 20 of file Entity.cpp.

10.34.3 Member Function Documentation

10.34.3.1 virtual auto samchon::protocol::Entity::TAG ( ) const -> std::string [pure virtual]

A tag name when represented by XML.

Returns

A tag name

Implemented in samchon::protocol::InvokeParameter, samchon::protocol::InvokeHistory, samchon::protocol::Invoke, samchon::namtree::NTCriteria, samchon::protocol::ExternalSystemArray, samchon::protocol::ExternalSystem, samchon::protocol::ExternalSystemRole, samchon::namtree::NTParameter, samchon::namtree::NTParameterDetermined, samchon::protocol::SystemRole, samchon::protocol::master::ParallelSlaveSystemMediator, samchon::protocol::master::DistributedSlaveSystemMediator, samchon::example::chat_service::ChatRoom, samchon::namtree::NTSide, samchon::protocol::InvokeHistoryArray, samchon::namtree::NTParameterArray, samchon::example::chat_service::ChatMessage, and samchon::library::FTInstance.

10.34.3.2 void Entity::construct (std::shared_ptr< library::XML >) [pure virtual]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

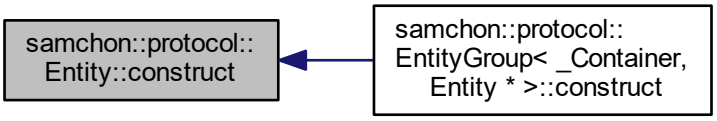
xml	An xml used to construct data of entity

Implemented in samchon::protocol::InvokeParameter, samchon::protocol::master::DistributedSystem, samchon::namtree::NTCriteria, samchon::namtree::NTFile, samchon::protocol::InvokeHistory, samchon::protocol::EntityGroup< _Container, _ETy, _Ty >, samchon::protocol::EntityGroup< _Container, std::unique_ptr< Entity > >, samchon::protocol::EntityGroup< _Container, std::shared_ptr< Entity > >, samchon::protocol::EntityGroup< _Container, Entity * >, samchon::protocol::master::ParallelSystem, samchon::protocol::EntityList< _Ty >, samchon::protocol::ExternalSystem, samchon::protocol::ExternalSystemRole, samchon::protocol::EntityArray< _Ty >, samchon::protocol::Invoke, samchon::protocol::master::PRInvokeHistory, samchon::protocol::master::DistributedSystemArray, samchon::example::chat_service::ChatMessage, samchon::namtree::INTExplore, samchon::namtree::NTParameterDetermined, samchon::protocol::master::ParallelSystemArrayMediator, samchon::protocol::master::DistributedSystemArrayMediator, samchon::namtree::NTParameter, samchon::protocol::SystemRole, samchon::namtree::NTSide, samchon::protocol::ExternalClientArray, samchon::protocol::ExternalServer, samchon::protocol::master::DistributedClientArrayMediator, samchon::protocol::master::ParallelClientArrayMediator, samchon::protocol::master::DistributedClientArray, samchon::protocol::master::DistributedServer, and samchon::protocol::master::ParallelServer.

Definition at line 23 of file Entity.cpp.

Referenced by samchon::protocol::EntityGroup< _Container, Entity * >::construct().

Here is the caller graph for this function:



10.34.3.3 auto Entity::key ( ) const -> std::string [virtual]

Get a key that can identify the Entity uniquely.

If identifier of the Entity is not atomic value, returns a string represents the composite identifier. If identifier of the Entity is not string, converts the identifier to string and returns the string.

Returns

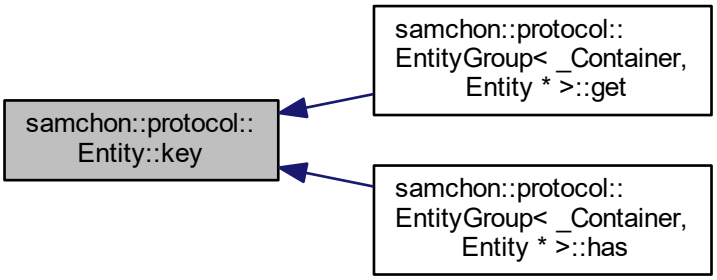
An identifier

Reimplemented in samchon::protocol::InvokeParameter, samchon::protocol::InvokeHistory, samchon::protocol::ExternalSystem, samchon::protocol::ExternalSystemRole, samchon::library::FTInstance, samchon::namtree::NTParameter, samchon::namtree::NTParameterDetermined, and samchon::protocol::SystemRole.

Definition at line 15 of file Entity.cpp.

Referenced by samchon::protocol::EntityGroup< _Container, Entity * >::get(), and samchon::protocol::EntityGroup< _Container, Entity * >::has().

Here is the caller graph for this function:



10.34.3.4 auto Entity::toXML ( ) const -> std::shared_ptr<library::XML> [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

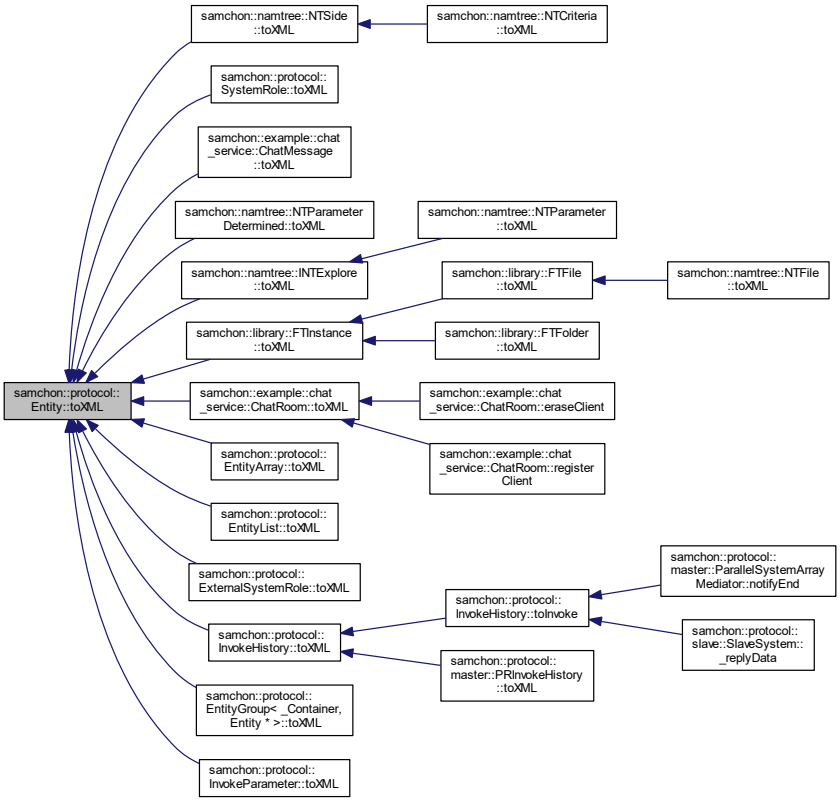
An XML object representing the Entity.

Reimplemented in samchon::protocol::InvokeParameter, samchon::protocol::EntityGroup< _Container, _ETy, _Ty >, samchon::protocol::EntityGroup< _Container, std::unique_ptr< Entity > >, samchon::protocol::EntityGroup< _Container, std::shared_ptr< Entity > >, samchon::protocol::EntityGroup< _Container, Entity * >, samchon::protocol::InvokeHistory, samchon::protocol::Invoke, samchon::protocol::master::DistributedSystem, samchon::namtree::NTCriteria, samchon::namtree::NTFile, samchon::protocol::ExternalSystem, samchon::protocol::ExternalSystemRole, samchon::protocol::master::ParallelSystem, samchon::protocol::master::DistributedSystemArray, samchon::protocol::EntityList< _Ty >, samchon::protocol::master::DistributedSystemRole, samchon::protocol::EntityArray< _Ty >, samchon::example::chat_service::ChatRoom, samchon::protocol::master::PRInvokeHistory, samchon::library::FTInstance, samchon::namtree::INTExplore, samchon::namtree::NTParameter, samchon::namtree::NTParameterDetermined, samchon::example::chat_service::ChatMessage, samchon::protocol::master::DistributedSystemArrayMediator, samchon::protocol::master::ParallelSystemArrayMediator, samchon::protocol::SystemRole, samchon::protocol::ExternalClientArray, samchon::protocol::ExternalServer, samchon::library::FTFolder, samchon::namtree::NTSide, samchon::library::FTFile, samchon::protocol::master::DSInvokeHistory, samchon::protocol::master::DistributedClientArrayMediator, samchon::protocol::master::ParallelClientArrayMediator, samchon::protocol::master::DistributedClientArray, samchon::protocol::master::DistributedServer, and samchon::protocol::master::ParallelServer.

Definition at line 30 of file Entity.cpp.

Referenced by samchon::namtree::NTSide::toXML(), samchon::protocol::SystemRole::toXML(), samchon::example::chat_service::ChatMessage::toXML(), samchon::namtree::NTParameterDetermined::toXML(), samchon::namtree::INTExplore::toXML(), samchon::library::FTInstance::toXML(), samchon::example::chat_service::ChatRoom::toXML(), samchon::protocol::EntityArray< _Ty >::toXML(), samchon::protocol::EntityList< _Ty >::toXML(), samchon::protocol::ExternalSystemRole::toXML(), samchon::protocol::InvokeHistory::toXML(), samchon::protocol::EntityGroup< _Container, Entity * >::toXML(), and samchon::protocol::InvokeParameter::toXML().

Here is the caller graph for this function:



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

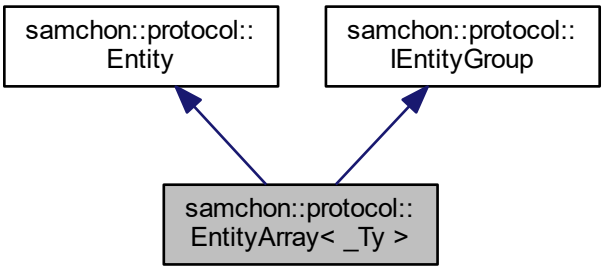
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/Entity.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/Entity.cpp

10.35 samchon::protocol::EntityArray< _Ty > Class Template Reference

An Entity and a static array containing Entity objects.

```
#include <EntityArray.hpp>
```

Collaboration diagram for samchon::protocol::EntityArray< _Ty >:



Public Member Functions

- EntityArray ()

  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML > xml) override

  Construct data of the Entity from an XML object.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the EntityArray.

Additional Inherited Members

10.35.1 Detailed Description

template<typename _Ty>class samchon::protocol::EntityArray< _Ty >

An Entity and a static array containing Entity objects.

Template Parameters

_Ty	A type of children Entity. Must be a class derived from an Entity.

EntityArray is a static array containing children objects derived from an Entity class.

The EntityArray has advantages of performance like fast iteration or fast access because the EntityArray is serialized. However, as EntityArray is serialized, allocated memory space for each child is fixed and cannot contain derived a entity class from the child entity type.

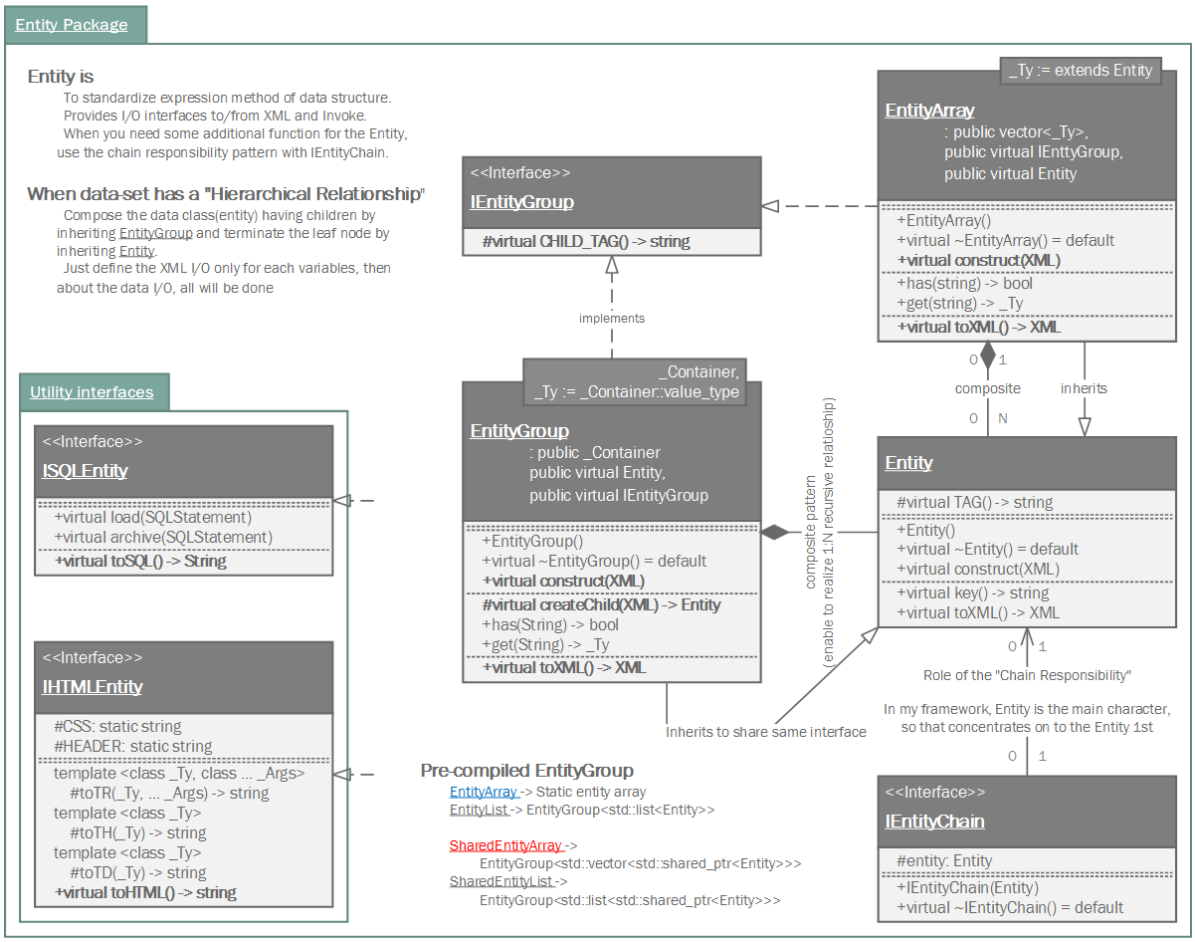
- EntityArray<Animal> is specified
- Dog is extended from the Animal
- EntityArray<Animal> cannot contain Dog. You can try insertion of the Dog class into the EntityArray, but only memory size of Animal is allocated, so that derived member variables and methods are all truncated.

Thus, it's suitable for when fast accessment and iteration like an entity class containing mathmatical or statistical data and methods which doesn't seemed to be inherited.

[Inherited]

Entity is a class for standardization of expression method using on network I/O by XML. If Invoke is a standard message protocol of Samchon Framework which must be kept, Entity is a recommended semi-protocol of message for expressing a data class. Following the semi-protocol Entity is not imposed but encouraged.

As we could get advantages from standardization of message for network I/O with Invoke, we can get additional advantage from standardizing expression method of data class with Entity. We do not need to know a part of network communication. Thus, with the Entity, we can only concentrate on entity's own logics and relationships between another entities. Entity does not need to how network communications are being done.



Example source

```
#include <iostream>

#include <samchon/protocol/Entity.hpp>
#include <samchon/protocol/IHTMLEntity.hpp>
#include <samchon/protocol/SharedEntityArray.hpp>

#include <samchon/library/XML.hpp>

#ifdef _WIN64
#   ifdef _DEBUG
#       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "x64/Release/SamchonFramework.lib")
#   endif
#else
#   ifdef _DEBUG
#       pragma comment(lib, "Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "Release/SamchonFramework.lib")
#   endif
#endif

using namespace std;
using namespace samchon::library;
using namespace samchon::protocol;

```

```
class Member
    : public Entity, public virtual IHTMLEntity
{
protected:
    typedef Entity super;

    string id;
    string name;
    int age;
    int grade;

public:
    /* ----------------------------------------------------------------------
        CONSTRUCTORS
    ---------------------------------------------------------------------- */
    Member()
        : super(), IHTMLEntity()
    {
    };
    Member(const string &id, const string &name, int age, int grade)
        : super(), IHTMLEntity()
    {
        this->id = id;
        this->name = name;
        this->age = age;
        this->grade = grade;
    };
    virtual ~Member() = default;

    virtual void construct(shared_ptr<XML> xml) override
    {
        this->id = xml->getProperty("id");
        this->name = xml->getProperty("name");
        this->age = xml->getProperty<int>("age");
        this->grade = xml->getProperty<int>("grade");
    };

    /* ----------------------------------------------------------------------
        GETTERSs
    ---------------------------------------------------------------------- */
    virtual auto key() const -> std::string override
    {
        return this->id;
    };

    /* ----------------------------------------------------------------------
        XML EXPORTERS
    ---------------------------------------------------------------------- */
    virtual auto TAG() const -> string override
    {
        return "member";
    };
    virtual auto toXML() const -> shared_ptr<XML>
    {
        shared_ptr<XML> &xml = super::toXML();
        xml->setProperty("id", id);
        xml->setProperty("name", name);
        xml->setProperty("age", age);
        xml->setProperty("grade", grade);

        return move(xml);
    };
    virtual auto toHTML() const -> string
    {
        return toTR(id, name, age, grade);
    };
};

class MemberArray
    : public SharedEntityArray<Member>,
    public virtual IHTMLEntity
{
protected:
    typedef SharedEntityArray<Member> super;

    string application;
    int department;
    Member *chief;

public:
    /* ----------------------------------------------------------------------
        CONSTRUCTORS
    ---------------------------------------------------------------------- */
    MemberArray()
        : super(), IHTMLEntity()
    {
        this->chief = nullptr;
```

```
    };
    virtual ~MemberArray() = default;

    // You don't need to consider children(Member) objects
    // Just concentrate on constructing MemberArray's own member variables
    virtual void construct(shared_ptr<XML> xml) override
    {
        super::construct(xml);

        this->application = xml->getProperty("application");
        this->department = xml->getProperty<int>("department");

        if(xml->hasProperty("chief") == true && this->has( xml->getProperty("chief") ) == true)
            this->chief = this->get( xml->getProperty("cheif") ).get();
    };

protected:
    //FACTORY METHOD FOR MEMBER
    virtual auto createChild(shared_ptr<XML> = nullptr) -> Member* override
    {
        return new Member();
    };

    /* ----------------------------------------------------------------------
        XML EXPORTERS
    ---------------------------------------------------------------------- */
public:
    virtual auto TAG() const -> string override
    {
        return "memberArray";
    };
    virtual auto CHILD_TAG() const -> string override
    {
        return "member";
    };

    // You don't need to consider children(Member) objects
    // Just concentrate on expressing MemberArray's own member variables
    virtual auto toXML() const -> shared_ptr<XML>
    {
        shared_ptr<XML> &xml = super::toXML();
        xml->setProperty("application", application);
        xml->setProperty("department", department);

        if(chief != nullptr)
            xml->setProperty("cheif", chief->key());

        return move(xml);
    };
    virtual auto toHTML() const -> string
    {
        string html = "<table>\n";
        html += toTH("ID", "Name", "Age", "Grade") + "\n";

        for (size_t i = 0; i < 2; i++)
            html += at(i)->toHTML() + "\n";

        html += "</table>";
        return move(html);
    };
};

void main()
{
    string str = string("") +
        "<memberArray application='framework' department='7' cheif='samchon'>\n" +
        "    <member id='samchon' name='Jeongho Nam' age='27' grade='5' />" +
        "    <member id='submaster' name='No Name' age='100' grade='4' />" +
        "    <member id='john' name='John Doe' age='33' grade='2' />" +
        "    <member id='bad_man' name='Bad Man' age='44' grade='-1' />" +
        "    <member id='guest' name='Guest' age='0' grade='0' />" +
        "</memberArray>";
    shared_ptr<XML> xml(new XML(str));

    MemberArray memberArray;
    memberArray.construct(xml);

    memberArray.emplace_back(new Member("freshman", "A fresh man", 20, 2));
    memberArray.emplace_back(new Member("senior", "A senior", 70, 2));

    cout << memberArray.toXML()->toString() << endl << endl;
    cout << memberArray.toHTML() << endl;
    system("pause");
}
```

Result of the example

```
D:\OneDrive\Project\Samchon\framework\cpp\Debug\entity.exe
<memberArray application="framework" department="7">
        <member age="27" grade="5" id="samchon" name="Jeongho" />
        <member age="100" grade="4" id="submaster" name="No" />
        <member age="33" grade="2" id="john" name="John" />
        <member age="44" grade="-1" id="bad_man" name="Bad" />
        <member age="0" grade="0" id="guest" name="Guest" />
        <member age="20" grade="2" id="freshman" name="A fresh man" />
        <member age="70" grade="2" id="senior" name="A senior" />
</memberArray>

<table>
        <th>
                <td>ID</td>
                <td>Name</td>
                <td>Age</td>
                <td>Grade</td>
        </th>
        <tr>
                <td>samchon</td>
                <td>Jeongho</td>
                <td>27</td>
                <td>5</td>
        </tr>
        <tr>
                <td>submaster</td>
                <td>No</td>
                <td>100</td>
                <td>4</td>
        </tr>
</table>
Press any key to continue . . .
```

Note

I say repeatedly. Expression method of Entity is recommended, but not imposed. It's a semi protocol for network I/O but not a essential protocol must be kept. The expression method of Entity, using on network I/O, is expressed by XML string.

If your own network system has a critical performance issue on communication data class, it would be better to using binary communication (with ByteArray or boost::serialization). Don't worry about the problem! Invoke also provides methods for binary data (ByteArray).

See also

protocol

Author

Jeongho Nam

Warning

If a data structure has recursive and hierarchical relationship, never use the EntityArray. EntityArray's memory allocation is static. The recursive relationship causes infinite memory allocation. It must generates stack overflow.

Use EntityGroup instead.

- class NTCriteria : public EntityArray<NTCriteria> causes a stack overflow.
- class NTCriteria : public SharedEntityArray<NTCriteria> doesn't cause the stack overflow.

Definition at line 47 of file EntityArray.hpp.

10.35.2 Constructor & Destructor Documentation

10.35.2.1 template<typename _Ty > samchon::protocol::EntityArray< _Ty >::EntityArray ( )

Default Constructor.

10.35.3 Member Function Documentation

10.35.3.1 template<typename _Ty > virtual void samchon::protocol::EntityArray< _Ty >::construct (std::shared_ptr< library::XML > *xml*) [inline], [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityArray's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityArray::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityArray::toXML().

Constructs only data of EntityArray's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

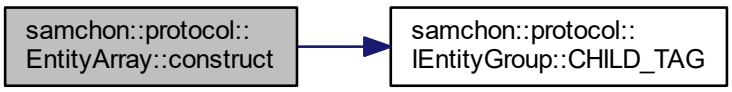
xml	An xml used to construct data of entity

Implements samchon::protocol::Entity.

Definition at line 76 of file EntityArray.hpp.

References samchon::protocol::IEntityGroup::CHILD_TAG().

Here is the call graph for this function:



10.35.3.2 template<typename _Ty > virtual auto samchon::protocol::EntityArray< _Ty >::toXML ( ) const -> std::shared_ptr<library::XML> [inline],[override],[virtual]

Get an XML object represents the EntityArray.

Archives the EntityArray's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityArray::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityArray::toXML().

Archives only data of EntityArray's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

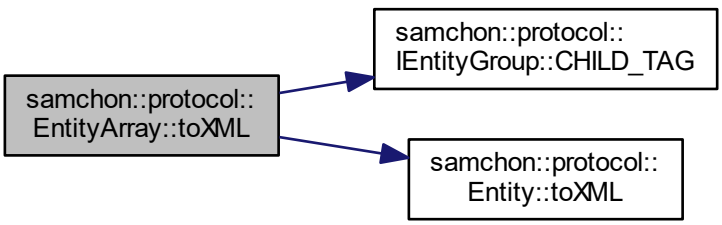
An XML object representing the Entity.

Reimplemented from samchon::protocol::Entity.

Definition at line 104 of file EntityArray.hpp.

References samchon::protocol::IEntityGroup::CHILD_TAG(), and samchon::protocol::Entity::toXML().

Here is the call graph for this function:



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

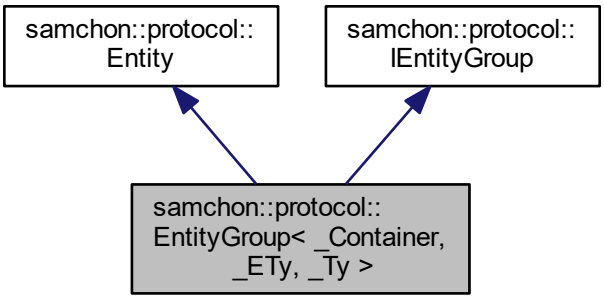
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/EntityArray.hpp

10.36 samchon::protocol::EntityGroup< _Container, _ETy, _Ty > Class Template Reference

An Entity and a container of children Entity objects.

`#include <EntityGroup.hpp>`

Collaboration diagram for samchon::protocol::EntityGroup< _Container, _ETy, _Ty >:



Public Member Functions

- EntityGroup ()
  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML > xml)
  Construct data of the Entity from an XML object.
- auto has (const std::string &key) const -> bool
  Indicates whether a container has an object having the specified identifier.
- auto get (const std::string &key) -> value_type &
  Access the element by specified identifier(key).
- auto get (const std::string &key) const -> const value_type &
  Access the const element by specified identifier(key).
- virtual auto toXML () const -> std::shared_ptr< library::XML >
  Get an XML object represents the EntityGroup.

Protected Member Functions

- virtual auto createChild (std::shared_ptr< library::XML >) -> entity_type *=0
  Factory method of a child Entity.

10.36.1 Detailed Description

template<typename _Container, typename _ETy = Entity, typename _Ty = _Container::value_type>class samchon::protocol::EntityGroup< _Container, _ETy, _Ty >

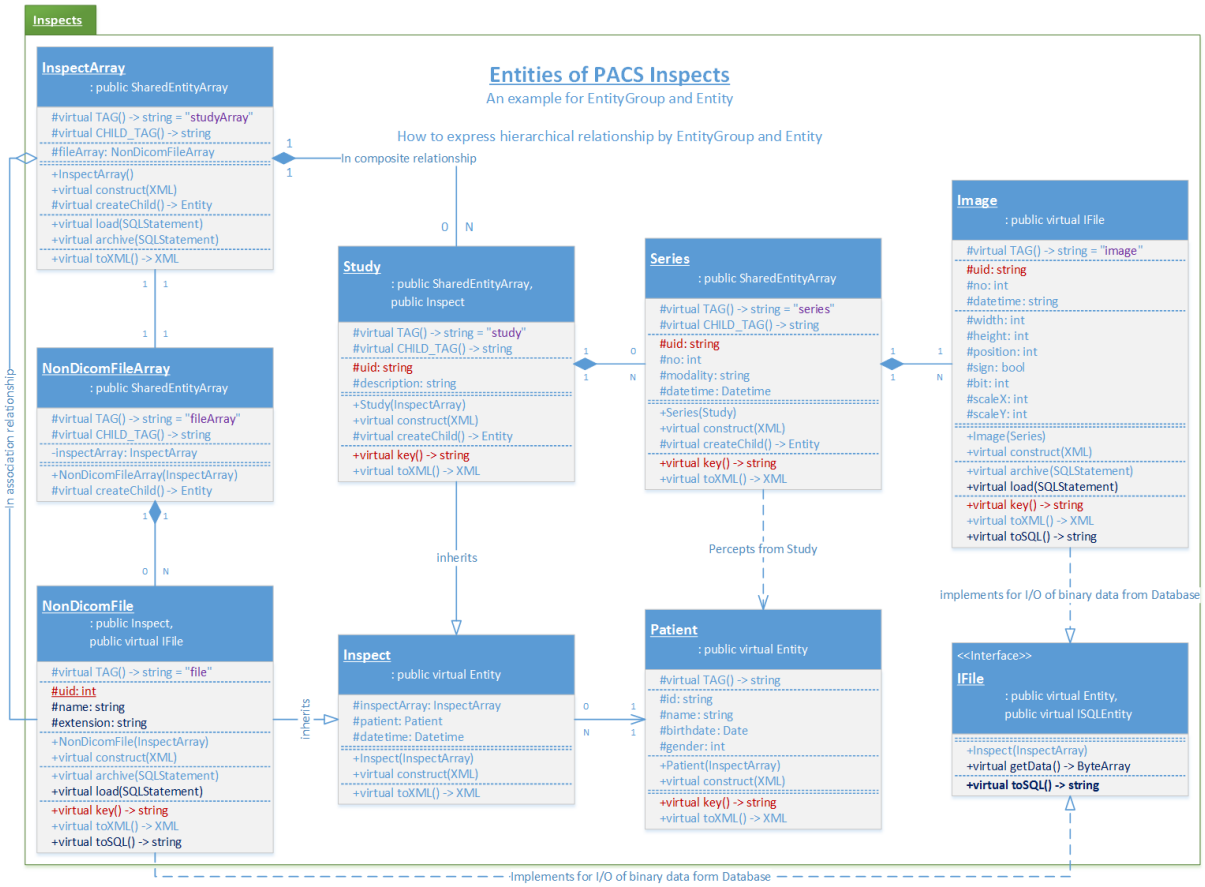
An Entity and a container of children Entity objects.

Template Parameters

_Container	A type of container containing children entity objects.
_Ety	A type of children entity. It must be a class derived from an Entity class, or Entity class itself.
_Ty	A type of children element of _Container. Using default template parameter is recommended.

EntityGroup is a template class for containinig children Entity objects, and also another type of an Entity, too. You can realize hierarchical relationship. Although some entities have complicated hierarchical relationship, you can deduct a optimal solution easily with EntityGroup and Entity.

If an entity has some subordinate entities of same type, they are in "Composite relationship". Make the entity to be EmntityGroup and subordinate entities to be children of the entity. When those relationships are continued, continue to create classes dervied from EntityGroup. When those relationshiop meets a terminal node, then make the terminal node to be an Entity.



EntityGroup is an Entity, and a container of children Entity objects at the same time. If children type, of a class derived from an EntityGroup, is itself, you can realize hierarchical and recursive relationship. The relationship is called as "Composite pattern".

- FTFolder extends FTInstance and SharedEntityArray<FTInstance>
- NTCriteria extends SharedEntityArray<NTCriteria>

Note

As a freelancer developer and architect I am, I even design DB I/O to follow the format representing Entity and EntityGroup by XML. Below T-SQL script also follows the standard format of expressing Entity with XML by procedure and "FOR XML AUTO" statement.

```
SET
  ANSI_NULLS,
  QUOTED_IDENTIFIER,
  CONCAT_NULL_YIELDS_NULL,
  ANSI_WARNINGS,
  ANSI_PADDING
ON;

USE OraQ;
IF OBJECT_ID('v_studyStatus') IS NOT NULL           DROP VIEW v_studyStatus;
IF OBJECT_ID('v_nonDicomFileStatus') IS NOT NULL    DROP VIEW v_nonDicomFileStatus;
IF OBJECT_ID('v_candidateStudy') IS NOT NULL        DROP VIEW v_candidateStudy;
IF OBJECT_ID('v_candidateFile') IS NOT NULL         DROP VIEW v_candidateFile;
IF OBJECT_ID('goCandidateInspectList') IS NOT NULL  DROP PROCEDURE goCandidateInspectList;
GO

CREATE VIEW v_studyStatus
AS
    WITH cte AS
    (
        SELECT S.studyUID uid, S.type type, ROW_NUMBER() OVER (PARTITION BY S.studyUID ORDER BY H.datetime DESC) no
        FROM History.history H, History.study S
        WHERE
            H.uid = S.uid AND type <> 1
    )
    SELECT
        A.uid,
        CASE WHEN B.type IS NULL THEN
            A.type
            ELSE 1
        END status
    FROM
    (
        SELECT uid, type
        FROM cte
        WHERE no = 1
    ) A LEFT OUTER JOIN
    (
        SELECT studyUID uid, 1 type
        FROM History.study
        GROUP BY studyUID
        HAVING MAX(type) = 1
    ) B ON A.uid = B.uid
GO
CREATE VIEW v_nonDicomFileStatus
AS
    WITH cte AS
    (
        SELECT
            F.fileUID uid, F.type type,
            ROW_NUMBER() OVER (PARTITION BY F.fileUID ORDER BY H.datetime DESC) no
        FROM History.history H, History.nonDicomFile F
        WHERE
            H.uid = F.uid AND type <> 1
    )
    SELECT
        A.uid,
        CASE WHEN B.type IS NULL THEN
            A.type
            ELSE 1
        END status
    FROM
    (
        SELECT uid, type
        FROM cte
        WHERE no = 1
    ) A LEFT OUTER JOIN
    (
        SELECT fileUID uid, 1 type
        FROM History.nonDicomFile
        GROUP BY fileUID
        HAVING MAX(type) = 1
    ) B ON A.uid = B.uid
GO
CREATE VIEW v_candidateStudy
AS
    SELECT DISTINCT
        P.id patientID,
            P.name patientName, P.gender patientGender,
            CAST(P.birthdate AS DATE) patientBirthdate,
        S.uid,
            S.description,
            CONVERT(VARCHAR, CAST(S.datetime AS DATETIME), 120) AS datetime,
            H.status
    FROM
        Hospital.patient P, Inspect.media M, Inspect.mediaStudyPair A,
```

```
        Inspect.study S, v_studyStatus H
    WHERE
        P.id = M.patientID AND M.uid = A.mediaUID AND A.studyUID = S.uid AND
        S.uid = H.uid
GO
CREATE VIEW v_candidateFile
AS
    SELECT DISTINCT
        P.id patientID,
            P.name patientName, P.gender patientGender,
            CAST(P.birthdate AS DATE) patientBirthdate,
        M.uid mediaUID,
            F.uid, F.name, F.extension,
             CONVERT(VARCHAR, CAST(F.datetime AS DATETIME), 120) AS datetime,
             H.status
    FROM
        Hospital.patient P, Inspect.media M,
        Inspect.nonDicomFile F, v_nonDicomFileStatus H
    WHERE
        P.id = M.patientID AND M.uid = F.mediaUID AND F.uid = H.uid
GO

CREATE PROCEDURE goCandidateInspectList
AS
    DECLARE @xml XML =
    (
        SELECT *
        FROM
        (
            SELECT N'candidate' service
        ) inspect
        FOR XML AUTO
    )
    DECLARE @studyList XML =
    (
        SELECT study.*, series.*, image.*
        FROM
            v_candidateStudy study
                LEFT OUTER JOIN Inspect.series series
                    ON study.uid = series.studyUID
                LEFT OUTER JOIN Inspect.image image
                    ON series.uid = image.seriesUID
        FOR XML AUTO, ROOT(N'studyList')
    )
    DECLARE @fileList XML =
    (
        SELECT * FROM v_candidateFile
        FOR XML RAW(N'file'), ROOT(N'fileList')
    )

    IF(@studyList IS NOT NULL)  SET @xml.modify('insert sql:variable("@studyList") into (/inspect)[1]')
    IF(@fileList IS NOT NULL)   SET @xml.modify('insert sql:variable("@fileList") into (/inspect)[1]')

    SELECT @xml;
GO
```

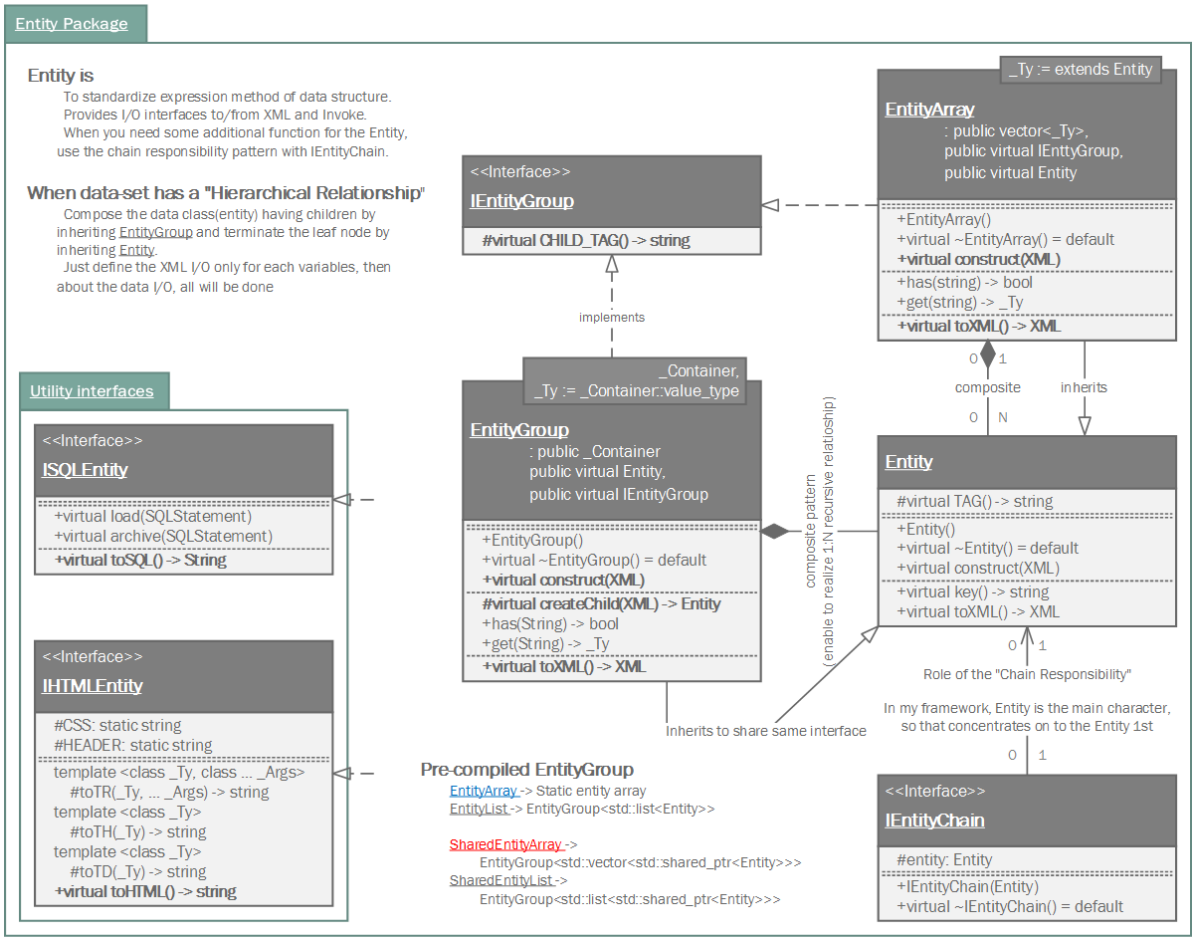
Warning

EntityGroup contains children entity elements as type of pointer. Because children entity objects are not serialized and referenced by pointer, its iteration and accessment is not fast. If it needs higher performance, then use EntityArray (static array for children entity) instead.

[Inherited]

Entity is a class for standardization of expression method using on network I/O by XML. If Invoke is a standard message protocol of Samchon Framework which must be kept, Entity is a recommended semi-protocol of message for expressing a data class. Following the semi-protocol Entity is not imposed but encouraged.

As we could get advantages from standardization of message for network I/O with Invoke, we can get additional advantage from standardizing expression method of data class with Entity. We do not need to know a part of network communication. Thus, with the Entity, we can only concentrate on entity's own logics and relationships between another entities. Entity does not need to how network communications are being done.



Example source

```
#include <iostream>

#include <samchon/protocol/Entity.hpp>
#include <samchon/protocol/IHTMLEntity.hpp>
#include <samchon/protocol/SharedEntityArray.hpp>

#include <samchon/library/XML.hpp>

#ifdef _WIN64
#   ifdef _DEBUG
#       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "x64/Release/SamchonFramework.lib")
#   endif
#else
#   ifdef _DEBUG
#       pragma comment(lib, "Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "Release/SamchonFramework.lib")
#   endif
#endif

using namespace std;
using namespace samchon::library;
using namespace samchon::protocol;

class Member
    : public Entity, public virtual IHTMLEntity
{
protected:
    typedef Entity super;

    string id;
    string name;
    int age;
    int grade;

public:
    /* ----------------------------------------------------------------------
```

```
        CONSTRUCTORS
    ------------------------------------------------------------------- */
    Member()
        : super(), IHTMLEntity()
    {
    };
    Member(const string &id, const string &name, int age, int grade)
        : super(), IHTMLEntity()
    {
        this->id = id;
        this->name = name;
        this->age = age;
        this->grade = grade;
    };
    virtual ~Member() = default;

    virtual void construct(shared_ptr<XML> xml) override
    {
        this->id = xml->getProperty("id");
        this->name = xml->getProperty("name");
        this->age = xml->getProperty<int>("age");
        this->grade = xml->getProperty<int>("grade");
    };

    /* -------------------------------------------------------------------
        GETTERSs
    ------------------------------------------------------------------- */
    virtual auto key() const -> std::string override
    {
        return this->id;
    };

    /* -------------------------------------------------------------------
        XML EXPORTERS
    ------------------------------------------------------------------- */
    virtual auto TAG() const -> string override
    {
        return "member";
    };
    virtual auto toXML() const -> shared_ptr<XML>
    {
        shared_ptr<XML> &xml = super::toXML();
        xml->setProperty("id", id);
        xml->setProperty("name", name);
        xml->setProperty("age", age);
        xml->setProperty("grade", grade);

        return move(xml);
    };
    virtual auto toHTML() const -> string
    {
        return toTR(id, name, age, grade);
    };
};

class MemberArray
    : public SharedEntityArray<Member>,
    public virtual IHTMLEntity
{
protected:
    typedef SharedEntityArray<Member> super;

    string application;
    int department;
    Member *chief;

public:
    /* -------------------------------------------------------------------
        CONSTRUCTORS
    ------------------------------------------------------------------- */
    MemberArray()
        : super(), IHTMLEntity()
    {
        this->chief = nullptr;
    };
    virtual ~MemberArray() = default;

    // You don't need to consider children(Member) objects
    // Just concentrate on constructing MemberArray's own member variables
    virtual void construct(shared_ptr<XML> xml) override
    {
        super::construct(xml);

        this->application = xml->getProperty("application");
        this->department = xml->getProperty<int>("department");

        if(xml->hasProperty("chief") == true && this->has( xml->getProperty("chief") ) == true)
```

```
            this->chief = this->get( xml->getProperty("cheif") ).get();
    };

protected:
    //FACTORY METHOD FOR MEMBER
    virtual auto createChild(shared_ptr<XML> = nullptr) -> Member* override
    {
        return new Member();
    };

    /* ----------------------------------------------------------------------
        XML EXPORTERS
    ---------------------------------------------------------------------- */
public:
    virtual auto TAG() const -> string override
    {
        return "memberArray";
    };
    virtual auto CHILD_TAG() const -> string override
    {
        return "member";
    };

    // You don't need to consider children(Member) objects
    // Just concentrate on expressing MemberArray's own member variables
    virtual auto toXML() const -> shared_ptr<XML>
    {
        shared_ptr<XML> &xml = super::toXML();
        xml->setProperty("application", application);
        xml->setProperty("department", department);

        if(chief != nullptr)
            xml->setProperty("cheif", chief->key());

        return move(xml);
    };
    virtual auto toHTML() const -> string
    {
        string html = "<table>\n";
        html += toTH("ID", "Name", "Age", "Grade") + "\n";

        for (size_t i = 0; i < 2; i++)
            html += at(i)->toHTML() + "\n";

        html += "</table>";
        return move(html);
    };
};

void main()
{
    string str = string("") +
        "<memberArray application='framework' department='7' cheif='samchon'>\n" +
        "   <member id='samchon' name='Jeongho Nam' age='27' grade='5' />" +
        "   <member id='submaster' name='No Name' age='100' grade='4' />" +
        "   <member id='john' name='John Doe' age='33' grade='2' />" +
        "   <member id='bad_man' name='Bad Man' age='44' grade='-1' />" +
        "   <member id='guest' name='Guest' age='0' grade='0' />" +
        "</memberArray>";
    shared_ptr<XML> xml(new XML(str));

    MemberArray memberArray;
    memberArray.construct(xml);

    memberArray.emplace_back(new Member("freshman", "A fresh man", 20, 2));
    memberArray.emplace_back(new Member("senior", "A senior", 70, 2));

    cout << memberArray.toXML()->toString() << endl << endl;
    cout << memberArray.toHTML() << endl;
    system("pause");
}
```

Result of the example

```
D:\OneDrive\Project\Samchon\framework\cpp\Debug\entity.exe
<memberArray application="framework" department="7">
        <member age="27" grade="5" id="samchon" name="Jeongho" />
        <member age="100" grade="4" id="submaster" name="No" />
        <member age="33" grade="2" id="john" name="John" />
        <member age="44" grade="-1" id="bad_man" name="Bad" />
        <member age="0" grade="0" id="guest" name="Guest" />
        <member age="20" grade="2" id="freshman" name="A fresh man" />
        <member age="70" grade="2" id="senior" name="A senior" />
</memberArray>

<table>
        <th>
                <td>ID</td>
                <td>Name</td>
                <td>Age</td>
                <td>Grade</td>
        </th>
        <tr>
                <td>samchon</td>
                <td>Jeongho</td>
                <td>27</td>
                <td>5</td>
        </tr>
        <tr>
                <td>submaster</td>
                <td>No</td>
                <td>100</td>
                <td>4</td>
        </tr>
</table>
Press any key to continue . . .
```

Note

I say repeatedly. Expression method of Entity is recommended, but not imposed. It's a semi protocol for network I/O but not a essential protocol must be kept. The expression method of Entity, using on network I/O, is expressed by XML string.

If your own network system has a critical performance issue on communication data class, it would be better to using binary communication (with ByteArray or boost::serialization). Don't worry about the problem! Invoke also provides methods for binary data (ByteArray).

See also

protocol

Author

Jeongho Nam

Definition at line 54 of file EntityGroup.hpp.

10.36.2 Constructor & Destructor Documentation

10.36.2.1 template<typename _Container, typename _ETy = Entity, typename _Ty = _Container::value_type> samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::EntityGroup ( ) [inline]

Default Constructor.

Definition at line 71 of file EntityGroup.hpp.

10.36.3 Member Function Documentation

10.36.3.1 template<typename _Container, typename _ETy = Entity, typename _Ty = _Container::value_type> virtual void samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::construct (std::shared_ptr< library::XML > *xml*) [inline],[virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

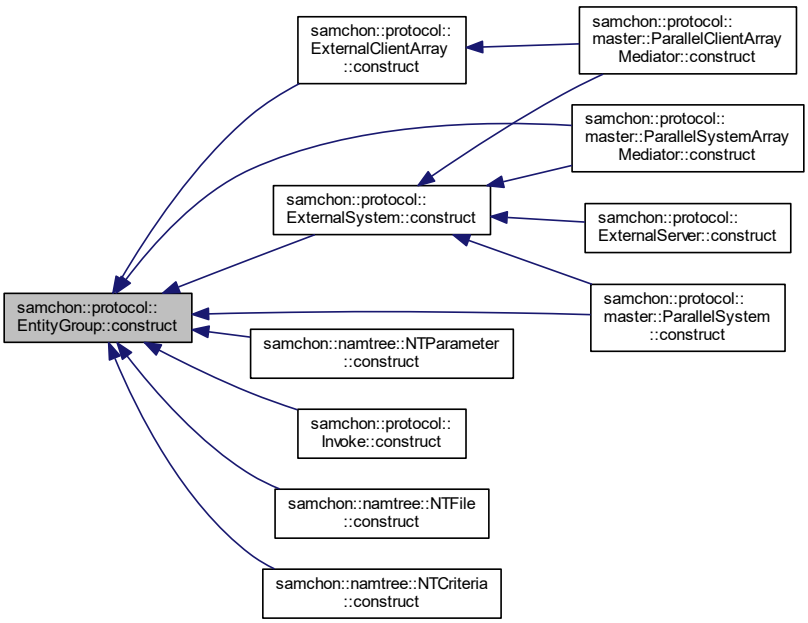
Implements samchon::protocol::Entity.

Reimplemented in samchon::protocol::master::DistributedSystem, samchon::namtree::NTCriteria, samchon::protocol::master::ParallelSystem, samchon::protocol::ExternalSystem, samchon::protocol::Invoke, samchon::protocol::master::DistributedSystemArray, samchon::protocol::master::ParallelSystemArrayMediator, samchon::protocol::master::DistributedSystemArrayMediator, samchon::namtree::NTParameter, samchon::protocol::ExternalClientArray, samchon::protocol::ExternalServer, samchon::protocol::master::DistributedClientArrayMediator, samchon::protocol::master::ParallelClientArrayMediator, samchon::protocol::master::DistributedClientArray, samchon::protocol::master::DistributedServer, and samchon::protocol::master::ParallelServer.

Definition at line 92 of file EntityGroup.hpp.

Referenced by samchon::protocol::ExternalClientArray::construct(), samchon::namtree::NTParameter::construct(), samchon::protocol::master::ParallelSystemArrayMediator::construct(), samchon::protocol::Invoke::construct(), samchon::protocol::ExternalSystem::construct(), samchon::protocol::master::ParallelSystem::construct(), samchon::namtree::NTFile::construct(), and samchon::namtree::NTCriteria::construct().

Here is the caller graph for this function:



10.36.3.2 template<typename _Container, typename _ETy = Entity, typename _Ty = _Container::value_type> virtual auto samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::createChild (std::shared_ptr< library::XML >) -> entity_type * [protected], [pure virtual]

Factory method of a child Entity.

EntityGroup::createChild() is a factory method creating a new child Entity which is belonged to the Entity↩Group. This method is called by EntityGroup::construct(). The children construction methods Entity↩::construct() will be called by abstract method of the EntityGroup::construct().

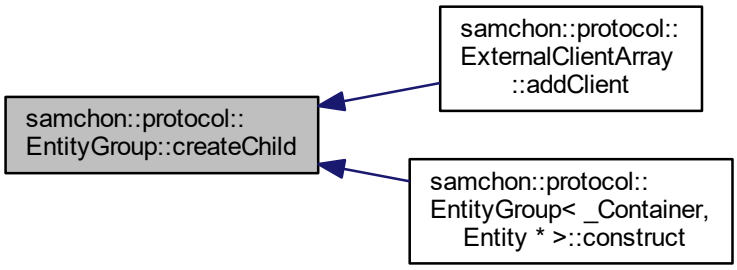
Returns

A new child Entity belongs to EntityGroup.

Implemented in samchon::namtree::NTCriteria, samchon::protocol::master::ParallelSystem, samchon::protocol::Invoke, samchon::namtree::NTParameter, samchon::namtree::NTParameterArray, samchon::protocol::slave::ParallelSystem, samchon::protocol::master::DSInvokeHistoryList, samchon::protocol::master::PRInvokeHistoryArray, and samchon::protocol::InvokeHistoryArray.

Referenced by samchon::protocol::ExternalClientArray::addClient(), and samchon::protocol::EntityGroup< _Container, Entity * >::construct().

Here is the caller graph for this function:



10.36.3.3 template<typename _Container, typename _ETy = Entity, typename _Ty = _Container::value_type> auto samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::has (const std::string & *key*) const -> bool [inline]

Indicates whether a container has an object having the specified identifier.

Parameters

key	An identifier of an Entity

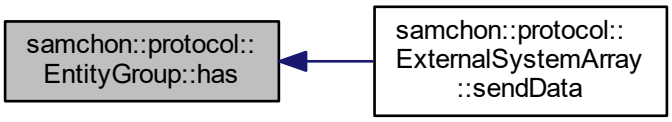
Returns

If there's the object then true, otherwise false

Definition at line 157 of file EntityGroup.hpp.

Referenced by samchon::protocol::ExternalSystemArray::sendData().

Here is the caller graph for this function:



10.36.3.4 template<typename _Container, typename _ETy = Entity, typename _Ty = _Container::value_type> auto samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::get (const std::string & *key*) -> value_type& [inline]

Access the element by specified identifier(key).

Parameters

key	the identifier of the element wants to access

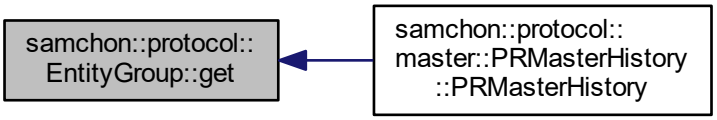
Returns

The element having the key, or throw exception if there is none.

Definition at line 172 of file EntityGroup.hpp.

Referenced by samchon::protocol::master::PRMasterHistory::PRMasterHistory().

Here is the caller graph for this function:



10.36.3.5 template<typename _Container, typename _ETy = Entity, typename _Ty = _Container::value_type> auto samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::get (const std::string & *key*) const -> const value_type& [inline]

Access the const element by specified identifier(key).

Parameters

key	the identifier of the element wants to access

Returns

The const element having the key, or throw exception if there is none.

Definition at line 187 of file EntityGroup.hpp.

10.36.3.6 template<typename _Container, typename _ETy = Entity, typename _Ty = _Container::value_type> virtual auto samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::toXML ( ) const -> std::shared_ptr<library::XML> [inline], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

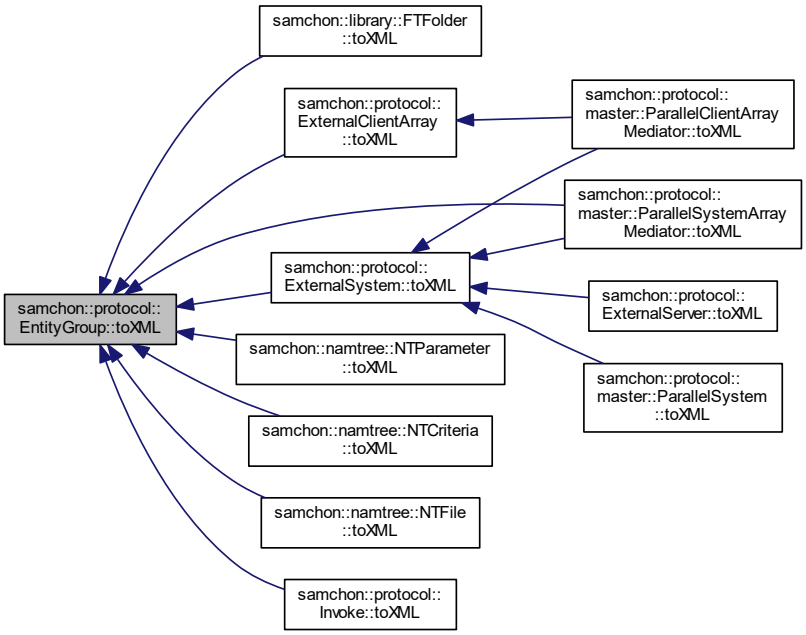
Reimplemented from samchon::protocol::Entity.

Reimplemented in samchon::protocol::Invoke, samchon::protocol::master::DistributedSystem, samchon::namtree::NTCriteria, samchon::protocol::ExternalSystem, samchon::protocol::master::ParallelSystem, samchon::protocol::master::DistributedSystemArray, samchon::namtree::NTParameter, samchon::protocol::master::DistributedSystemArrayMediator, samchon::protocol::master::ParallelSystemArrayMediator, samchon::protocol::ExternalClientArray, samchon::protocol::ExternalServer, samchon::library::FTFolder, samchon::protocol::master::DistributedClientArrayMediator, samchon::protocol::master::ParallelClientArrayMediator, samchon::protocol::master::DistributedClientArray, samchon::protocol::master::DistributedServer, and samchon::protocol::master::ParallelServer.

Definition at line 214 of file EntityGroup.hpp.

Referenced by samchon::library::FTFolder::toXML(), samchon::protocol::ExternalClientArray::toXML(), samchon::protocol::master::ParallelSystemArrayMediator::toXML(), samchon::namtree::NTParameter::toXML(), samchon::protocol::ExternalSystem::toXML(), samchon::namtree::NTCriteria::toXML(), samchon::namtree::NTFile::toXML(), and samchon::protocol::Invoke::toXML().

Here is the caller graph for this function:



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

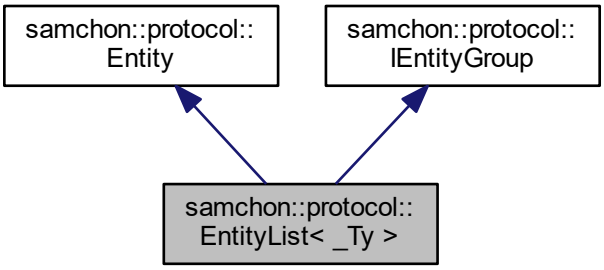
– D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/EntityGroup.hpp

10.37 samchon::protocol::EntityList< _Ty > Class Template Reference

An Entity and a static list containing Entity objects.

`#include <EntityList.hpp>`

Collaboration diagram for samchon::protocol::EntityList< _Ty >:



Public Member Functions

- EntityList ()
  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML > xml) override
  Construct data of the Entity from an XML object.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override
  Get an XML object represents the EntityList.

Additional Inherited Members

10.37.1 Detailed Description

template<typename _Ty>class samchon::protocol::EntityList< _Ty >

An Entity and a static list containing Entity objects.

Template Parameters

_Ty	A type of children Entity. Must be a class derived from an Entity.

EntityList is a static array containing children objects derived from an Entity class.

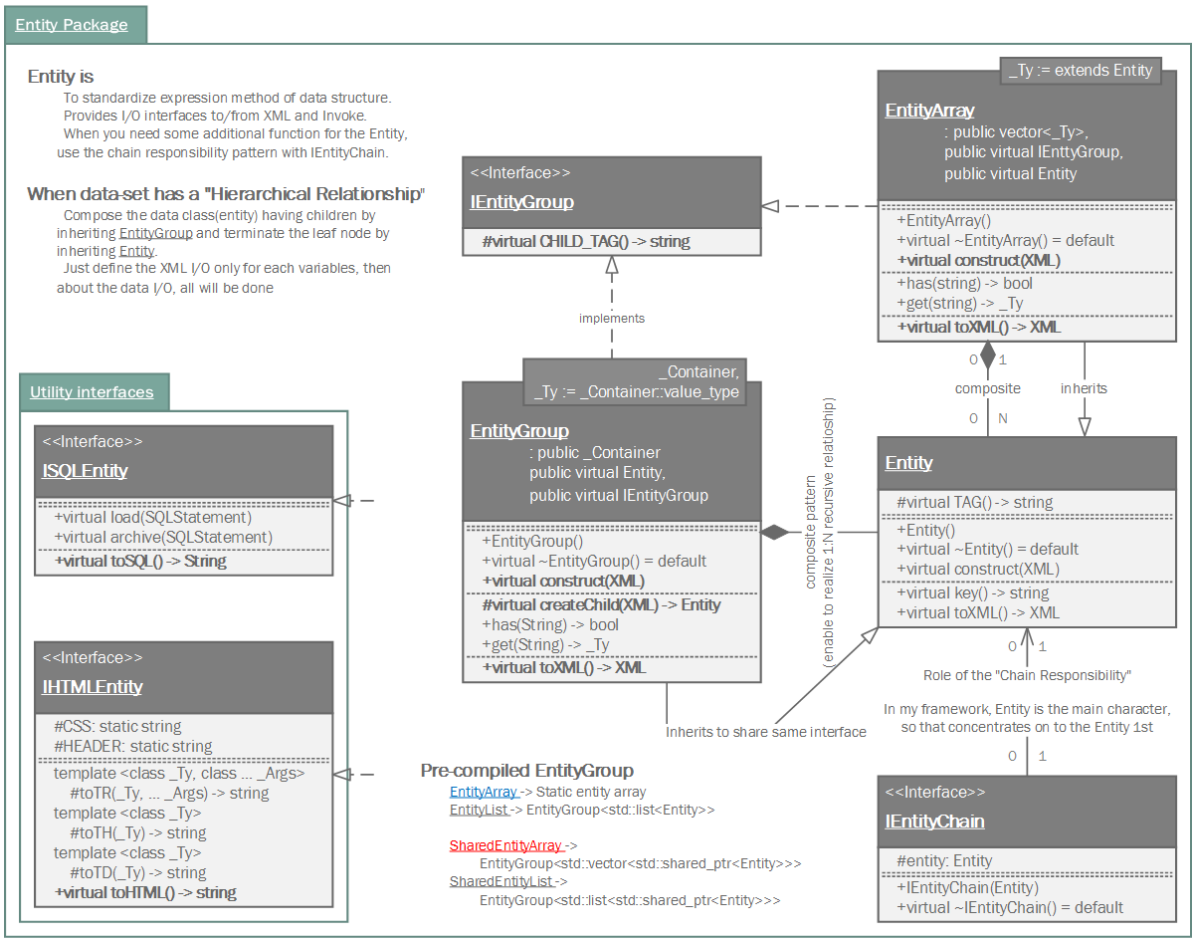
EntityList, its allocated memory space for each child is fixed and cannot contain derived a entity class from the child entity type. It has advantages on performance by direct accessment and static memory allocation for child, however, its increasement of performance is not eminent.

- EntityList<Animal> is specified
- Dog is extended from the Animal
- EntityList<Animal> cannot contain Dog. You can try insertion of the Dog class into the EntityList, but only memory size of Animal is allocated, so that derived member variables and methods are all truncated.

[Inherited]

Entity is a class for standardization of expression method using on network I/O by XML. If Invoke is a standard message protocol of Samchon Framework which must be kept, Entity is a recommended semi-protocol of message for expressing a data class. Following the semi-protocol Entity is not imposed but encouraged.

As we could get advantages from standardization of message for network I/O with Invoke, we can get additional advantage from standardizing expression method of data class with Entity. We do not need to know a part of network communication. Thus, with the Entity, we can only concentrate on entity's own logics and relationships between another entities. Entity does not need to how network communications are being done.



Example source

```
#include <iostream>

#include <samchon/protocol/Entity.hpp>
#include <samchon/protocol/IHTMLEntity.hpp>
#include <samchon/protocol/SharedEntityArray.hpp>

#include <samchon/library/XML.hpp>

#ifdef _WIN64
#   ifdef _DEBUG
#       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "x64/Release/SamchonFramework.lib")
#   endif
#else
#   ifdef _DEBUG
#       pragma comment(lib, "Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "Release/SamchonFramework.lib")
#   endif
#endif

using namespace std;
using namespace samchon::library;
using namespace samchon::protocol;

```

```
class Member
    : public Entity, public virtual IHTMLEntity
{
protected:
    typedef Entity super;

    string id;
    string name;
    int age;
    int grade;

public:
    /* ----------------------------------------------------------------------
        CONSTRUCTORS
    ---------------------------------------------------------------------- */
    Member()
        : super(), IHTMLEntity()
    {
    };
    Member(const string &id, const string &name, int age, int grade)
        : super(), IHTMLEntity()
    {
        this->id = id;
        this->name = name;
        this->age = age;
        this->grade = grade;
    };
    virtual ~Member() = default;

    virtual void construct(shared_ptr<XML> xml) override
    {
        this->id = xml->getProperty("id");
        this->name = xml->getProperty("name");
        this->age = xml->getProperty<int>("age");
        this->grade = xml->getProperty<int>("grade");
    };

    /* ----------------------------------------------------------------------
        GETTERSs
    ---------------------------------------------------------------------- */
    virtual auto key() const -> std::string override
    {
        return this->id;
    };

    /* ----------------------------------------------------------------------
        XML EXPORTERS
    ---------------------------------------------------------------------- */
    virtual auto TAG() const -> string override
    {
        return "member";
    };
    virtual auto toXML() const -> shared_ptr<XML>
    {
        shared_ptr<XML> &xml = super::toXML();
        xml->setProperty("id", id);
        xml->setProperty("name", name);
        xml->setProperty("age", age);
        xml->setProperty("grade", grade);

        return move(xml);
    };
    virtual auto toHTML() const -> string
    {
        return toTR(id, name, age, grade);
    };
};

class MemberArray
    : public SharedEntityArray<Member>,
    public virtual IHTMLEntity
{
protected:
    typedef SharedEntityArray<Member> super;

    string application;
    int department;
    Member *chief;

public:
    /* ----------------------------------------------------------------------
        CONSTRUCTORS
    ---------------------------------------------------------------------- */
    MemberArray()
        : super(), IHTMLEntity()
    {
        this->chief = nullptr;
```

```
    };
    virtual ~MemberArray() = default;

    // You don't need to consider children(Member) objects
    // Just concentrate on constructing MemberArray's own member variables
    virtual void construct(shared_ptr<XML> xml) override
    {
        super::construct(xml);

        this->application = xml->getProperty("application");
        this->department = xml->getProperty<int>("department");

        if(xml->hasProperty("chief") == true && this->has( xml->getProperty("chief") ) == true)
            this->chief = this->get( xml->getProperty("cheif") ).get();
    };

protected:
    //FACTORY METHOD FOR MEMBER
    virtual auto createChild(shared_ptr<XML> = nullptr) -> Member* override
    {
        return new Member();
    };

    /* ----------------------------------------------------------------------
        XML EXPORTERS
    ---------------------------------------------------------------------- */
public:
    virtual auto TAG() const -> string override
    {
        return "memberArray";
    };
    virtual auto CHILD_TAG() const -> string override
    {
        return "member";
    };

    // You don't need to consider children(Member) objects
    // Just concentrate on expressing MemberArray's own member variables
    virtual auto toXML() const -> shared_ptr<XML>
    {
        shared_ptr<XML> &xml = super::toXML();
        xml->setProperty("application", application);
        xml->setProperty("department", department);

        if(chief != nullptr)
            xml->setProperty("cheif", chief->key());

        return move(xml);
    };
    virtual auto toHTML() const -> string
    {
        string html = "<table>\n";
        html += toTH("ID", "Name", "Age", "Grade") + "\n";

        for (size_t i = 0; i < 2; i++)
            html += at(i)->toHTML() + "\n";

        html += "</table>";
        return move(html);
    };
};

void main()
{
    string str = string("") +
        "<memberArray application='framework' department='7' cheif='samchon'>\n" +
        "    <member id='samchon' name='Jeongho Nam' age='27' grade='5' />" +
        "    <member id='submaster' name='No Name' age='100' grade='4' />" +
        "    <member id='john' name='John Doe' age='33' grade='2' />" +
        "    <member id='bad_man' name='Bad Man' age='44' grade='-1' />" +
        "    <member id='guest' name='Guest' age='0' grade='0' />" +
        "</memberArray>";
    shared_ptr<XML> xml(new XML(str));

    MemberArray memberArray;
    memberArray.construct(xml);

    memberArray.emplace_back(new Member("freshman", "A fresh man", 20, 2));
    memberArray.emplace_back(new Member("senior", "A senior", 70, 2));

    cout << memberArray.toXML()->toString() << endl << endl;
    cout << memberArray.toHTML() << endl;
    system("pause");
}
```

Result of the example

```
D:\OneDrive\Project\Samchon\framework\cpp\Debug\entity.exe
<memberArray application="framework" department="7">
        <member age="27" grade="5" id="samchon" name="Jeongho" />
        <member age="100" grade="4" id="submaster" name="No" />
        <member age="33" grade="2" id="john" name="John" />
        <member age="44" grade="-1" id="bad_man" name="Bad" />
        <member age="0" grade="0" id="guest" name="Guest" />
        <member age="20" grade="2" id="freshman" name="A fresh man" />
        <member age="70" grade="2" id="senior" name="A senior" />
</memberArray>

<table>
        <th>
                <td>ID</td>
                <td>Name</td>
                <td>Age</td>
                <td>Grade</td>
        </th>
        <tr>
                <td>samchon</td>
                <td>Jeongho</td>
                <td>27</td>
                <td>5</td>
        </tr>
        <tr>
                <td>submaster</td>
                <td>No</td>
                <td>100</td>
                <td>4</td>
        </tr>
</table>
Press any key to continue . . .
```

Note

I say repeatedly. Expression method of Entity is recommended, but not imposed. It's a semi protocol for network I/O but not a essential protocol must be kept. The expression method of Entity, using on network I/O, is expressed by XML string.

If your own network system has a critical performance issue on communication data class, it would be better to using binary communication (with ByteArray or boost::serialization). Don't worry about the problem! Invoke also provides methods for binary data (ByteArray).

See also

protocol

Author

Jeongho Nam

Warning

If a data structure has recursive and hierarchical relationship, never use the EntityList. EntityList's memory allocation is static. The recursive relationship causes infinite memory allocation. It must generates stack overflow.

Use EntityGroup instead.

- class NTCriteria : public EntityList<NTCriteria> causes a stack overflow.
- class NTCriteria : public SharedEntityList<NTCriteria> doesn't cause the stack overflow.

Warning

Unlike EntityArray containing children serialized, EntityList's static memory allocation and accessment for a child, its advantage doesn't seemed to have eminent advantage on performance. As there's not meaningful advantage but has only disadvantage that cannot containing derived objects, I'm planning to depreciate the EntityList in next generation.

Use SharedEntityList instead will be better I think.

Definition at line 52 of file EntityList.hpp.

10.37.2 Constructor & Destructor Documentation

10.37.2.1 template<typename _Ty > samchon::protocol::EntityList< _Ty >::EntityList ( )

Default Constructor.

10.37.3 Member Function Documentation

10.37.3.1 template<typename _Ty > virtual void samchon::protocol::EntityList< _Ty >::construct (std::shared_ptr< library::XML > *xml*) [inline], [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityList's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityList::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityList::toXML().

Constructs only data of EntityList's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

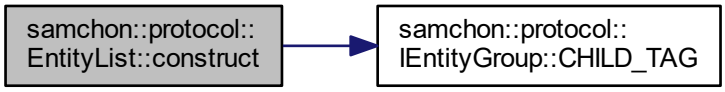
xml	An xml used to construct data of entity

Implements samchon::protocol::Entity.

Definition at line 81 of file EntityList.hpp.

References samchon::protocol::IEntityGroup::CHILD_TAG().

Here is the call graph for this function:



10.37.3.2 template<typename _Ty > virtual auto samchon::protocol::EntityList< _Ty >::toXML ( ) const -> std::shared_ptr<library::XML> [inline],[override],[virtual]

Get an XML object represents the EntityList.

Archives the EntityList's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityList::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityList::toXML().

Archives only data of EntityList's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

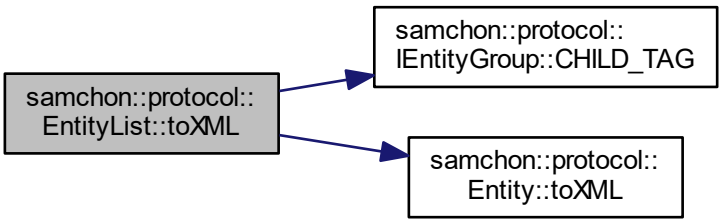
An XML object representing the Entity.

Reimplemented from samchon::protocol::Entity.

Definition at line 109 of file EntityList.hpp.

References samchon::protocol::IEntityGroup::CHILD_TAG(), and samchon::protocol::Entity::toXML().

Here is the call graph for this function:



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

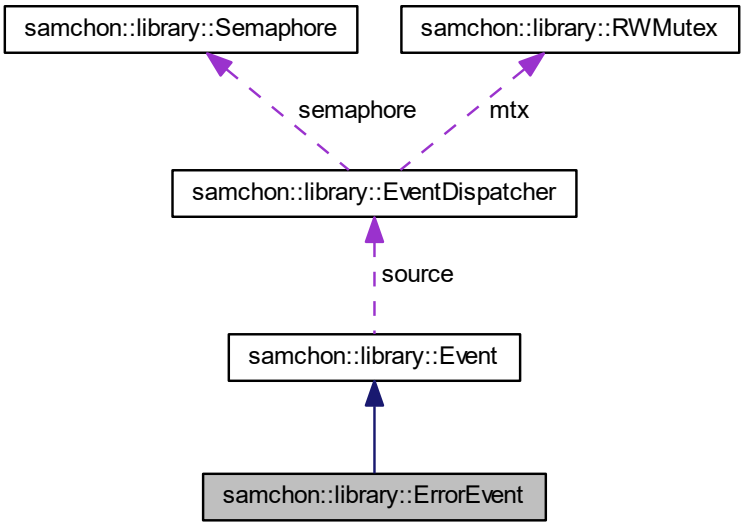
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/EntityList.hpp

10.38 samchon::library::ErrorEvent Class Reference

Event representing an error.

`#include <ErrorEvent.hpp>`

Collaboration diagram for samchon::library::ErrorEvent:



Public Member Functions

- ErrorEvent (EventDispatcher *, const std::string &)
  Construct from source and error-id.
- auto getMessage () const -> std::string
  Get error-id.

Protected Attributes

- std::string message
  Error message.

10.38.1 Detailed Description

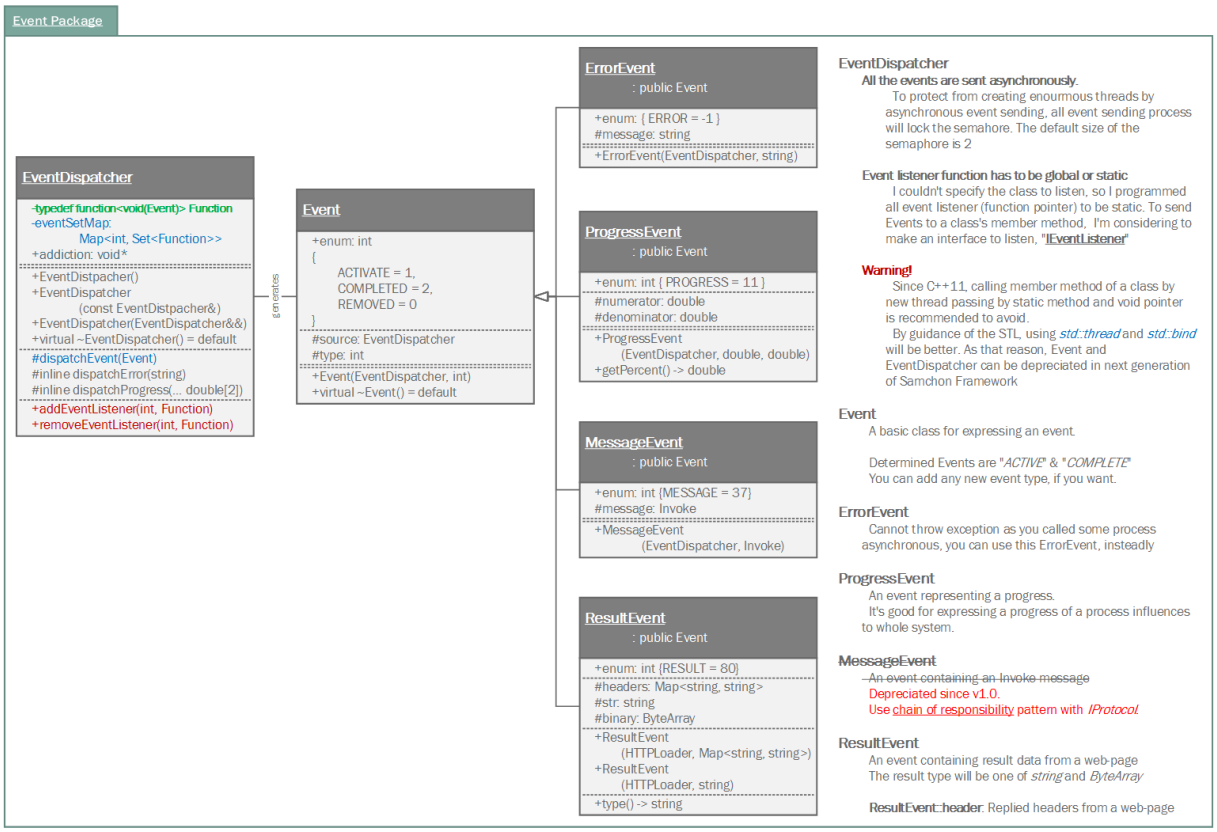
Event representing an error.

[Inherited]

The Event class is used as the base class for the creation of Event objects, which are passed as parameters to event listeners when an event occurs.

The properties of the Event class carry basic information about an event, such as the event's type or source (who made the event) of the event.

For many events, such as the events represented by the Event class constants, this basic information is sufficient. Other events, however, may require more detailed information.



Example source

```
1 #include <iostream>
2 #include <thread>
3 #include <chrono>
4
5 #include <samchon/library/EventDispatcher.hpp>
6 #include <samchon/library/Event.hpp>
7 #include <samchon/library/ProgressEvent.hpp>
8
9 #ifdef _WIN64
10 #   ifdef _DEBUG
11 #       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
12 #   else
13 #       pragma comment(lib, "x64/Release/SamchonFramework.lib")
14 #   endif
15 #else
16 #   ifdef _DEBUG
17 #       pragma comment(lib, "Debug/SamchonFramework.lib")
18 #   else
19 #       pragma comment(lib, "Release/SamchonFramework.lib")
20 #   endif
21 #endif
22
23 using namespace std;
24 using namespace samchon::library;
25
26 class Process
27     : public EventDispatcher
28 {
29 public:
30     Process()
31     {
32
33     };
34
35     void run()
36     {
37         dispatchEvent(shared_ptr<Event>(new Event(this, Event::ACTIVATE)));
38
39         for (size_t i = 0; i < 100; i++)
40         {
41             this_thread::sleep_for(chrono::seconds(1));
42             dispatchEvent(shared_ptr<Event>(new ProgressEvent(this, i + 1, 100)));
43         }
44
```

```
        dispatchEvent(shared_ptr<Event>(new Event(this, Event::COMPLETE)));
    };
};

void handleActivate(shared_ptr<Event> event)
{
    cout << "Activated" << endl;
}
void handleComplete(shared_ptr<Event> event)
{
    cout << "Completed" << endl;
}
void handleProgress(shared_ptr<Event> event)
{
    cout << dynamic_pointer_cast<ProgressEvent>(event)->getPercent() * 100.0 << "%" << endl;
}

void main()
{
    Process process;

    process.addEventListener(Event::ACTIVATE, handleActivate);
    process.addEventListener(Event::COMPLETE, handleComplete);
    process.addEventListener(ProgressEvent::PROGRESS, handleProgress);

    process.run();
    system("pause");
}
```

Deprecated

Event is a candidate to be deprecated.

Since C++11, calling member method of a class by a new thread passing by static method and using void pointer are recommeded to avoid. As the reason, using *std::thread* and *std::bind will* be better.

- std::thread: http://www.cplusplus.com/reference/thread/thread/
- std::bind: http://www.cplusplus.com/reference/functional/bind/

See also

samchon::library

Author

Jeongho Nam

Definition at line 19 of file ErrorEvent.hpp.

10.38.2 Constructor & Destructor Documentation

10.38.2.1 ErrorEvent::ErrorEvent (EventDispatcher *, const std::string &)

Construct from source and error-id.

The event object owns its source and type

Parameters

source	Source of the event; who made the event
id	An error-id

Definition at line 7 of file ErrorEvent.cpp.

References message.

10.38.3 Member Function Documentation

10.38.3.1 auto ErrorEvent::getMessage ( ) const -> std::string

Get error-id.

Definition at line 12 of file ErrorEvent.cpp.

References message.

10.38.4 Member Data Documentation

10.38.4.1 std::string samchon::library::ErrorEvent::message [protected]

Error message.

Definition at line 36 of file ErrorEvent.hpp.

Referenced by ErrorEvent(), and getMessage().

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

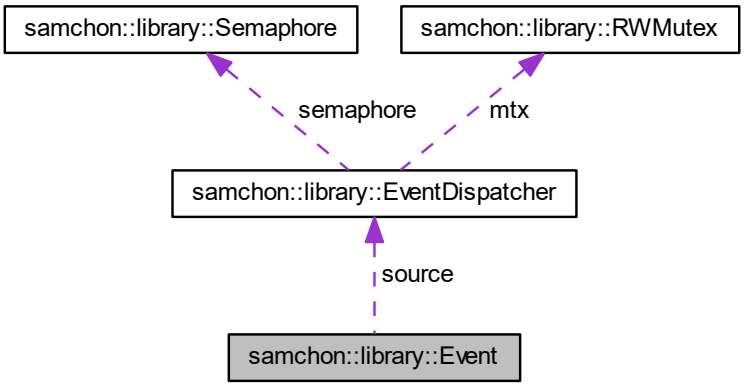
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/ErrorEvent.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/ErrorEvent.cpp

10.39 samchon::library::Event Class Reference

Represent an event running on background.

`#include <Event.hpp>`

Collaboration diagram for samchon::library::Event:



Public Member Functions

- Event (EventDispatcher *, int)

  Construct from source and type.
- auto getSource () const -> EventDispatcher *

  Get source of the Event.
- auto getType () const -> int

  Get type of the Event.

Protected Attributes

- EventDispatcher * source

  Source of the event.
- int type

  Type of the event.

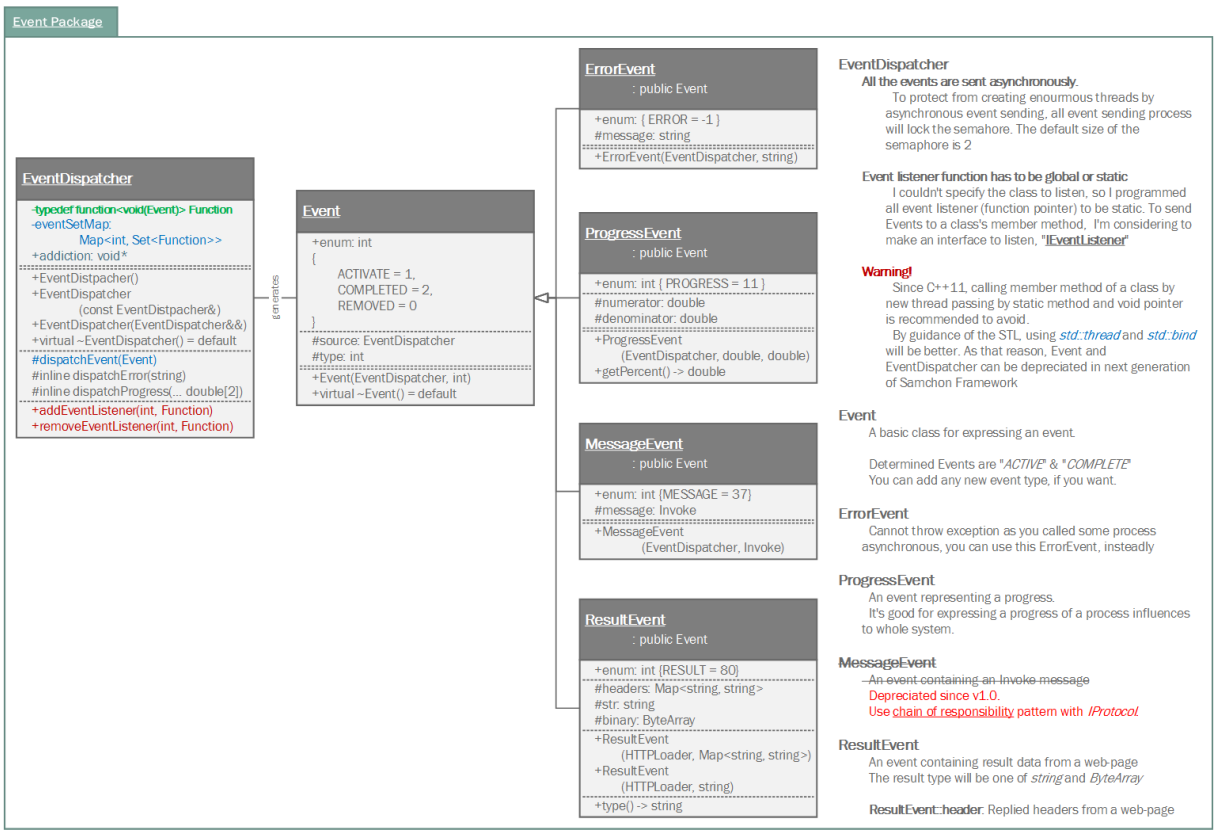
10.39.1 Detailed Description

Represent an event running on background.

The Event class is used as the base class for the creation of Event objects, which are passed as parameters to event listeners when an event occurs.

The properties of the Event class carry basic information about an event, such as the event's type or source (who made the event) of the event.

For many events, such as the events represented by the Event class constants, this basic information is sufficient. Other events, however, may require more detailed information.



Example source

```
1 #include <iostream>
2 #include <thread>
3 #include <chrono>
4
5 #include <samchon/library/EventDispatcher.hpp>
6 #include <samchon/library/Event.hpp>
7 #include <samchon/library/ProgressEvent.hpp>
8
9 #ifdef _WIN64
10 #   ifdef _DEBUG
11 #       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
12 #   else
13 #       pragma comment(lib, "x64/Release/SamchonFramework.lib")
14 #   endif
15 #else
16 #   ifdef _DEBUG
17 #       pragma comment(lib, "Debug/SamchonFramework.lib")
18 #   else
19 #       pragma comment(lib, "Release/SamchonFramework.lib")
20 #   endif
21 #endif
22
23 using namespace std;
24 using namespace samchon::library;
25
26 class Process
27     : public EventDispatcher
```

```
28 {
29 public:
30     Process()
31     {
32
33     };
34
35     void run()
36     {
37         dispatchEvent(shared_ptr<Event>(new Event(this, Event::ACTIVATE)));
38
39         for (size_t i = 0; i < 100; i++)
40         {
41             this_thread::sleep_for(chrono::seconds(1));
42             dispatchEvent(shared_ptr<Event>(new ProgressEvent(this, i + 1, 100)));
43         }
44
45         dispatchEvent(shared_ptr<Event>(new Event(this, Event::COMPLETE)));
46     };
47 };
48
49 void handleActivate(shared_ptr<Event> event)
50 {
51     cout << "Activated" << endl;
52 }
53 void handleComplete(shared_ptr<Event> event)
54 {
55     cout << "Completed" << endl;
56 }
57 void handleProgress(shared_ptr<Event> event)
58 {
59     cout << dynamic_pointer_cast<ProgressEvent>(event)->getPercent() * 100.0 << "%" << endl;
60 }
61
62 void main()
63 {
64     Process process;
65
66     process.addEventListener(Event::ACTIVATE, handleActivate);
67     process.addEventListener(Event::COMPLETE, handleComplete);
68     process.addEventListener(ProgressEvent::PROGRESS, handleProgress);
69
70     process.run();
71     system("pause");
72 }
```

Deprecated

Event is a candidate to be deprecated.

Since C++11, calling member method of a class by a new thread passing by static method and using void pointer are recommeded to avoid. As the reason, using *std::thread* and *std::bind will* be better.

- std::thread: http://www.cplusplus.com/reference/thread/thread/
- std::bind: http://www.cplusplus.com/reference/functional/bind/

See also

samchon::library

Author

Jeongho Nam

Definition at line 42 of file Event.hpp.

10.39.2 Constructor & Destructor Documentation

10.39.2.1 Event::Event (EventDispatcher * *source*, int *type*)

Construct from source and type.

The event object owns its source and type

Parameters

source	Source of the event; who made the event
type	Type of the event

Definition at line 5 of file Event.cpp.

References source, and type.

10.39.3 Member Function Documentation

10.39.3.1 auto Event::getSource ( ) const -> EventDispatcher*

Get source of the Event.

Definition at line 11 of file Event.cpp.

References source.

10.39.3.2 auto Event::getType ( ) const -> int

Get type of the Event.

Definition at line 15 of file Event.cpp.

References type.

10.39.4 Member Data Documentation

10.39.4.1 EventDispatcher* samchon::library::Event::source [protected]

Source of the event.

Definition at line 56 of file Event.hpp.

Referenced by Event(), and getSource().

10.39.4.2 int samchon::library::Event::type [protected]

Type of the event.

Definition at line 61 of file Event.hpp.

Referenced by Event(), and getType().

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

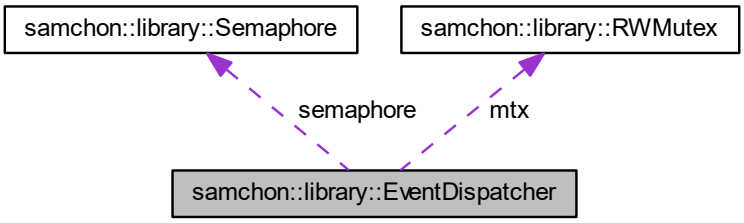
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/Event.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/Event.cpp

10.40 samchon::library::EventDispatcher Class Reference

Abstract class for dispatching Event.

```
#include <EventDispatcher.hpp>
```

Collaboration diagram for samchon::library::EventDispatcher:



Public Member Functions

- EventDispatcher ()
  Default Constructor.
- EventDispatcher (const EventDispatcher &eventDispatcher)
  Copy Constructor.
- EventDispatcher (EventDispatcher &&eventDispatcher)
  Move Constructor.
- void addEventListener (int, void(∗listener)(std::shared_ptr< Event >))
  Register an event listener.
- void removeEventListener (int, void(∗listener)(std::shared_ptr< Event >))
  Remove a registered event listener.

Protected Member Functions

- auto dispatchEvent (std::shared_ptr< Event >) -> bool
  Dispatches an event to all listeners.
- auto dispatchProgressEvent (size_t x, size_t size) -> bool
  Convenient method of dispatching a progress event.

Private Attributes

- std::map< int, std::set< void(∗)(std::shared_ptr< Event >)> > eventSetMap
  A container storing listeners.
- RWMutex mtx
  A rw_mutex for concurrency.
- Semaphore semaphore
  A semaphore for restricting thread size.

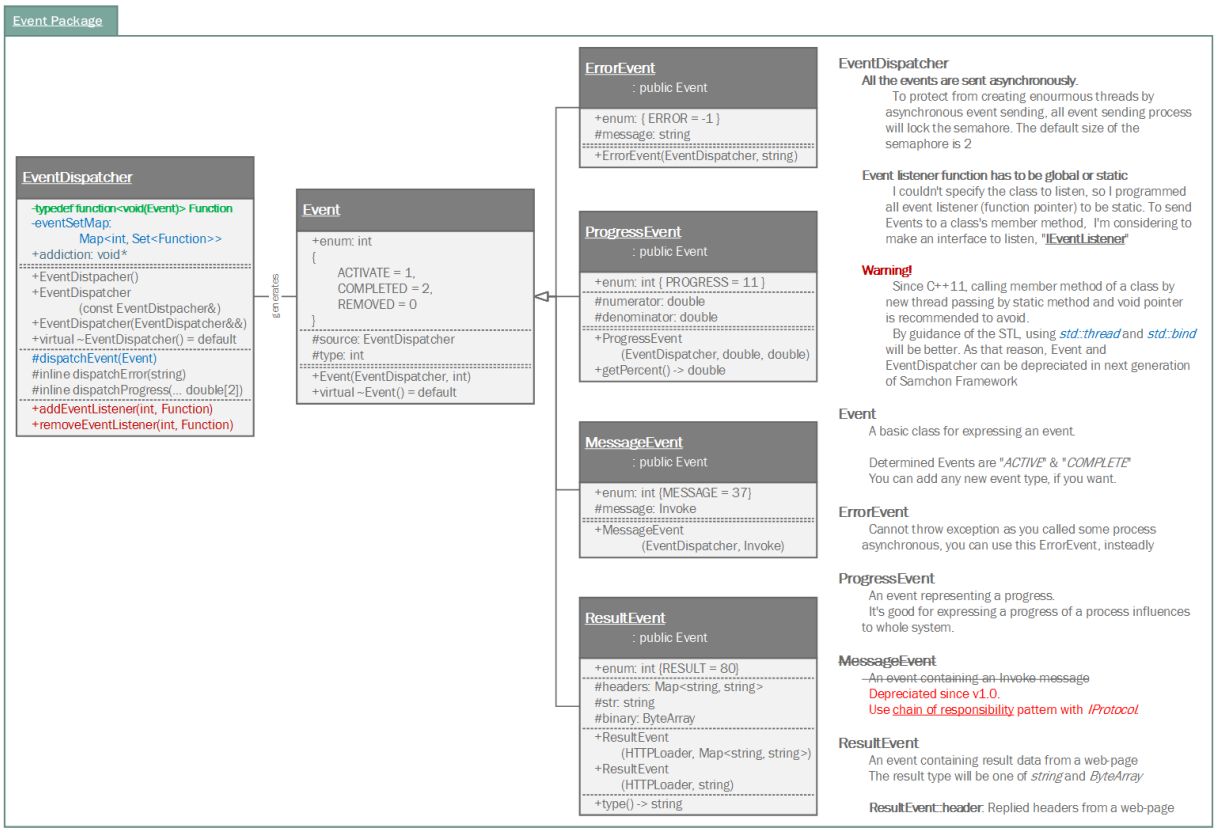
10.40.1 Detailed Description

Abstract class for dispatching Event.

EventDispatcher is the base class for all classes that dispatch events.

All the events are sent asynchronously. To avoid from creating tooo enourmouse threads dispatching events, all event sending processes will acuiqre a semaphore. The default permitted size of the semaphore is 2.

- Number of thread pools used to sending events is 2.



Example source

```
#include <iostream>
#include <thread>
#include <chrono>

#include <samchon/library/EventDispatcher.hpp>
#include <samchon/library/Event.hpp>
#include <samchon/library/ProgressEvent.hpp>

#ifdef _WIN64
#   ifdef _DEBUG
#       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "x64/Release/SamchonFramework.lib")
#   endif
#else
#   ifdef _DEBUG
#       pragma comment(lib, "Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "Release/SamchonFramework.lib")
#   endif
#endif

using namespace std;
using namespace samchon::library;

class Process
    : public EventDispatcher
{
public:
    Process()
    {

    };

    void run()
    {
        dispatchEvent(shared_ptr<Event>(new Event(this, Event::ACTIVATE)));

        for (size_t i = 0; i < 100; i++)
        {
            this_thread::sleep_for(chrono::seconds(1));
            dispatchEvent(shared_ptr<Event>(new ProgressEvent(this, i + 1, 100)))
    ;
        }
```

```

        dispatchEvent(shared_ptr<Event>(new Event(this, Event::COMPLETE)));
    };
};

void handleActivate(shared_ptr<Event> event)
{
    cout << "Activated" << endl;
}
void handleComplete(shared_ptr<Event> event)
{
    cout << "Completed" << endl;
}
void handleProgress(shared_ptr<Event> event)
{
    cout << dynamic_pointer_cast<ProgressEvent>(event)->getPercent() * 100.0 << "%" << endl;
}

void main()
{
    Process process;

    process.addEventListener(Event::ACTIVATE, handleActivate);
    process.addEventListener(Event::COMPLETE, handleComplete);
    process.addEventListener(ProgressEvent::PROGRESS, handleProgress);

    process.run();
    system("pause");
}
```

Deprecated

EventDispatcher is a candidate to be deprecated.

Since C++11, calling member method of a class by a new thread passing by static method and using void pointer are recommeded to avoid. As the reason, using *std::thread* and *std::bind* will be better.

- std::thread: http://www.cplusplus.com/reference/thread/thread/
- std::bind: http://www.cplusplus.com/reference/functional/bind/

Todo

Adjust new optimal size of semaphore representing size of backgrounds' thread.

Find another way to adding listeners of member method without using void pointer. About the problem, pull request from a forked repository is planned to come.

Test

Change listeners (function pointer) to have a new parameter, void pointer.

Bug

When EventDispatcher is deleted, error on sending events are occured.

See also

samchon::library

Author

Jeongho Nam

Definition at line 59 of file EventDispatcher.hpp.

10.40.2 Constructor & Destructor Documentation

10.40.2.1 EventDispatcher::EventDispatcher ( )

Default Constructor.

Definition at line 24 of file EventDispatcher.cpp.

10.40.2.2 EventDispatcher::EventDispatcher (const EventDispatcher & *eventDispatcher*)

Copy Constructor.

Copying an EventDispatcher instance does not copy the event listeners attached to it. (If your newly created node needs an event listener, you must attach the listener after creating the node.)

Parameters

eventDispatcher	The object to copy

Definition at line 27 of file EventDispatcher.cpp.

10.40.2.3 EventDispatcher::EventDispatcher (EventDispatcher && *eventDispatcher*)

Move Constructor.

Parameters

eventDispatcher	The object to move

Definition at line 31 of file EventDispatcher.cpp.

10.40.3 Member Function Documentation

10.40.3.1 void EventDispatcher::addEventListener (int *type*, void(∗)(std::shared_ptr< Event >) *listener*)

Register an event listener.

Registers an event listener object with an EventDispatcher object so that the listener receives notification of an event.

Warning

> Copying an EventDispatcher instance does not copy the event listeners attached to it. (If your newly created node needs an event listener, you must attach the listener after creating the node.) However, if you move an EventDispatcher instance, the event listeners attached to it move along with it.
> If you no longer need an event listener, remove it by calling removeEventListener, or EventDispatcher already try to send events to the no longer needed listener and it can cause some confliction.

Parameters

type	The type of event.
listener	The listener function processes the event.

Definition at line 39 of file EventDispatcher.cpp.

10.40.3.2 void EventDispatcher::removeEventListener (int *type*, void(∗)(std::shared_ptr< Event >) *listener*)

Remove a registered event listener.

Removes a listener from the EventDispatcher object. If there is no matching listener registered with the EventDispatcher object, a call to this method has no effect

Parameters

type	The type of event.
listener	The listener function to remove.

Definition at line 46 of file EventDispatcher.cpp.

10.40.3.3 auto EventDispatcher::dispatchEvent (std::shared_ptr< Event >) -> bool [protected]

Dispatches an event to all listeners.

Dispatches an event into the event flow in the background. The Event::source is the EventDispatcher object upon which the dispatchEvent.

Parameters

event	The Event object that is dispatched into the event flow.

Returns

Whether there's some listener to listen the event

Definition at line 57 of file EventDispatcher.cpp.

10.40.3.4 auto EventDispatcher::dispatchProgressEvent (size_t *x*, size_t *size*) -> bool [protected]

Convenient method of dispatching a progress event.

Dispatches a progress event into the event flow in the background The Event::source is the EventDispatcher object upon with the dispatchProgressEvent

Parameters

x	The number of current progress
size	The number of total progress

Returns

Whether there's some listener to listen the progress event

See also

ProgressEvent
EventDispatcher::dispatchEvent

Definition at line 77 of file EventDispatcher.cpp.

10.40.4 Member Data Documentation

10.40.4.1 std::map<int, std::set<void(*)(std::shared_ptr<Event>)> > samchon::library::EventDispatcher::eventSetMap [private]

A container storing listeners.

Definition at line 66 of file EventDispatcher.hpp.

10.40.4.2 RWMutex samchon::library::EventDispatcher::mtx [private]

A rw_mutex for concurrency.

Definition at line 71 of file EventDispatcher.hpp.

10.40.4.3 Semaphore samchon::library::EventDispatcher::semaphore [private]

A semaphore for restricting thread size.

Definition at line 76 of file EventDispatcher.hpp.

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

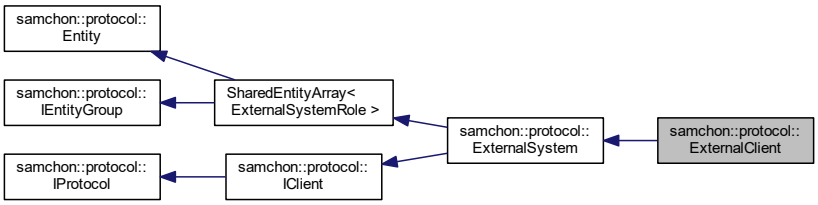
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/EventDispatcher.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/EventDispatcher.cpp

10.41 samchon::protocol::ExternalClient Class Reference

A network driver for an external client.

```
#include <ExternalClient.hpp>
```

Collaboration diagram for samchon::protocol::ExternalClient:



Public Member Functions

- ExternalClient ()

  Default Constructor.
- virtual void start () override

  Start interaction.

Additional Inherited Members

10.41.1 Detailed Description

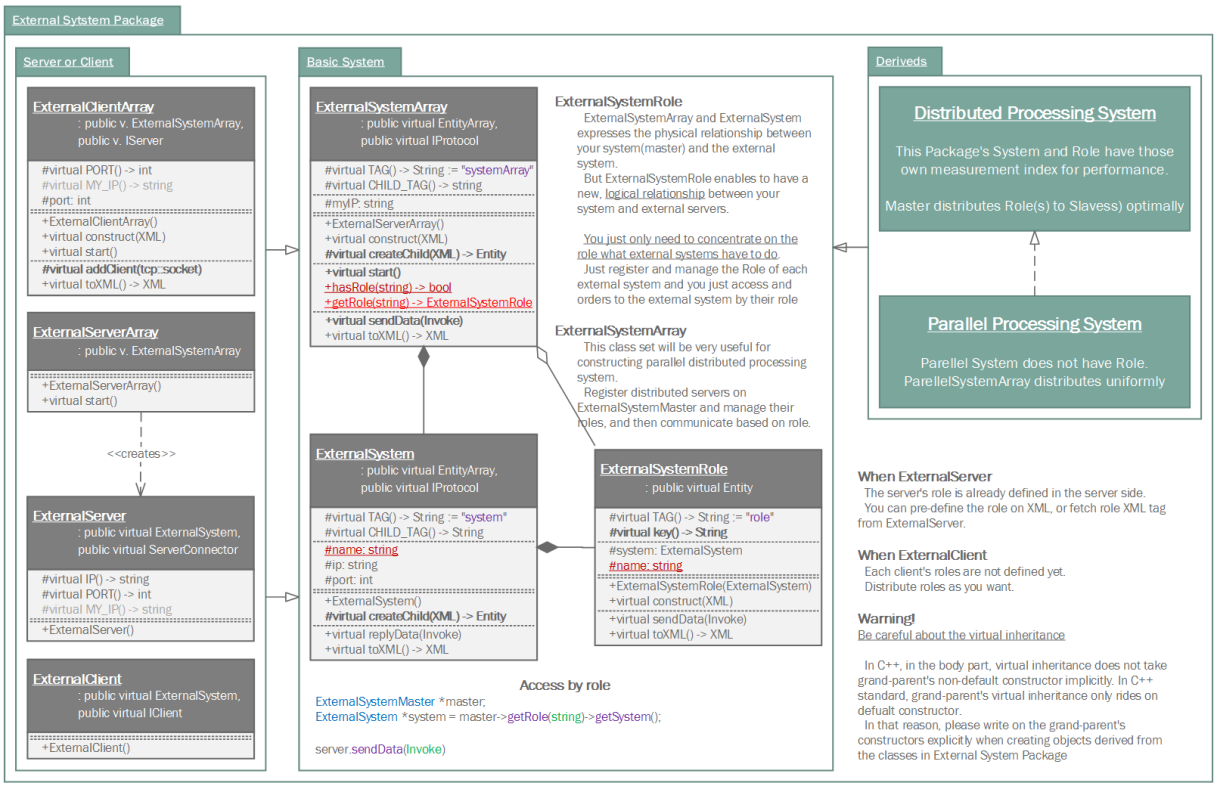
A network driver for an external client.

ExternalClient is an ExternalSystem specialized in client driver.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, ExternalSystem is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystem↩ Role.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an External↩ SystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an External↩ System. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 22 of file ExternalClient.hpp.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 ExternalClient::ExternalClient ( )

Default Constructor.

Definition at line 13 of file ExternalClient.cpp.

10.41.3 Member Function Documentation

10.41.3.1 void ExternalClient::start ( ) [override],[virtual]

Start interaction.

An abstract method starting interaction with an external system.

If an external systems are a server, starts connection and listening Inovoke message, else clients, just starts listening only. You also can addict your own procudures of starting the driver, but if you directly override method of abstract ExternalSystem, be careful about virtual inheritance.

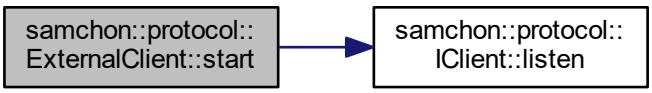
Implements samchon::protocol::ExternalSystem.

Definition at line 17 of file ExternalClient.cpp.

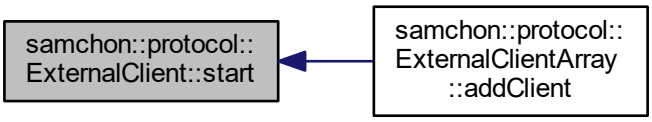
References samchon::protocol::IClient::listen().

Referenced by samchon::protocol::ExternalClientArray::addClient().

Here is the call graph for this function:



Here is the caller graph for this function:



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

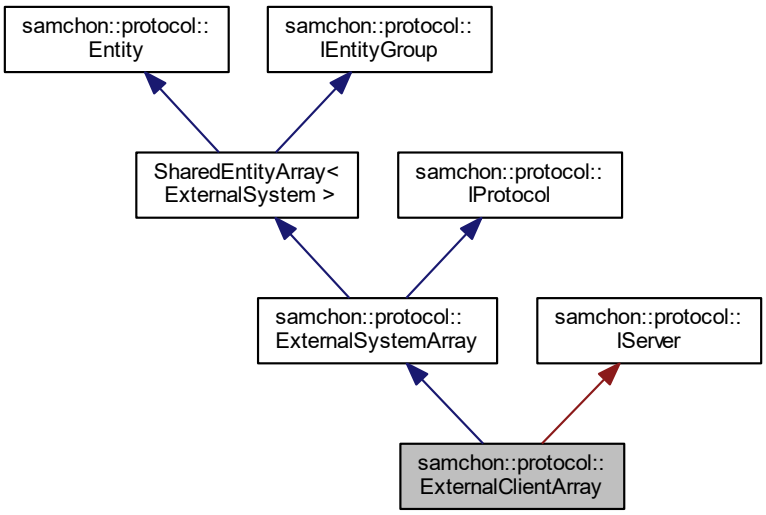
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalClient.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalClient.cpp

10.42 samchon::protocol::ExternalClientArray Class Reference

An array of ExternalClient(s)

```
#include <ExternalClientArray.hpp>
```

Collaboration diagram for samchon::protocol::ExternalClientArray:



Public Member Functions

- ExternalClientArray ()
  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override
  Construct data of the Entity from an XML object.
- virtual void start () override
  Start interaction.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override
  Get an XML object represents the EntityGroup.

Protected Member Functions

- virtual auto PORT () const -> int
  Port number of the server.
- virtual auto MY_IP () const -> std::string
  (optional) Server's IP
- virtual void addClient (Socket *) override
  Handling connection of a physical client.

Protected Attributes

- std::string myIP
  A custom ip address of my system to bind.
- int port
  A port number for accepting external clients.

Additional Inherited Members

10.42.1 Detailed Description

An array of ExternalClient(s)

ExternalClientArray is an ExternalSystemArray specialized in clients' driver.

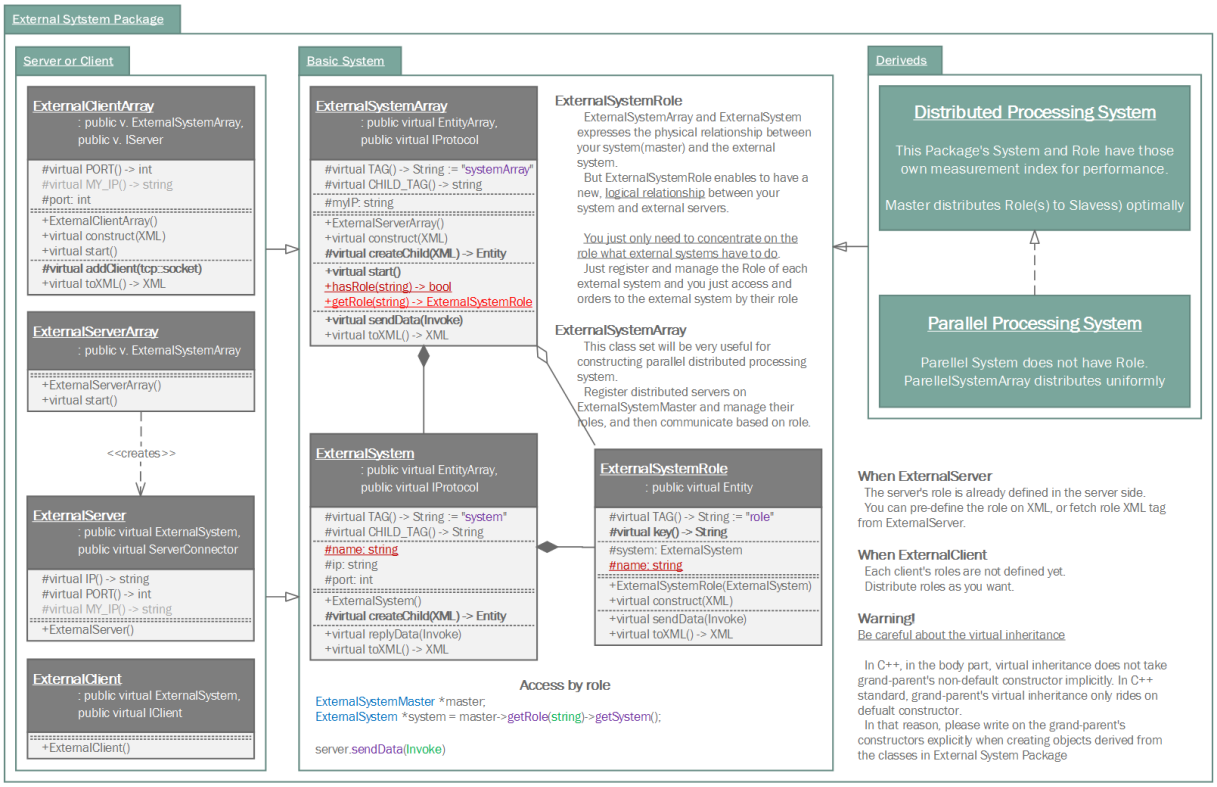
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

- ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 22 of file ExternalClientArray.hpp.

10.42.2 Constructor & Destructor Documentation

10.42.2.1 ExternalClientArray::ExternalClientArray ( )

Default Constructor.

Definition at line 16 of file ExternalClientArray.cpp.

10.42.3 Member Function Documentation

10.42.3.1 void ExternalClientArray::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Reimplemented from samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

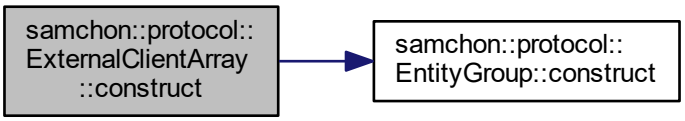
Reimplemented in samchon::protocol::master::DistributedClientArrayMediator, samchon::protocol::master::ParallelClientArrayMediator, and samchon::protocol::master::DistributedClientArray.

Definition at line 21 of file ExternalClientArray.cpp.

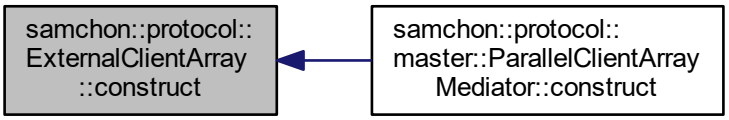
References samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::construct(), myIP, and port.

Referenced by samchon::protocol::master::ParallelClientArrayMediator::construct().

Here is the call graph for this function:



Here is the caller graph for this function:



10.42.3.2 void ExternalClientArray::start ( ) [override], [virtual]

Start interaction.

An abstract method starting interaction with external systems.

If external systems are servers, starts connection to them, else clients, opens a server and accepts the external systems. You can addict your own procudures of starting drivers, but if you directly override method of abstract ExternalSystemArray, be careful about virtual inheritance.

- DistributedClientArray::start() calls DistributedSystemArray::start() and ExternalClientArray:start()

Implements samchon::protocol::ExternalSystemArray.

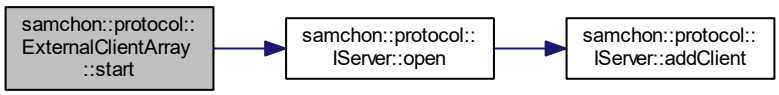
Reimplemented in samchon::protocol::master::DistributedClientArrayMediator, samchon::protocol::master↩ ::ParallelClientArrayMediator, and samchon::protocol::master::DistributedClientArray.

Definition at line 33 of file ExternalClientArray.cpp.

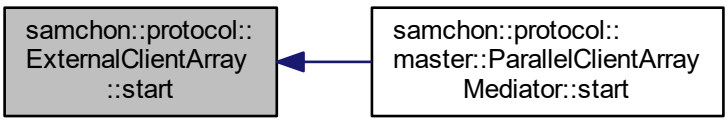
References samchon::protocol::IServer::open().

Referenced by samchon::protocol::master::ParallelClientArrayMediator::start().

Here is the call graph for this function:



Here is the caller graph for this function:



10.42.3.3 auto ExternalClientArray::PORT ( ) const -> int [protected], [virtual]

Port number of the server.

Reimplemented from samchon::protocol::IServer.

Definition at line 41 of file ExternalClientArray.cpp.

References port.

10.42.3.4 auto ExternalClientArray::MY_IP ( ) const -> std::string [protected], [virtual]

(optional) Server's IP

Reimplemented from samchon::protocol::IServer.

Definition at line 45 of file ExternalClientArray.cpp.

References myIP.

10.42.3.5 void ExternalClientArray::addClient (Socket *) [override],[protected],[virtual]

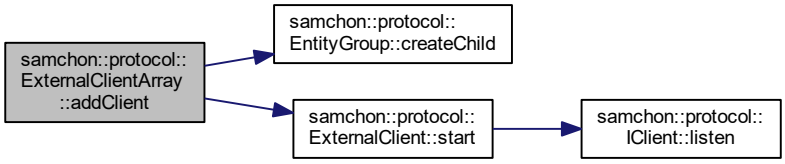
Handling connection of a physical client.

Implements samchon::protocol::IServer.

Definition at line 50 of file ExternalClientArray.cpp.

References samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::createChild(), samchon::protocol↩ ::ExternalSystem::ip, port, samchon::protocol::ExternalSystem::port, samchon::protocol::IClient::socket, and samchon::protocol::ExternalClient::start().

Here is the call graph for this function:



10.42.3.6 auto ExternalClientArray::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

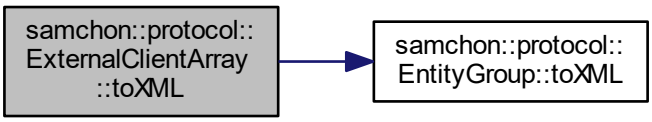
Reimplemented in samchon::protocol::master::DistributedClientArrayMediator, samchon::protocol::master::ParallelClientArrayMediator, and samchon::protocol::master::DistributedClientArray.

Definition at line 72 of file ExternalClientArray.cpp.

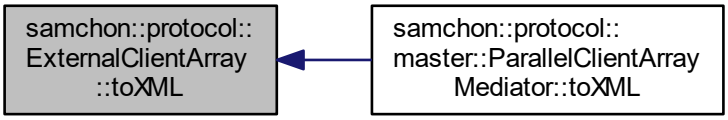
References myIP, and samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::toXML().

Referenced by samchon::protocol::master::ParallelClientArrayMediator::toXML().

Here is the call graph for this function:



Here is the caller graph for this function:



10.42.4 Member Data Documentation

10.42.4.1 std::string samchon::protocol::ExternalClientArray::myIP [protected]

A custom ip address of my system to bind.

Definition at line 32 of file ExternalClientArray.hpp.

Referenced by construct(), MY_IP(), and toXML().

10.42.4.2 int samchon::protocol::ExternalClientArray::port [protected]

A port number for accepting external clients.

Definition at line 37 of file ExternalClientArray.hpp.

Referenced by addClient(), construct(), and PORT().

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

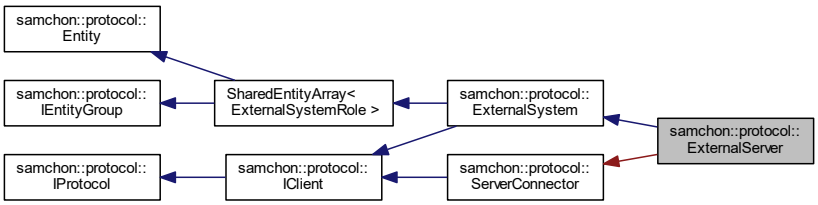
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalClientArray.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalClientArray.cpp

10.43 samchon::protocol::ExternalServer Class Reference

A network driver for an external server.

`#include <ExternalServer.hpp>`

Collaboration diagram for samchon::protocol::ExternalServer:



Public Member Functions

- ExternalServer ()
  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override
  Construct data of the Entity from an XML object.
- virtual void start () override
  Start interaction.
- virtual auto getIP () const -> std::string override
  Destinatio IP.
- virtual auto getPort () const -> int override
  Destination port.
- virtual auto getMyIP () const -> std::string override
  (optional) My IP, if you want to bind
- virtual auto toXML () const -> std::shared_ptr< library::XML > override
  Get an XML object represents the EntityGroup.

Protected Attributes

- std::string myIP
  A custom ip address of my system to bind.

Additional Inherited Members

10.43.1 Detailed Description

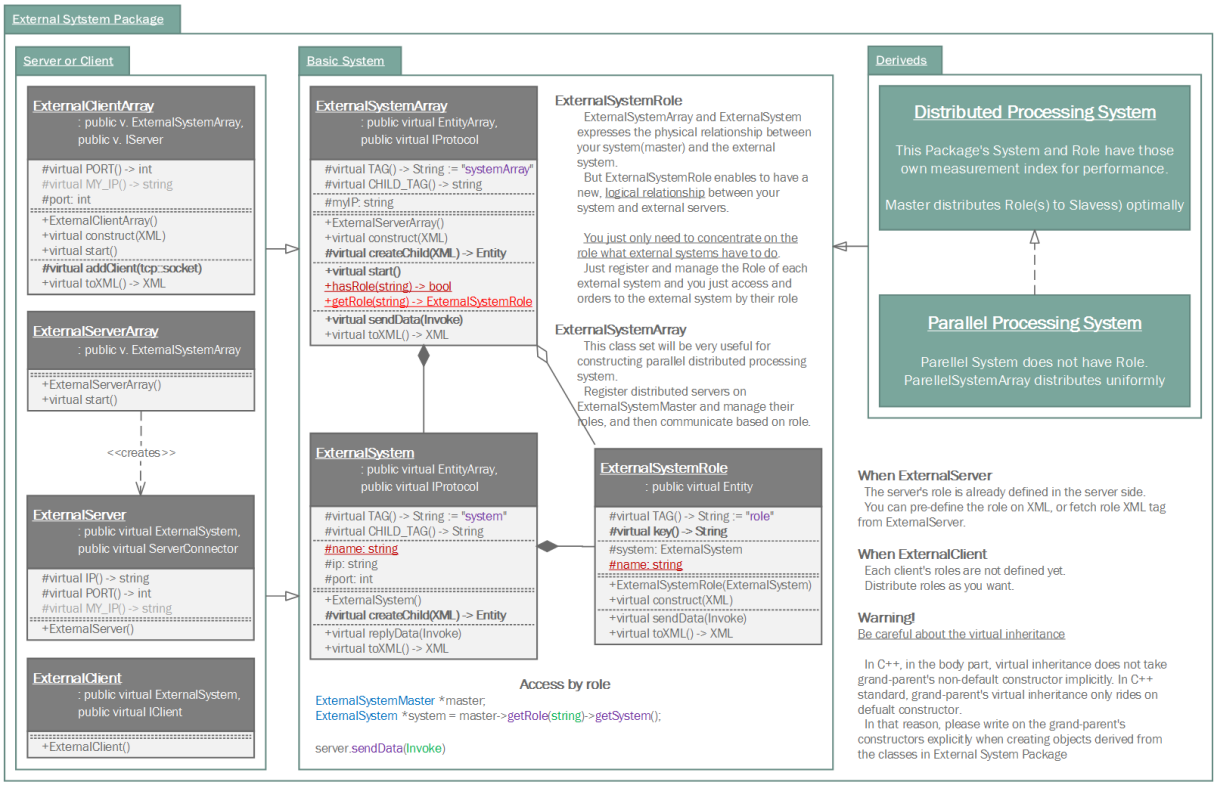
A network driver for an external server.

ExternalServer is an ExternalSystem specialized in server driver.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, ExternalSystem is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystem↩ Role.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an External↩ SystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an External↩ System. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 20 of file ExternalServer.hpp.

10.43.2 Constructor & Destructor Documentation

10.43.2.1 ExternalServer::ExternalServer ( )

Default Constructor.

Definition at line 13 of file ExternalServer.cpp.

10.43.3 Member Function Documentation

10.43.3.1 void ExternalServer::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

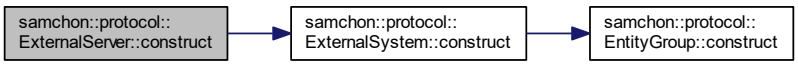
Reimplemented from samchon::protocol::ExternalSystem.

Reimplemented in samchon::protocol::master::DistributedServer, and samchon::protocol::master::ParallelServer.

Definition at line 18 of file ExternalServer.cpp.

References samchon::protocol::ExternalSystem::construct(), and myIP.

Here is the call graph for this function:



10.43.3.2 void ExternalServer::start ( ) [override], [virtual]

Start interaction.

An abstract method starting interaction with an external system.

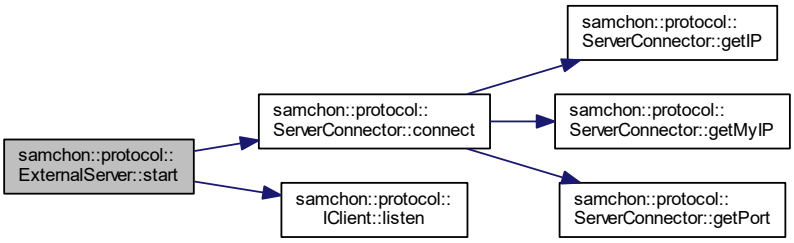
If an external systems are a server, starts connection and listening Inovoke message, else clients, just starts listening only. You also can addict your own procudures of starting the driver, but if you directly override method of abstract ExternalSystem, be careful about virtual inheritance.

Implements samchon::protocol::ExternalSystem.

Definition at line 28 of file ExternalServer.cpp.

References samchon::protocol::ServerConnector::connect(), and samchon::protocol::IClient::listen().

Here is the call graph for this function:



10.43.3.3 auto ExternalServer::getIP ( ) const -> std::string [override], [virtual]

Destinatio IP.

Implements samchon::protocol::ServerConnector.

Definition at line 38 of file ExternalServer.cpp.

References samchon::protocol::ExternalSystem::ip.

10.43.3.4 auto ExternalServer::getPort ( ) const -> int [override], [virtual]

Destination port.

Implements samchon::protocol::ServerConnector.

Definition at line 42 of file ExternalServer.cpp.

References samchon::protocol::ExternalSystem::port.

10.43.3.5 auto ExternalServer::getMyIP ( ) const -> std::string [override], [virtual]

(optional) My IP, if you want to bind

Reimplemented from samchon::protocol::ServerConnector.

Definition at line 47 of file ExternalServer.cpp.

References myIP.

10.43.3.6 auto ExternalServer::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

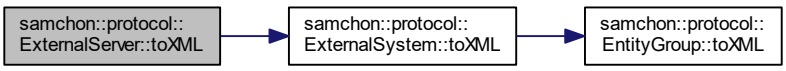
Reimplemented from samchon::protocol::ExternalSystem.

Reimplemented in samchon::protocol::master::DistributedServer, and samchon::protocol::master::ParallelServer.

Definition at line 55 of file ExternalServer.cpp.

References myIP, and samchon::protocol::ExternalSystem::toXML().

Here is the call graph for this function:



10.43.4 Member Data Documentation

10.43.4.1 std::string samchon::protocol::ExternalServer::myIP [protected]

A custom ip address of my system to bind.

Definition at line 31 of file ExternalServer.hpp.

Referenced by construct(), getMyIP(), and toXML().

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

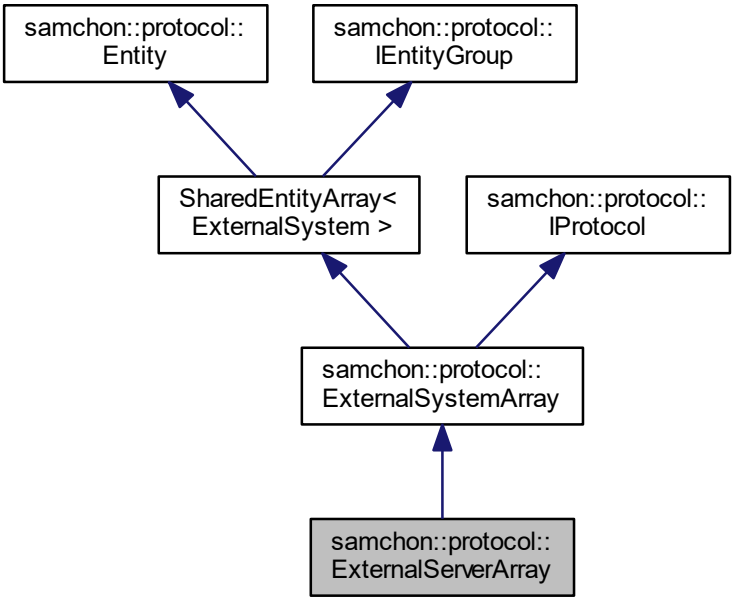
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalServer.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalServer.cpp

10.44 samchon::protocol::ExternalServerArray Class Reference

An array of ExternalServer(s)

```
#include <ExternalServerArray.hpp>
```

Collaboration diagram for samchon::protocol::ExternalServerArray:



Public Member Functions

- virtual void start () override

  Start interaction.

Additional Inherited Members

10.44.1 Detailed Description

An array of ExternalServer(s)

ExternalServerArray is an ExternalSystemArray specialized in servers' driver.

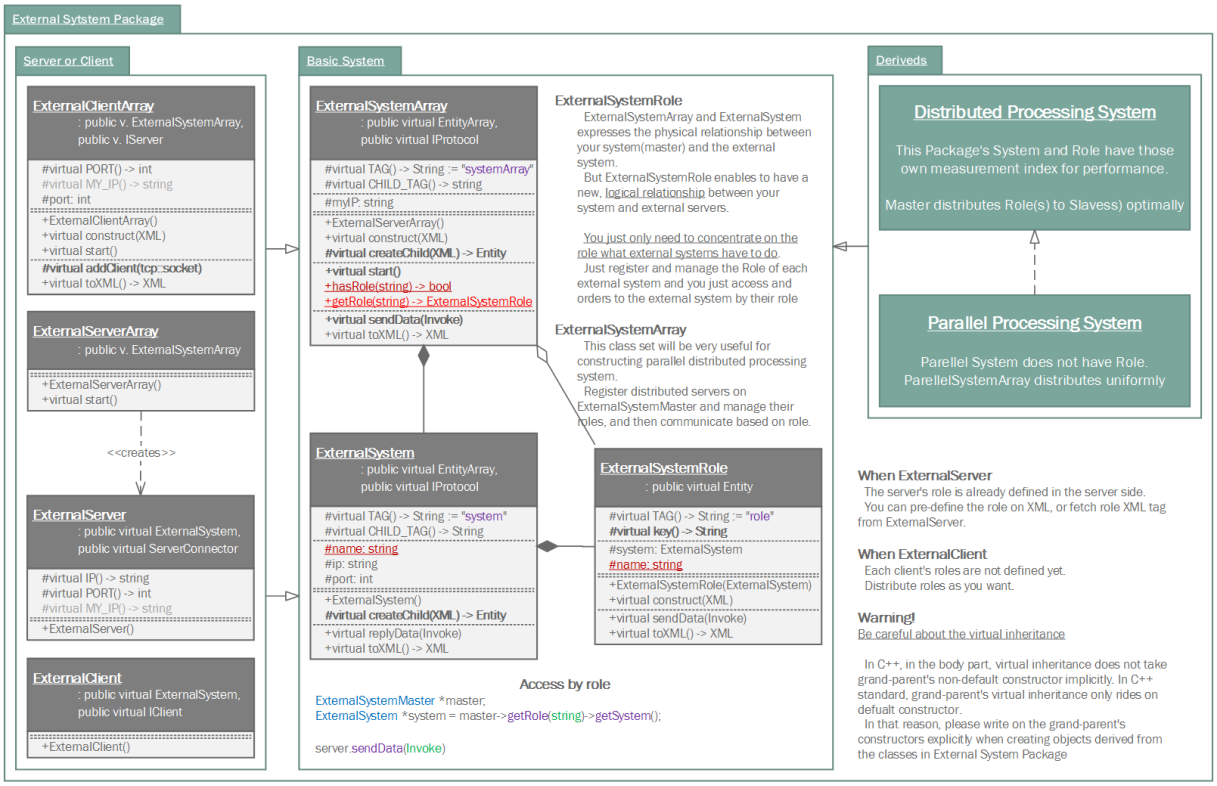
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

- ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 21 of file ExternalServerArray.hpp.

10.44.2 Member Function Documentation

10.44.2.1 void ExternalServerArray::start ( ) [override], [virtual]

Start interaction.

An abstract method starting interaction with external systems.

If external systems are servers, starts connection to them, else clients, opens a server and accepts the external systems. You can addict your own procudures of starting drivers, but if you directly override method of abstract ExternalSystemArray, be careful about virtual inheritance.

- DistributedClientArray::start() calls DistributedSystemArray::start() and ExternalClientArray:start()

Implements samchon::protocol::ExternalSystemArray.

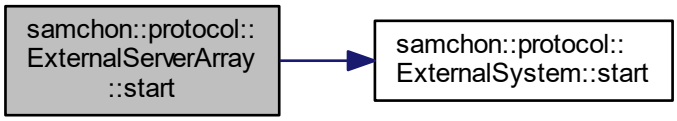
Reimplemented in samchon::protocol::master::DistributedServerArrayMediator, samchon::protocol::master↩ ::ParallelServerArrayMediator, and samchon::protocol::master::DistributedServerArray.

Definition at line 15 of file ExternalServerArray.cpp.

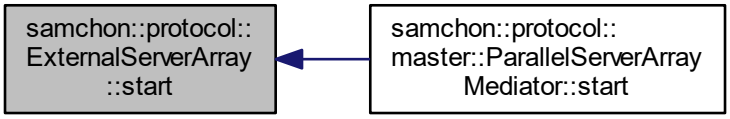
References samchon::protocol::ExternalSystem::start().

Referenced by samchon::protocol::master::ParallelServerArrayMediator::start().

Here is the call graph for this function:



Here is the caller graph for this function:



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

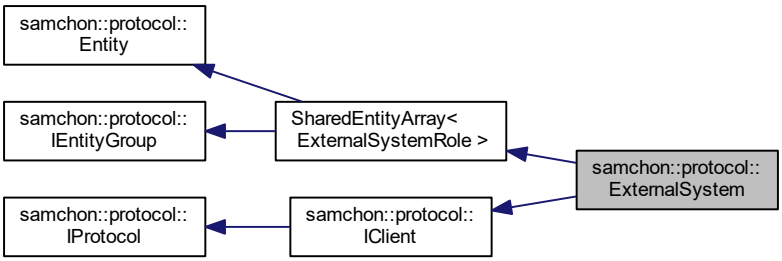
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalServerArray.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalServerArray.cpp

10.45 samchon::protocol::ExternalSystem Class Reference

A network driver for an external system.

```
#include <ExternalSystem.hpp>
```

Collaboration diagram for samchon::protocol::ExternalSystem:



Public Member Functions

- ExternalSystem ()

  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override

  Construct data of the Entity from an XML object.
- virtual void start ()=0

  Start interaction.
- virtual auto key () const -> std::string override

  Get a key that can identify the Entity uniquely.
- virtual void replyData (std::shared_ptr< Invoke >) override

  Handling replied message from an external system.
- virtual auto TAG () const -> std::string override

  A tag name when represented by XML.
- virtual auto CHILD_TAG () const -> std::string override

  A tag name of children.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the EntityGroup.

Protected Attributes

- std::string name

  A name can identify an external system.
- std::string ip

  An ip address of an external system.
- int port

  A port number of an external system.

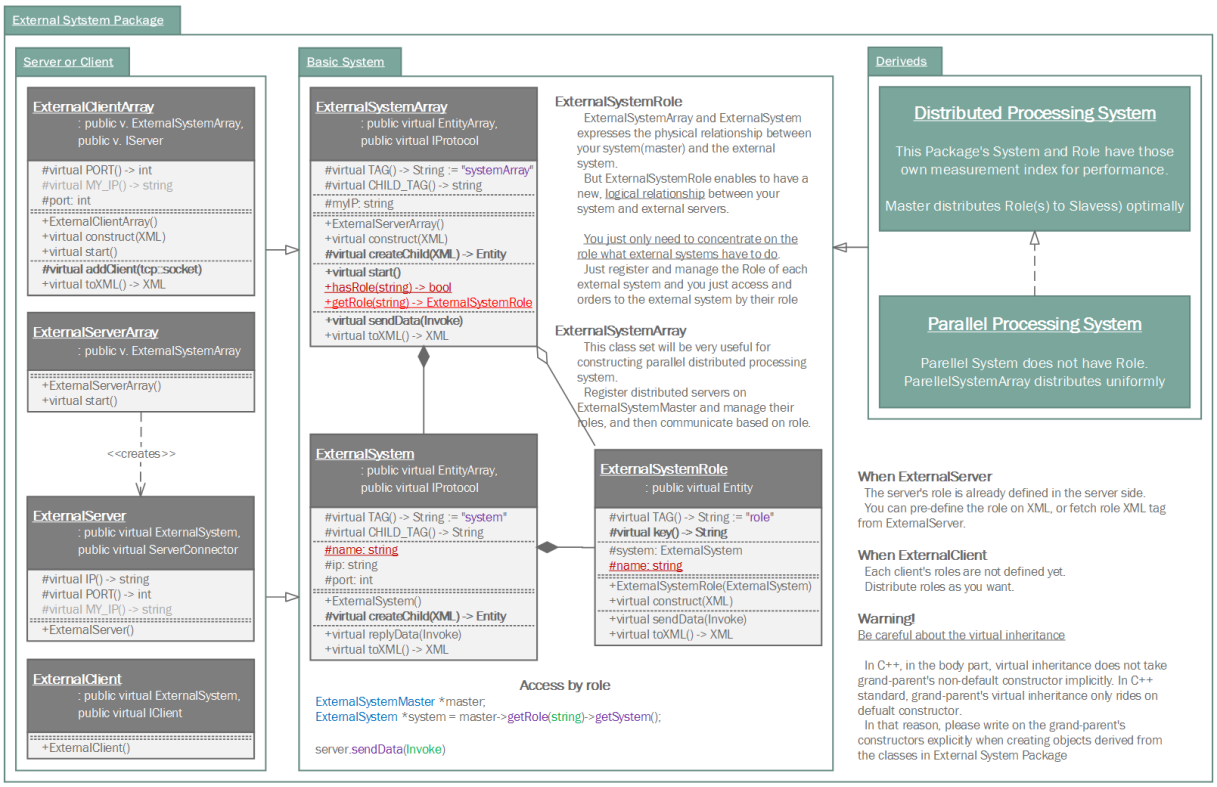
Additional Inherited Members

10.45.1 Detailed Description

A network driver for an external system.

ExternalSystem is a boundary class interacting with an external system by network communication. Also, ExternalSystem is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystem↩ Role.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an External↩ SystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an External↩ System. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 46 of file ExternalSystem.hpp.

10.45.2 Constructor & Destructor Documentation

10.45.2.1 ExternalSystem::ExternalSystem ( )

Default Constructor.

Definition at line 15 of file ExternalSystem.cpp.

10.45.3 Member Function Documentation

10.45.3.1 void ExternalSystem::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Reimplemented from samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

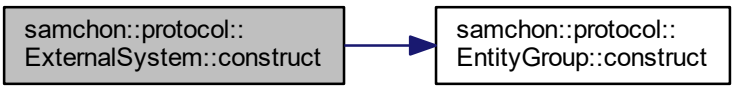
Reimplemented in samchon::protocol::master::DistributedSystem, samchon::protocol::master::ParallelSystem, samchon::protocol::ExternalServer, samchon::protocol::master::DistributedServer, and samchon::protocol::master::ParallelServer.

Definition at line 20 of file ExternalSystem.cpp.

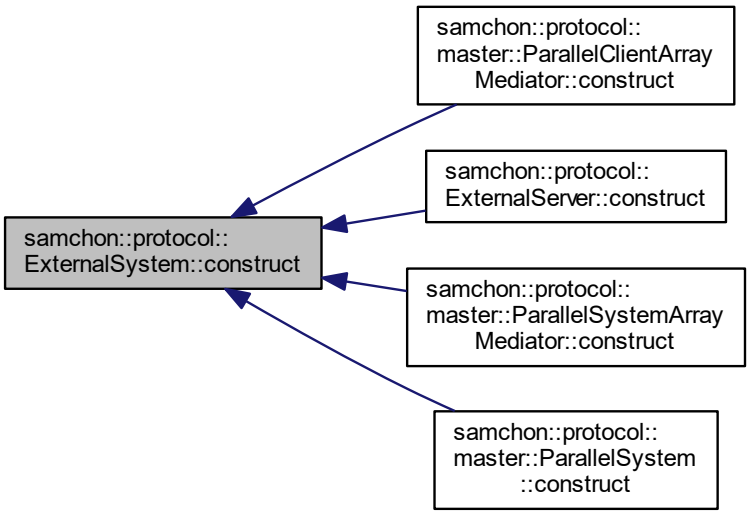
References samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::construct(), ip, name, and port.

Referenced by samchon::protocol::master::ParallelClientArrayMediator::construct(), samchon::protocol::ExternalServer::construct(), samchon::protocol::master::ParallelSystemArrayMediator::construct(), and samchon::protocol::master::ParallelSystem::construct().

Here is the call graph for this function:



Here is the caller graph for this function:



10.45.3.2 virtual void samchon::protocol::ExternalSystem::start ( ) [pure virtual]

Start interaction.

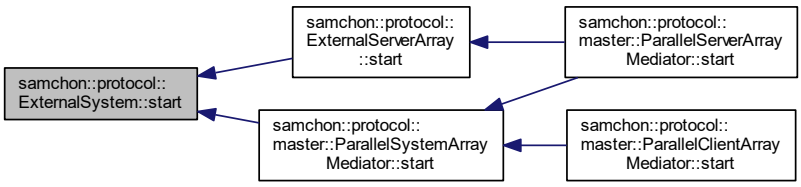
An abstract method starting interaction with an external system.

If an external systems are a server, starts connection and listening Inovoke message, else clients, just starts listening only. You also can addict your own procudures of starting the driver, but if you directly override method of abstract ExternalSystem, be careful about virtual inheritance.

Implemented in samchon::protocol::ExternalServer, and samchon::protocol::ExternalClient.

Referenced by samchon::protocol::ExternalServerArray::start(), and samchon::protocol::master::Parallel↩SystemArrayMediator::start().

Here is the caller graph for this function:



10.45.3.3 auto ExternalSystem::key ( ) const -> std::string [override], [virtual]

Get a key that can identify the Entity uniquely.

If identifier of the Entity is not atomic value, returns a string represents the composite identifier. If identifier of the Entity is not string, converts the identifier to string and returns the string.

Returns

An identifier

Reimplemented from samchon::protocol::Entity.

Definition at line 33 of file ExternalSystem.cpp.

References name.

10.45.3.4 void ExternalSystem::replyData (std::shared_ptr< Invoke >) [override], [virtual]

Handling replied message from an external system.

If there's a related ExternalSystemRole, shifts responsibility to the ExternalSystemRole.

Parameters

invoke	Replied Invoke message

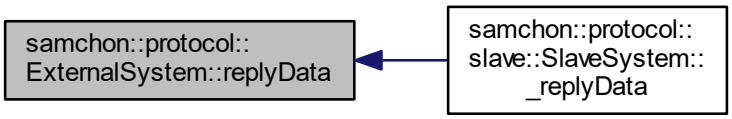
Reimplemented from samchon::protocol::IProtocol.

Reimplemented in samchon::protocol::master::DistributedSystem, and samchon::protocol::master::DistributedSlaveSystemMediator.

Definition at line 41 of file ExternalSystem.cpp.

Referenced by samchon::protocol::slave::SlaveSystem::_replyData().

Here is the caller graph for this function:



10.45.3.5 auto ExternalSystem::TAG ( ) const -> std::string [override], [virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Reimplemented in samchon::protocol::master::ParallelSlaveSystemMediator, and samchon::protocol::master::DistributedSlaveSystemMediator.

Definition at line 53 of file ExternalSystem.cpp.

10.45.3.6 auto ExternalSystem::CHILD_TAG ( ) const -> std::string [override], [virtual]

A tag name of children.

< TAG>
<CHILD_TAG />
<CHILD_TAG />
</TAG>

Implements samchon::protocol::IEntityGroup.

Definition at line 57 of file ExternalSystem.cpp.

10.45.3.7 auto ExternalSystem::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

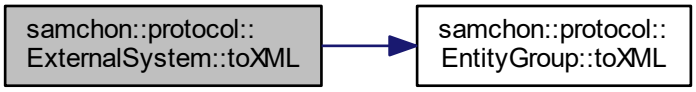
Reimplemented in samchon::protocol::master::DistributedSystem, samchon::protocol::master::ParallelSystem, samchon::protocol::ExternalServer, samchon::protocol::master::DistributedServer, and samchon::protocol::master::ParallelServer.

Definition at line 62 of file ExternalSystem.cpp.

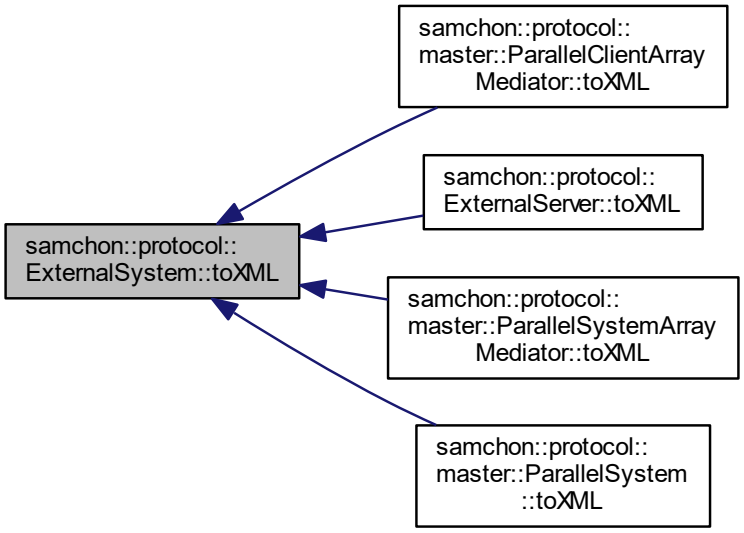
References ip, name, port, and samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::toXML().

Referenced by samchon::protocol::master::ParallelClientArrayMediator::toXML(), samchon::protocol::ExternalServer::toXML(), samchon::protocol::master::ParallelSystemArrayMediator::toXML(), and samchon::protocol::master::ParallelSystem::toXML().

Here is the call graph for this function:



Here is the caller graph for this function:



10.45.4 Member Data Documentation

10.45.4.1 std::string samchon::protocol::ExternalSystem::name [protected]

A name can identify an external system.

The name must be unique in ExternalSystemArray.

Definition at line 58 of file ExternalSystem.hpp.

Referenced by construct(), key(), and toXML().

10.45.4.2 std::string samchon::protocol::ExternalSystem::ip [protected]

An ip address of an external system.

Definition at line 63 of file ExternalSystem.hpp.

Referenced by samchon::protocol::ExternalClientArray::addClient(), construct(), samchon::protocol::↩ ExternalServer::getIP(), and toXML().

10.45.4.3 int samchon::protocol::ExternalSystem::port [protected]

A port number of an external system.

Definition at line 68 of file ExternalSystem.hpp.

Referenced by samchon::protocol::ExternalClientArray::addClient(), construct(), samchon::protocol::↩ ExternalServer::getPort(), and toXML().

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

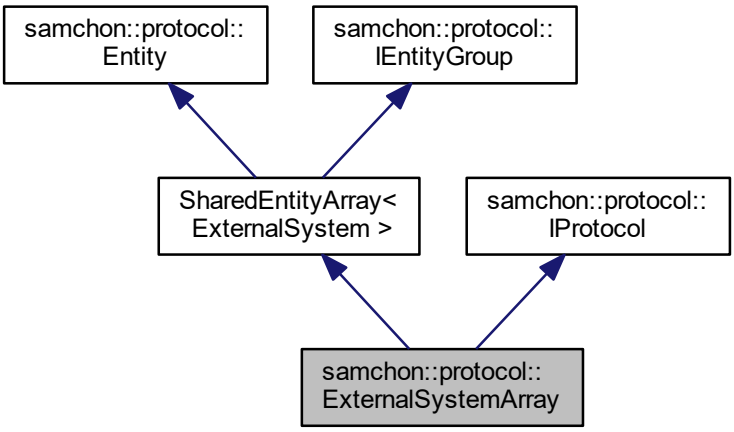
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalSystem.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalSystem.cpp

10.46 samchon::protocol::ExternalSystemArray Class Reference

An array of ExternalSystem(s).

`#include <ExternalSystemArray.hpp>`

Collaboration diagram for samchon::protocol::ExternalSystemArray:



Public Member Functions

- ExternalSystemArray ()
  Default Constructor.
- virtual void start ()=0
  Start interaction.
- auto hasRole (const std::string &) const -> bool
  Test whether has a role.
- auto getRole (const std::string &) const -> std::shared_ptr< ExternalSystemRole >
  Get a role.
- virtual void sendData (std::shared_ptr< Invoke >) override
  Sends an Invoke message to related ExternalSystem(s).
- virtual void replyData (std::shared_ptr< Invoke >) override
  Handle replied Invoke message.
- virtual auto TAG () const -> std::string override
  A tag name when represented by XML.
- virtual auto CHILD_TAG () const -> std::string override
  A tag name of children.

Additional Inherited Members

10.46.1 Detailed Description

An array of ExternalSystem(s).

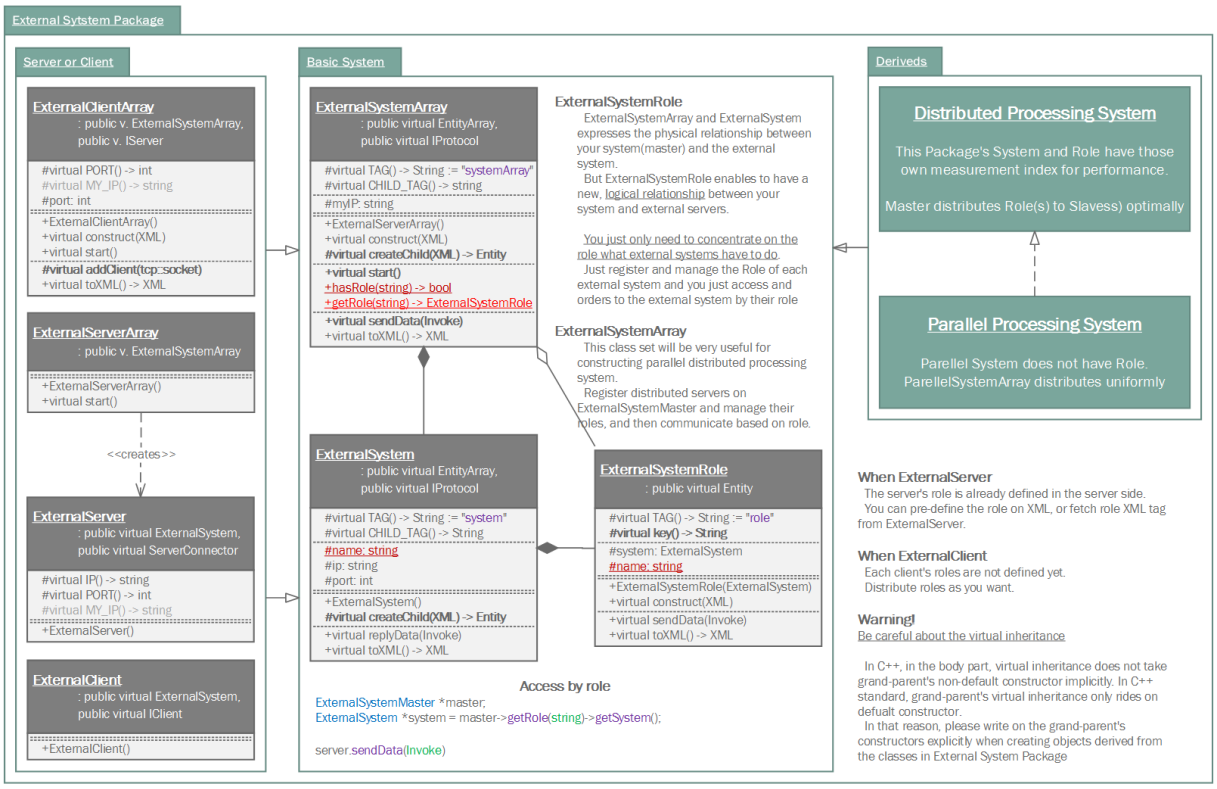
ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly,

not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

– ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 39 of file ExternalSystemArray.hpp.

10.46.2 Constructor & Destructor Documentation

10.46.2.1 ExternalSystemArray::ExternalSystemArray ( )

Default Constructor.

Definition at line 18 of file ExternalSystemArray.cpp.

10.46.3 Member Function Documentation

10.46.3.1 virtual void samchon::protocol::ExternalSystemArray::start ( ) [pure virtual]

Start interaction.

An abstract method starting interaction with external systems.

If external systems are servers, starts connection to them, else clients, opens a server and accepts the external systems. You can addict your own procudures of starting drivers, but if you directly override method of abstract ExternalSystemArray, be careful about virtual inheritance.

- DistributedClientArray::start() calls DistributedSystemArray::start() and ExternalClientArray:start()

Implemented in samchon::protocol::master::DistributedSystemArray, samchon::protocol::master::↩ DistributedSystemArrayMediator, samchon::protocol::master::ParallelSystemArrayMediator, samchon↩ ::protocol::ExternalClientArray, samchon::protocol::master::DistributedClientArrayMediator, samchon↩ ::protocol::master::ParallelClientArrayMediator, samchon::protocol::master::DistributedClientArray, samchon↩ ::protocol::master::DistributedServerArrayMediator, samchon::protocol::master::ParallelServerArray↩ Mediator, samchon::protocol::master::DistributedServerArray, and samchon::protocol::ExternalServerArray.

10.46.3.2 auto ExternalSystemArray::hasRole (const std::string & *name*) const -> bool

Test whether has a role.

Parameters

name	Name of an ExternalSystemRole

Returns

Whether has or not

Definition at line 26 of file ExternalSystemArray.cpp.

10.46.3.3 auto ExternalSystemArray::getRole (const std::string & *name*) const -> std::shared_ptr<ExternalSystemRole>

Get a role.

Parameters

name	Name of an ExternalSystemRole

Returns

A shared pointer of specialized role

Definition at line 34 of file ExternalSystemArray.cpp.

10.46.3.4 void ExternalSystemArray::sendData (std::shared_ptr< Invoke >) [override], [virtual]

Sends an Invoke message to related ExternalSystem(s).

If listener of the Invoke message is registered on some ExternalSystemRole(s), sends the Invoke message to related ExternalSystem(s) (containing the ExternalSystemRole(s)). If cannot find any ExternalSystemRole, sends the Invoke message to all ExternalSystem(s) belongs to the ExternalSystemArray.

Parameters

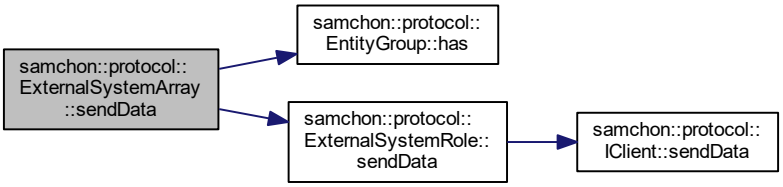
invoke	Invoke message to send

Reimplemented from samchon::protocol::IProtocol.

Definition at line 46 of file ExternalSystemArray.cpp.

References samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::has(), and samchon::protocol::↩ ExternalSystemRole::sendData().

Here is the call graph for this function:



10.46.3.5 void ExternalSystemArray::replyData (std::shared_ptr< Invoke >) [override], [virtual]

Handle replied Invoke message.

Nothing is defined in the ExternalSystemArray::replyData(). It's only for avoid the method to have a status "not overriden, abstract".

Parameters

invoke	Invoke message to send

Reimplemented from samchon::protocol::IProtocol.

Definition at line 65 of file ExternalSystemArray.cpp.

10.46.3.6 auto ExternalSystemArray::TAG ( ) const -> std::string [override], [virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Definition at line 73 of file ExternalSystemArray.cpp.

10.46.3.7 auto ExternalSystemArray::CHILD_TAG ( ) const -> std::string [override], [virtual]

A tag name of children.

< TAG>
<CHILD_TAG />
<CHILD_TAG />
</TAG>

Implements samchon::protocol::IEntityGroup.

Definition at line 77 of file ExternalSystemArray.cpp.

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

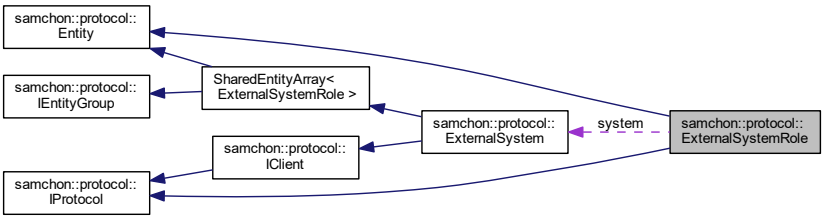
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalSystemArray.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalSystemArray.cpp

10.47 samchon::protocol::ExternalSystemRole Class Reference

A role belongs to an external system.

`#include <ExternalSystemRole.hpp>`

Collaboration diagram for samchon::protocol::ExternalSystemRole:



Public Member Functions

- ExternalSystemRole (ExternalSystem *)

  Construct from external system driver.
- virtual void construct (std::shared_ptr< library::XML >) override

  Construct data of the Entity from an XML object.
- auto getSystem () const -> ExternalSystem *

  Get an external system driver.
- virtual auto key () const -> std::string override

  Get a key that can identify the Entity uniquely.
- auto hasSendListener (const std::string &) const -> bool

  Test whether has a listener for send in the role.
- auto hasReplyListener (const std::string &) const -> bool

  Test whether has a listener for reply in the role.
- virtual void sendData (std::shared_ptr< Invoke >) override

  Send a message.
- virtual auto TAG () const -> std::string override

  A tag name when represented by XML.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the Entity.

Protected Attributes

- std::string name

  A name representing the role.
- std::set< std::string > replyListeners

  Listeners to reply in the role.
- std::set< std::string > sendListeners

  Listeners to send in the role.

Private Attributes

- ExternalSystem * system

  A driver of external system containing the ExternalSystemRole.

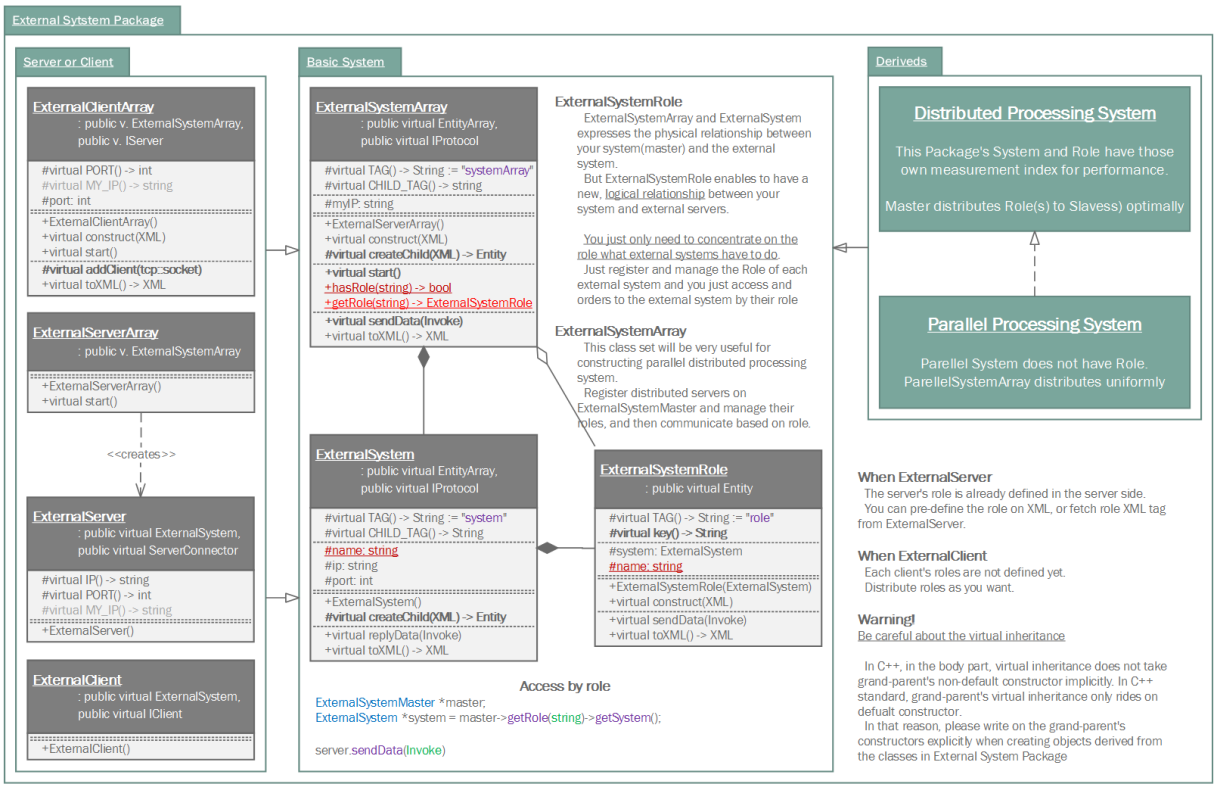
10.47.1 Detailed Description

A role belongs to an external system.

ExternalSystemRole is a 'control' class groupping methods, handling Invoke messages interacting with an external system that the ExternalSystemRole is belonged to, by a subject or unit of a module.

ExternalSystemRole can be a "logical proxy" for an ExternalSystem which is containing the ExternalSystem↩ Role. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystem↩ Role acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 39 of file ExternalSystemRole.hpp.

10.47.2 Constructor & Destructor Documentation

10.47.2.1 ExternalSystemRole::ExternalSystemRole (ExternalSystem * *system*)

Construct from external system driver.

Parameters

system	A driver of external system the ExternalSystemRole is belonged to.

Definition at line 11 of file ExternalSystemRole.cpp.

References system.

10.47.3 Member Function Documentation

10.47.3.1 void ExternalSystemRole::construct (std::shared_ptr< library::XML >) [override], [virtual]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Implements samchon::protocol::Entity.

Definition at line 17 of file ExternalSystemRole.cpp.

References name, replyListeners, and sendListeners.

10.47.3.2 auto ExternalSystemRole::getSystem ( ) const -> ExternalSystem*

Get an external system driver.

Definition at line 38 of file ExternalSystemRole.cpp.

References system.

10.47.3.3 auto ExternalSystemRole::key ( ) const -> std::string [override], [virtual]

Get a key that can identify the Entity uniquely.

If identifier of the Entity is not atomic value, returns a string represents the composite identifier. If identifier of the Entity is not string, converts the identifier to string and returns the string.

Returns

An identifier

Reimplemented from samchon::protocol::Entity.

Definition at line 42 of file ExternalSystemRole.cpp.

References name.

10.47.3.4 auto ExternalSystemRole::hasSendListener (const std::string &) const -> bool

Test whether has a listener for send in the role.

Definition at line 47 of file ExternalSystemRole.cpp.

10.47.3.5 auto ExternalSystemRole::hasReplyListener (const std::string &) const -> bool

Test whether has a listener for reply in the role.

Definition at line 51 of file ExternalSystemRole.cpp.

10.47.3.6 void ExternalSystemRole::sendData (std::shared_ptr< Invoke >) [override], [virtual]

Send a message.

Sends Invoke message to a network system or shifts the responsibility to related chain.

Parameters

invoke	An Invoke message to send to a network system

Reimplemented from samchon::protocol::IProtocol.

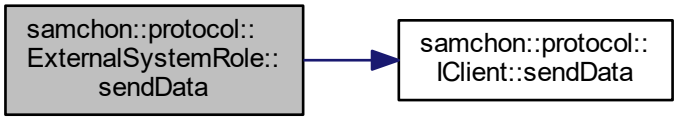
Reimplemented in samchon::protocol::master::DistributedSystemRole.

Definition at line 59 of file ExternalSystemRole.cpp.

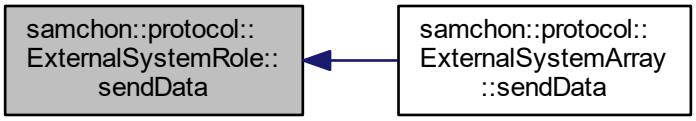
References samchon::protocol::IClient::sendData(), and system.

Referenced by samchon::protocol::ExternalSystemArray::sendData().

Here is the call graph for this function:



Here is the caller graph for this function:



10.47.3.7 auto ExternalSystemRole::TAG ( ) const -> std::string [override], [virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Definition at line 67 of file ExternalSystemRole.cpp.

10.47.3.8 auto ExternalSystemRole::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

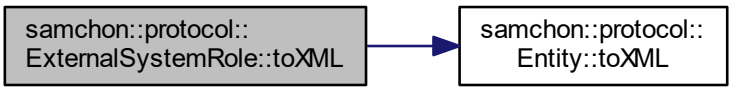
Reimplemented from samchon::protocol::Entity.

Reimplemented in samchon::protocol::master::DistributedSystemRole.

Definition at line 72 of file ExternalSystemRole.cpp.

References name, replyListeners, sendListeners, and samchon::protocol::Entity::toXML().

Here is the call graph for this function:



10.47.4 Member Data Documentation

10.47.4.1 ExternalSystem* samchon::protocol::ExternalSystemRole::system [private]

A driver of external system containing the ExternalSystemRole.

Definition at line 50 of file ExternalSystemRole.hpp.

Referenced by ExternalSystemRole(), getSystem(), and sendData().

10.47.4.2 std::string samchon::protocol::ExternalSystemRole::name [protected]

A name representing the role.

Definition at line 56 of file ExternalSystemRole.hpp.

Referenced by construct(), key(), and toXML().

10.47.4.3 std::set<std::string> samchon::protocol::ExternalSystemRole::replyListeners [protected]

Listeners to reply in the role.

Definition at line 61 of file ExternalSystemRole.hpp.

Referenced by construct(), and toXML().

10.47.4.4 std::set<std::string> samchon::protocol::ExternalSystemRole::sendListeners [protected]

Listeners to send in the role.

Definition at line 66 of file ExternalSystemRole.hpp.

Referenced by construct(), and toXML().

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

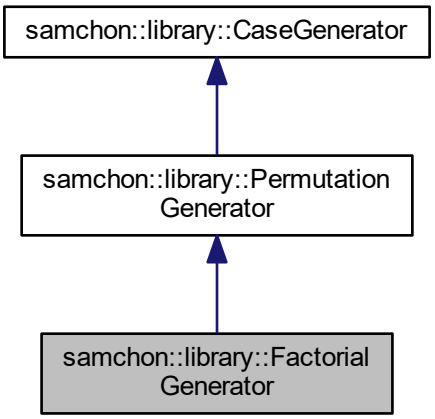
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalSystemRole.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ExternalSystemRole.cpp

10.48 samchon::library::FactorialGenerator Class Reference

Factorical case generator.

```
#include <FactorialGenerator.hpp>
```

Collaboration diagram for samchon::library::FactorialGenerator:



Public Member Functions

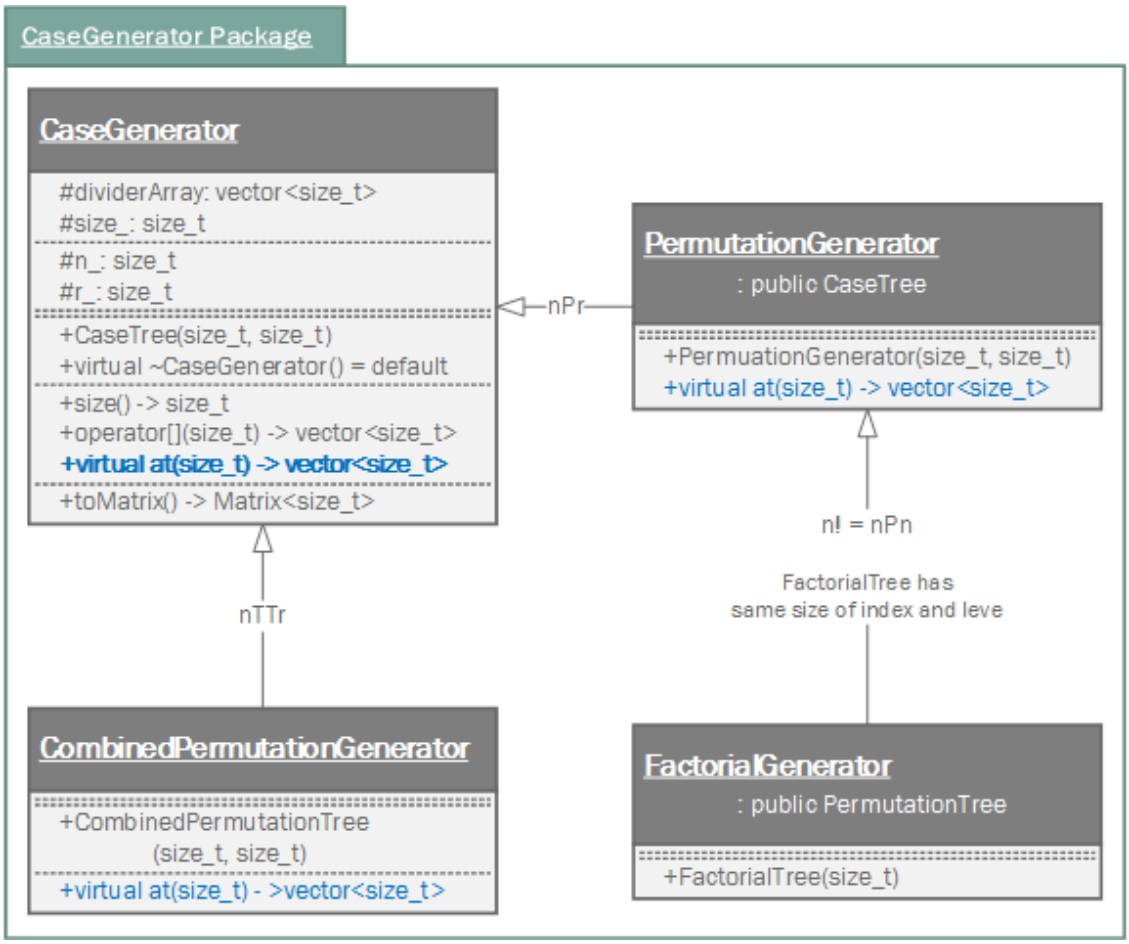
- FactorialGenerator (size_t size)
  Construct from size N.

Additional Inherited Members

10.48.1 Detailed Description

Factorical case generator.

n! = nPn



See also

samchon::library

Author

Jeongho Nam

Definition at line 21 of file FactorialGenerator.hpp.

10.48.2 Constructor & Destructor Documentation

10.48.2.1 FactorialGenerator::FactorialGenerator (size_t *size*)

Construct from size N.

Parameters

n	Size of elements of each case to be arranged.

Definition at line 6 of file FactorialGenerator.cpp.

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/FactorialGenerator.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/FactorialGenerator.cpp

10.49 samchon::protocol::FlashPolicyServer Class Reference

A flash policy server.

```
#include <FlashPolicyServer.hpp>
```

Public Member Functions

- FlashPolicyServer ()
  Default constructor.
- FlashPolicyServer (std::shared_ptr< library::XML >)
  Construct from custom policy.
- void openServer ()
  Open server.

Protected Attributes

- std::shared_ptr< library::XML > policy
  cross-domain-policy

Private Member Functions

- void accept (Socket *)
  Accept client.

10.49.1 Detailed Description

A flash policy server.

FlashPolicyServer is a policy server replying cross-domain-policy to flash clients.

Author

Jeongho Nam

Definition at line 21 of file FlashPolicyServer.hpp.

10.49.2 Constructor & Destructor Documentation

10.49.2.1 FlashPolicyServer::FlashPolicyServer ( )

Default constructor.

The cross-domain-policy is to accept all, any client.

Definition at line 14 of file FlashPolicyServer.cpp.

10.49.2.2 samchon::protocol::FlashPolicyServer::FlashPolicyServer (std::shared_ptr< library::XML >)

Construct from custom policy.

Parameters

policy	Custom cross-domain-policy to adjust

10.49.3 Member Function Documentation

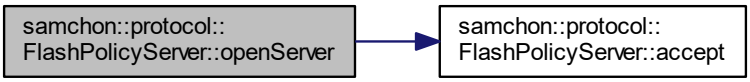
10.49.3.1 void FlashPolicyServer::openServer ( )

Open server.

Definition at line 32 of file FlashPolicyServer.cpp.

References accept().

Here is the call graph for this function:



10.49.3.2 void FlashPolicyServer::accept (Socket *) [private]

Accept client.

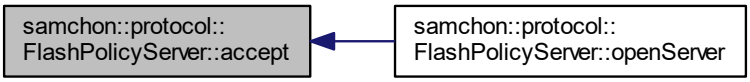
Accepts flash client and replies cross-domain-policy

Definition at line 46 of file FlashPolicyServer.cpp.

References policy.

Referenced by openServer().

Here is the caller graph for this function:



10.49.4 Member Data Documentation

10.49.4.1 std::shared_ptr<library::XML> samchon::protocol::FlashPolicyServer::policy [protected]

cross-domain-policy

Definition at line 27 of file FlashPolicyServer.hpp.

Referenced by accept().

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

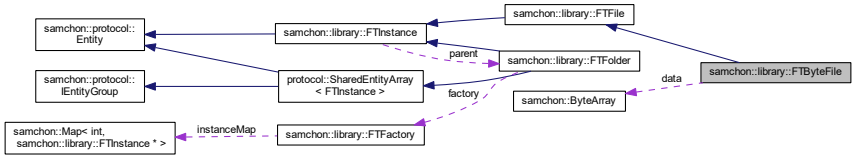
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/FlashPolicyServer.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/FlashPolicyServer.cpp

10.50 samchon::library::FTByteFile Class Reference

A binary file.

```
#include <FTByteFile.hpp>
```

Collaboration diagram for samchon::library::FTByteFile:



Public Member Functions

- FTByteFile (FTFolder ∗)

  Construct from parent folder.

- auto getData () -> ByteArray

  Get binaray data of the file.

Protected Attributes

- ByteArray data

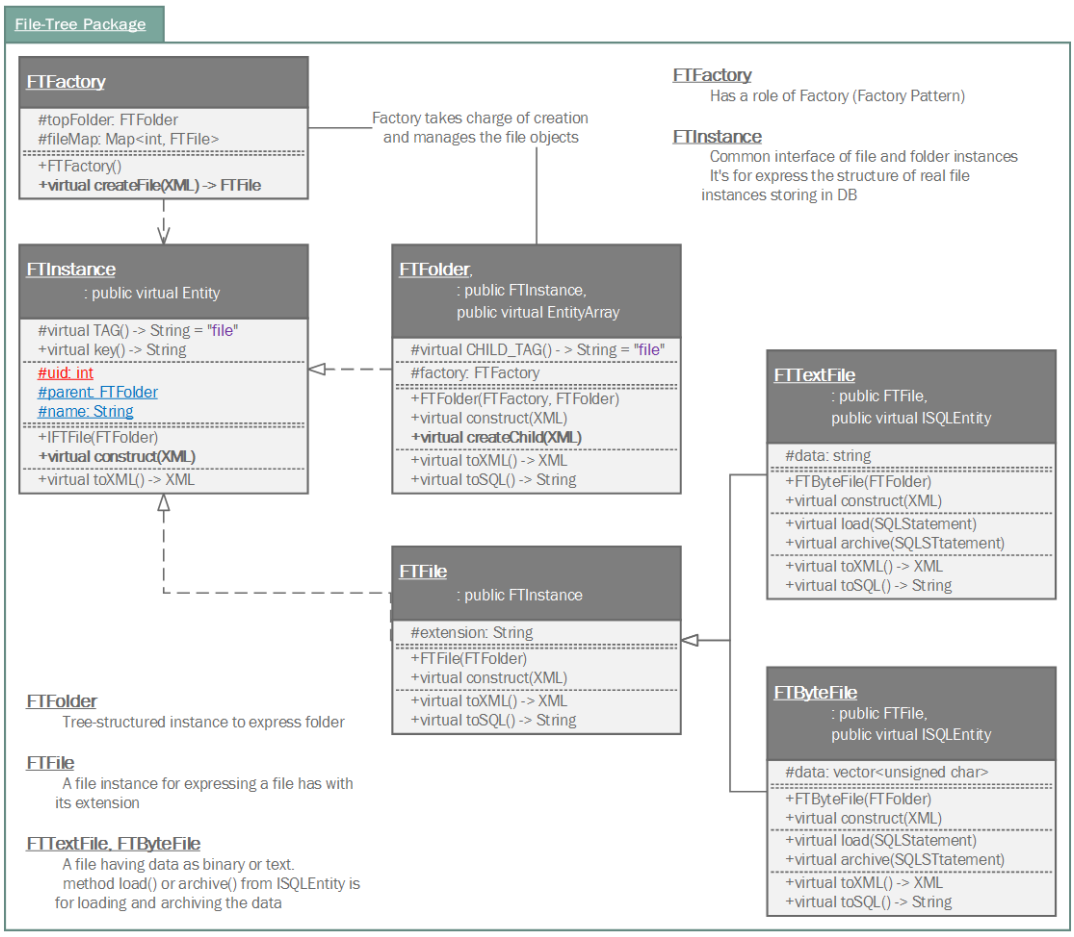
  Binary data of the file.

Additional Inherited Members

10.50.1 Detailed Description

A binary file.

A file having data as binary.



See also

samchon::library

Author

Jeongho Nam

Definition at line 22 of file FTByteFile.hpp.

10.50.2 Constructor & Destructor Documentation

10.50.2.1 samchon::library::FTByteFile::FTByteFile (FTFolder *)

Construct from parent folder.

Parameters

folder	Belonged folder

10.50.3 Member Function Documentation

10.50.3.1 auto samchon::library::FTByteFile::getData ( ) -> ByteArray

Get binaray data of the file.

10.50.4 Member Data Documentation

10.50.4.1 ByteArray samchon::library::FTByteFile::data [protected]

Binary data of the file.

Definition at line 32 of file FTByteFile.hpp.

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

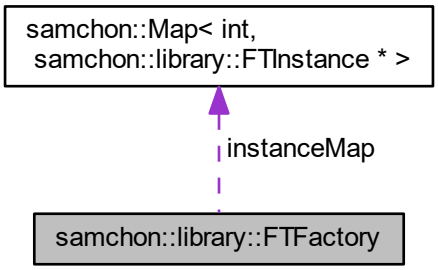
– D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/FTByteFile.hpp

10.51 samchon::library::FTFactory Class Reference

A Factory and manager for file-tree.

`#include <FTFactory.hpp>`

Collaboration diagram for samchon::library::FTFactory:



Public Member Functions

– FTFactory ()

Default Constructor.

– virtual auto createFile (FTFolder *, std::shared_ptr< XML >) -> FTFile *=0

Factory method of a file.

– void registerInstance (FTInstance *)

Register file instance to map.

Protected Attributes

– int application

An application id.

– int category

A categori id.

– std::string member

Account id of a member.

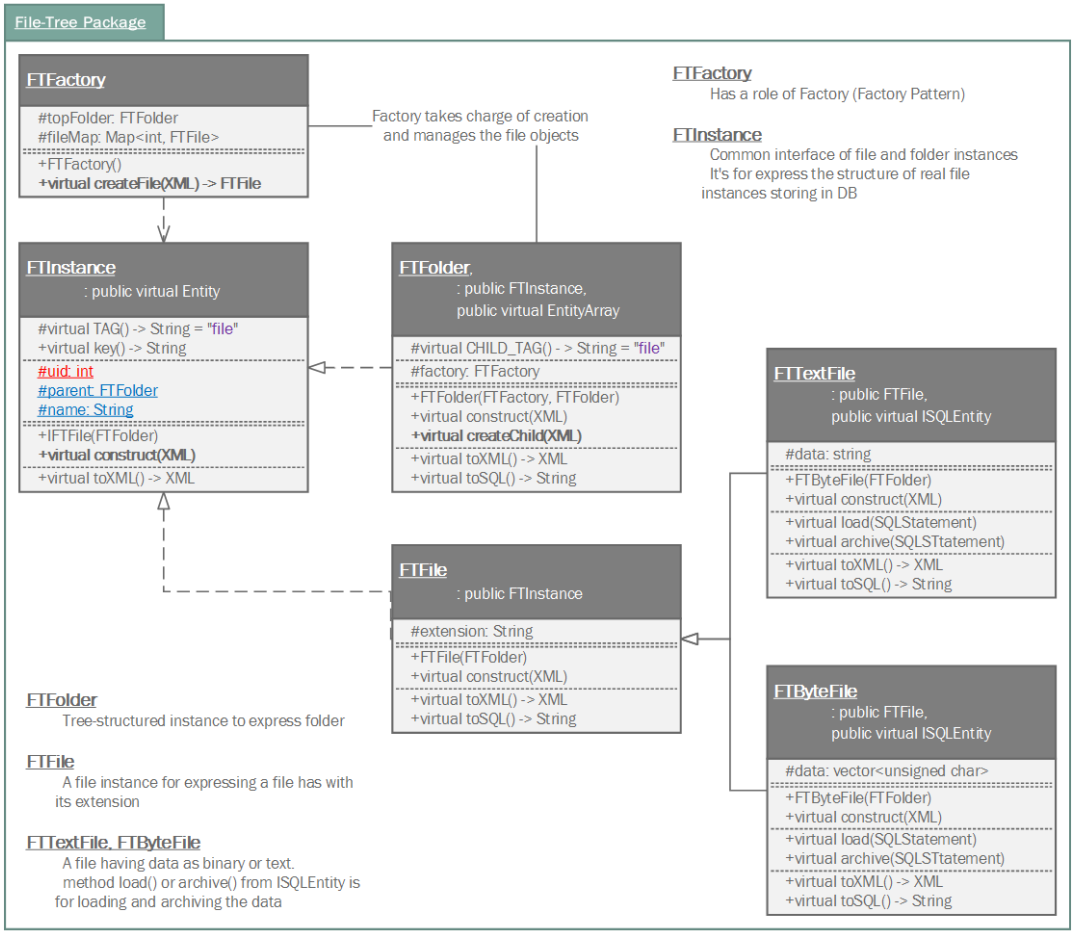
– Map< int, FTInstance * > instanceMap

Map of files.

10.51.1 Detailed Description

A Factory and manager for file-tree.

FTFactory is a class taking a role of creating FTFile instances and managing them by having their direct pointers. FTFactory also has a role of archiving and loading from DB.



See also

samchon::library

Author

Jeongho Nam

Definition at line 29 of file FTFactory.hpp.

10.51.2 Constructor & Destructor Documentation

10.51.2.1 FTFactory::FTFactory ( )

Default Constructor.

Definition at line 12 of file FTFactory.cpp.

10.51.3 Member Function Documentation

10.51.3.1 virtual auto samchon::library::FTFactory::createFile (FTFolder *, std::shared_ptr< XML >) -> FTFile * [pure virtual]

Factory method of a file.

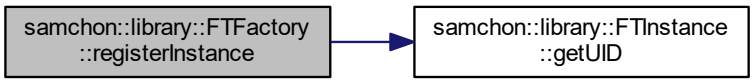
10.51.3.2 void FTFactory::registerInstance (FTInstance * *instance*)

Register file instance to map.

Definition at line 16 of file FTFactory.cpp.

References samchon::library::FTInstance::getUID().

Here is the call graph for this function:



10.51.4 Member Data Documentation

10.51.4.1 int samchon::library::FTFactory::application [protected]

An application id.

An enumeration code of an application that managed FTInstance(s) are belonged to.

Definition at line 37 of file FTFactory.hpp.

10.51.4.2 int samchon::library::FTFactory::category [protected]

A categori id.

An enumeration code of a category that managed FTInstance(s) are belonged to.

Definition at line 44 of file FTFactory.hpp.

10.51.4.3 std::string samchon::library::FTFactory::member [protected]

Account id of a member.

Definition at line 51 of file FTFactory.hpp.

10.51.4.4 Map<int, FTInstance*> samchon::library::FTFactory::instanceMap [protected]

Map of files.

- key: uid
- value: pointer of file

Definition at line 60 of file FTFactory.hpp.

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

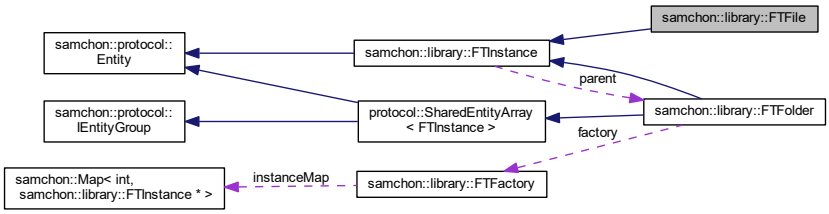
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/FTFactory.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/FTFactory.cpp

10.52 samchon::library::FTFile Class Reference

A file.

```
#include <FTFile.hpp>
```

Collaboration diagram for samchon::library::FTFile:



Public Member Functions

- FTFile (FTFolder *)

  Construct from parent folder.
- auto getExtension () const -> std::string

  Get extension.
- virtual auto toXML () const -> std::shared_ptr< library::XML >

  Get an XML object represents the Entity.

Protected Attributes

- std::string extension

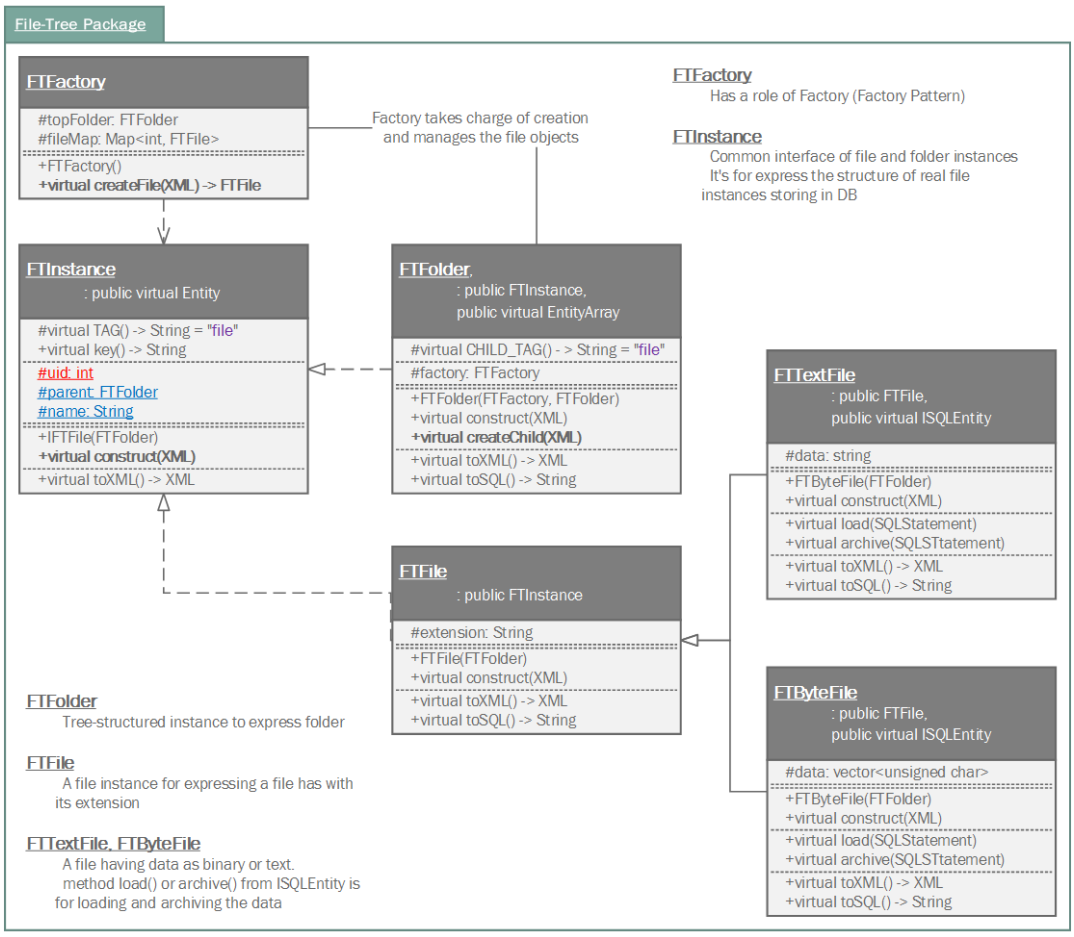
  An extension of a file.

Additional Inherited Members

10.52.1 Detailed Description

A file.

An abstract class represents a file instance with extension.



See also

samchon::library

Author

Jeongho Nam

Definition at line 20 of file FTFile.hpp.

10.52.2 Constructor & Destructor Documentation

10.52.2.1 FTFile::FTFile (FTFolder * *parent*)

Construct from parent folder.

Parameters

folder	Belonged folder

Definition at line 12 of file FTFile.cpp.

10.52.3 Member Function Documentation

10.52.3.1 auto FTFile::getExtension ( ) const -> std::string

Get extension.

Definition at line 23 of file FTFile.cpp.

References extension.

10.52.3.2 auto FTFile::toXML ( ) const -> std::shared_ptr<library::XML> [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::library::FTInstance.

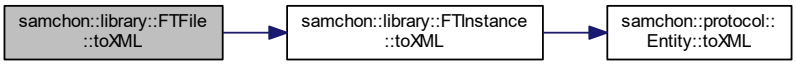
Reimplemented in samchon::namtree::NTFile.

Definition at line 28 of file FTFile.cpp.

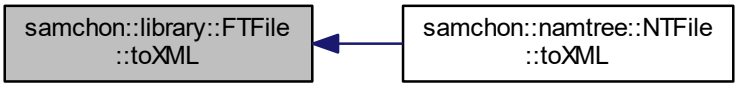
References extension, and samchon::library::FTInstance::toXML().

Referenced by samchon::namtree::NTFile::toXML().

Here is the call graph for this function:



Here is the caller graph for this function:



10.52.4 Member Data Documentation

10.52.4.1 std::string samchon::library::FTFile::extension [protected]

An extension of a file.

FTFile's identifier is composited by name and extension.

Definition at line 31 of file FTFile.hpp.

Referenced by getExtension(), and toXML().

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

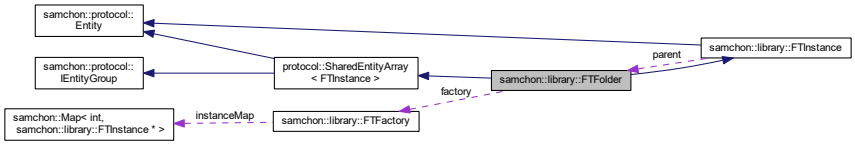
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/FTFile.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/FTFile.cpp

10.53 samchon::library::FTFolder Class Reference

A folder.

`#include <FTFolder.hpp>`

Collaboration diagram for samchon::library::FTFolder:



Public Member Functions

- FTFolder (FTFactory *, FTFolder *)

  Construct from factory and parent folder.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the Entity.

Protected Member Functions

- virtual auto CHILD_TAG () const -> std::string override

  A tag name of children.

Protected Attributes

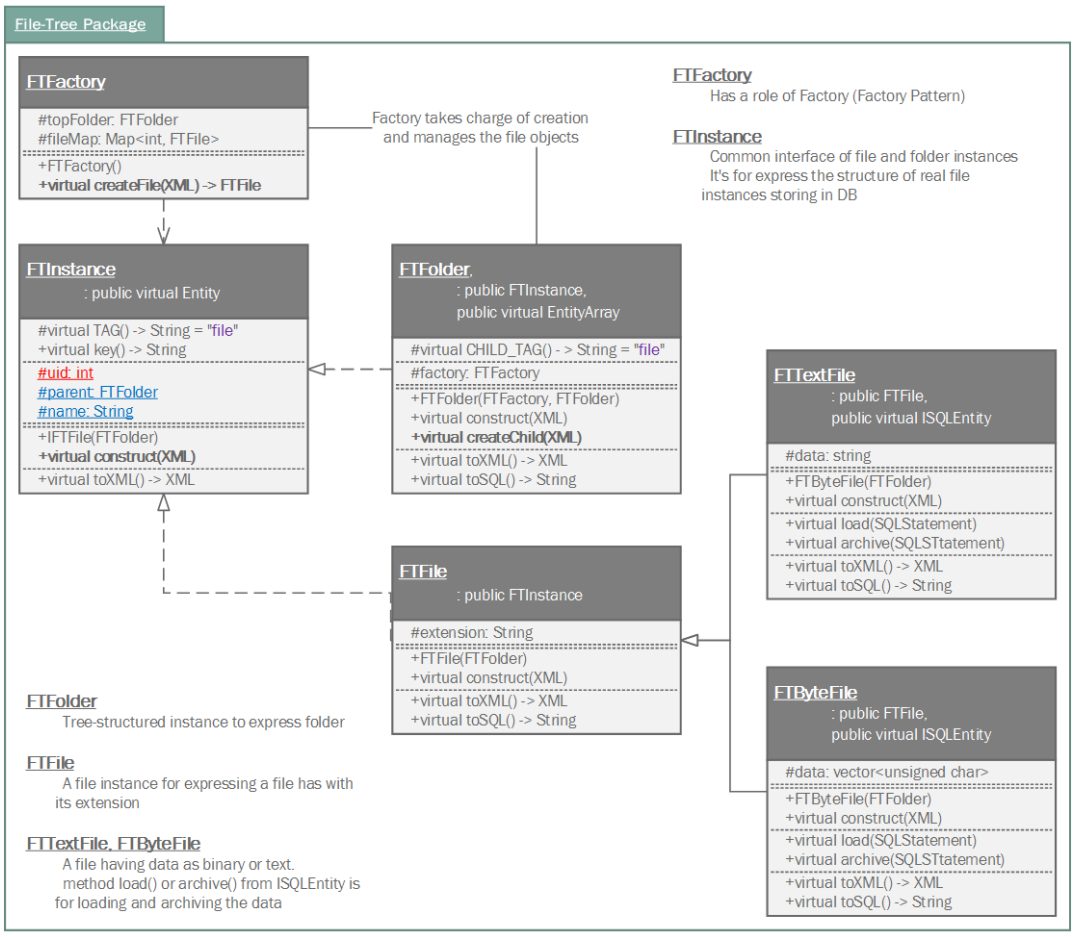
- FTFactory * factory

  Factory instance for creating sub files.

10.53.1 Detailed Description

A folder.

A folder object has recursive and hierarchical relationship.



See also

samchon::library

Author

Jeongho Nam

Definition at line 23 of file FTFolder.hpp.

10.53.2 Constructor & Destructor Documentation

10.53.2.1 FTFolder::FTFolder (FTFactory * *factory*, FTFolder * *parent*)

Construct from factory and parent folder.

Parameters

factory	Factory instance
parent	Parent folder that this folder is belonged to

Definition at line 15 of file FTFolder.cpp.

References factory.

10.53.3 Member Function Documentation

10.53.3.1 auto FTFolder::CHILD_TAG ( ) const -> std::string [override], [protected], [virtual]

A tag name of children.

< TAG>
<CHILD_TAG />
<CHILD_TAG />
</TAG>

Implements samchon::protocol::IEntityGroup.

Definition at line 13 of file FTFolder.cpp.

10.53.3.2 auto FTFolder::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

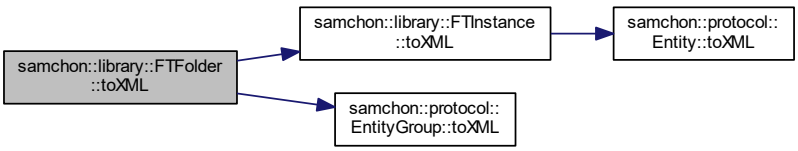
An XML object representing the Entity.

Reimplemented from samchon::library::FTInstance.

Definition at line 39 of file FTFolder.cpp.

References samchon::library::FTInstance::toXML(), and samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::toXML().

Here is the call graph for this function:



10.53.4 Member Data Documentation

10.53.4.1 FTFactory* samchon::library::FTFolder::factory [protected]

Factory instance for creating sub files.

Definition at line 36 of file FTFolder.hpp.

Referenced by FTFolder().

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

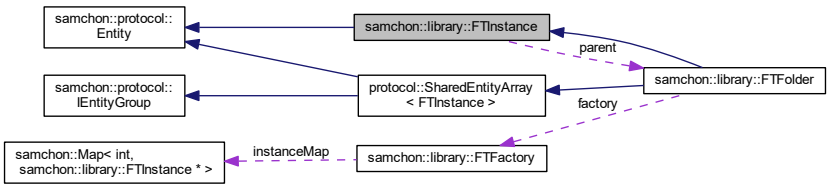
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/FTFolder.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/FTFolder.cpp

10.54 samchon::library::FTInstance Class Reference

An interface of file-tree.

`#include <FTInstance.hpp>`

Collaboration diagram for samchon::library::FTInstance:



Public Member Functions

- FTInstance (FTFolder *)
  Construct from parent folder.
- virtual auto key () const -> std::string
  Get a key that can identify the Entity uniquely.
- auto getUID () const -> int
  Get uid.
- auto getParent () const -> FTFolder *
  Get parent folder.
- auto getName () const -> std::string
  Get name.
- auto getComment () const -> std::string
  Get comment.
- virtual auto toXML () const -> std::shared_ptr< library::XML >
  Get an XML object represents the Entity.

Protected Member Functions

- virtual auto TAG () const -> std::string
  A tag name when represented by XML.

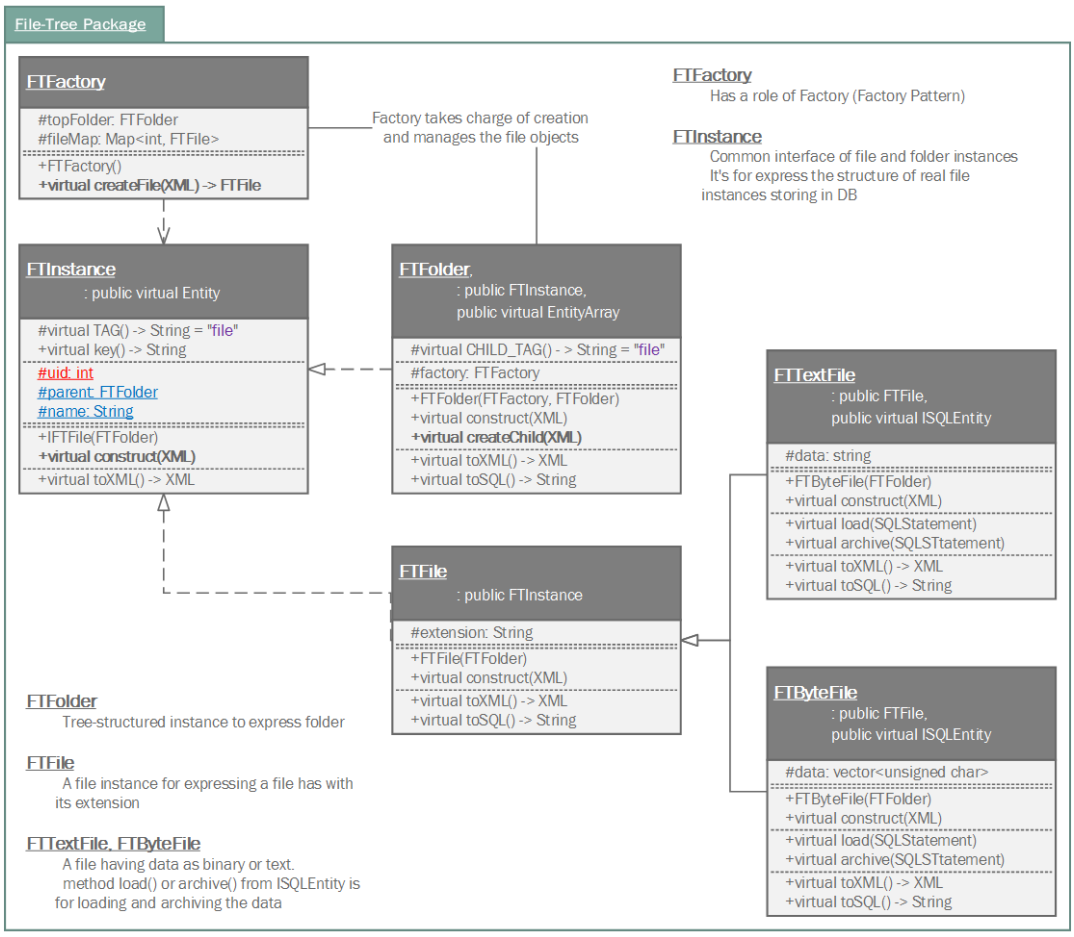
Protected Attributes

- FTFolder * parent
  Parent folder containing the instance.
- int uid
  Key, an unique id of file.
- std::string name
  Name of the file.
- std::string comment
  Comment of the file.

10.54.1 Detailed Description

An interface of file-tree.

An abstract class for folder and file instances.



See also

samchon::library

Author

Jeongho Nam

Definition at line 22 of file FTInstance.hpp.

10.54.2 Constructor & Destructor Documentation

10.54.2.1 FTInstance::FTInstance (FTFolder * *parent*)

Construct from parent folder.

Parameters

parent	Parent folder that this instance is belonged to

Definition at line 13 of file FTInstance.cpp.

References parent.

10.54.3 Member Function Documentation

10.54.3.1 auto FTInstance::TAG ( ) const -> std::string [protected], [virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Definition at line 11 of file FTInstance.cpp.

10.54.3.2 auto FTInstance::key ( ) const -> std::string [virtual]

Get a key that can identify the Entity uniquely.

If identifier of the Entity is not atomic value, returns a string represents the composite identifier. If identifier of the Entity is not string, converts the identifier to string and returns the string.

Returns

An identifier

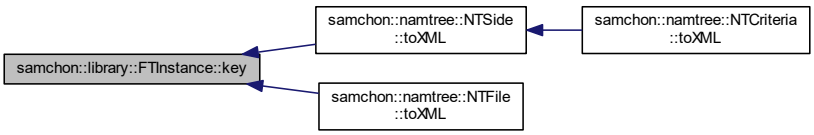
Reimplemented from samchon::protocol::Entity.

Definition at line 25 of file FTInstance.cpp.

References uid.

Referenced by samchon::namtree::NTSide::toXML(), and samchon::namtree::NTFile::toXML().

Here is the caller graph for this function:



10.54.3.3 auto FTInstance::getUID ( ) const -> int

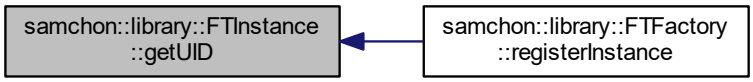
Get uid.

Definition at line 29 of file FTInstance.cpp.

References uid.

Referenced by samchon::library::FTFactory::registerInstance().

Here is the caller graph for this function:



10.54.3.4 auto FTInstance::getParent ( ) const -> FTFolder*

Get parent folder.

Definition at line 33 of file FTInstance.cpp.

References parent.

10.54.3.5 auto FTInstance::getName ( ) const -> std::string

Get name.

Definition at line 37 of file FTInstance.cpp.

References name.

10.54.3.6 auto FTInstance::getComment ( ) const -> std::string

Get comment.

Definition at line 41 of file FTInstance.cpp.

References comment.

10.54.3.7 auto FTInstance::toXML ( ) const -> std::shared_ptr<library::XML> [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::Entity.

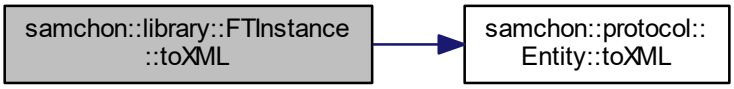
Reimplemented in samchon::namtree::NTFile, samchon::library::FTFolder, and samchon::library::FTFile.

Definition at line 46 of file FTInstance.cpp.

References comment, samchon::protocol::Entity::toXML(), and uid.

Referenced by samchon::library::FTFile::toXML(), and samchon::library::FTFolder::toXML().

Here is the call graph for this function:



Here is the caller graph for this function:

10.54.4 Member Data Documentation

10.54.4.1 FTFolder* samchon::library::FTInstance::parent [protected]

Parent folder containing the instance.

Definition at line 34 of file FTInstance.hpp.

Referenced by FTInstance(), and getParent().

10.54.4.2 int samchon::library::FTInstance::uid [protected]

Key, an unique id of file.

Definition at line 39 of file FTInstance.hpp.

Referenced by getUID(), key(), and toXML().

10.54.4.3 std::string samchon::library::FTInstance::name [protected]

Name of the file.

Definition at line 44 of file FTInstance.hpp.

Referenced by getName().

10.54.4.4 std::string samchon::library::FTInstance::comment [protected]

Comment of the file.

Definition at line 49 of file FTInstance.hpp.

Referenced by getComment(), and toXML().

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

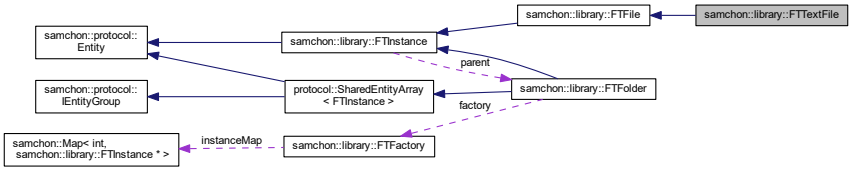
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/FTInstance.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/FTInstance.cpp

10.55 samchon::library::FTTextFile Class Reference

A text file.

```
#include <FTTextFile.hpp>
```

Collaboration diagram for samchon::library::FTTextFile:



Public Member Functions

- FTTextFile (FTFolder ∗)

  Construct from parent folder.

- auto getData () const -> std::string

  Get string content.

Protected Attributes

- std::string data

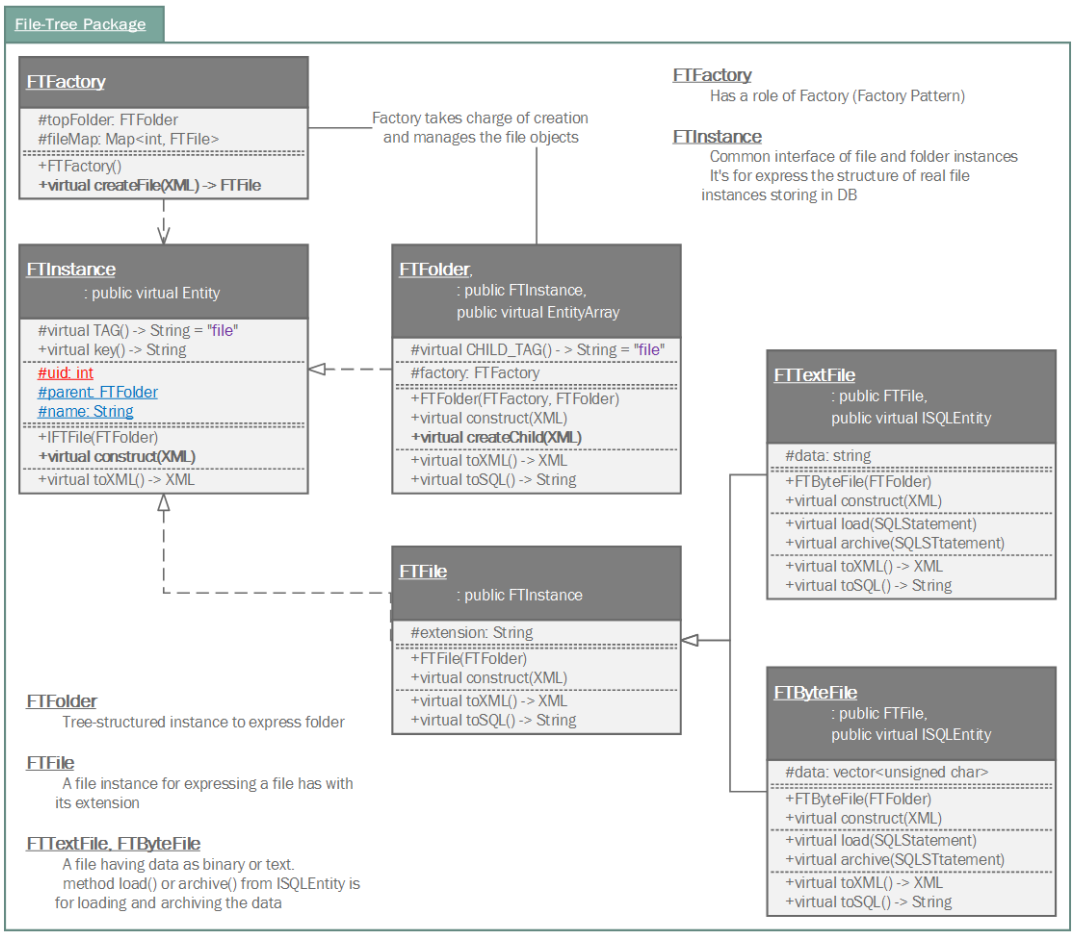
  Text recorded in the file.

Additional Inherited Members

10.55.1 Detailed Description

A text file.

A file having data as text.



See also

samchon::library

Author

Jeongho Nam

Definition at line 20 of file FTTextFile.hpp.

10.55.2 Constructor & Destructor Documentation

10.55.2.1 samchon::library::FTTextFile::FTTextFile (FTFolder *)

Construct from parent folder.

Parameters

folder	Belonged folder

10.55.3 Member Function Documentation

10.55.3.1 auto samchon::library::FTTextFile::getData ( ) const -> std::string

Get string content.

10.55.4 Member Data Documentation

10.55.4.1 std::string samchon::library::FTTextFile::data [protected]

Text recorded in the file.

Definition at line 30 of file FTTextFile.hpp.

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/FTTextFile.hpp

10.56 samchon::example::tsp::GAParameters Struct Reference

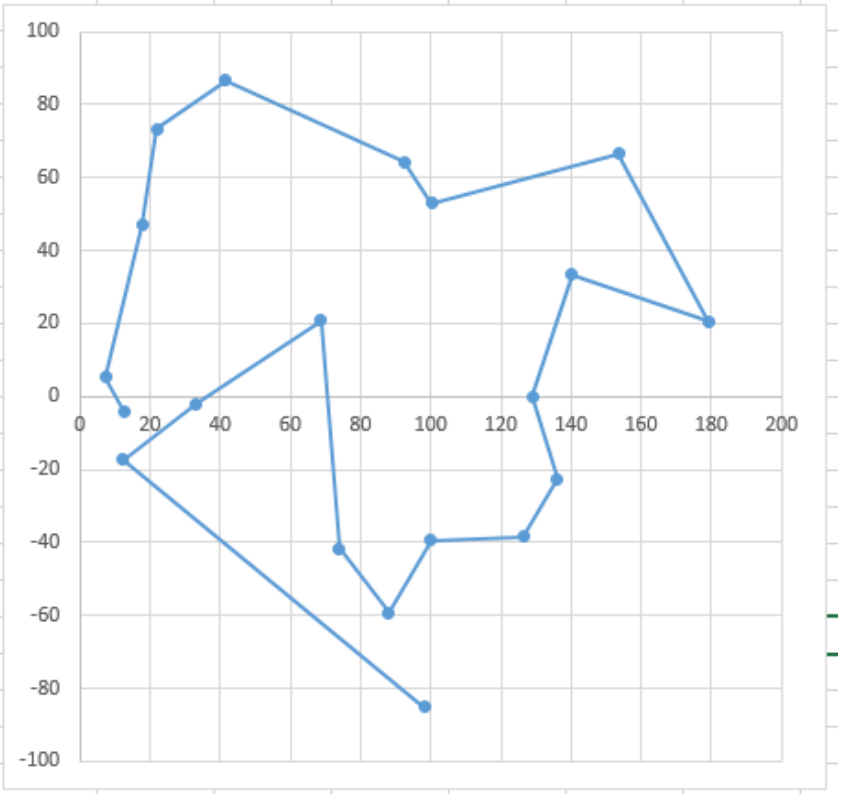
Parameters for Genetic-Algorithm.

`#include <Scheduler.hpp>`

10.56.1 Detailed Description

Parameters for Genetic-Algorithm.

Distance: 64,896.70 km		
uid	longitude	latitude
18	12.5233	-3.95529
15	7.23275	5.18342
7	17.7528	46.8665
2	22.0434	73.4133
20	41.6615	86.5372
13	92.427	64.3369
1	100.345	53.1067
9	153.521	66.5119
3	179.118	20.5073
10	140.113	33.3537
17	128.885	0.054607
4	136.012	-22.4915
6	126.352	-38.5031
16	99.9775	-39.3696
11	87.8859	-59.3663
14	73.8477	-41.5867
5	68.6563	20.8528
12	33.2162	-1.92823
8	12.1839	-17.3552
19	98.0967	-85.195



See also

samchon::library::GeneticAlgorithm

Author

Jeongho Nam

Definition at line 23 of file Scheduler.hpp.

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/tsp/Scheduler.hpp

10.57 samchon::library::GAParameters Struct Reference

A structure containing parameters of genetic algorithm.

```
#include <GAParameters.hpp>
```

Public Attributes

- size_t generation
  A number of generation of evolution.
- size_t population
  A number of population in a generation.
- size_t tournament
  A number of tournament in a selection.
- double mutationRate
  Rate of mutate ocurrence.

10.57.1 Detailed Description

A structure containing parameters of genetic algorithm.

The number and rate parameters in the structure affects success of evolution of genetic algorithm.

See also

library::GeneticAlgorithm
library::GAPopulation

Author

Jeongho Nam

Definition at line 18 of file GAParameters.hpp.

10.57.2 Member Data Documentation

10.57.2.1 size_t samchon::library::GAParameters::generation

A number of generation of evolution.

Definition at line 23 of file GAParameters.hpp.

10.57.2.2 size_t samchon::library::GAParameters::population

A number of population in a generation.

Definition at line 28 of file GAParameters.hpp.

10.57.2.3 size_t samchon::library::GAParameters::tournament

A number of tournament in a selection.

Definition at line 33 of file GAParameters.hpp.

10.57.2.4 double samchon::library::GAParameters::mutationRate

Rate of mutate ocurrence.

A mutation rate determines the percentage of occurence of mutation in a sequence list.

- Value from 0.0 to 1.0

Note

- When mutationRate is too high, it is hard to ancitipate studying on genetic algorithm.
- When mutationRate is too low and initial set of genes(GeneArray) is far away from optimal, the evolution tends to wandering outside of he optimal.

Definition at line 47 of file GAParameters.hpp.

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/GAParameters.hpp

10.58 samchon::library::GAPopulation< GeneArray, Compare > Class Template Reference

A population of a generation in G.A.

`#include <GAPopulation.hpp>`

Public Member Functions

- GAPopulation (std::shared_ptr< GeneArray > geneArray, size_t size)
  Construct from a GeneArray and size of the population.
- auto fitTest () const -> std::shared_ptr< GeneArray >
  Test fitness of each GeneArray in the population.

Private Member Functions

- GAPopulation (size_t size)
  Private Constructor with population.

Private Attributes

- std::vector< std::shared_ptr< GeneArray > > children
  Genes representing the population.

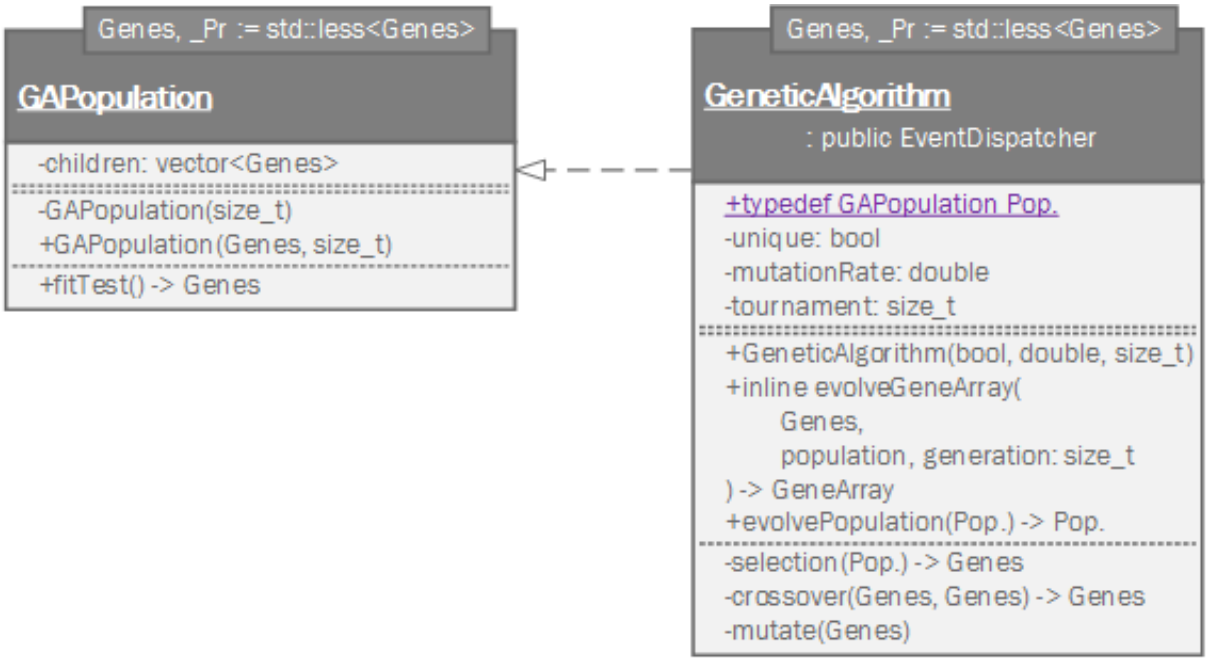
10.58.1 Detailed Description

template<typename GeneArray, typename Compare = std::less<GeneArray>>class samchon::library::GAPopulation< GeneArray, Compare >

A population of a generation in G.A.

GAPopulation is a class representing population of candidate genes(sequence listing) having an array of GeneArray as a member. GAPopulation also manages initial set of genes and handles fitting test direclty by the method *fitTest*.

The success of evolution of genetic alogirhtm is depend on the GAPopulation's initial set and fitting test. (GeneArray and Compare.)



Template Parameters

GeneArray	

An array(std::vector) containing genes as elments; sequnce listing

The GeneArray must be a type of *std::vector*.

Template Parameters

Compare	A comparison class (or struct) returns whether left gene is more optimal.
	
	Default template parameter of Compare is *std::less<GeneArray>*. It means to compare two std::vector (GeneArray must be a std::vector). Thus, you've to keep follwing rules.
	
	If you don't want to follow the rules or want a custom comparison class, you have to realize a comparison class. The following code is an example realizing the comparison class.
	
	`- GeneArray is inherited from <i>std::vector</i>`
	`- GeneArray has custom <i>auto operator<(const GeneArray &) const -> bool</i>`

```
template <typename _Ty>
struct MyCompare
{
auto operator()(const _Ty &newObj, const _Ty &prevObj) const -> bool;
};
```

Warning

Be careful for the mistakes of direction or position of Compare.

Most of logical errors failed to access optimal solution are occured by those mistakens.

See also

library::GeneticAlgorithm
samchon::library

Author

Jeonghoh Nam

Definition at line 58 of file GAPopulation.hpp.

10.58.2 Constructor & Destructor Documentation

10.58.2.1 template<typename GeneArray , typename Compare = std::less<GeneArray>> samchon::library::GAPopulation< GeneArray, Compare >::GAPopulation (size_t *size*) [inline], [private]

Private Constructor with population.

Private constructor of GAPopulation does not create children (candidate genes) but fills only *nullptr* repeatedly following the size.

The private constructor is only for and handled by GeneticAlgorithm.

Parameters

size	Size of the population

Definition at line 79 of file GAPopulation.hpp.

10.58.2.2 template<typename GeneArray , typename Compare = std::less<GeneArray>> samchon::library::GAPopulation< GeneArray, Compare >::GAPopulation (std::shared_ptr< GeneArray > *geneArray*, size_t *size*) [inline]

Construct from a GeneArray and size of the population.

Constructor with initial set of genes and size of the population.

The public constructor creates GeneArray(s) as population (size) having shuffled genes which are came from the initial set of genes (geneArray).

Note

The success of genetic algorithm is depend on the initial genes(geneArray).

Parameters

geneArray	An initial sequence listing
size	The size of population to have as children

Definition at line 98 of file GAPopulation.hpp.

10.58.3 Member Function Documentation

10.58.3.1 template<typename GeneArray , typename Compare = std::less<GeneArray>> auto samchon::library::GAPopulation< GeneArray, Compare >::fitTest ( ) const -> std::shared_ptr<GeneArray> [inline]

Test fitness of each GeneArray in the population.

Returns

The best GeneArray in the population

Definition at line 118 of file GAPopulation.hpp.

Referenced by samchon::library::GeneticAlgorithm< GeneArray, Compare >::selection().

Here is the caller graph for this function:

10.58.4 Member Data Documentation

10.58.4.1 template<typename GeneArray , typename Compare = std::less<GeneArray>> std::vector<std::shared_ptr<GeneArray> > samchon::library::GAPopulation< GeneArray, Compare >::children [private]

Genes representing the population.

Definition at line 67 of file GAPopulation.hpp.

Referenced by samchon::library::GeneticAlgorithm< GeneArray, Compare >::selection().

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/GAPopulation.hpp

10.59 samchon::library::GeneticAlgorithm< GeneArray, Compare > Class Template Reference

A genetic algorithm class.

`#include <GeneticAlgorithm.hpp>`

Public Member Functions

- GeneticAlgorithm (bool unique, double mutationRate=0.015, size_t tournament=10)
  Construct from parameters of Genetic Algorithm.
- auto evolveGeneArray (std::shared_ptr< GeneArray > geneArray, size_t population, size_t generation) const -> std::shared_ptr< GeneArray >
  Evolve a GeneArray.
- auto evolvePopulation (std::shared_ptr< MyPopulation > population) const -> std::shared_ptr< My↩Population >
  Evolve population, a mass of GeneArray(es)

Private Member Functions

- auto selection (std::shared_ptr< MyPopulation > population) const -> std::shared_ptr< GeneArray >
  Select the best GeneArray in population from tournament.
- auto crossover (std::shared_ptr< GeneArray > &parent1, std::shared_ptr< GeneArray > &parent2) const -> std::shared_ptr< GeneArray >
  Create a new GeneArray by crossing over two GeneArray(s)
- void mutate (std::shared_ptr< GeneArray > geneArray) const
  Cause a mutation on the GeneArray.

Private Attributes

- bool unique
  Whether each element (Gene) is unique in their GeneArray.
- double mutationRate
  Rate of mutate ocurrence.
- size_t tournament
  Size of tournament in selection.

10.59.1 Detailed Description

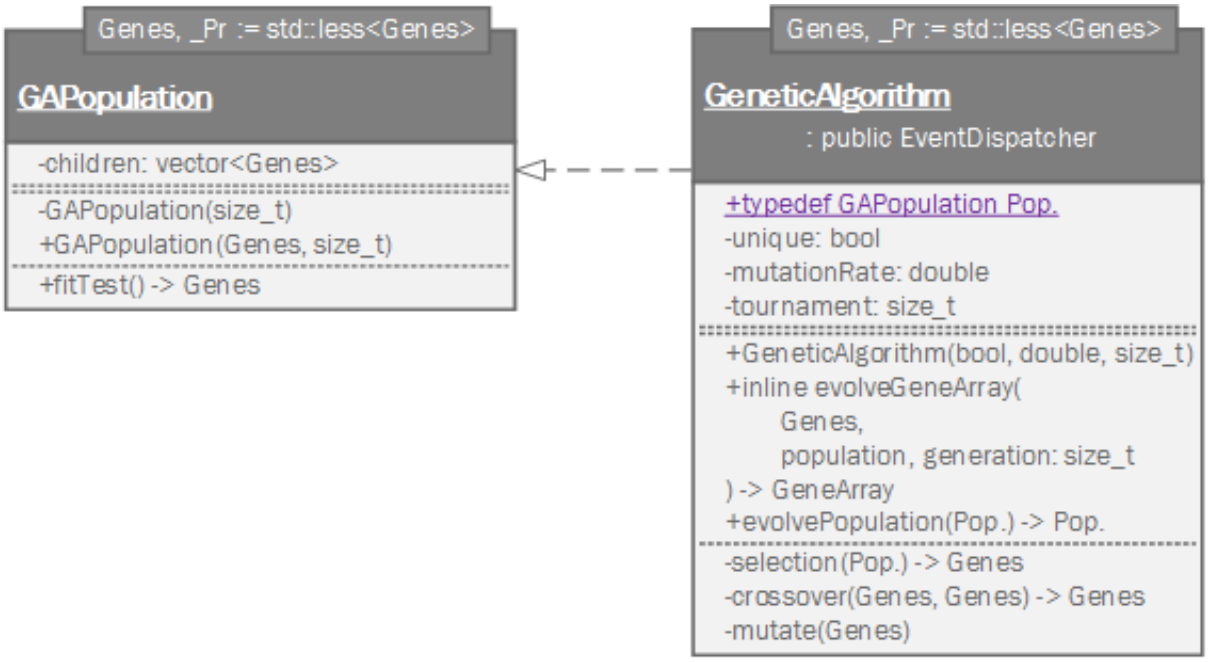
template<typename GeneArray, typename Compare = std::less<GeneArray>>class samchon::library::GeneticAlgorithm< GeneArray, Compare >

A genetic algorithm class.

In the field of artificial intelligence, a genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a metaheuristic) is routinely used to generate useful solutions to optimization and search problems.

Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

- Referenced Wikipedia: https://en.wikipedia.org/wiki/Genetic_algorithm



Template Parameters

GeneArray	An array(std::vector) containing genes as elments; sequnce listing. The GeneArray must be a type of *std::vector*.

Compare	A comparison class (or struct) returns whether left gene is more optimal. Default template parameter of Compare is *std::less<GeneArray>*. It means to compare two std::vector (GeneArray must be a std::vector). Thus, you've to keep follwing rules. If you don't want to follow the rules or want a custom comparison class, you have to realize a comparison class. The following code is an example realizing the comparison class. - GeneArray is inherited from std::vector - GeneArray has custom auto operator<(const GeneArray &) const -> bool

```
template <typename _Ty>
struct MyCompare
{
auto operator()(const _Ty &newObj, const _Ty &prevObj) const -> bool;
};
```

Example source

TSP - Traveling Salesman Problem

examples/tsp/Scheduler.hpp

```
1 #pragma once
2 #include <vector>
3 #include <memory>
4
5 namespace samchon
6 {
7     namespace example
8     {
9         namespace tsp
10         {
11             class Travel;
12
23             struct GAParameters
24             {
25                 double mutationRate;
26                 size_t tornament;
27
28                 size_t population;
29                 size_t generation;
30             };
31
38             class Scheduler
39             {
40             private:
44                 std::shared_ptr<Travel> travel;
45
49                 struct GAParameters gaParameters;
50
51             public:
55                 Scheduler(std::shared_ptr<Travel>, const struct GAParameters &);
56
64                 auto optimize() -> std::shared_ptr<Travel>;
65             };
66         };
67     };
68 };
```

examples/tsp/Travel.hpp

```
1 #pragma once
2 #include <vector>
3 #include <string>
4
5 namespace samchon
6 {
7     namespace example
8     {
9         namespace tsp
10         {
11             class GeometryPoint;
12
24             class Travel
25                 : public std::vector<GeometryPoint*>
26             {
```

```
27              private:
28                  typedef std::vector<GeometryPoint*> super;
29
36                  double distance;
37
38              public:
39                  /* ------------------------------------------------------------
40                      CONSTRUCTORS
41                  ------------------------------------------------------------ */
45                  Travel();
46
55                  Travel(const Travel&);
56
60                  Travel(Travel&&);
61
62              private:
63                  /* ------------------------------------------------------------
64                      CALCULATORS
65                  ------------------------------------------------------------ */
69                  auto calcDistance() const -> double;
70
71              public:
79                  auto operator<(const Travel &) const -> bool;
80
81                  /* ------------------------------------------------------------
82                      EXPORTER
83                  ------------------------------------------------------------ */
96                  auto toString() const -> std::string;
97              };
98          };
99      };
100 };
```

examples/tsp/GeometryPoint.hpp

```
1 #pragma once
2 #include <string>
3
4 namespace samchon
5 {
6     namespace example
7     {
8         namespace tsp
9         {
19              class GeometryPoint
20              {
21              private:
25                  int uid;
26
30                  double longitude;
31
35                  double latitude;
36
37              public:
38                  /* ------------------------------------------------------------
39                      CONSTRUCTORS
40                  ------------------------------------------------------------ */
47                  GeometryPoint(int);
48
55                  GeometryPoint(int, double, double);
56
57                  /* ------------------------------------------------------------
58                      CALCULATOR
59                  ------------------------------------------------------------ */
66                  auto calcDistance(const GeometryPoint &) const -> double;
67
68                  /* ------------------------------------------------------------
69                      EXPORTER
70                  ------------------------------------------------------------ */
79                  auto toString() const -> std::string;
80              };
81          };
82      };
83 };
```

examples/tsp/main.cpp

```
1 #include <iostream>
2 #include <Windows.h>
3
4 #include "Scheduler.hpp"
5 #include "Travel.hpp"
6 #include "GeometryPoint.hpp"
7
8 using namespace std;
9 using namespace samchon::example::tsp;
```

```

#ifdef _WIN64
#   ifdef _DEBUG
#       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "x64/Release/SamchonFramework.lib")
#   endif
#else
#   ifdef _DEBUG
#       pragma comment(lib, "Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "Release/SamchonFramework.lib")
#   endif
#endif

void toClipboard(const string &);

void main()
{
    //GEOMETRY COORPORATES
    shared_ptr<Travel> travel(new Travel());
    for(int i = 0; i < 20; i++)
        travel->push_back(new GeometryPoint(i + 1));

    //OPTIMIZING
    struct GAParameters gaParameters = {.03, 50, 100, 300};

    Scheduler scheduler(travel, gaParameters);
    travel = scheduler.optimize();

    //PRINTING
    string &str = travel->toString();
    toClipboard(str);
    cout << str << endl;

    system("pause");
}

void toClipboard(const string &str)
{
    OpenClipboard(0);
    EmptyClipboard();
    HGLOBAL hg = GlobalAlloc(GMEM_MOVEABLE, str.size());

    if (!hg)
    {
        CloseClipboard();
        return;
    }
    memcpy(GlobalLock(hg), str.c_str(), str.size());

    GlobalUnlock(hg);
    SetClipboardData(CF_TEXT, hg);
    CloseClipboard();
    GlobalFree(hg);
}
```

examples/tsp/Scheduler.cpp

```
#include "Scheduler.hpp"
#   include "Travel.hpp"

#include <samchon/library/GeneticAlgorithm.hpp>

using namespace std;
using namespace samchon::library;
using namespace samchon::example::tsp;

Scheduler::Scheduler(shared_ptr<Travel> travel, const GAParameters &gaParameteres)
{
    this->travel = travel;
    this->gaParameters = gaParameteres;
}
auto Scheduler::optimize() -> shared_ptr<Travel>
{
    GeneticAlgorithm<Travel> geneticAlgorithm
    (
        true,
        gaParameters.mutationRate,
        gaParameters.tornament
    );

    travel =
        geneticAlgorithm.evolveGeneArray
        (
            travel,
            gaParameters.population,
```

```
            gaParameters.generation
        );
    return travel;
}
```

examples/tsp/Travel.cpp

```
#include "Travel.hpp"
#   include "GeometryPoint.hpp"

#include <samchon/library/StringUtil.hpp>

using namespace std;
using namespace samchon::library;
using namespace samchon::example::tsp;

/* ---------------------------------------------------------
    CONSTRUCTORS
--------------------------------------------------------- */
Travel::Travel()
    : super()
{
    distance = INT_MIN;
}
Travel::Travel(const Travel &travel)
    : super(travel)
{
    distance = INT_MIN;
}
Travel::Travel(Travel &&travel)
    : super(move(travel))
{
    distance = travel.distance;
}

/* ---------------------------------------------------------
    CALCULATORS
--------------------------------------------------------- */
auto Travel::calcDistance() const -> double
{
    if(this->distance != INT_MIN)
        return this->distance;

    double distance = 0.0;
    for(size_t i = 1; i < size(); i++)
        distance += at(i-1)->calcDistance(*at(i));

    ((Travel*)this)->distance = distance;
    return distance;
}
auto Travel::operator<(const Travel &travel) const -> bool
{
    return this->calcDistance() < travel.calcDistance();
}

/* ---------------------------------------------------------
    EXPORTER
--------------------------------------------------------- */
auto Travel::toString() const -> string
{
    string str =
        "Distance: " + StringUtil::numberFormat(calcDistance()) + " km\n" +
        "uid    longitude   latitude\n";

    for(size_t i = 0; i < size(); i++)
        str += at(i)->toString() + "\n";

    return move(str);
}
```

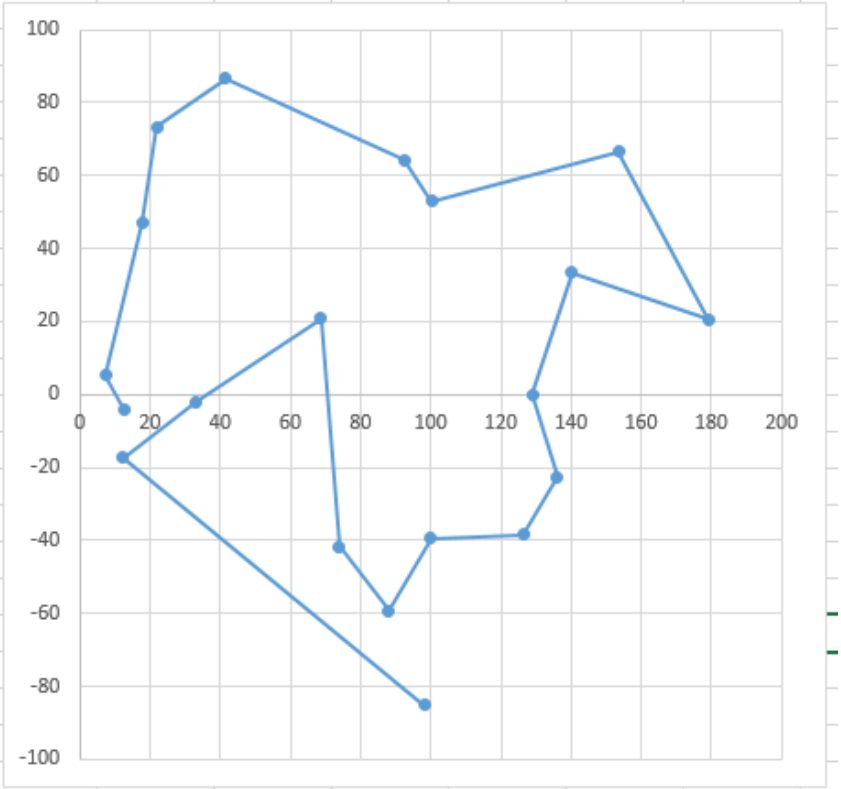
examples/tsp/GeometryPoint.cpp

```
#include "GeometryPoint.hpp"

#include <random>
#include <cmath>
#include <samchon/library/Math.hpp>
#include <samchon/library/StringUtil.hpp>

using namespace std;
using namespace samchon::library;
using namespace samchon::example::tsp;

/* ---------------------------------------------------------
    CONSTRUCTORS
--------------------------------------------------------- */
```

```
GeometryPoint::GeometryPoint(int uid)
{
    this->uid = uid;
    this->longitude = Math::random() * 180.0;
    this->latitude = Math::random() * 180 - 90.0;
}
GeometryPoint::GeometryPoint(int uid, double longitude, double latitude)
{
    this->uid = uid;
    this->longitude = longitude;
    this->latitude = latitude;
}

/* ----------------------------------------------------------
    CALCULATOR
---------------------------------------------------------- */
auto GeometryPoint::calcDistance(const GeometryPoint &point) const -> double
{
    if (longitude == point.longitude && latitude == point.latitude)
        return 0.0;

    double latitude_radian1 = Math::degree_to_radian(this->latitude);
    double latitude_radian2 = Math::degree_to_radian(point.latitude);
    double theta = this->longitude - point.longitude;

    double val =
        sin(latitude_radian1) * sin(latitude_radian2)
        + cos(latitude_radian1) * cos(latitude_radian2) * cos(
    Math::degree_to_radian(theta));

    val = acos(val);
    val = Math::radian_to_degree(val);
    val = val * 60 * 1.1515;
    val = val * 1.609344;

    return val;
}

/* ----------------------------------------------------------
    EXPORTER
---------------------------------------------------------- */
auto GeometryPoint::toString() const -> string
{
    return StringUtil::substitute
        (
            "{1}\t{2}\t{3}",
            uid, longitude, latitude
        );
}
```

Result of the example

Distance: 64,896.70 km		
uid	longitude	latitude
18	12.5233	-3.95529
15	7.23275	5.18342
7	17.7528	46.8665
2	22.0434	73.4133
20	41.6615	86.5372
13	92.427	64.3369
1	100.345	53.1067
9	153.521	66.5119
3	179.118	20.5073
10	140.113	33.3537
17	128.885	0.054607
4	136.012	-22.4915
6	126.352	-38.5031
16	99.9775	-39.3696
11	87.8859	-59.3663
14	73.8477	-41.5867
5	68.6563	20.8528
12	33.2162	-1.92823
8	12.1839	-17.3552
19	98.0967	-85.195



Warning

Be careful for the mistakes of direction or position of Compare.

Most of logical errors failed to access optimal solution are occured by those mistakens.

See also

library::GAPopulation
samchon::library

Author

Jeongho Nam

Definition at line 83 of file GeneticAlgorithm.hpp.

10.59.2 Constructor & Destructor Documentation

10.59.2.1 template<typename GeneArray, typename Compare = std::less<GeneArray>> samchon::library::GeneticAlgorithm< GeneArray, Compare >::GeneticAlgorithm (bool *unique*, double *mutationRate* = 0.015, size_t *tournament* = 10) [inline]

Construct from parameters of Genetic Algorithm.

Parameters

unique	Whether each Gene is unique in their GeneArray
mutationRate	Rate of mutation
tournament	Size of tournament in selection
elitism	Whether to keep the elitest GeneArray

Definition at line 120 of file GeneticAlgorithm.hpp.

References samchon::library::GeneticAlgorithm< GeneArray, Compare >::mutationRate, samchon↩ ::library::GeneticAlgorithm< GeneArray, Compare >::tournament, and samchon::library::GeneticAlgorithm< GeneArray, Compare >::unique.

10.59.3 Member Function Documentation

10.59.3.1 template<typename GeneArray, typename Compare = std::less<GeneArray>> auto samchon::library::GeneticAlgorithm< GeneArray, Compare >::evolveGeneArray (std::shared_ptr< GeneArray > *geneArray*, size_t *population*, size_t *generation*) const -> std::shared_ptr<GeneArray> [inline]

Evolve a GeneArray.

Convinient method accessing to evolvePopulation().

Parameters

geneArray	An initial set of genes; sequence listing
population	Size of population in a generation
generation	Size of generation in evolution

Returns

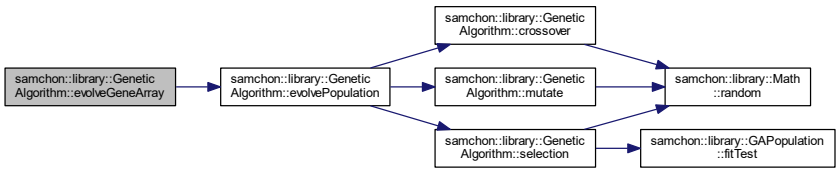
A evolved GeneArray optimally

Definition at line 138 of file GeneticAlgorithm.hpp.

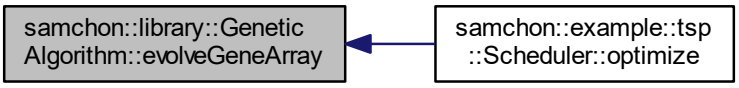
References samchon::library::GeneticAlgorithm< GeneArray, Compare >::evolvePopulation().

Referenced by samchon::example::tsp::Scheduler::optimize().

Here is the call graph for this function:



Here is the caller graph for this function:



10.59.3.2 template<typename GeneArray, typename Compare = std::less<GeneArray>> auto samchon::library::GeneticAlgorithm< GeneArray, Compare >::evolvePopulation (std::shared_ptr< MyPopulation > *population*) const -> std::shared_ptr<MyPopulation> [inline]

Evolve population, a mass of GeneArray(es)

Parameters

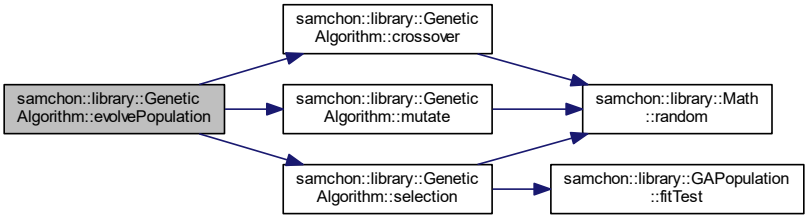
population	An initial population

Definition at line 154 of file GeneticAlgorithm.hpp.

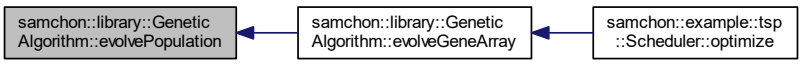
References samchon::library::GeneticAlgorithm< GeneArray, Compare >::crossover(), samchon::library::GeneticAlgorithm< GeneArray, Compare >::mutate(), and samchon::library::GeneticAlgorithm< GeneArray, Compare >::selection().

Referenced by samchon::library::GeneticAlgorithm< GeneArray, Compare >::evolveGeneArray().

Here is the call graph for this function:



Here is the caller graph for this function:



10.59.3.3 template<typename GeneArray, typename Compare = std::less<GeneArray>> auto samchon::library::GeneticAlgorithm< GeneArray, Compare >::selection (std::shared_ptr< MyPopulation > *population*) const -> std::shared_ptr<GeneArray> [inline], [private]

Select the best GeneArray in population from tournament.

Selection is the stage of a genetic algorithm in which individual genomes are chosen from a population for later breeding (using crossover operator). A generic selection procedure may be implemented as follows:

1. The fitness function is evaluated for each individual, providing fitness values, which are then normalized. Normalization means dividing the fitness value of each individual by the sum of all fitness values, so that the sum of all resulting fitness values equals 1.
2. The population is sorted by descending fitness values.
3. Accumulated normalized fitness values are computed (the accumulated fitness value of an individual is the sum of its own fitness value plus the fitness values of all the previous individuals). The accumulated fitness of the last individual should be 1 (otherwise something went wrong in the normalization step).
4. A random number R between 0 and 1 is chosen.
5. The selected individual is the first one whose accumulated normalized value is greater than R.

– Referenced Wekipedia: https://en.wikipedia.org/wiki/Selection_(genetic_algorithm)

Parameters

population	The target of tournament

Returns

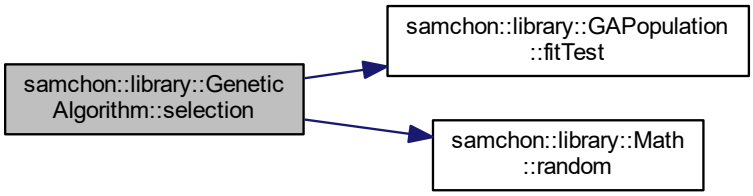
The best genes derived by the tournament

Definition at line 211 of file GeneticAlgorithm.hpp.

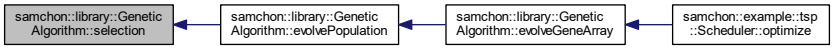
References samchon::library::GAPopulation< GeneArray, Compare >::children, samchon::library::GA↩ Population< GeneArray, Compare >::fitTest(), and samchon::library::Math::random().

Referenced by samchon::library::GeneticAlgorithm< GeneArray, Compare >::evolvePopulation().

Here is the call graph for this function:



Here is the caller graph for this function:



10.59.3.4 template<typename GeneArray, typename Compare = std::less<GeneArray>> auto samchon::library::GeneticAlgorithm< GeneArray, Compare >::crossover (std::shared_ptr< GeneArray > & *parent1*, std::shared_ptr< GeneArray > & *parent2*) const -> std::shared_ptr<GeneArray> [inline], [private]

Create a new GeneArray by crossing over two GeneArray(s)

crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based.

Cross over is a process of taking more than one parent solutions and producing a child solution from them. There are methods for selection of the chromosomes.

- Referenced Wikipedia: https://en.wikipedia.org/wiki/Crossover_(genetic_↩ algorithm)

Parameters

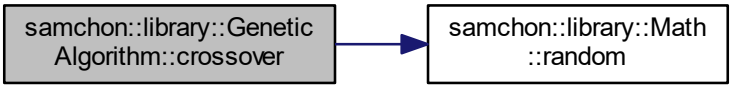
parent1	A parent sequence listing
parent2	A parent sequence listing

Definition at line 240 of file GeneticAlgorithm.hpp.

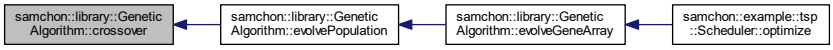
References samchon::library::Math::random().

Referenced by samchon::library::GeneticAlgorithm< GeneArray, Compare >::evolvePopulation().

Here is the call graph for this function:



Here is the caller graph for this function:



10.59.3.5 template<typename GeneArray, typename Compare = std::less<GeneArray>> void samchon::library::GeneticAlgorithm< GeneArray, Compare >::mutate (std::shared_ptr< GeneArray > *geneArray*) const [inline], [private]

Cause a mutation on the GeneArray.

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. It is analogous to biological mutation.

Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence GA can come to better solution by using mutation.

Mutation occurs during evolution according to a user-definable mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search.

- Referenced Wikipedia: https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm)

Note

Muttion is pursuing diversity. Mutation is useful for avoiding the following problem.

When initial set of genes(GeneArray) is far away from optimail, without mutation (only with selection and crossover), the genetic algorithm has a tend to wandering outside of the optimal.

Genes in the GeneArray will be swapped following percentage of the mutationRate.

Parameters

geneArray	A container of genes to mutate

See also

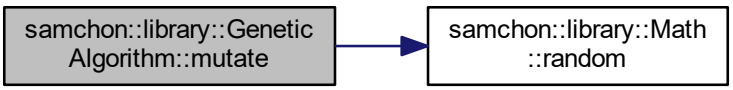
mutationRate;

Definition at line 312 of file GeneticAlgorithm.hpp.

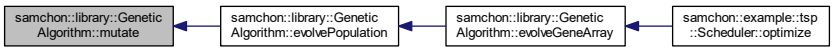
References samchon::library::GeneticAlgorithm< GeneArray, Compare >::mutationRate, and samchon::library::Math::random().

Referenced by samchon::library::GeneticAlgorithm< GeneArray, Compare >::evolvePopulation().

Here is the call graph for this function:



Here is the caller graph for this function:



10.59.4 Member Data Documentation

10.59.4.1 template<typename GeneArray, typename Compare = std::less<GeneArray>> bool samchon::library::GeneticAlgorithm< GeneArray, Compare >::unique [private]

Whether each element (Gene) is unique in their GeneArray.

Definition at line 91 of file GeneticAlgorithm.hpp.

Referenced by samchon::library::GeneticAlgorithm< GeneArray, Compare >::GeneticAlgorithm().

10.59.4.2 template<typename GeneArray, typename Compare = std::less<GeneArray>> double samchon::library::GeneticAlgorithm< GeneArray, Compare >::mutationRate [private]

Rate of mutate ocurrence.

The mutationRate determines the percentage of occurence of mutation in a GeneArray.

Note

- When mutationRate is too high, it is hard to ancitipate studying on genetic algorithm.
- When mutationRate is too low and initial set of genes(GeneArray) is far away from optimal, the evolution tends to wandering outside of he optimal.

Definition at line 104 of file GeneticAlgorithm.hpp.

Referenced by samchon::library::GeneticAlgorithm< GeneArray, Compare >::GeneticAlgorithm(), and samchon::library::GeneticAlgorithm< GeneArray, Compare >::mutate().

10.59.4.3 template<typename GeneArray, typename Compare = std::less<GeneArray>> size_t samchon::library::GeneticAlgorithm< GeneArray, Compare >::tournament [private]

Size of tournament in selection.

Definition at line 109 of file GeneticAlgorithm.hpp.

Referenced by samchon::library::GeneticAlgorithm< GeneArray, Compare >::GeneticAlgorithm().

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/GeneticAlgorithm.hpp

10.60 samchon::example::tsp::GeometryPoint Class Reference

A geometry coordinates (point)

```
#include <GeometryPoint.hpp>
```

Public Member Functions

- GeometryPoint (int)

  Construct from uid.
- GeometryPoint (int, double, double)

  Construct from uid and geometry coordinates.
- auto calcDistance (const GeometryPoint &) const -> double

  Calculate distance between target Branch.
- auto toString () const -> std::string

  Convert the Branch to String.

Private Attributes

- int uid

  An unique id for uniqueness.
- double longitude

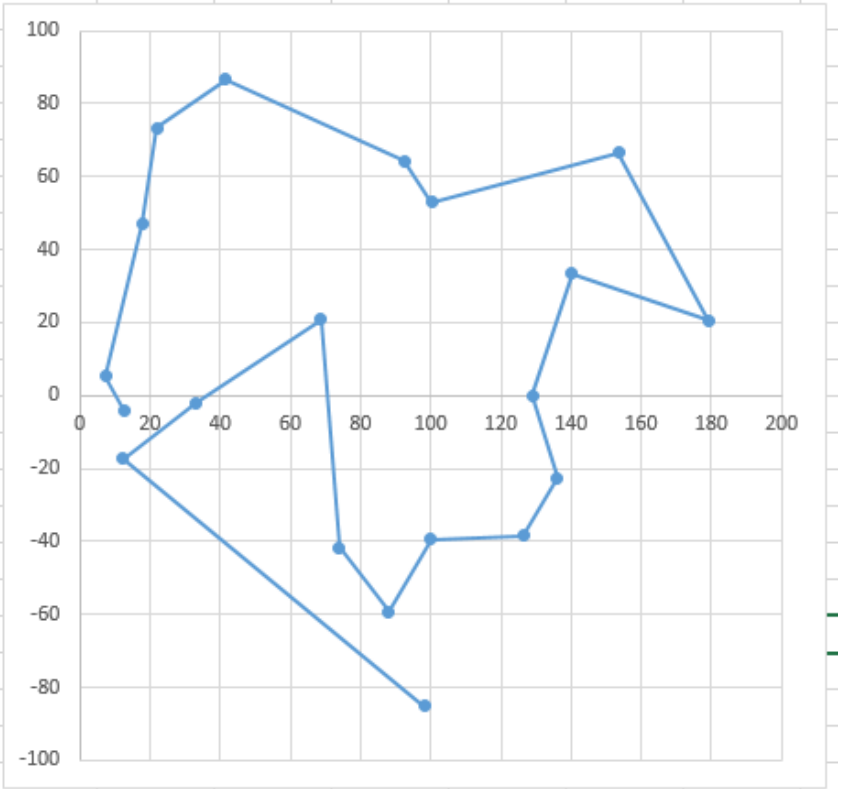
  The longitude; coordinates X.
- double latitude

  The latitude, coordinates Y.

10.60.1 Detailed Description

A geometry coordinates (point)

Distance: 64,896.70 km		
uid	longitude	latitude
18	12.5233	-3.95529
15	7.23275	5.18342
7	17.7528	46.8665
2	22.0434	73.4133
20	41.6615	86.5372
13	92.427	64.3369
1	100.345	53.1067
9	153.521	66.5119
3	179.118	20.5073
10	140.113	33.3537
17	128.885	0.054607
4	136.012	-22.4915
6	126.352	-38.5031
16	99.9775	-39.3696
11	87.8859	-59.3663
14	73.8477	-41.5867
5	68.6563	20.8528
12	33.2162	-1.92823
8	12.1839	-17.3552
19	98.0967	-85.195



Author

Jeongho Nam

Definition at line 19 of file GeometryPoint.hpp.

10.60.2 Constructor & Destructor Documentation

10.60.2.1 GeometryPoint::GeometryPoint (int *uid*)

Construct from uid.

Geometry coordinates, longitude and latitude will have a random value between

- Longitude between 0 and 180.
- Latitude between -90 and 90.

Definition at line 15 of file GeometryPoint.cpp.

10.60.2.2 GeometryPoint::GeometryPoint (int *uid,* double *longitude,* double *latitude*)

Construct from uid and geometry coordinates.

Parameters

longitude	Longitude, a coordinate of the geometry point
latitude	Latitude, a coordinate of the geometry point

Definition at line 21 of file GeometryPoint.cpp.

10.60.3 Member Function Documentation

10.60.3.1 auto GeometryPoint::calcDistance (const GeometryPoint & *point*) const -> double

Calculate distance between target Branch.

Referenced from http://thesunrises.tistory.com/958 (Geometry points to killometers)

Parameters

branch	The target Branch to calculate

Definition at line 31 of file GeometryPoint.cpp.

10.60.3.2 auto GeometryPoint::toString ( ) const -> std::string

Convert the Branch to String.

Have of form of tab and enter delimeters for Excel.

- `{$uid}  {$longitude}  {$latitude}`

Returns

A string represents the GeometryPoint

Definition at line 55 of file GeometryPoint.cpp.

10.60.4 Member Data Documentation

10.60.4.1 int samchon::example::tsp::GeometryPoint::uid [private]

An unique id for uniqueness.

Definition at line 25 of file GeometryPoint.hpp.

10.60.4.2 double samchon::example::tsp::GeometryPoint::longitude [private]

The longitude; coordinates X.

Definition at line 30 of file GeometryPoint.hpp.

10.60.4.3 double samchon::example::tsp::GeometryPoint::latitude [private]

The latitude, coordinates Y.

Definition at line 35 of file GeometryPoint.hpp.

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

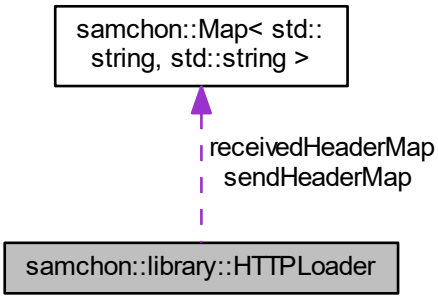
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/tsp/GeometryPoint.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/tsp/GeometryPoint.cpp

10.61 samchon::library::HTTPLoader Class Reference

A http, web-page loader.

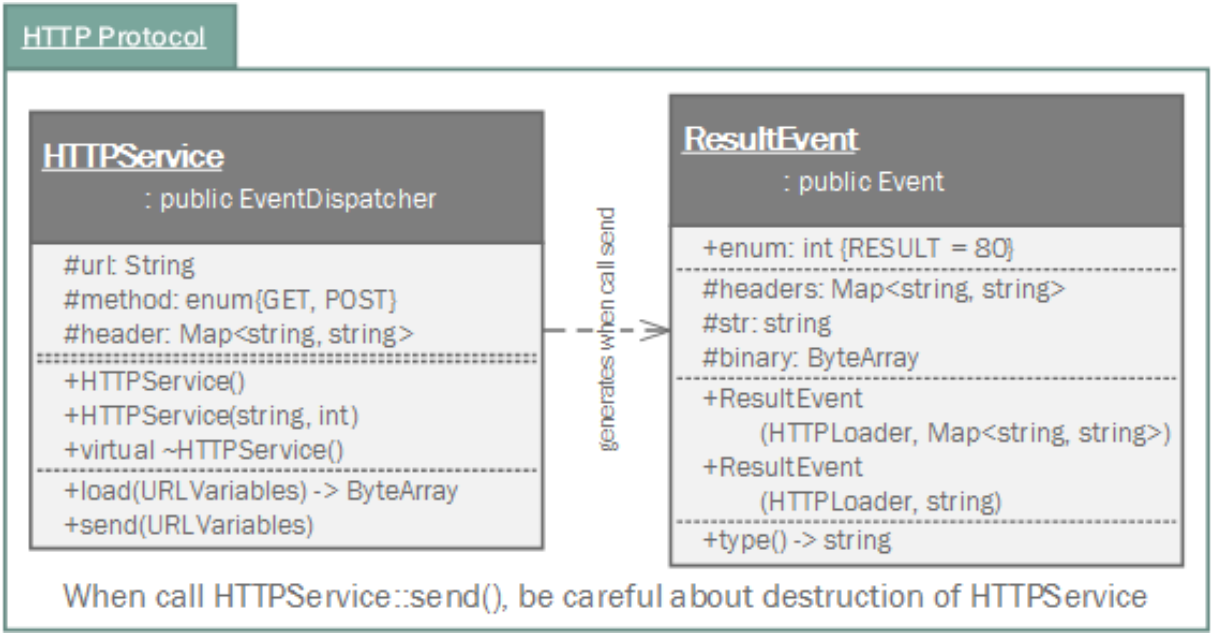
```
#include <HTTPLoader.hpp>
```

Collaboration diagram for samchon::library::HTTPLoader:



10.61.1 Detailed Description

A http, web-page loader.



Note

Depreciated

See also

samchon::library

Author

Jeongho Nam

Definition at line 25 of file HTTPLoader.hpp.

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

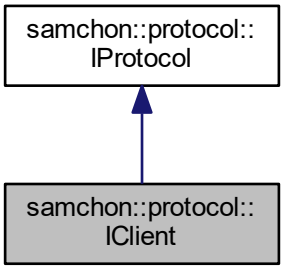
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/HTTPLoader.hpp

10.62 samchon::protocol::IClient Class Reference

An interface for a client.

`#include <IClient.hpp>`

Collaboration diagram for samchon::protocol::IClient:



Public Member Functions

- IClient ()
  Default Constructor.
- virtual void listen ()
  Listens message from a related system.
- virtual void sendData (std::shared_ptr< Invoke >)
  Sends message to a related system.

Protected Member Functions

- virtual auto BUFFER_SIZE () const -> size_t
  Buffer size of network I/O.
- virtual void _replyData (std::shared_ptr< Invoke >)
  A method for pre-processing replied Invoke message.

Protected Attributes

- Socket * socket
  Socket for network I/O.
- std::mutex * sendMtx
  A mutex for sending message.

10.62.1 Detailed Description

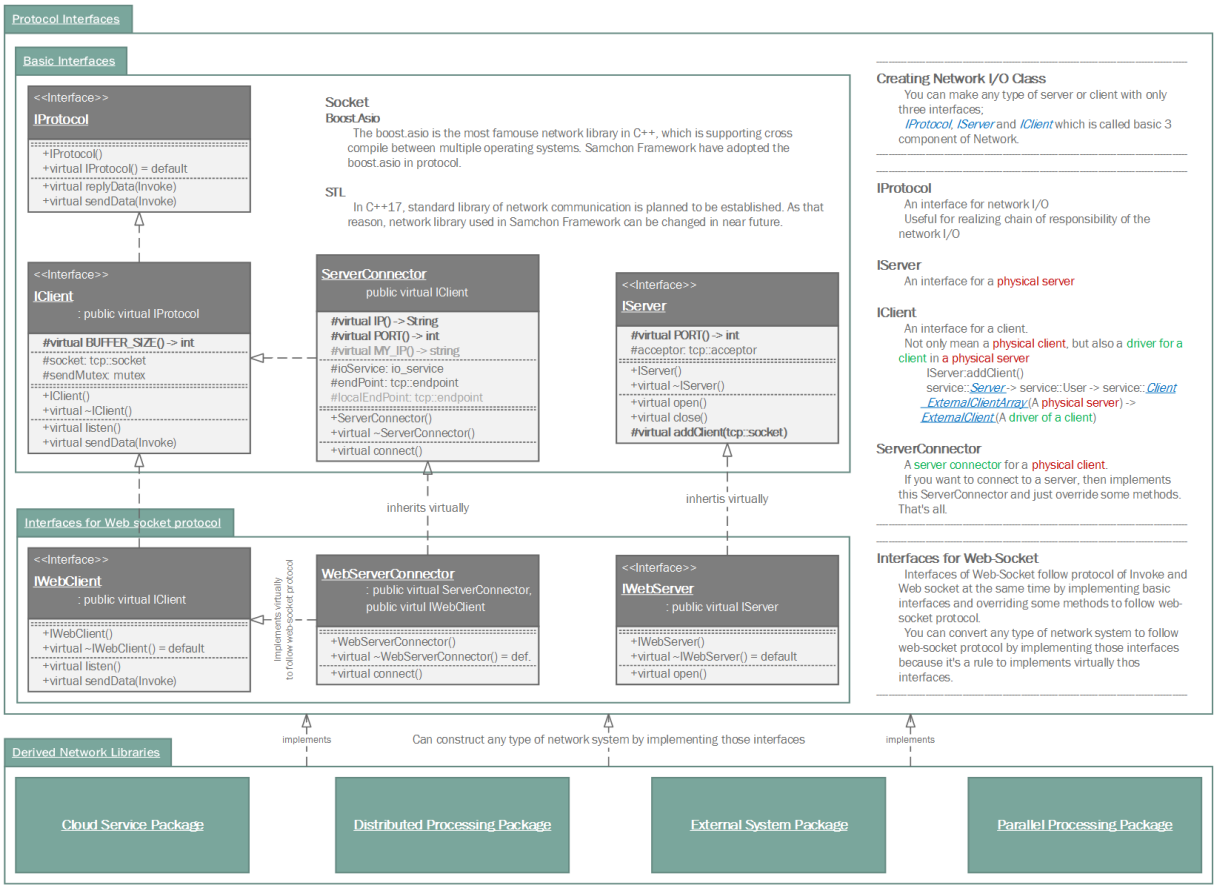
An interface for a client.

IClient is an interface not only for a physical client, but also a driver for a client in a physical server.

- IServer:addClient()
  - service::Server -> service::User -> service::Client
  - ExternalClientArray (A physical server) -> ExternalClient (A driver of a client)

IClient is one of the basic 3 + 1 components that can make any type of network system in Samchon Framework with IProtocol and IServer. Looking around classes in Samchon Framework, you can see all classes related with client are implemented from the IClient.

IClient takes a role of real network I/O. IClient sends data to otherside network system from IProtocol::sendData(). IClient also gets data from otherside network system, generates standard Invoke message from parsing the replied data and shift the Invoke message to IProtocol::replyData().



Author

Jeongho Nam

Definition at line 50 of file IClient.hpp.

10.62.2 Constructor & Destructor Documentation

10.62.2.1 IClient::IClient ( )

Default Constructor.

Definition at line 28 of file IClient.cpp.

10.62.3 Member Function Documentation

10.62.3.1 auto IClient::BUFFER_SIZE ( ) const -> size_t [protected], [virtual]

Buffer size of network I/O.

Definition at line 23 of file IClient.cpp.

10.62.3.2 void IClient::listen ( ) [virtual]

Listens message from a related system.

Note

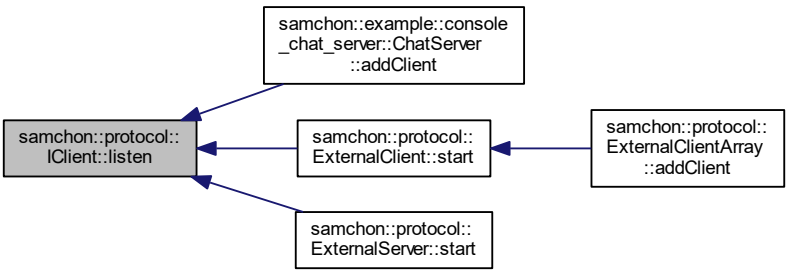
It holds (monopolies) a thread.

Reimplemented in samchon::protocol::IWebClient.

Definition at line 43 of file IClient.cpp.

Referenced by samchon::example::console_chat_server::ChatServer::addClient(), samchon::protocol::↩ ExternalClient::start(), and samchon::protocol::ExternalServer::start().

Here is the caller graph for this function:



10.62.3.3 void IClient::sendData (std::shared_ptr< Invoke >) [virtual]

Sends message to a related system.

Parameters

invoke	Invoke message to send

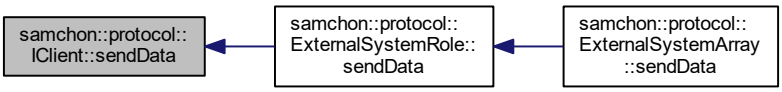
Reimplemented from samchon::protocol::IProtocol.

Reimplemented in samchon::protocol::master::DistributedSystem, samchon::protocol::service::Client, samchon::protocol::IWebClient, and samchon::protocol::slave::SlaveSystem.

Definition at line 224 of file IClient.cpp.

Referenced by samchon::protocol::ExternalSystemRole::sendData().

Here is the caller graph for this function:



10.62.3.4 void IClient::_replyData (std::shared_ptr< Invoke >) [protected],[virtual]

A method for pre-processing replied Invoke message.

You can realize a pre-processing about the replied Invoke message by overriding the IClient::_replyData().

Archiving a history log, filtering and truncating, let handling the message with a new thread, calculating performance by estimating elapsed time about the Invoke message, and so on. Anything you can define pre-processing works before IProtocol::replyData().

- Archive history log -> cloud service
- Estimating performance -> distributed and parallel processing system

Parameters

invoke	An invoke message to be pre-processed and shifted to IProtocol::replyData().

Reimplemented in samchon::protocol::master::ParallelSystem, samchon::protocol::slave::SlaveSystem, and samchon::protocol::slave::ParallelSystem.

Definition at line 245 of file IClient.cpp.

10.62.4 Member Data Documentation

10.62.4.1 Socket* samchon::protocol::IClient::socket [protected]

Socket for network I/O.

Definition at line 57 of file IClient.hpp.

Referenced by samchon::protocol::ExternalClientArray::addClient(), samchon::example::console_chat_server::ChatClient::ChatClient(), samchon::protocol::ServerConnector::connect(), and samchon::protocol::service::Client::sendData().

10.62.4.2 std::mutex* samchon::protocol::IClient::sendMtx [protected]

A mutex for sending message.

Definition at line 62 of file IClient.hpp.

Referenced by samchon::protocol::service::Client::sendData().

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

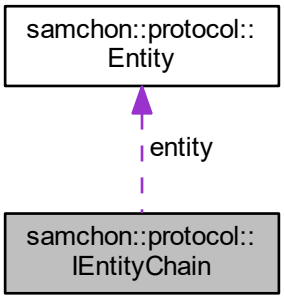
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IClient.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IClient.cpp

10.63 samchon::protocol::IEntityChain Class Reference

A chain of entity.

```
#include <IEntityChain.hpp>
```

Collaboration diagram for samchon::protocol::IEntityChain:



Public Member Functions

- IEntityChain (Entity *)

  Default Constructor.

Protected Attributes

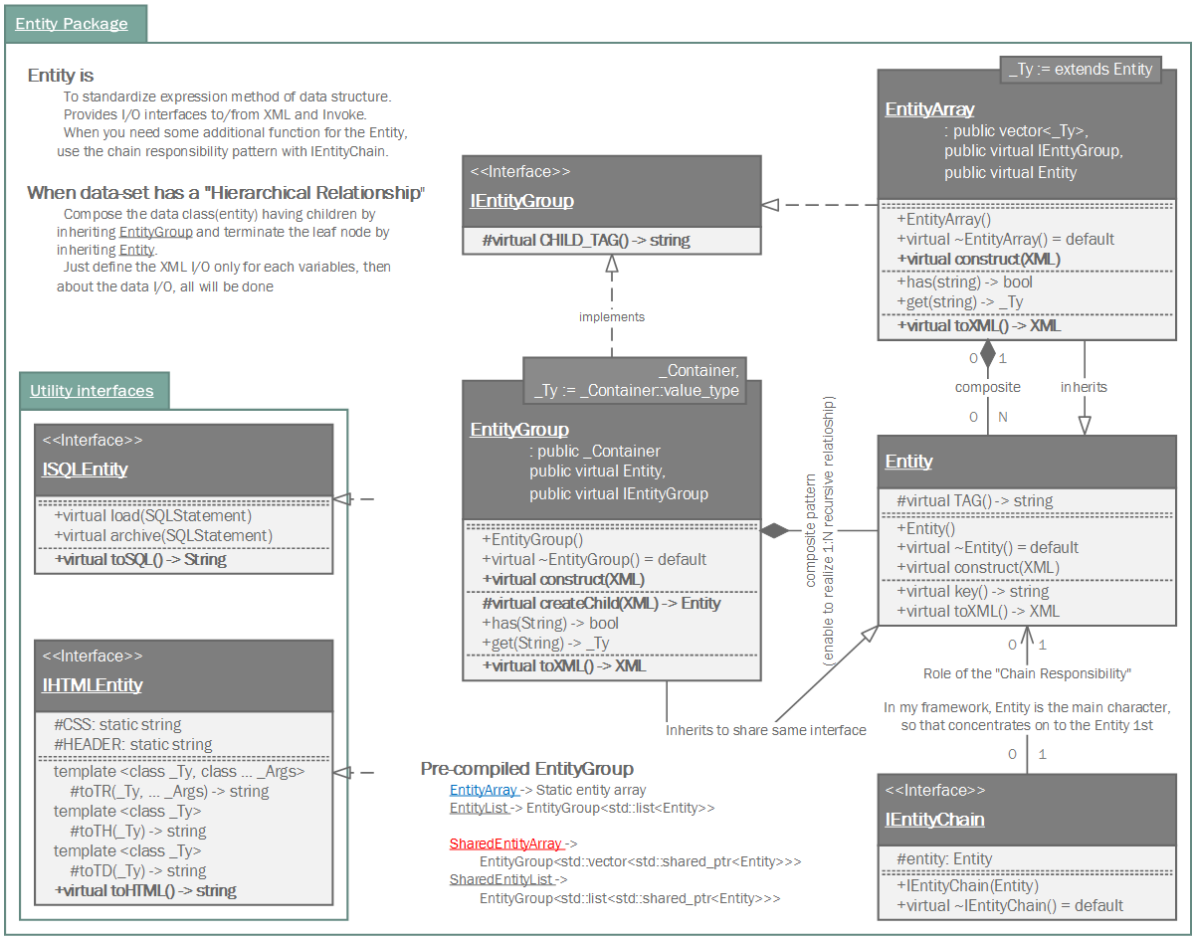
- Entity * entity

  An entity related with the chain.

10.63.1 Detailed Description

A chain of entity.

IEntityChain is an interface takes role of control about an entity by chain of responsibliy pattern. It's suitable for a supplementary processes are required on an entity like calculation or historical regression about an entity.



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 24 of file IEntityChain.hpp.

10.63.2 Constructor & Destructor Documentation

10.63.2.1 IEntityChain::IEntityChain (Entity * *entity*)

Default Constructor.

Parameters

entity	A related entity.

Definition at line 6 of file IEntityChain.cpp.

References entity.

10.63.3 Member Data Documentation

10.63.3.1 Entity* samchon::protocol::IEntityChain::entity [protected]

An entity related with the chain.

Definition at line 30 of file IEntityChain.hpp.

Referenced by IEntityChain().

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IEntityChain.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IEntityChain.cpp

10.64 samchon::protocol::IEntityGroup Class Reference

An iternface for entity group.

```
#include <IEntityGroup.hpp>
```

Public Member Functions

- IEntityGroup ()
  Default Constructor.

Protected Member Functions

- virtual auto CHILD_TAG () const -> std::string=0
  A tag name of children.

10.64.1 Detailed Description

An iternface for entity group.

IEntityGroup is an interface for a container of entity. It does not have any special role or function. The IEntityGorup is only for discriminate an entity is single instance or gorup of entity.

- Group of entity is divided into EntityArray and EntityGroup.
  - Even they are template classes so that cannot be discriminated globally.
- dynamic_cast<IEntityGroup>(*some entity instance*)
  - If true then, it's not single entity instance, but group of entity.
  - False then, it's a single instance.

Author

Jeongho Nam

Definition at line 32 of file IEntityGroup.hpp.

10.64.2 Constructor & Destructor Documentation

10.64.2.1 IEntityGroup::IEntityGroup ( )

Default Constructor.

Definition at line 5 of file IEntityGroup.cpp.

10.64.3 Member Function Documentation

10.64.3.1 virtual auto samchon::protocol::IEntityGroup::CHILD_TAG ( ) const -> std::string [protected], [pure virtual]

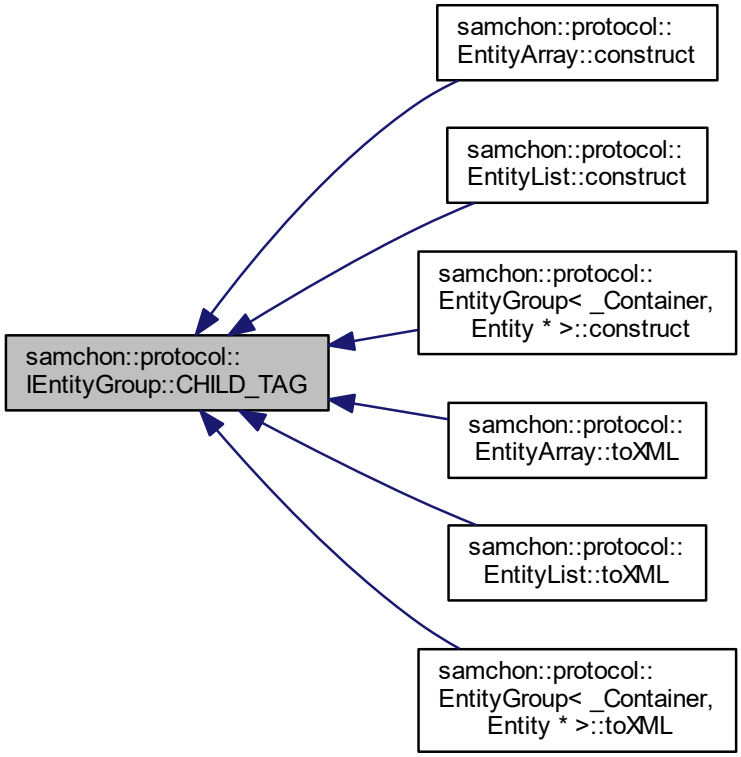
A tag name of children.

< TAG>
<CHILD_TAG />
<CHILD_TAG />
</TAG>

Implemented in samchon::protocol::Invoke, samchon::namtree::NTCriteria, samchon::protocol::ExternalSystemArray, samchon::protocol::ExternalSystem, samchon::namtree::NTParameter, samchon::example::chat_service::ChatRoomArray, samchon::protocol::InvokeHistoryArray, samchon::namtree::NTParameterArray, and samchon::library::FTFolder.

Referenced by samchon::protocol::EntityArray< _Ty >::construct(), samchon::protocol::EntityList< _Ty >::construct(), samchon::protocol::EntityGroup< _Container, Entity * >::construct(), samchon::protocol::EntityArray< _Ty >::toXML(), samchon::protocol::EntityList< _Ty >::toXML(), and samchon::protocol::EntityGroup< _Container, Entity * >::toXML().

Here is the caller graph for this function:



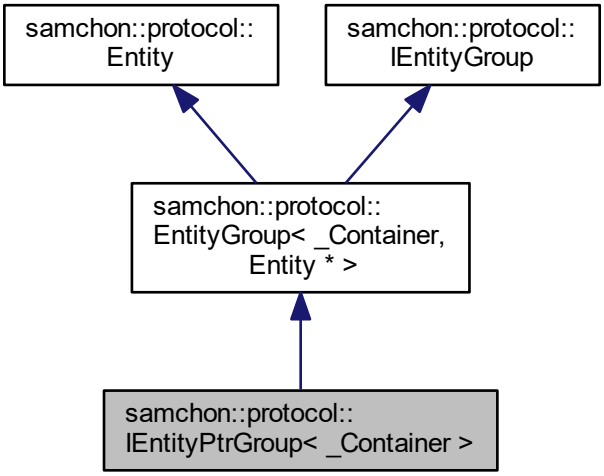
The documentation for this class was generated from the following files:

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IEntityGroup.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IEntityGroup.cpp

10.65 samchon::protocol::IEntityPtrGroup< _Container > Class Template Reference

`#include <IEntityPtrGroup.hpp>`

Collaboration diagram for samchon::protocol::IEntityPtrGroup< _Container >:



Additional Inherited Members

10.65.1 Detailed Description

template<typename _Container>class samchon::protocol::IEntityPtrGroup< _Container >

EntityPtrGroup is an EntityGroup having children Entity as their pointers.

Definition at line 14 of file IEntityPtrGroup.hpp.

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IEntityPtrGroup.hpp

10.66 samchon::protocol::IHTMLEntity Class Reference

An interface supporting conversion to html.

`#include <IHTMLEntity.hpp>`

Public Member Functions

- IHTMLEntity ()
  Default Constructor.
- virtual auto toHTML () const -> std::string=0
  Get an html string represents the Entity.

Static Public Member Functions

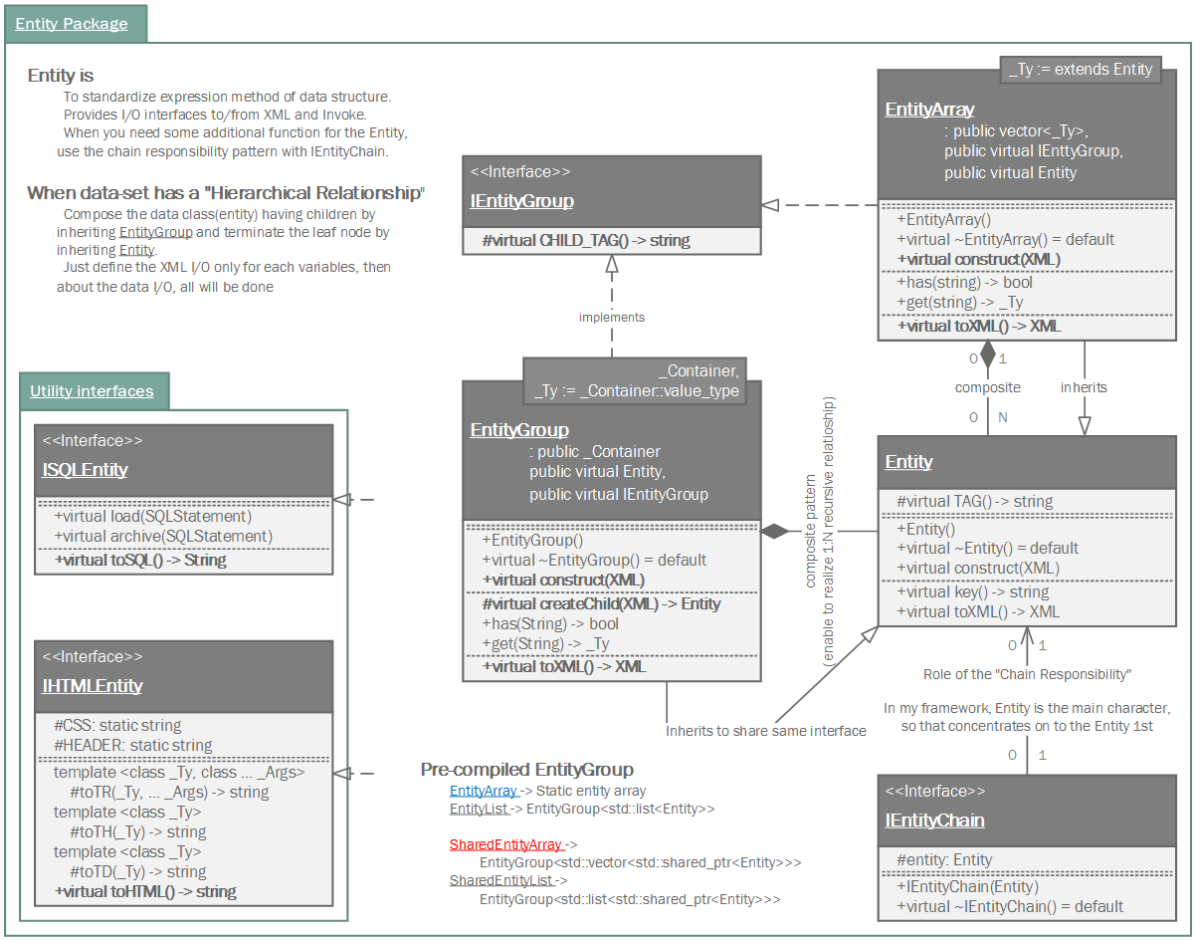
- template<typename _Ty , typename... _Args>
  static auto toTH (const _Ty &val, const _Args &...args) -> std::string
  Get a string represents a <th> tag.
- template<typename _Ty , typename... _Args>
  static auto toTR (const _Ty &val, const _Args &...args) -> std::string
  Get a string represents a <tr> tag.

10.66.1 Detailed Description

An interface supporting conversion to html.

IHTMEntity is an interface supporting conversion method to html tags. The IHTMLEntity is used to documentation or printing data of the entity to web-screen; In C++, documentation is main reason to implementing the IHTMLEntity and in JS, printing on web-screen is main reason.

IHTMLEntity provides abstract method toHTML() and varadic template methods for constituting a table. I'm planning to support lots of utility methods for html tags but I've not exprienced developing the homepage or html service. If you're familiar with the html, please contribute to the IHTMLEntity interface.



Example source

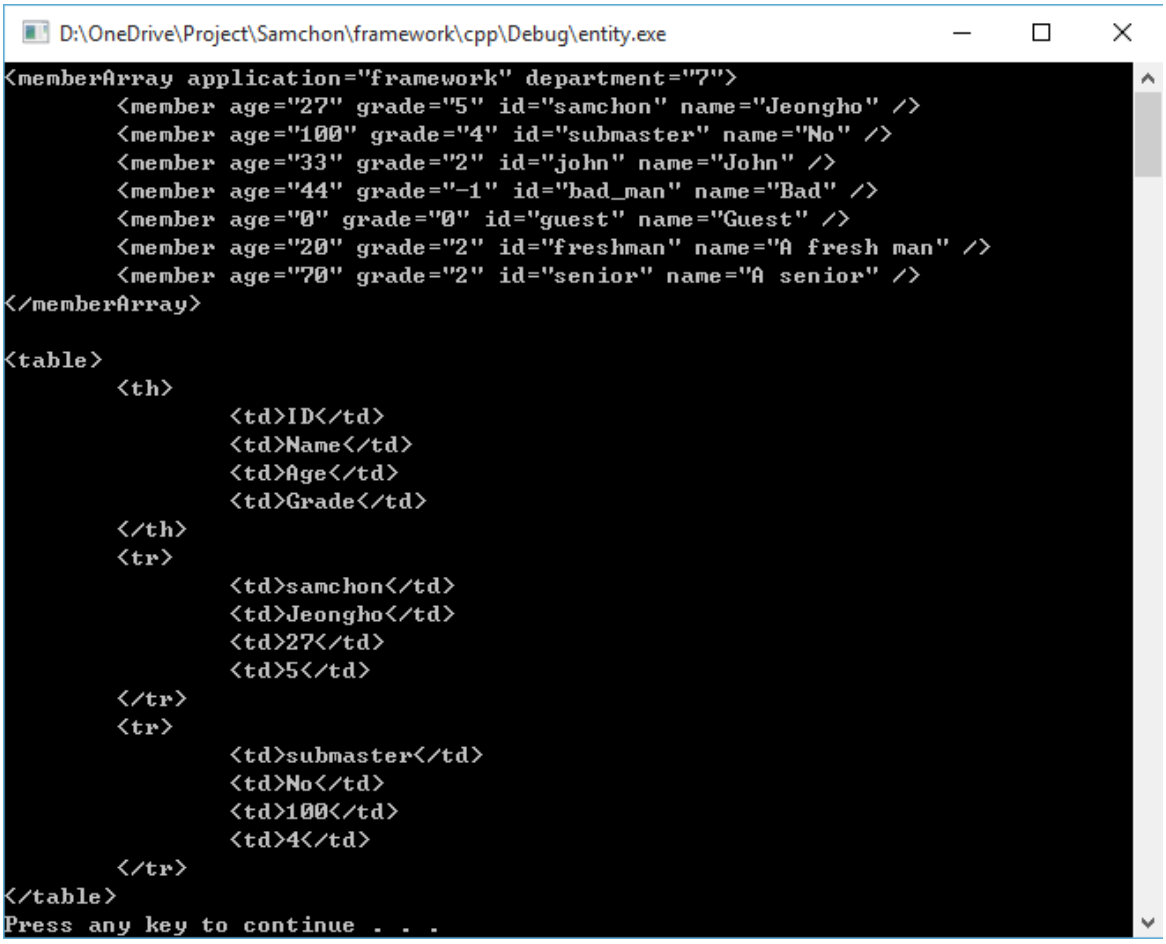
```
#include <iostream>

#include <samchon/protocol/Entity.hpp>
#include <samchon/protocol/IHTMLEntity.hpp>
#include <samchon/protocol/SharedEntityArray.hpp>

#include <samchon/library/XML.hpp>

```

```
#ifdef _WIN64
#   ifdef _DEBUG
#       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "x64/Release/SamchonFramework.lib")
#   endif
#else
#   ifdef _DEBUG
#       pragma comment(lib, "Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "Release/SamchonFramework.lib")
#   endif
#endif

using namespace std;
using namespace samchon::library;
using namespace samchon::protocol;

class Member
    : public Entity, public virtual IHTMLEntity
{
protected:
    typedef Entity super;

    string id;
    string name;
    int age;
    int grade;

public:
    /* ----------------------------------------------------------------------
        CONSTRUCTORS
    ---------------------------------------------------------------------- */
    Member()
        : super(), IHTMLEntity()
    {
    };
    Member(const string &id, const string &name, int age, int grade)
        : super(), IHTMLEntity()
    {
        this->id = id;
        this->name = name;
        this->age = age;
        this->grade = grade;
    };
    virtual ~Member() = default;

    virtual void construct(shared_ptr<XML> xml) override
    {
        this->id = xml->getProperty("id");
        this->name = xml->getProperty("name");
        this->age = xml->getProperty<int>("age");
        this->grade = xml->getProperty<int>("grade");
    };

    /* ----------------------------------------------------------------------
        GETTERSs
    ---------------------------------------------------------------------- */
    virtual auto key() const -> std::string override
    {
        return this->id;
    };

    /* ----------------------------------------------------------------------
        XML EXPORTERS
    ---------------------------------------------------------------------- */
    virtual auto TAG() const -> string override
    {
        return "member";
    };
    virtual auto toXML() const -> shared_ptr<XML>
    {
        shared_ptr<XML> &xml = super::toXML();
        xml->setProperty("id", id);
        xml->setProperty("name", name);
        xml->setProperty("age", age);
        xml->setProperty("grade", grade);

        return move(xml);
    };
    virtual auto toHTML() const -> string
    {
        return toTR(id, name, age, grade);
    };
};

class MemberArray
```

```
    : public SharedEntityArray<Member>,
    public virtual IHTMLEntity
{
protected:
    typedef SharedEntityArray<Member> super;

    string application;
    int department;
    Member *chief;

public:
    /* ----------------------------------------------------------------------
        CONSTRUCTORS
    ---------------------------------------------------------------------- */
    MemberArray()
        : super(), IHTMLEntity()
    {
        this->chief = nullptr;
    };
    virtual ~MemberArray() = default;

    // You don't need to consider children(Member) objects
    // Just concentrate on constructing MemberArray's own member variables
    virtual void construct(shared_ptr<XML> xml) override
    {
        super::construct(xml);

        this->application = xml->getProperty("application");
        this->department = xml->getProperty<int>("department");

        if(xml->hasProperty("chief") == true && this->has( xml->getProperty("chief") ) == true)
            this->chief = this->get( xml->getProperty("cheif") ).get();
    };

protected:
    //FACTORY METHOD FOR MEMBER
    virtual auto createChild(shared_ptr<XML> = nullptr) -> Member* override
    {
        return new Member();
    };

    /* ----------------------------------------------------------------------
        XML EXPORTERS
    ---------------------------------------------------------------------- */
public:
    virtual auto TAG() const -> string override
    {
        return "memberArray";
    };
    virtual auto CHILD_TAG() const -> string override
    {
        return "member";
    };

    // You don't need to consider children(Member) objects
    // Just concentrate on expressing MemberArray's own member variables
    virtual auto toXML() const -> shared_ptr<XML>
    {
        shared_ptr<XML> &xml = super::toXML();
        xml->setProperty("application", application);
        xml->setProperty("department", department);

        if(chief != nullptr)
            xml->setProperty("cheif", chief->key());

        return move(xml);
    };
    virtual auto toHTML() const -> string
    {
        string html = "<table>\n";
        html += toTH("ID", "Name", "Age", "Grade") + "\n";

        for (size_t i = 0; i < 2; i++)
            html += at(i)->toHTML() + "\n";

        html += "</table>";
        return move(html);
    };
};

void main()
{
    string str = string("") +
        "<memberArray application='framework' department='7' cheif='samchon'>\n" +
        "   <member id='samchon' name='Jeongho Nam' age='27' grade='5' />" +
        "   <member id='submaster' name='No Name' age='100' grade='4' />" +
        "   <member id='john' name='John Doe' age='33' grade='2' />" +
```

```
        "   <member id='bad_man' name='Bad Man' age='44' grade='-1' />" +
        "   <member id='guest' name='Guest' age='0' grade='0' />" +
        "</memberArray>";
    shared_ptr<XML> xml(new XML(str));

    MemberArray memberArray;
    memberArray.construct(xml);

    memberArray.emplace_back(new Member("freshman", "A fresh man", 20, 2));
    memberArray.emplace_back(new Member("senior", "A senior", 70, 2));

    cout << memberArray.toXML()->toString() << endl << endl;
    cout << memberArray.toHTML() << endl;
    system("pause");
}
```

Result of the example



Author

Jeongho Nam

Definition at line 38 of file IHTMLEntity.hpp.

10.66.2 Constructor & Destructor Documentation

10.66.2.1 IHTMLEntity::IHTMLEntity ( )

Default Constructor.

Definition at line 9 of file IHTMLEntity.cpp.

10.66.3 Member Function Documentation

10.66.3.1 template<typename _Ty , typename... _Args> static auto samchon::protocol::IHTMLEntity::toTH (const _Ty & *val,* const _Args &... *args*) -> std::string [inline], [static]

Get a string represents a <th> tag.

Template Parameters

_Ty	Type of an argument to be contained with a <td> tag.
_Args	Left varadic template arguments' type

Returns a string represents a title columns with <th> tag by varadic template method.

Supported parameters: that can converted to string

- number
  - (unsigned) short
  - (unsigned) long
  - (unsigned) long long
  - (unsigned) int
  - float
  - double
  - long double
- string
  - std::string
  - WeakString

Parameters

val	An element to be contained with <td> a tag.
args	Left argument to be contained with <td> tags.

Returns

A string of <th> tag represents title columns.

Definition at line 86 of file IHTMLEntity.hpp.

10.66.3.2 template<typename _Ty , typename... _Args> static auto samchon::protocol::IHTMLEntity::toTR (const _Ty & *val*, const _Args &... *args*) -> std::string [inline], [static]

Get a string represents a <tr> tag.

Template Parameters

_Ty	Type of an argument to be contained with a <tr> tag.
_Args	Left varadic template arguments' type

Returns a string represents a data columns of a row with <tr> tag by varadic template method.

Supported parameters: that can converted to string

- number
  - (unsigned) short
  - (unsigned) long
  - (unsigned) long long
  - (unsigned) int
  - float
  - double
  - long double
- string
  - std::string
  - WeakString

Parameters

val	An element to be contained with <tr> a tag.
args	Left argument to be contained with <tr> tags.

Returns

A string of <th> tag represents data columns of a row.

Definition at line 139 of file IHTMLEntity.hpp.

10.66.3.3 virtual auto samchon::protocol::IHTMLEntity::toHTML ( ) const -> std::string [pure virtual]

Get an html string represents the Entity.

Returns

A string represents the Entity by html tags.

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IHTMLEntity.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IHTMLEntity.cpp

10.67 samchon::IndexPair< _Ty > Class Template Reference

A pair of index and its value(_Ty)

`#include <IndexPair.hpp>`

Public Member Functions

- auto getIndex () const -> size_t
  Get index.
- auto getValue () -> _Ty &
  Get const value.
- auto getValue () const -> const _Ty &
  Get value.

10.67.1 Detailed Description

template<class _Ty>class samchon::IndexPair< _Ty >

A pair of index and its value(_Ty)

IndexPair is a std::pair<size_t, _Ty> class.

IndexPair expresses index and value. It's useful for expressing finding literals, specified object, calculating mininum or maximum with its position.

[Inherited]

This class couples together a pair of values, which may be of different types (T1 and T2). The individual values can be accessed through its public members first and second.

Pairs are a particular case of tuple.

Referenced comments of std::pair

- http://www.cplusplus.com/reference/utility/pair/

Author

Jeongho Nam

Definition at line 29 of file IndexPair.hpp.

10.67.2 Member Function Documentation

10.67.2.1 template<class _Ty> auto samchon::IndexPair< _Ty >::getIndex ( ) const -> size_t [inline]

Get index.

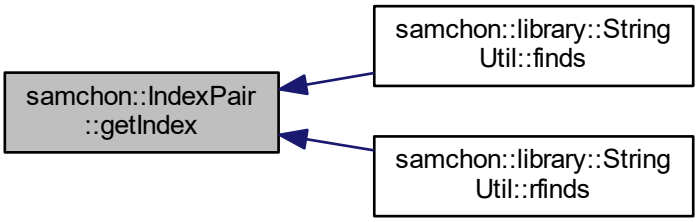
Returns

Index of the pair

Definition at line 46 of file IndexPair.hpp.

Referenced by samchon::library::StringUtil::finds(), and samchon::library::StringUtil::rfinds().

Here is the caller graph for this function:



10.67.2.2 template<class _Ty> auto samchon::IndexPair< _Ty >::getValue ( ) -> _Ty& [inline]

Get const value.

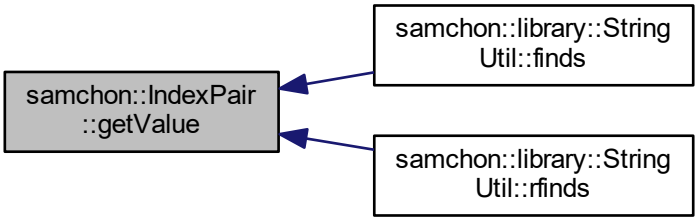
Returns

Reference of value of the pair

Definition at line 56 of file IndexPair.hpp.

Referenced by samchon::library::StringUtil::finds(), and samchon::library::StringUtil::rfinds().

Here is the caller graph for this function:



10.67.2.3 template<class _Ty> auto samchon::IndexPair< _Ty >::getValue ( ) const -> const _Ty& [inline]

Get value.

Returns

Const reference of value of the pair

Definition at line 66 of file IndexPair.hpp.

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/IndexPair.hpp

10.68 samchon::example::packer::Instance Class Reference

A physical instance.

`#include <Instance.hpp>`

Public Member Functions

- Instance (const std::string &name, int price, int volume, int weight)
  Construct from instance.
- auto getName () const -> std::string
  Get name.
- auto getPrice () const -> int
  Get price.
- auto getVolume () const -> int
  Get volume.
- auto getWeight () const -> int
  Get weight.
- virtual auto toString () const -> std::string
  Return a string represents the Instance.

Protected Attributes

- std::string name

  Name represent the Instance.
- int price

  Price of an instance -> 1,000 won.
- int volume

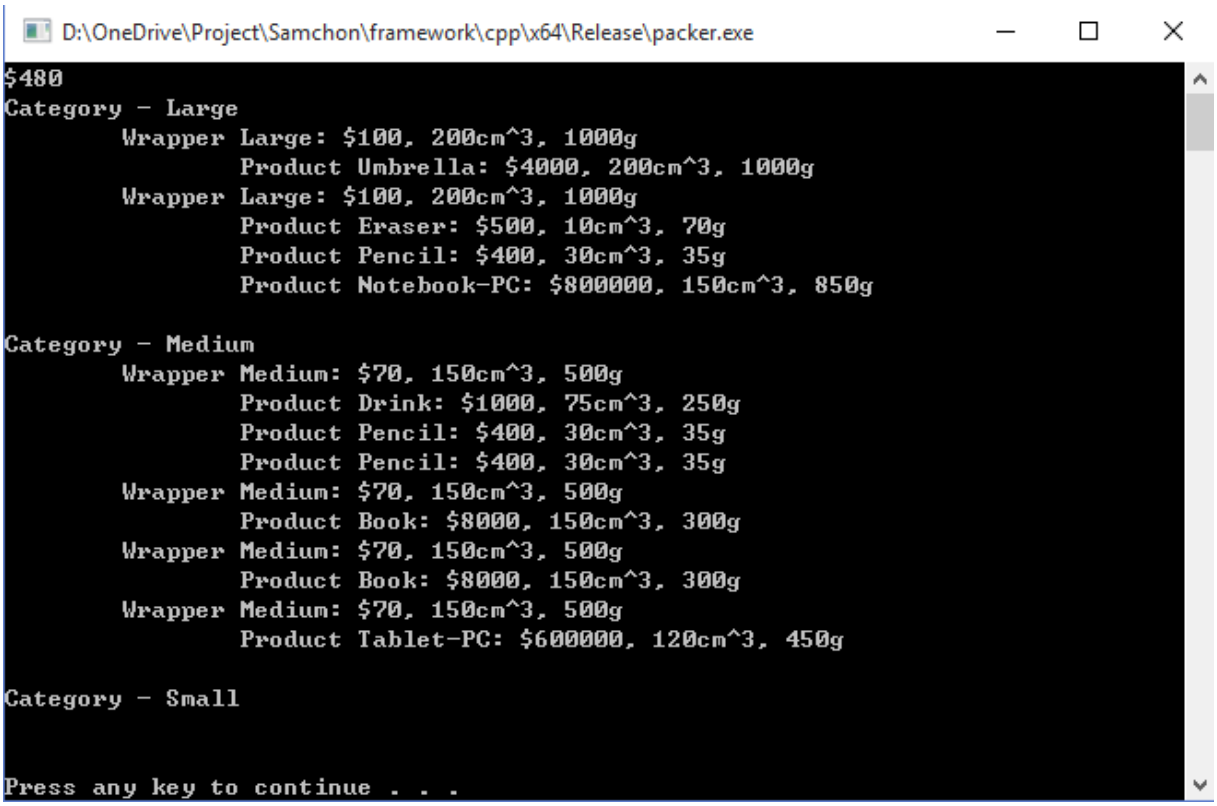
  Volume of an instance -> 130 cm^3.
- int weight

  Weight of an instance -> 1,200 g.

10.68.1 Detailed Description

A physical instance.

A physical instance having its own name, price, volume and weight.



Author

Jeongho Nam

Definition at line 21 of file Instance.hpp.

10.68.2 Constructor & Destructor Documentation

10.68.2.1 Instance::Instance (const std::string & *name*, int *price*, int *volume*, int *weight*)

Construct from instance.

Parameters

name	Name of the instance
price	Price of the instance
volume	Volume of the instance
weight	Weight of the instance

Definition at line 12 of file Instance.cpp.

10.68.3 Member Function Documentation

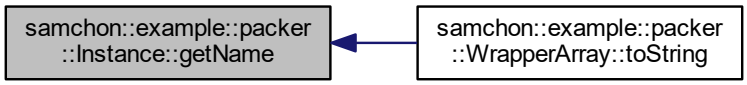
10.68.3.1 auto Instance::getName ( ) const -> std::string

Get name.

Definition at line 23 of file Instance.cpp.

Referenced by samchon::example::packer::WrapperArray::toString().

Here is the caller graph for this function:



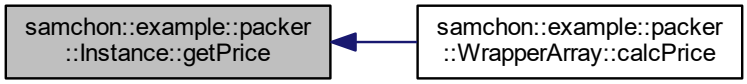
10.68.3.2 auto Instance::getPrice ( ) const -> int

Get price.

Definition at line 27 of file Instance.cpp.

Referenced by samchon::example::packer::WrapperArray::calcPrice().

Here is the caller graph for this function:



10.68.3.3 auto Instance::getVolume ( ) const -> int

Get volume.

Definition at line 31 of file Instance.cpp.

10.68.3.4 auto Instance::getWeight ( ) const -> int

Get weight.

Definition at line 35 of file Instance.cpp.

10.68.3.5 auto Instance::toString ( ) const -> std::string [virtual]

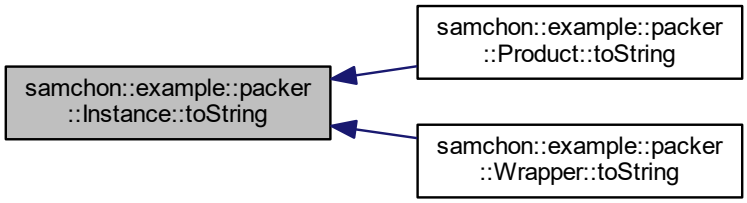
Return a string represents the Instance.

Reimplemented in samchon::example::packer::Wrapper, and samchon::example::packer::Product.

Definition at line 43 of file Instance.cpp.

Referenced by samchon::example::packer::Product::toString(), and samchon::example::packer::Wrapper↩ ::toString().

Here is the caller graph for this function:



10.68.4 Member Data Documentation

10.68.4.1 std::string samchon::example::packer::Instance::name [protected]

Name represent the Instance.

Definition at line 27 of file Instance.hpp.

10.68.4.2 int samchon::example::packer::Instance::price [protected]

Price of an instance -> 1,000 won.

Definition at line 32 of file Instance.hpp.

10.68.4.3 int samchon::example::packer::Instance::volume [protected]

Volume of an instance -> 130 cm^3.

Definition at line 37 of file Instance.hpp.

10.68.4.4 int samchon::example::packer::Instance::weight [protected]

Weight of an instance -> 1,200 g.

Definition at line 42 of file Instance.hpp.

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

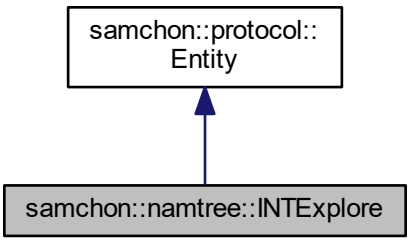
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/packer/Instance.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/packer/Instance.cpp

10.69 samchon::namtree::INTExplore Class Reference

An interface for exploration.

`#include <INTExplore.hpp>`

Collaboration diagram for samchon::namtree::INTExplore:



Public Member Functions

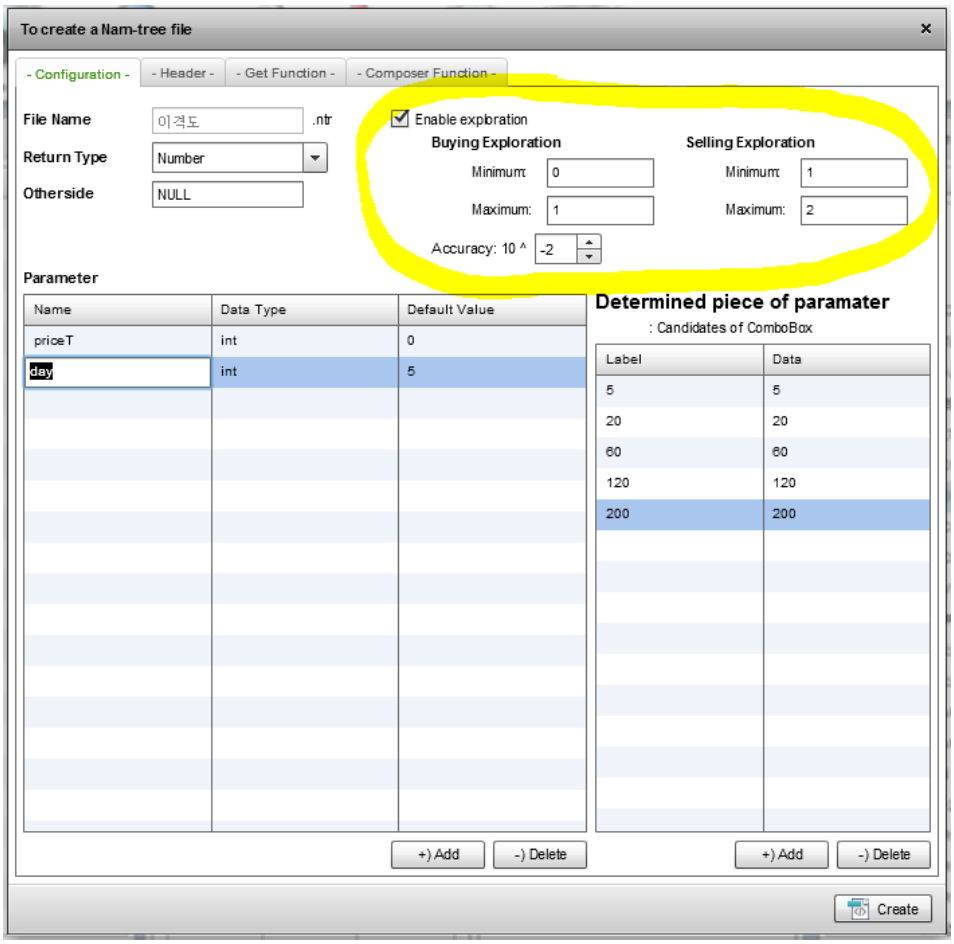
- INTExplore ()
  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >)
  Construct data of the Entity from an XML object.
- auto getMinimum () const -> double
  Get minimum.
- auto getMaximum () const -> double
  Get maximum.
- auto getSection () const -> unsigned int
  Get section.
- auto getPrecision () const -> int
  Get precision.
- virtual auto toXML () const -> std::shared_ptr< library::XML >
  Get an XML object represents the Entity.

Protected Attributes

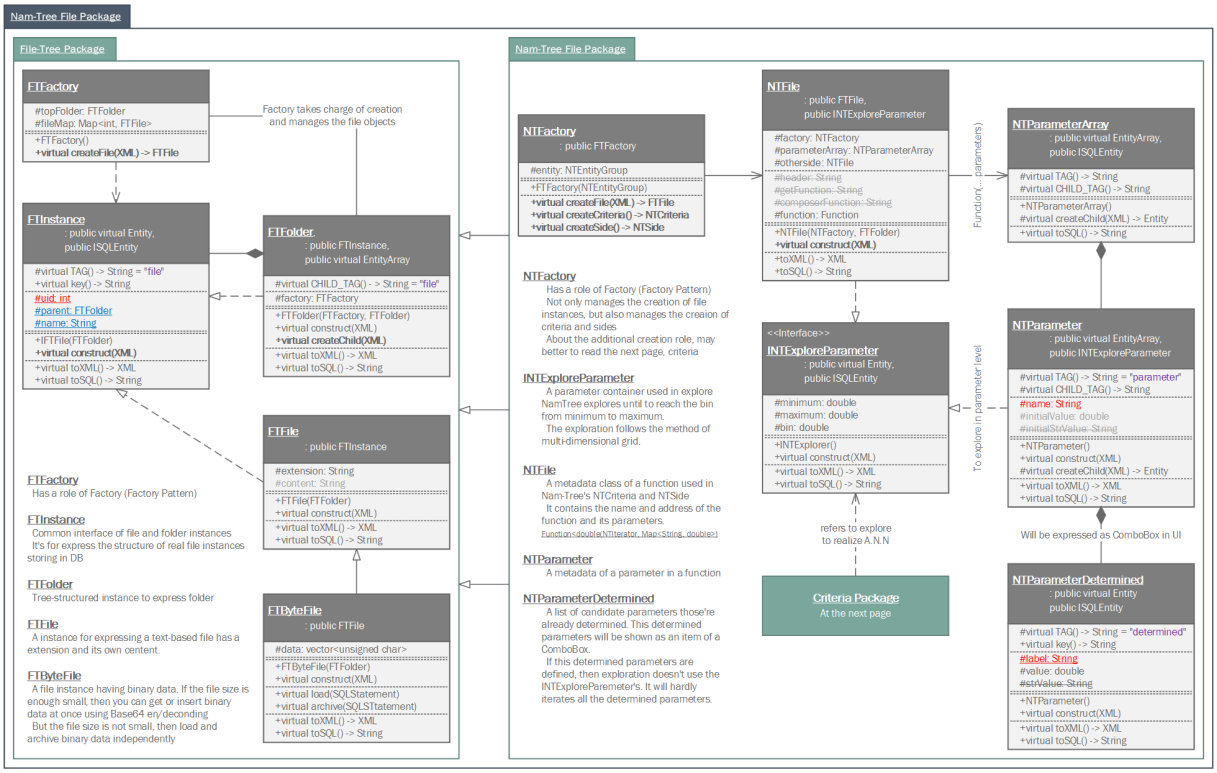
- double minimum
  Minimum value.
- double maximum
  Maximum value.
- unsigned int section
  Section number of exploration.
- int precision
  Target precision: $10^{precision}$

10.69.1 Detailed Description

An interface for exploration.



INTExplore is an interface containing some parameters for grid optimization method.



See also

samchon::namtree

Author

Jeongho Nam

Definition at line 25 of file INTExplore.hpp.

10.69.2 Constructor & Destructor Documentation

10.69.2.1 INTExplore::INTExplore ( )

Default Constructor.

Definition at line 12 of file INTExplore.cpp.

References maximum, minimum, precision, and section.

10.69.3 Member Function Documentation

10.69.3.1 void INTExplore::construct (std::shared_ptr< library::XML >) [virtual]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Implements samchon::protocol::Entity.

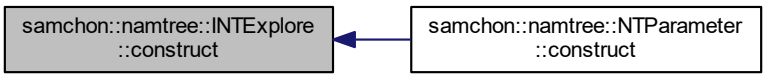
Reimplemented in samchon::namtree::NTFile, and samchon::namtree::NTParameter.

Definition at line 20 of file INTExplore.cpp.

References maximum, minimum, precision, and section.

Referenced by samchon::namtree::NTParameter::construct().

Here is the caller graph for this function:



10.69.3.2 auto INTExplore::getMinimum ( ) const -> double

Get minimum.

Definition at line 37 of file INTExplore.cpp.

References minimum.

10.69.3.3 auto INTExplore::getMaximum ( ) const -> double

Get maximum.

Definition at line 41 of file INTExplore.cpp.

References maximum.

10.69.3.4 auto INTExplore::getSection ( ) const -> unsigned int

Get section.

Definition at line 45 of file INTExplore.cpp.

References section.

10.69.3.5 auto INTExplore::getPrecision ( ) const -> int

Get precision.

Definition at line 49 of file INTExplore.cpp.

References precision.

10.69.3.6 auto INTExplore::toXML ( ) const -> std::shared_ptr<library::XML> [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::Entity.

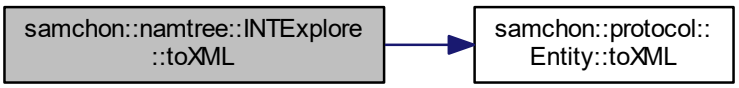
Reimplemented in samchon::namtree::NTFile, and samchon::namtree::NTParameter.

Definition at line 54 of file INTExplore.cpp.

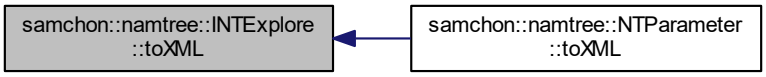
References maximum, minimum, precision, section, and samchon::protocol::Entity::toXML().

Referenced by samchon::namtree::NTParameter::toXML().

Here is the call graph for this function:



Here is the caller graph for this function:



10.69.4 Member Data Documentation

10.69.4.1 double samchon::namtree::INTExplore::minimum [protected]

Minimum value.

Definition at line 35 of file INTExplore.hpp.

Referenced by construct(), getMinimum(), INTExplore(), and toXML().

10.69.4.2 double samchon::namtree::INTExplore::maximum [protected]

Maximum value.

Definition at line 40 of file INTExplore.hpp.

Referenced by construct(), getMaximum(), INTExplore(), and toXML().

10.69.4.3 unsigned int samchon::namtree::INTExplore::section [protected]

Section number of exploration.

Definition at line 45 of file INTExplore.hpp.

Referenced by construct(), getSection(), INTExplore(), and toXML().

10.69.4.4 int samchon::namtree::INTExplore::precision [protected]

Target precision: $10^{precision}$

Definition at line 50 of file INTExplore.hpp.

Referenced by construct(), getPrecision(), INTExplore(), and toXML().

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

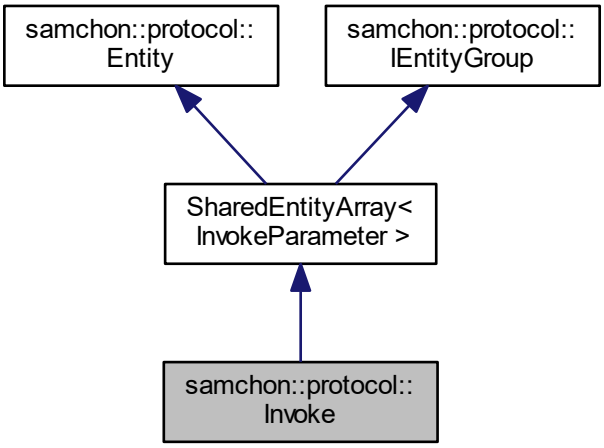
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/INTExplore.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/INTExplore.cpp

10.70 samchon::protocol::Invoke Class Reference

Standard message of network I/O.

```
#include <Invoke.hpp>
```

Collaboration diagram for samchon::protocol::Invoke:



Public Member Functions

- Invoke ()
  Default Constructor.
- Invoke (const std::string &listener)
  Construct from a listener.
- virtual void construct (std::shared_ptr< library::XML >) override
  Construct data of the Entity from an XML object.
- template<typename _Ty , typename... _Args>
  Invoke (const std::string &listener, const _Ty &val, const _Args &...args)
  Construct from arguments.
- auto getListener () const -> std::string
  Get listener.
- void setListener (const std::string &)
  Set listener.
- virtual auto TAG () const -> std::string override
  A tag name when represented by XML.
- virtual auto CHILD_TAG () const -> std::string override
  A tag name of children.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override
  Get an XML object represents the EntityGroup.
- auto toSQL () const -> std::string
  Get a string of sql statement used to archive history log.

Protected Member Functions

- virtual auto createChild (std::shared_ptr< library::XML >) -> InvokeParameter *override
  Factory method of a child Entity.

Protected Attributes

- std::string listener
  Represent who listens, often be a function name.

10.70.1 Detailed Description

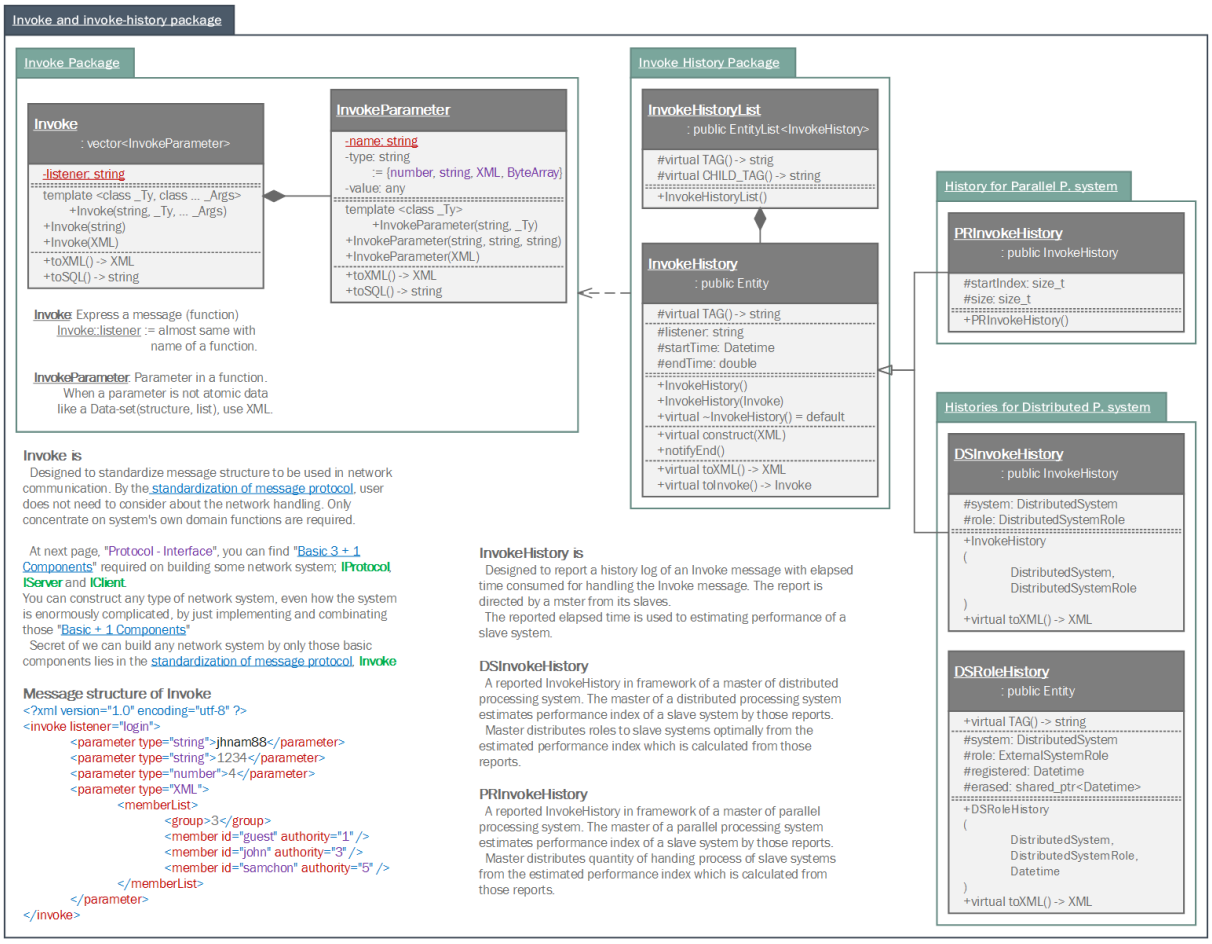
Standard message of network I/O.

Invoke is a class used in network I/O in protocol package of Samchon Framework.

The Invoke message has an XML structure like the result screen of provided example in below. We can enjoy lots of benefits by the normalized and standardized message structure used in network I/O.

The greatest advantage is that we can make any type of network system, even how the system is enourmously complicated. As network communication message is standardized, we only need to concentrate on logical relationships between network systems. We can handle each network system like a object (class) in OOD. And those relationships can be easily designed by using design pattern.

In Samchon Framework, you can make any type of network system with basic 3 + 1 componenets (IProtocol, IServer and IClient + ServerConnector), by implemens or inherits them, like designing classes of S/W architecture.

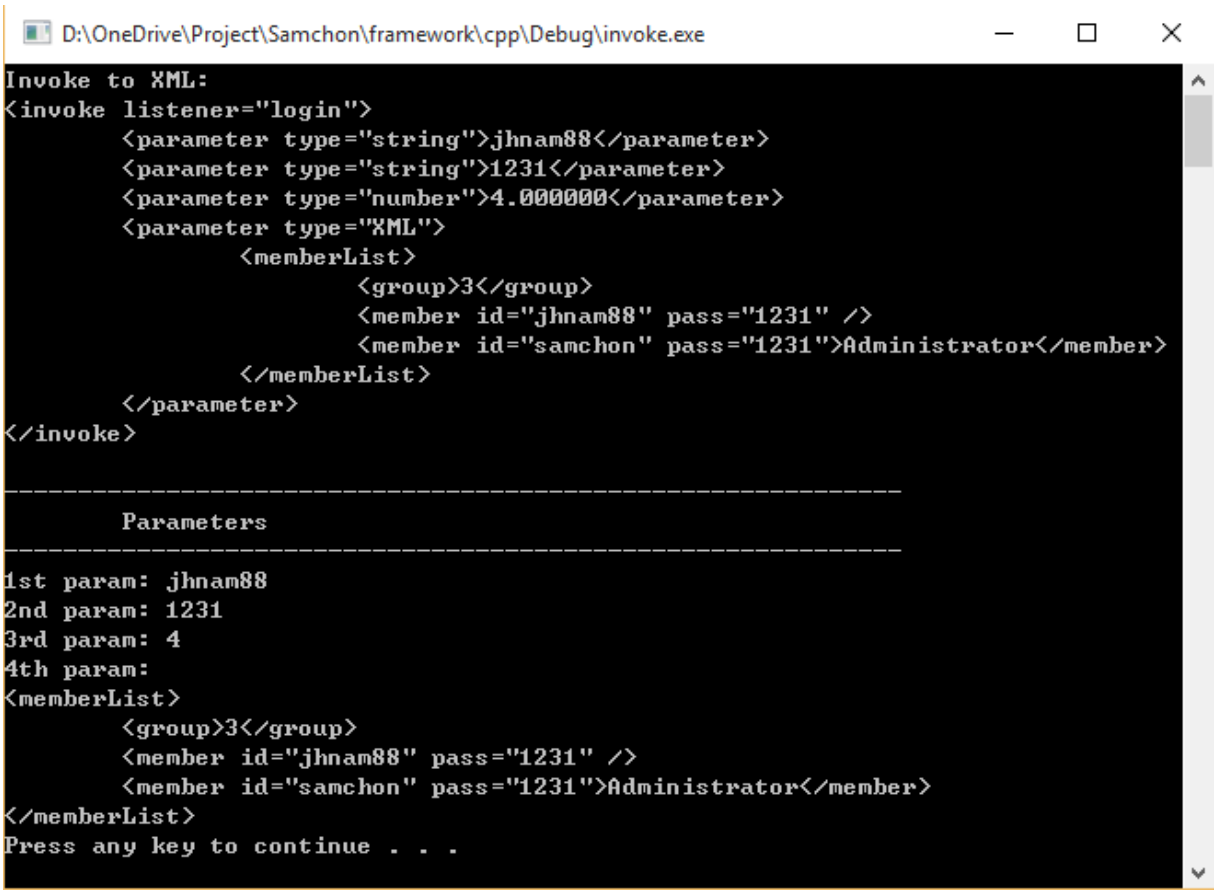


Example source

```
#include <iostream>

#include <samchon/library/XML.hpp>
#include <samchon/protocol/Invoke.hpp>

using namespace std;
using namespace samchon;
using namespace samchon::library;
using namespace samchon::protocol;

#ifdef _WIN64
#   ifdef _DEBUG
#       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "x64/Release/SamchonFramework.lib")
#   endif
```

```
#else
#   ifdef _DEBUG
#       pragma comment(lib, "Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "Release/SamchonFramework.lib")
#   endif
#endif

void main()
{
    string str = string("") +
        "<memberList>\n" +
        "   <member id='jhnam88' pass='1231' />\n" +
        "   <member id='samchon' pass='1231'>Administrator</member>\n" +
        "   <group>3</group>\n" +
        "</memberList>";

    shared_ptr<XML> xml(new XML(str));
    shared_ptr<Invoke> invoke(new Invoke("login", "jhnam88", "1231", 4, xml));

    cout << "Invoke to XML: " << endl;
    cout << invoke->toXML()->toString() << endl << endl;

    cout << "------------------------------------------------------------" << endl;
    cout << "   Parameters" << endl;
    cout << "------------------------------------------------------------" << endl;
    cout << "1st param: " << invoke->at(0)->getValue<string>() << endl;
    cout << "2nd param: " << invoke->at(1)->getValue<string>() << endl;
    cout << "3rd param: " << invoke->at(2)->getValue<string>() << endl;
    cout << "4th param: " << endl << invoke->at(3)->getValue<shared_ptr<XML>>()->toString() << endl;

    system("pause");
}
```

Result of the example



See also

samchon::protocol
protocol::IProtocol

Author

Jeongho Nam

Definition at line 45 of file Invoke.hpp.

10.70.2 Constructor & Destructor Documentation

10.70.2.1 Invoke::Invoke ( )

Default Constructor.

Definition at line 14 of file Invoke.cpp.

10.70.2.2 Invoke::Invoke (const std::string & *listener*)

Construct from a listener.

Parameters

listener	Represents who listens the Invoke message. Almost same with Function name

Definition at line 18 of file Invoke.cpp.

References listener.

10.70.2.3 template<typename _Ty , typename... _Args> samchon::protocol::Invoke::Invoke (const std::string & *listener,* const _Ty & *val,* const _Args &... *args*) [inline]

Construct from arguments.

Template Parameters

_Ty	Type of an argument which represents a parameter
_Args	Left varadic template arguments' types

Creates Invoke and InvokeParameter(s) at the same time by varadic template method.

By the varadic template constructor, you can't specify name of each InvokeParameter, but specify type and value of each InvokeParameter. If you try to record the Invoke to Database, the name of InvokeParameter will be *NULL*.

Note

By the varadic template constructor, name of InovkeParameter(s) will be omitted. Because of name, an identifier of an InvokeParameter, is omitted, you can't access to InvokeParameter by Invoke::has() or Invoke↩ ::get().

Parameters

listener	A string represents who listens the Invoke message. Almost same with name of a function.
val	A value to be a parameter of Invoke
args	Left arguments to be parameters of Invoke

Definition at line 106 of file Invoke.hpp.

10.70.3 Member Function Documentation

10.70.3.1 void Invoke::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

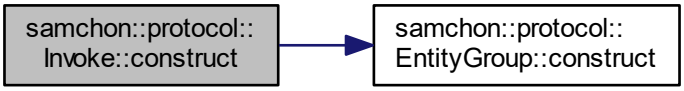
xml	An xml used to construct data of entity

Reimplemented from samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

Definition at line 24 of file Invoke.cpp.

References samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::construct(), and listener.

Here is the call graph for this function:



10.70.3.2 auto Invoke::createChild (std::shared_ptr< library::XML >) -> InvokeParameter *override [override], [protected], [virtual]

Factory method of a child Entity.

EntityGroup::createChild() is a factory method creating a new child Entity which is belonged to the Entity↩Group. This method is called by EntityGroup::construct(). The children construction methods Entity↩::construct() will be called by abstract method of the EntityGroup::construct().

Returns

A new child Entity belongs to EntityGroup.

Implements samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

Definition at line 30 of file Invoke.cpp.

10.70.3.3 auto Invoke::getListener ( ) const -> std::string

Get listener.

Definition at line 65 of file Invoke.cpp.

References listener.

10.70.3.4 void Invoke::setListener (const std::string &)

Set listener.

Definition at line 70 of file Invoke.cpp.

References listener.

10.70.3.5 auto Invoke::TAG ( ) const -> std::string [override], [virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Definition at line 78 of file Invoke.cpp.

10.70.3.6 auto Invoke::CHILD_TAG ( ) const -> std::string [override], [virtual]

A tag name of children.

< TAG>
<CHILD_TAG />
<CHILD_TAG />
</TAG>

Implements samchon::protocol::IEntityGroup.

Definition at line 82 of file Invoke.cpp.

10.70.3.7 auto Invoke::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

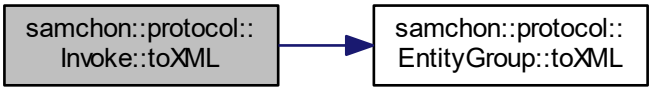
An XML object representing the Entity.

Reimplemented from samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

Definition at line 87 of file Invoke.cpp.

References listener, and samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::toXML().

Here is the call graph for this function:



10.70.3.8 auto Invoke::toSQL ( ) const -> std::string

Get a string of sql statement used to archive history log.

Definition at line 95 of file Invoke.cpp.

10.70.4 Member Data Documentation

10.70.4.1 std::string samchon::protocol::Invoke::listener [protected]

Represent who listens, often be a function name.

Definition at line 55 of file Invoke.hpp.

Referenced by construct(), getListener(), Invoke(), setListener(), and toXML().

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

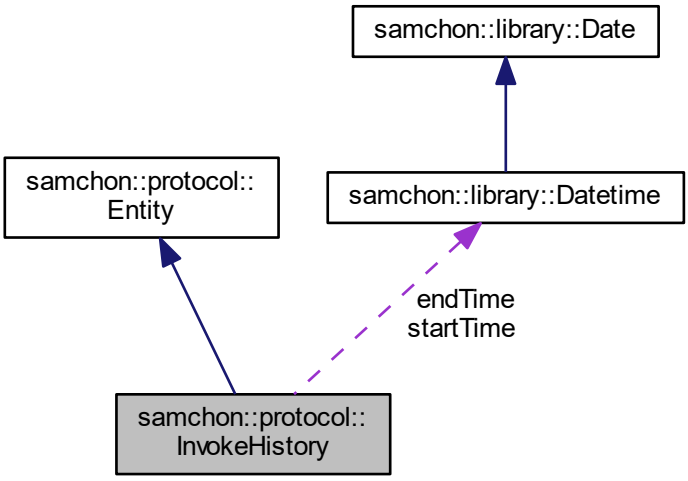
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/Invoke.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/Invoke.cpp

10.71 samchon::protocol::InvokeHistory Class Reference

A history of an Invoke message.

```
#include <InvokeHistory.hpp>
```

Collaboration diagram for samchon::protocol::InvokeHistory:



Public Member Functions

- InvokeHistory ()
  Default Constructor.
- InvokeHistory (std::shared_ptr< Invoke >)
  Construct from an Invoke message.
- virtual void construct (std::shared_ptr< library::XML >) override
  Construct data of the Entity from an XML object.
- virtual void notifyEnd ()
  Notify end of the process.
- virtual auto key () const -> std::string override
  Get a key that can identify the Entity uniquely.
- auto getUID () const -> size_t
  Get uid.
- auto getListener () const -> std::string
  Get listener.
- auto getStartTime () const -> library::Datetime
  Get start time.
- auto getEndTime () const -> library::Datetime
  Get end time.
- auto calcElapsedTime () const -> long long
  Get elapsed time.
- virtual auto TAG () const -> std::string override
  A tag name when represented by XML.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override
  Get an XML object represents the Entity.
- auto toInvoke () const -> std::shared_ptr< Invoke >
  Get an Invoke message.

Protected Attributes

- size_t uid

An identifier.

– std::string listener

A listener of the Invoke message.

– library::Datetime startTime

Start time of the history.

– library::Datetime endTime

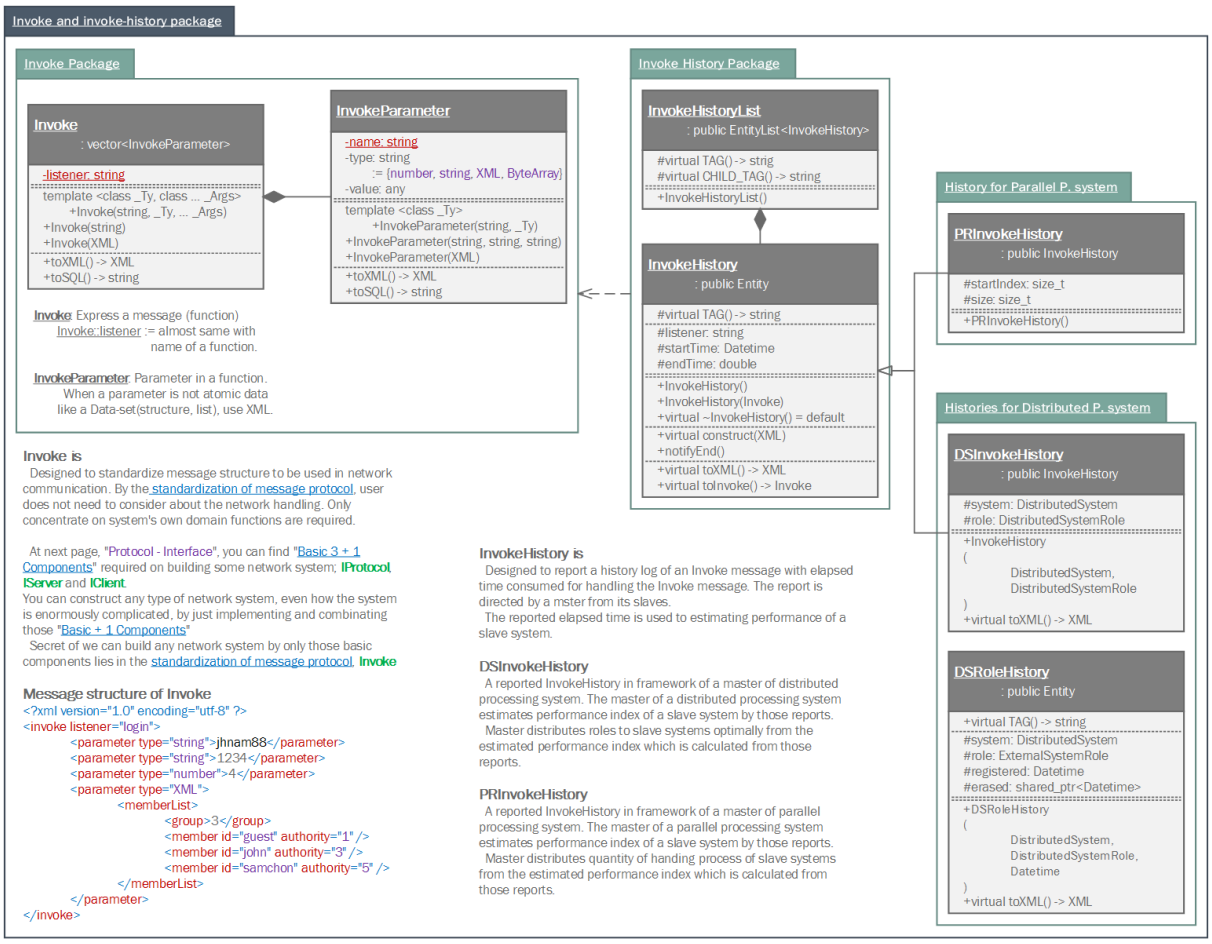
End time of the history.

10.71.1 Detailed Description

A history of an Invoke message.

InvokeHistory is a class for reporting history log of an Invoke message with elapsed time from a slave to its master.

With the elapsed time, consumed time for a process of handling the Invoke message, InvokeHistory is reported to the master. The master utilizies the elapsed time to estimating performances of each slave system. With the estimated performan index, master retrives the optimal solution of distributing processes.



See also

samchon::protocol
samchon::protocol::master
samchon::protocol::slave

Author

Jeongho Nam

Definition at line 35 of file InvokeHistory.hpp.

10.71.2 Constructor & Destructor Documentation

10.71.2.1 InvokeHistory::InvokeHistory ( )

Default Constructor.

Definition at line 13 of file InvokeHistory.cpp.

10.71.2.2 samchon::protocol::InvokeHistory::InvokeHistory (std::shared_ptr< Invoke >)

Construct from an Invoke message.

InvokeHistory does not archive entire Invoke message, only archives its listener.

Parameters

invoke	A message to archive its history log

10.71.3 Member Function Documentation

10.71.3.1 void InvokeHistory::construct (std::shared_ptr< library::XML >) [override],[virtual]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Implements samchon::protocol::Entity.

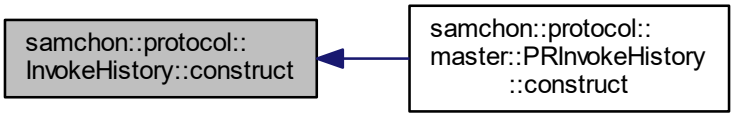
Reimplemented in samchon::protocol::master::PRInvokeHistory.

Definition at line 26 of file InvokeHistory.cpp.

References endTime, startTime, and uid.

Referenced by samchon::protocol::master::PRInvokeHistory::construct().

Here is the caller graph for this function:



10.71.3.2 void InvokeHistory::notifyEnd ( ) [virtual]

Notify end of the process.

Notifies end of a process handling the matched Invoke message to InvokeHistory.

InvokeHistory archives the end datetime and calculates elapsed time as nanoseconds.

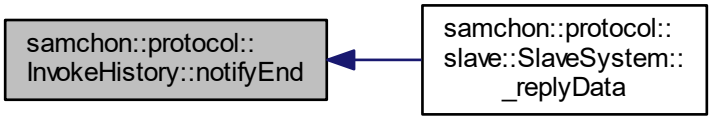
Reimplemented in samchon::protocol::master::PRInvokeHistory, and samchon::protocol::master::PR↩MasterHistory.

Definition at line 56 of file InvokeHistory.cpp.

References endTime.

Referenced by samchon::protocol::slave::SlaveSystem::_replyData().

Here is the caller graph for this function:



10.71.3.3 auto InvokeHistory::key ( ) const -> std::string [override], [virtual]

Get a key that can identify the Entity uniquely.

If identifier of the Entity is not atomic value, returns a string represents the composite identifier. If identifier of the Entity is not string, converts the identifier to string and returns the string.

Returns

An identifier

Reimplemented from samchon::protocol::Entity.

Definition at line 64 of file InvokeHistory.cpp.

References uid.

10.71.3.4 auto InvokeHistory::getUID ( ) const -> size_t

Get uid.

Definition at line 69 of file InvokeHistory.cpp.

References uid.

10.71.3.5 auto InvokeHistory::getListener ( ) const -> std::string

Get listener.

Definition at line 73 of file InvokeHistory.cpp.

References listener.

10.71.3.6 auto InvokeHistory::getStartTime ( ) const -> library::Datetime

Get start time.

Definition at line 77 of file InvokeHistory.cpp.

References startTime.

10.71.3.7 auto InvokeHistory::getEndTime ( ) const -> library::Datetime

Get end time.

Definition at line 81 of file InvokeHistory.cpp.

References endTime.

10.71.3.8 auto InvokeHistory::calcElapsedTime ( ) const -> long long

Get elapsed time.

Calculates elapsed time frrom start, end time and Returns it.

The elapsed time will be reported to a master and the master will utilize the elapsed time to estimating performance of the slave system, the InvokeHistory is belonged to.

Returns

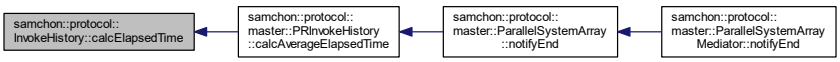
An elapsed time expressed as nano seconds.

Definition at line 86 of file InvokeHistory.cpp.

References endTime, and startTime.

Referenced by samchon::protocol::master::PRInvokeHistory::calcAverageElapsedTime().

Here is the caller graph for this function:



10.71.3.9 auto InvokeHistory::TAG ( ) const -> std::string [override], [virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Definition at line 94 of file InvokeHistory.cpp.

10.71.3.10 auto InvokeHistory::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value

<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::Entity.

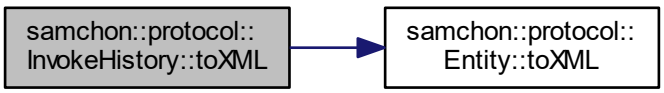
Reimplemented in samchon::protocol::master::PRInvokeHistory, and samchon::protocol::master::DS↩InvokeHistory.

Definition at line 99 of file InvokeHistory.cpp.

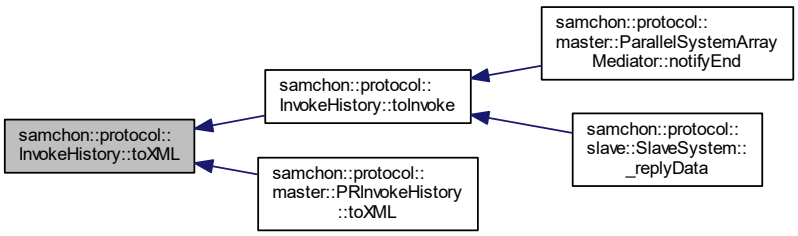
References endTime, startTime, samchon::protocol::Entity::toXML(), and uid.

Referenced by toInvoke(), and samchon::protocol::master::PRInvokeHistory::toXML().

Here is the call graph for this function:



Here is the caller graph for this function:



10.71.3.11 auto InvokeHistory::toInvoke ( ) const -> std::shared_ptr<Invoke>

Get an Invoke message.

Returns an Invoke message to report to a master that how much time was elapsed on a process handling the Invoke message. In master, those reports are used to estimate performance of each slave system.

Returns

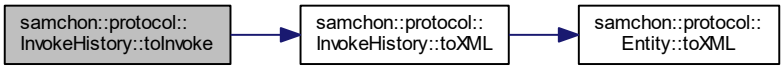
An Invoke message to report master.

Definition at line 110 of file InvokeHistory.cpp.

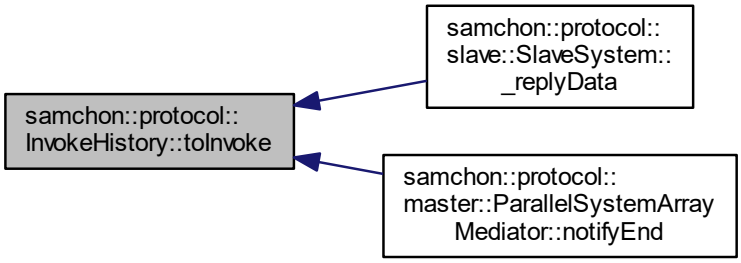
References toXML().

Referenced by samchon::protocol::slave::SlaveSystem::_replyData(), and samchon::protocol::master::↩ ParallelSystemArrayMediator::notifyEnd().

Here is the call graph for this function:



Here is the caller graph for this function:



10.71.4 Member Data Documentation

10.71.4.1 size_t samchon::protocol::InvokeHistory::uid [protected]

An identifier.

Definition at line 44 of file InvokeHistory.hpp.

Referenced by construct(), getUID(), key(), and toXML().

10.71.4.2 std::string samchon::protocol::InvokeHistory::listener [protected]

A listener of the Invoke message.

InvokeHistory does not archive entire data of an Invoke message. InvokeHistory only archives its listener. The first, formal reason is to save space, avoid wasting spaces.

The second, complicate reason is on an aspect of which systems are using the InvokeHistory class. Invoke↩ History is designed to let slave reports to master elapsed time of a process used to handling the Invoke message. If you want to archive entire history log of Invoke messages, then the subject should be master, not the slave using InvokeHistory classes.

Definition at line 59 of file InvokeHistory.hpp.

Referenced by getListener().

10.71.4.3 library::Datetime samchon::protocol::InvokeHistory::startTime [protected]

Start time of the history.

Means start time of a process handling the Invoke message. The start time not only has ordinary arguments represented Datetime (year to seconds), but also has very precise values under seconds, which is expressed as nano seconds (10^-9).

The precise start time will be used to calculate elapsed time with end time.

Definition at line 71 of file InvokeHistory.hpp.

Referenced by calcElapsedTime(), construct(), getStartTime(), and toXML().

10.71.4.4 library::Datetime samchon::protocol::InvokeHistory::endTime [protected]

End time of the history.

Means end time of a process handling the Invoke message. The end time not only has ordinary arguments represented Datetime (year to seconds), but also has very precise values under seconds, which is expressed as nano seconds (10^-9).

The precise end time will be used to calculate elapsed time with start time.

Definition at line 83 of file InvokeHistory.hpp.

Referenced by calcElapsedTime(), construct(), getEndTime(), notifyEnd(), and toXML().

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

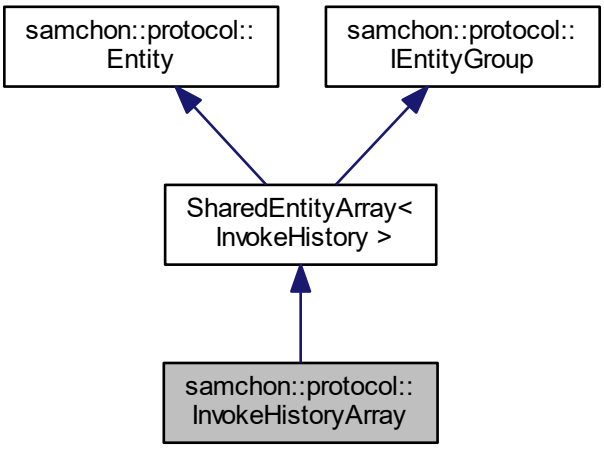
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/InvokeHistory.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/InvokeHistory.cpp

10.72 samchon::protocol::InvokeHistoryArray Class Reference

An array of InvokeHistory.

```
#include <InvokeHistoryArray.hpp>
```

Collaboration diagram for samchon::protocol::InvokeHistoryArray:



Public Member Functions

- InvokeHistoryArray ()
  Default Constructor.
- virtual auto TAG () const -> std::string
  A tag name when represented by XML.
- virtual auto CHILD_TAG () const -> std::string
  A tag name of children.

Protected Member Functions

- virtual auto createChild (std::shared_ptr< library::XML >) -> InvokeHistory *override
  Factory method of a child Entity.

10.72.1 Detailed Description

An array of InvokeHistory.

Author

Jeongho Nam

Definition at line 16 of file InvokeHistoryArray.hpp.

10.72.2 Constructor & Destructor Documentation

10.72.2.1 InvokeHistoryArray::InvokeHistoryArray ( )

Default Constructor.

Definition at line 7 of file InvokeHistoryArray.cpp.

10.72.3 Member Function Documentation

10.72.3.1 auto InvokeHistoryArray::createChild (std::shared_ptr< library::XML >) -> InvokeHistory *override [override], [protected], [virtual]

Factory method of a child Entity.

EntityGroup::createChild() is a factory method creating a new child Entity which is belonged to the EntityGroup. This method is called by EntityGroup::construct(). The children construction methods Entity::construct() will be called by abstract method of the EntityGroup::construct().

Returns

A new child Entity belongs to EntityGroup.

Implements samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

Reimplemented in samchon::protocol::master::PRInvokeHistoryArray.

Definition at line 11 of file InvokeHistoryArray.cpp.

10.72.3.2 auto InvokeHistoryArray::TAG ( ) const -> std::string [virtual]

A tag name when represented by XML.

Returns

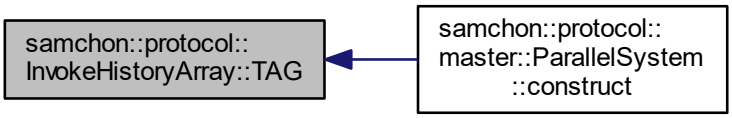
A tag name

Implements samchon::protocol::Entity.

Definition at line 16 of file InvokeHistoryArray.cpp.

Referenced by samchon::protocol::master::ParallelSystem::construct().

Here is the caller graph for this function:



10.72.3.3 auto InvokeHistoryArray::CHILD_TAG ( ) const -> std::string [virtual]

A tag name of children.

< TAG>
<CHILD_TAG />
<CHILD_TAG />
</TAG>

Implements samchon::protocol::IEntityGroup.

Definition at line 20 of file InvokeHistoryArray.cpp.

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

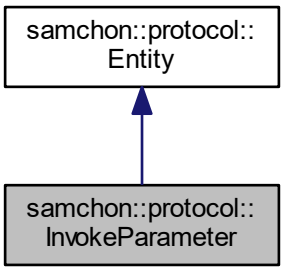
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/InvokeHistoryArray.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/InvokeHistoryArray.cpp

10.73 samchon::protocol::InvokeParameter Class Reference

A parameter of an Invoke.

```
#include <InvokeParameter.hpp>
```

Collaboration diagram for samchon::protocol::InvokeParameter:



Public Member Functions

- InvokeParameter (const std::string &, const std::string &, const std::string &)
  Construct from arguments.
- template<typename _Ty >
  InvokeParameter (const std::string &name, const _Ty &val)
  Construct with its name and a value.
- InvokeParameter (const std::string &, std::string &&)
  Construct from the name and a moved string.
- InvokeParameter (const std::string &, ByteArray &&)
  Construct from name and a moved ByteArray.
- virtual auto key () const -> std::string override
  Get a key that can identify the Entity uniquely.
- auto getName () const -> std::string
  Get name.
- auto getType () const -> std::string
  Get type.
- template<typename _Ty >
  auto getValue () const -> _Ty
  Get value.
- auto getvalueAsXML () const -> std::shared_ptr< library::XML >
  Get value as XML object.
- template<typename _Ty >
  auto referValue () const -> const _Ty &
  Reference value.
- template<typename _Ty >
  auto moveValue () -> _Ty
  Move value.

Protected Member Functions

- InvokeParameter ()
  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override
  Construct data of the Entity from an XML object.
- virtual auto TAG () const -> std::string override
  A tag name when represented by XML.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override
  Get an XML object represents the Entity.
- auto toSQL () const -> std::string
  Get a string of sql statement used to archive history log.

Protected Attributes

- std::string name

  A name can represent the parameter.
- std::string type

  Type of the parameter.
- std::string str

  A string value if the type is "string" or "number".
- std::shared_ptr< library::XML > xml

  An XML object if the type is "XML".
- std::unique_ptr< ByteArray > byteArray

  A binary value if the type is "ByteArray".

10.73.1 Detailed Description

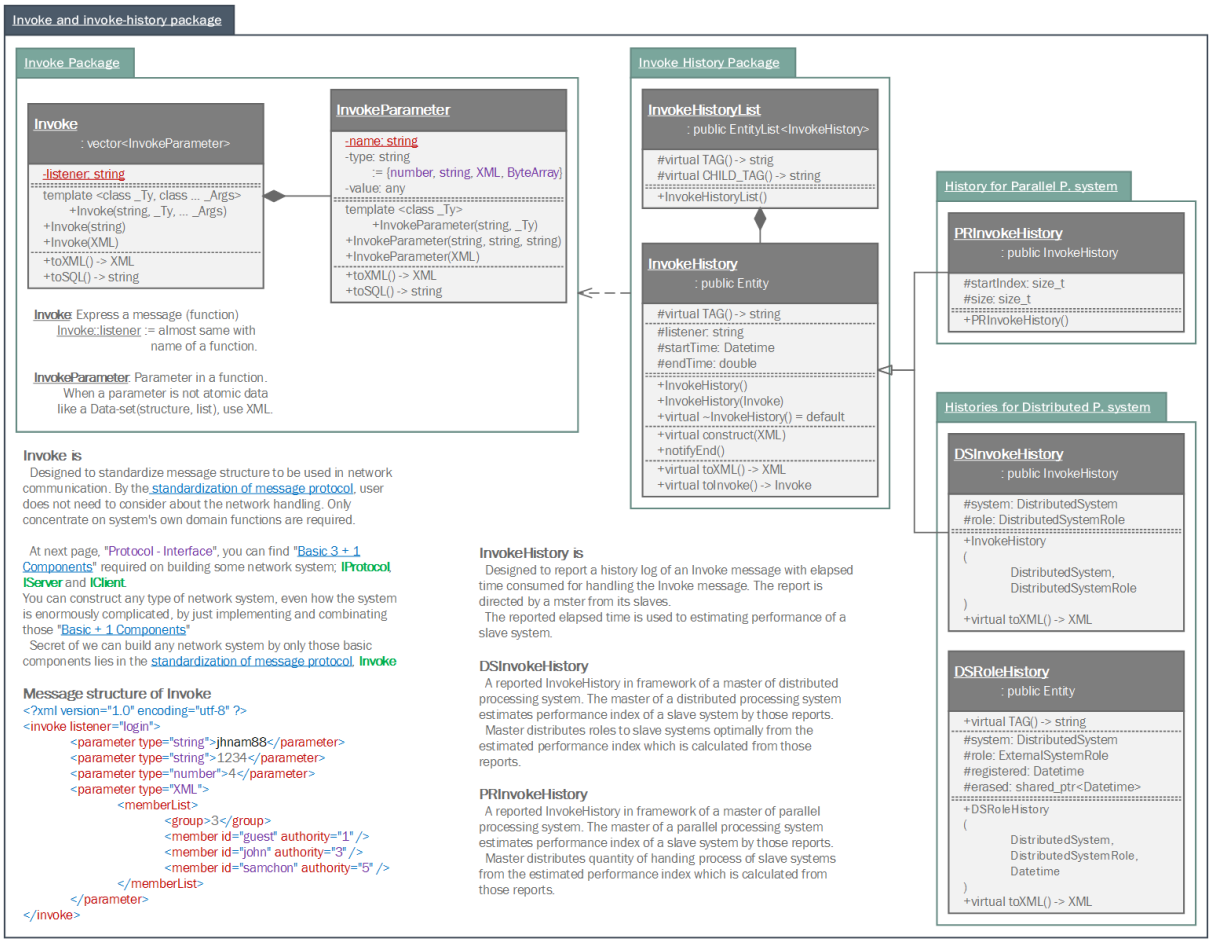
A parameter of an Invoke.

A parameter with its name, type and value in an Invoke message.

InvokeParameter supports 4 types pre-defined.

- number
- string
- XML
- ByteArray

You can specify InvokeParameter to have some type which is not one of the basic 4 types, but values of the custom type must be enable to expressed by one of those types; number, string, XML and ByteArray.



Note

A member variable void∗ representing any type of value is deprecated.

Since C++11, method of void pointer (Declare a variable as void pointer whose type can not specified. And convert the pointer type to specified one for each case) is recommended to avoid.

As that reason, the *void value was a candidate to be deprecated or to be replaced to boost::any. And the void value* is really deprecated since version v1.0.

See also

protocol::Invoke
samchon::protocol

Author

Jeongho Nam

Definition at line 56 of file InvokeParameter.hpp.

10.73.2 Constructor & Destructor Documentation

10.73.2.1 samchon::protocol::InvokeParameter::InvokeParameter (const std::string & , const std::string & , const std::string &)

Construct from arguments.

Specifies all arguments of an InvokeParameter. You can specify a custom type, that is not one of number or string, but value of the type must be enable to expressed by a string.

If you want to express an object or container group, use XML instead; *Invoke::InvokeParameter(string, XML)*.

Parameters

name	A name can represent the InvokeParameter
type	Type of value in the InvokeParameter
value	A value capsuled by a string

10.73.2.2 template<typename _Ty > samchon::protocol::InvokeParameter::InvokeParameter (const std::string & *name*, const _Ty & *val*) [inline]

Construct with its name and a value.

Type of the InvokeParameter will be determined automatically.

Type supported in the template constructor:

- number
  - (unsigned) short
  - (unsigned) long
  - (unsigned) long long
  - (unsigned) int
  - float
  - double
  - long double
- string
  - std::string
  - WeakString
- XML

    * std::shared_ptr<library::XML>
    * Entity; protocol::Entity::toXML()
  - ByteArray

Template Parameters

_Ty	Type of value

Definition at line 152 of file InvokeParameter.hpp.

10.73.2.3 samchon::protocol::InvokeParameter::InvokeParameter (const std::string & , std::string &&)

Construct from the name and a moved string.

10.73.2.4 samchon::protocol::InvokeParameter::InvokeParameter (const std::string & , ByteArray &&)

Construct from name and a moved ByteArray.

10.73.2.5 InvokeParameter::InvokeParameter ( ) [protected]

Default Constructor.

Definition at line 53 of file InvokeParameter.cpp.

10.73.3 Member Function Documentation

10.73.3.1 void InvokeParameter::construct (std::shared_ptr< library::XML >) [override], [protected], [virtual]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Implements samchon::protocol::Entity.

Definition at line 58 of file InvokeParameter.cpp.

References byteArray, str, and type.

10.73.3.2 auto InvokeParameter::key ( ) const -> std::string [override], [virtual]

Get a key that can identify the Entity uniquely.

If identifier of the Entity is not atomic value, returns a string represents the composite identifier. If identifier of the Entity is not string, converts the identifier to string and returns the string.

Returns

An identifier

Reimplemented from samchon::protocol::Entity.

Definition at line 119 of file InvokeParameter.cpp.

References name.

10.73.3.3 auto InvokeParameter::getName ( ) const -> std::string

Get name.

Definition at line 124 of file InvokeParameter.cpp.

References name.

10.73.3.4 auto InvokeParameter::getType ( ) const -> std::string

Get type.

Definition at line 128 of file InvokeParameter.cpp.

References type.

10.73.3.5 template<typename _Ty > auto samchon::protocol::InvokeParameter::getValue ( ) const -> _Ty [inline]

Get value.

Template Parameters

_Ty	Type of value to get

Definition at line 226 of file InvokeParameter.hpp.

10.73.3.6 auto InvokeParameter::getvalueAsXML ( ) const -> std::shared_ptr<library::XML>

Get value as XML object.

Same with getValue< std::shared_ptr<library::XML> >();

Definition at line 146 of file InvokeParameter.cpp.

References xml.

10.73.3.7 template<typename _Ty > auto samchon::protocol::InvokeParameter::referValue ( ) const -> const _Ty &

Reference value.

Template Parameters

_Ty	Type of value to reference

10.73.3.8 template<typename _Ty > auto samchon::protocol::InvokeParameter::moveValue ( ) -> _Ty

Move value.

Template Parameters

_Ty	Type of value to move

10.73.3.9 auto InvokeParameter::TAG ( ) const -> std::string [override],[protected],[virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Definition at line 172 of file InvokeParameter.cpp.

10.73.3.10 auto InvokeParameter::toXML ( ) const -> std::shared_ptr<library::XML> [override], [protected], [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

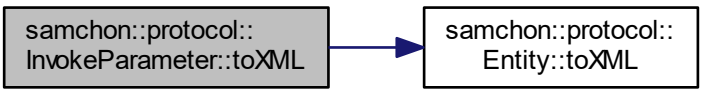
An XML object representing the Entity.

Reimplemented from samchon::protocol::Entity.

Definition at line 177 of file InvokeParameter.cpp.

References samchon::protocol::Entity::toXML(), type, and xml.

Here is the call graph for this function:



10.73.3.11 auto InvokeParameter::toSQL ( ) const -> std::string [protected]

Get a string of sql statement used to archive history log.

Definition at line 200 of file InvokeParameter.cpp.

10.73.4 Member Data Documentation

10.73.4.1 std::string samchon::protocol::InvokeParameter::name [protected]

A name can represent the parameter.

Definition at line 69 of file InvokeParameter.hpp.

Referenced by getName(), and key().

10.73.4.2 std::string samchon::protocol::InvokeParameter::type [protected]

Type of the parameter.

List of pre-defined types

- number
- string
- XML
- ByteArray

Definition at line 81 of file InvokeParameter.hpp.

Referenced by construct(), getType(), and toXML().

10.73.4.3 std::string samchon::protocol::InvokeParameter::str [protected]

A string value if the type is "string" or "number".

Definition at line 86 of file InvokeParameter.hpp.

Referenced by construct().

10.73.4.4 std::shared_ptr<library::XML> samchon::protocol::InvokeParameter::xml [protected]

An XML object if the type is "XML".

Definition at line 91 of file InvokeParameter.hpp.

Referenced by getvalueAsXML(), and toXML().

10.73.4.5 std::unique_ptr<ByteArray> samchon::protocol::InvokeParameter::byteArray [protected]

A binary value if the type is "ByteArray".

Definition at line 96 of file InvokeParameter.hpp.

Referenced by construct().

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/InvokeParameter.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/InvokeParameter.cpp

10.74 samchon::library::IOperator< _Ty > Class Template Reference

Interface for comparision operator.

```
#include <IOperator.hpp>
```

Public Member Functions

- IOperator ()
  Default Constructor.
- virtual auto operator< (const IOperator &) const -> bool=0
  Less.
- virtual auto operator== (const IOperator &) const -> bool=0
  Equal.

10.74.1 Detailed Description

template<typename _Ty>class samchon::library::IOperator< _Ty >

Interface for comparision operator.

IOperator is a template class makes enalbe to realizing all compare operators by only overriding operator< and operator== methods.

- [!=] -> [NOT ==]
- [<=] -> [< OR ==]
- [>] -> [NOT < AND NOT ==]
- [>=] -> [NOT <]

Methods operator< and operator== are abstract. Overrides them.

Note

If you want to realize all comparison operators without inheritance and specifying clear type of parameter, you can use following macros instead.

- OPERATOR_METHODS_INLINE(_Ty)
- OPERATOR_METHODS_HEADER(_Ty)
- OPERATOR_METHODS_BODY(_Class, _Ty)

See also

samchon::library

Author

Jeongho Nam

Definition at line 123 of file IOperator.hpp.

10.74.2 Constructor & Destructor Documentation

10.74.2.1 template<typename _Ty> samchon::library::IOperator< _Ty >::IOperator ( ) [inline]

Default Constructor.

Definition at line 129 of file IOperator.hpp.

10.74.3 Member Function Documentation

10.74.3.1 template<typename _Ty> virtual auto samchon::library::IOperator< _Ty >::operator< (const IOperator< _Ty > &) const -> bool [pure virtual]

Less.

Overrides method less.

Parameters

obj	The object to compare.

Returns

The object is less than this.

10.74.3.2 template<typename _Ty> virtual auto samchon::library::IOperator< _Ty >::operator== (const IOperator< _Ty > &) const -> bool [pure virtual]

Equal.

Overrides methods equal.

Parameters

obj	The object to compare.

Returns

The object is equal with this.

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/IOperator.hpp

10.75 samchon::protocol::IProtocol Class Reference

An interface of Invoke message chain.

```
#include <IProtocol.hpp>
```

Public Member Functions

- IProtocol ()

  Default Constructor.
- virtual void replyData (std::shared_ptr< Invoke >)

  Reply a message.
- virtual void sendData (std::shared_ptr< Invoke >)

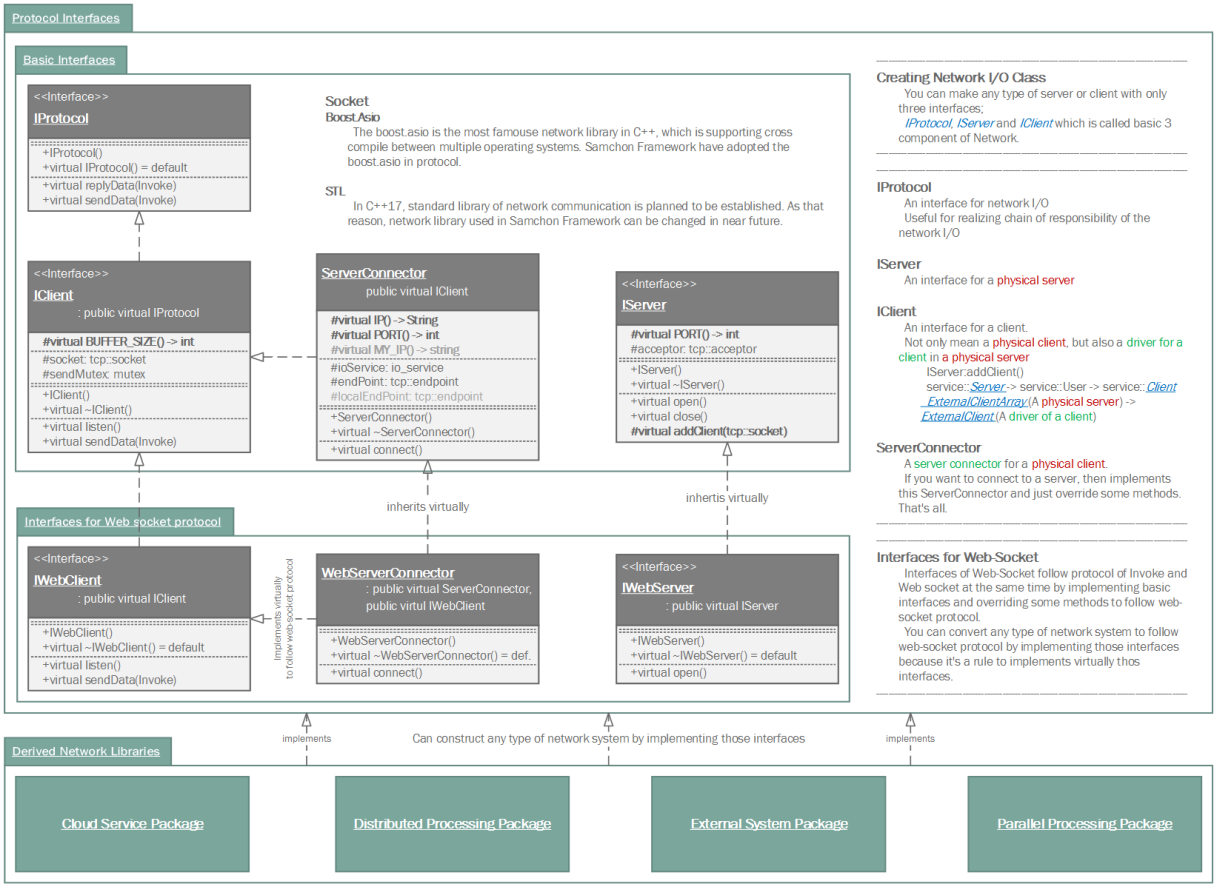
  Send a message.

10.75.1 Detailed Description

An interface of Invoke message chain.

IProtocol is an interface for Invoke message, which is standard message of network I/O in Samchon Framework, chain. The IProtocol interface is used to network drivers and some classes which are in a relationship of chain of responsibility with those network drivers.

IProtocol is one of the basic 3 + 1 components that can make any type of network system in Samchon Framework with IServer and IClient. Following the "chain of responsibility" pattern, looking around classes in Samchon Framework, you can see all related classes with network I/O are implemented from the IProtocol.



Author

Jeongho Nam

Definition at line 31 of file IProtocol.hpp.

10.75.2 Constructor & Destructor Documentation

10.75.2.1 IProtocol::IProtocol ( )

Default Constructor.

Definition at line 4 of file IProtocol.cpp.

10.75.3 Member Function Documentation

10.75.3.1 virtual void samchon::protocol::IProtocol::replyData (std::shared_ptr< Invoke >) [virtual]

Reply a message.

Handles a replied Invoke message or shifts the responsibility to related chain.

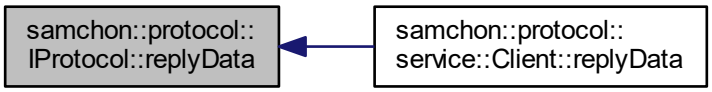
Parameters

invoke	An Invoke message gotten from a network system

Reimplemented in samchon::protocol::service::User, samchon::protocol::service::Client, samchon::protocol::master::DistributedSystem, samchon::protocol::ExternalSystemArray, samchon::protocol::ExternalSystem, and samchon::protocol::master::DistributedSlaveSystemMediator.

Referenced by samchon::protocol::service::Client::replyData().

Here is the caller graph for this function:



10.75.3.2 virtual void samchon::protocol::IProtocol::sendData (std::shared_ptr< Invoke >) [virtual]

Send a message.

Sends Invoke message to a network system or shifts the responsibility to related chain.

Parameters

invoke	An Invoke message to send to a network system

Reimplemented in samchon::protocol::service::User, samchon::protocol::master::DistributedSystem, samchon::protocol::service::Client, samchon::protocol::ExternalSystemRole, samchon::protocol::External↩SystemArray, samchon::protocol::master::DistributedSystemRole, samchon::protocol::service::Service, samchon::protocol::IClient, samchon::protocol::IWebClient, and samchon::protocol::slave::SlaveSystem.

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

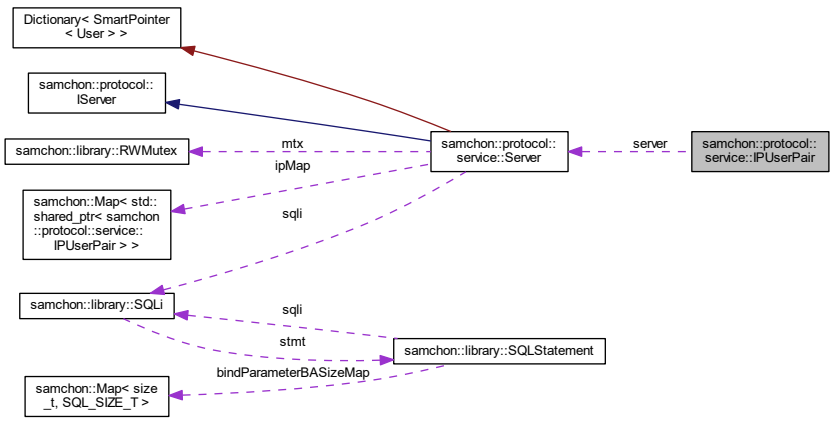
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IProtocol.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IProtocol.cpp

10.76 samchon::protocol::service::IPUserPair Class Reference

Relationship between IP address and User for issuing session.

`#include <IPUserPair.hpp>`

Collaboration diagram for samchon::protocol::service::IPUserPair:



Private Member Functions

- IPUserPair (Server *server, const std::string &)

  Construct from Server and IP address.
- auto getSessionID (Socket *, size_t) -> std::string

  Get a session ID.
- auto issueSessionID (size_t) const -> std::string

  Issue a session ID.

Private Attributes

- Server * server

  The Server containing IP addresses and User(s)
- std::string ip

  An IP address.
- std::set< const User * > userSet

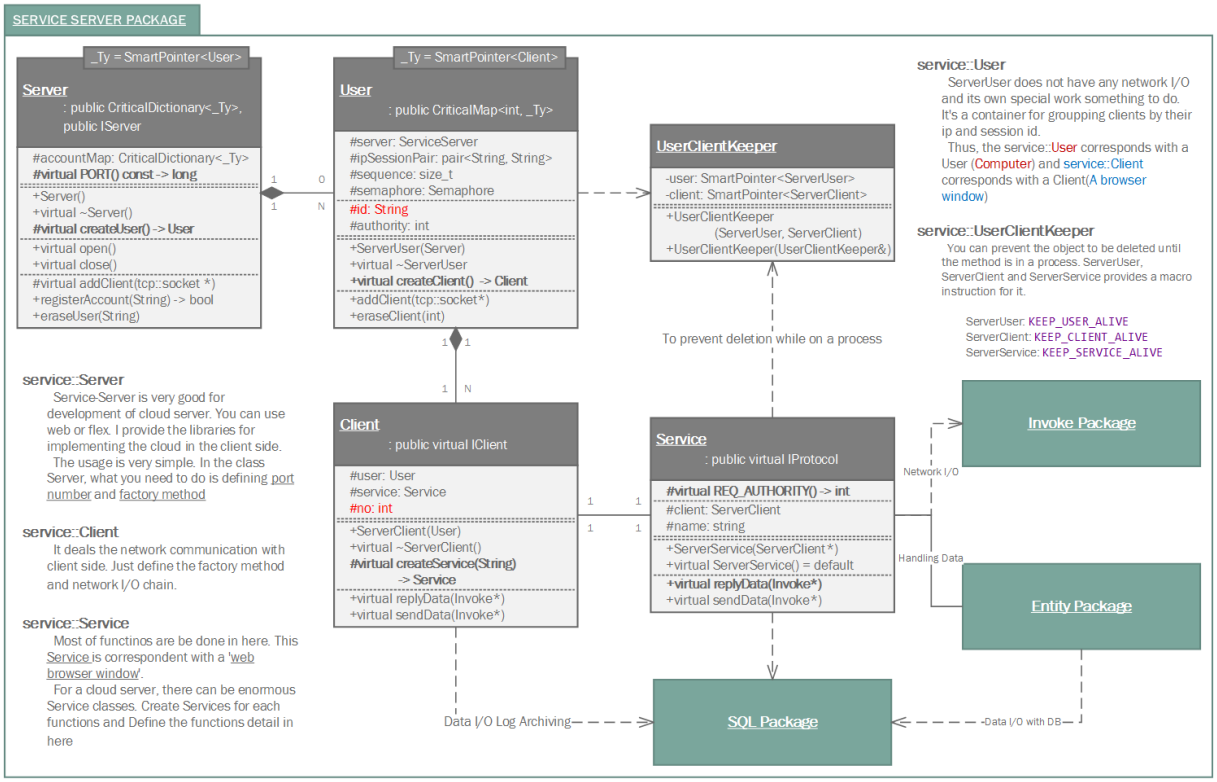
  User(s) belongs to the IP address.

10.76.1 Detailed Description

Relationship between IP address and User for issuing session.

IPUserPair is a class representing 1:N relationship between IP address and User.

The key role of IPUserPair is to issuing session ID, that is published to User(s), within a critical section of an IP address.



Note

Samchon Framework is for Cloud Server basically, so I've adopted Session ID to discriminate unique User to follow standard. But I think there can be another good strategies something like a using MAC address instead of publishing Session-ID to discriminate uniqueness of User. It's okay to adapt those strategies and rebuild my framework's source to be suitable for those.

Warning

When changing the strategy, you've to careful about policy of each system or libraries.

Followings are something to consider when adjusting MAC address for uniqueness.

- In server side and view of socket, boost::asio or candidate network i/o libary of stl does not allow accessing otherside system's MAC address. But basic socket of window or linux are allowed.
- In client side and view of browser, IE allows to get MAC address by javascript with ActiveX, but other browsers do not allow.

See also

samchon::protocol
samchon::protocol::service

Author

Jeongho Nam

Definition at line 50 of file IPUserPair.hpp.

10.76.2 Constructor & Destructor Documentation

10.76.2.1 IPUserPair::IPUserPair (Server * *server,* const std::string &) [private]

Construct from Server and IP address.

Definition at line 25 of file IPUserPair.cpp.

10.76.3 Member Function Documentation

10.76.3.1 auto IPUserPair::getSessionID (Socket * *socket,* size_t *sequence*) -> std::string [private]

Get a session ID.

Asks Whether client to have a session id stored in cookie.

If client has, get the session id from client, but if not, issues a new session ID and notify it(enrolled new session ID) to client. Additionally, if the session id was gotten from a client, inspect validate of the session ID.

Those tasks will hold a critical section of a global IP address

Warning

The session id must be unique

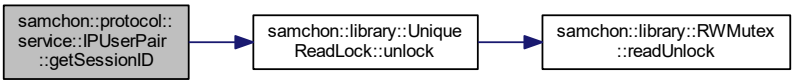
Returns

A new session ID that can ensure uniqueness

Definition at line 30 of file IPUserPair.cpp.

References samchon::library::UniqueReadLock::unlock().

Here is the call graph for this function:



10.76.3.2 auto IPUserPair::issueSessionID (size_t *sequence*) const -> std::string [private]

Issue a session ID.

Issues a session ID when a target client haven't issued seesion ID yet

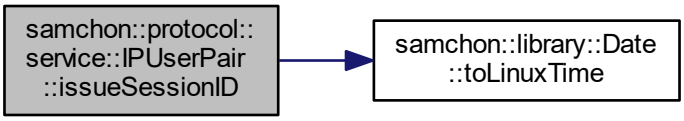
Default organization of the Session ID

- {$name}::{$port}::{$sequence}::{$linux_time}
  1. $name: Name of the Server; Server::NAME()
  2. $port: Port number of the Server; Server::PORT()
  3. $sequence: Sequence number of creation of User in Server
  4. $linux_time: Linux time of current time.
     If the object Server is re-constructed, $sequence can be duplicated. $linux_time is to avoid the duplication so that can ensure uniqueness on session id

Definition at line 81 of file IPUserPair.cpp.

References samchon::library::Date::toLinuxTime().

Here is the call graph for this function:



10.76.4 Member Data Documentation

10.76.4.1 Server* samchon::protocol::service::IPUserPair::server [private]

The Server containing IP addresses and User(s)

Definition at line 59 of file IPUserPair.hpp.

10.76.4.2 std::string samchon::protocol::service::IPUserPair::ip [private]

An IP address.

Definition at line 64 of file IPUserPair.hpp.

10.76.4.3 std::set<const User*> samchon::protocol::service::IPUserPair::userSet [private]

User(s) belongs to the IP address.

Definition at line 69 of file IPUserPair.hpp.

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/service/IPUserPair.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/service/IPUserPair.cpp

10.77 samchon::protocol::IServer Class Reference

An interface of a physical server.

```
#include <IServer.hpp>
```

Public Member Functions

- IServer ()
  Default Constructor.
- virtual void open ()
  Open the server.
- virtual void close ()
  Close the server.

Protected Member Functions

- virtual auto MY_IP () const -> std::string
  (optional) Server's IP
- virtual auto PORT () const -> int=NULL
  Port number of the server.
- virtual void addClient (Socket *)=0
  Handling connection of a physical client.

Protected Attributes

- Acceptor * acceptor
  An acceptor for clients.

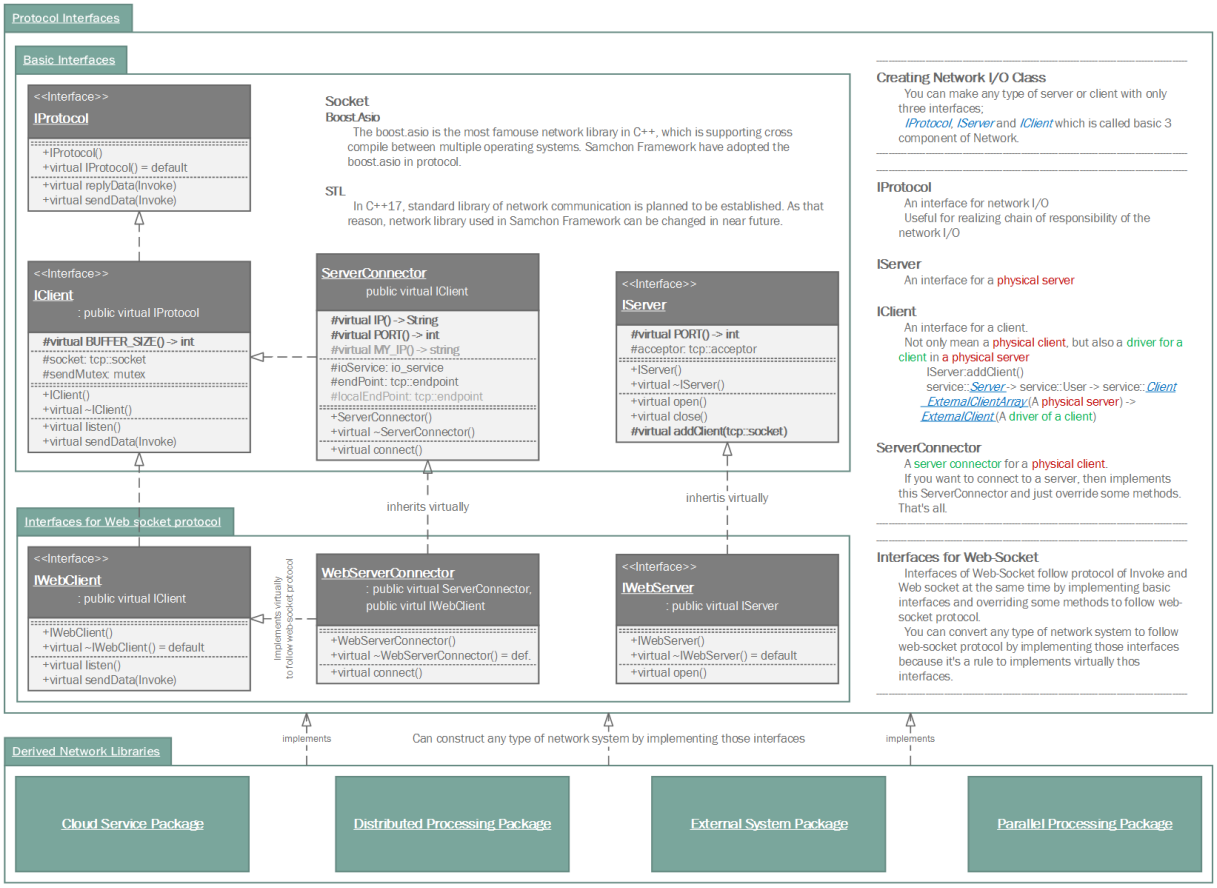
10.77.1 Detailed Description

An interface of a physical server.

IServer provides methods for opening a server.

IServer is one of the basic 3 + 1 components that can make any type of network system in Samchon Framework with IProtocol and IClient. Looking around classes in Samchon Framework, you can see all servers are implemented from the IServer.

When a client connects to the server, the abstract method IServer::addClient() is called with a new thread. If you want to accept only a client at a time, use OneToOneServer instead.



Example source

A simple chat server running on console

examples/console_chat_server/ChatServer.hpp

```
1 #pragma once
2 #include <samchon/protocol/IServer.hpp>
3 #include <samchon/protocol/IProtocol.hpp>
4
5 #include <set>
6 #include <samchon/library/RWMutex.hpp>
7
8 namespace samchon
9 {
10     namespace example
11     {
21         namespace console_chat_server
22         {
23             class ChatClient;
24
30             class ChatServer
31                 : public protocol::IServer,
32                 public protocol::IProtocol
33             {
34             private:
35                 typedef protocol::IServer super;
36
40                 std::set<ChatClient*> clientSet;
41
45                 library::RWMutex rwMutex;
46
47             public:
51                 ChatServer();
52                 virtual ~ChatServer() = default;
53
57                 void eraseClient(ChatClient *client);
58
59             protected:
63                 virtual auto PORT() const -> int override
64                 {
```

```
65                      return 37749;
66                  };
67
71                  virtual void addClient(protocol::Socket *socket) override;
72
73              public:
80                  virtual void replyData(std::shared_ptr<protocol::Invoke>) override;
81
88                  virtual void sendData(std::shared_ptr<protocol::Invoke>) override;
89              };
90          };
91      };
92 };
```

examples/console_chat_server/ChatServer.cpp

```
1 #include <samchon/examples/console_chat_server/ChatServer.hpp>
2 #include <samchon/examples/console_chat_server/ChatClient.hpp>
3
4 #include <thread>
5 #include <samchon/protocol/Invoke.hpp>
6
7 using namespace std;
8 using namespace samchon::library;
9 using namespace samchon::protocol;
10 using namespace samchon::example::console_chat_server;
11
12 ChatServer::ChatServer()
13     : super()
14 {
15 }
16 void ChatServer::addClient(Socket *socket)
17 {
18     ChatClient *client = new ChatClient(this, socket);
19
20     UniqueWriteLock uk(rwMutex);
21     clientSet.insert(client);
22
23     thread(&ChatClient::listen, client).detach();
24 }
25 void ChatServer::eraseClient(ChatClient *client)
26 {
27     UniqueWriteLock uk(rwMutex);
28     clientSet.erase(client);
29 }
30
31 void ChatServer::replyData(shared_ptr<Invoke> invoke)
32 {
33     sendData(invoke);
34 }
35 void ChatServer::sendData(shared_ptr<Invoke> invoke)
36 {
37     UniqueReadLock uk(rwMutex);
38
39     for(auto it = clientSet.begin(); it != clientSet.end(); it++)
40         (*it)->sendData(invoke);
41 }
```

examples/console_chat_server/ChatClient.hpp

```
1 #pragma once
2 #include <samchon/protocol/IClient.hpp>
3
4 namespace samchon
5 {
6     namespace example
7     {
8         namespace console_chat_server
9         {
10             class ChatServer;
11
21             class ChatClient
22                 : public protocol::IClient
23             {
24             private:
25                 typedef protocol::IClient super;
26
30                 ChatServer *server;
31
32             public:
39                 ChatClient(ChatServer*, protocol::Socket*);
40
45                 virtual ~ChatClient();
46
51                 virtual void replyData(std::shared_ptr<protocol::Invoke>) override;
52             };
```

```
        };
    };
};
```

examples/console_chat_server/ChatClient.cpp

```
#include <samchon/examples/console_chat_server/ChatClient.hpp>
#include <samchon/examples/console_chat_server/ChatServer.hpp>

#include <samchon/protocol/Invoke.hpp>

using namespace std;
using namespace samchon::library;
using namespace samchon::protocol;
using namespace samchon::example::console_chat_server;

ChatClient::ChatClient(ChatServer *server, Socket *socket)
    : super()
{
    this->server = server;
    this->socket = socket;
}
ChatClient::~ChatClient()
{
    server->eraseClient(this);
}

void ChatClient::replyData(shared_ptr<Invoke> invoke)
{
    server->replyData(invoke);
}
```

examples/console_chat_server/main.cpp

```
#include <iostream>
#include <samchon/examples/console_chat_server/ChatServer.hpp>

#ifdef _WIN64
#   ifdef _DEBUG
#       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "x64/Release/SamchonFramework.lib")
#   endif
#else
#   ifdef _DEBUG
#       pragma comment(lib, "Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "Release/SamchonFramework.lib")
#   endif
#endif

using namespace samchon::example::console_chat_server;

void main()
{
    ChatServer chatServer;
    chatServer.open();

    system("pause");
}
```

Author

Jeongho Nam

Definition at line 43 of file IServer.hpp.

10.77.2 Constructor & Destructor Documentation

10.77.2.1 IServer::IServer ( )

Default Constructor.

Definition at line 18 of file IServer.cpp.

10.77.3 Member Function Documentation

10.77.3.1 auto IServer::MY_IP ( ) const -> std::string [protected], [virtual]

(optional) Server's IP

Reimplemented in samchon::protocol::ExternalClientArray.

Definition at line 16 of file IServer.cpp.

10.77.3.2 virtual auto samchon::protocol::IServer::PORT ( ) const -> int=NULL [protected], [virtual]

Port number of the server.

Reimplemented in samchon::example::console_chat_server::ChatServer, samchon::protocol::External↩ ClientArray, and samchon::example::chat_service::ChatServer.

10.77.3.3 void IServer::open ( ) [virtual]

Open the server.

Note

It holds (monopolies) a thread.

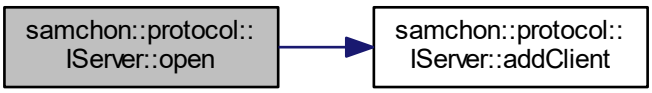
Reimplemented in samchon::protocol::IWebServer.

Definition at line 30 of file IServer.cpp.

References addClient().

Referenced by samchon::protocol::ExternalClientArray::start().

Here is the call graph for this function:



Here is the caller graph for this function:

10.77.3.4 void IServer::close ( ) [virtual]

Close the server.

Definition at line 71 of file IServer.cpp.

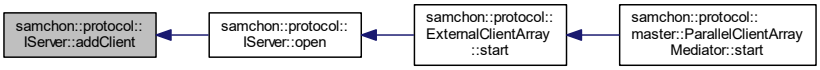
10.77.3.5 virtual void samchon::protocol::IServer::addClient (Socket *) [protected], [pure virtual]

Handling connection of a physical client.

Implemented in samchon::protocol::service::Server, samchon::example::console_chat_server::ChatServer, and samchon::protocol::ExternalClientArray.

Referenced by open().

Here is the caller graph for this function:



10.77.4 Member Data Documentation

10.77.4.1 Acceptor* samchon::protocol::IServer::acceptor [protected]

An acceptor for clients.

Definition at line 49 of file IServer.hpp.

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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IServer.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IServer.cpp

10.78 samchon::protocol::ISQLEntity Class Reference

An interface supporting DB-I/O.

```
#include <ISQLEntity.hpp>
```

Public Member Functions

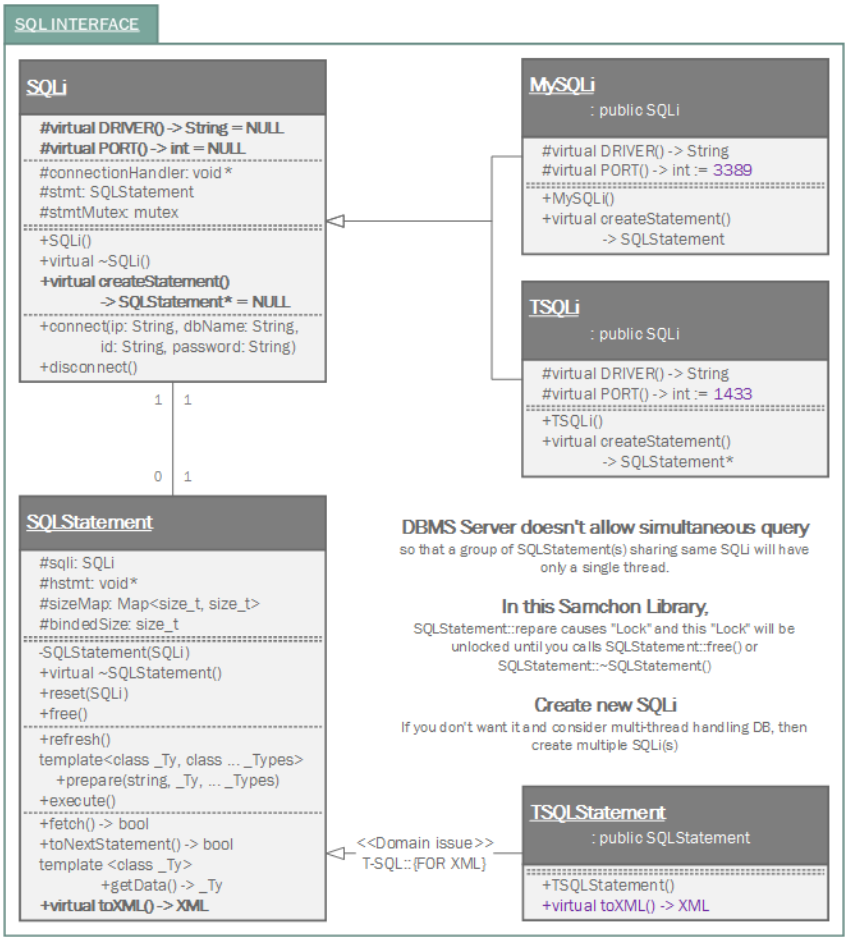
- ISQLEntity ()
  Default Constructor.
- virtual void load (std::shared_ptr< library::SQLStatement > stmt)
  Load data of entity from DB.
- virtual void archive (std::shared_ptr< library::SQLStatement > stmt)
  Archive data of entity to DB.
- virtual auto toSQL () const -> std::string
  Get a sql-statement string represents the entity.

10.78.1 Detailed Description

An interface supporting DB-I/O.

ISQLEntity is an interface for interacting with Database. By implementing load() and archive() methods, you can make an entity to interact with a Database.

If an entity implementing the ISQLEntity is a type of EntityGroup and its children also implemented the IS↩QLEntity, you can realize construction and archiving chain by each child entity to do their I/O procedures by their own.



Author

Jeongho Nam

Definition at line 32 of file ISQLEntity.hpp.

10.78.2 Constructor & Destructor Documentation

10.78.2.1 ISQLEntity::ISQLEntity ( )

Default Constructor.

Definition at line 9 of file ISQLEntity.cpp.

10.78.3 Member Function Documentation

10.78.3.1 void ISQLEntity::load (std::shared_ptr< library::SQLStatement > *stmt*) [virtual]

Load data of entity from DB.

Loads and constructs data of the entity from a SQLStatement.

If an entity implementing the ISQLEntity is a type of EntityGroup and its children also implemented the ISQLEntity, you can realize construction chain by each child entity fetches data of their own by themselves.

Parameters

stmt	SQLStatement storing data of the Entity

Definition at line 13 of file ISQLEntity.cpp.

10.78.3.2 void ISQLEntity::archive (std::shared_ptr< library::SQLStatement > *stmt*) [virtual]

Archive data of entity to DB.

Archives data of the entity to Database by a SQLStatement.

If an entity implementing the ISQLEntity is a type of EntityGroup and its children also implemented the ISQLEntity, you can realize archiving chain by each child entity archives data of their own by themselves.

Parameters

stmt	SQLStatement would store data of the Entity

Definition at line 16 of file ISQLEntity.cpp.

10.78.3.3 auto ISQLEntity::toSQL ( ) const -> std::string [virtual]

Get a sql-statement string represents the entity.

ISQLEntity::toSL() is an abstract method returns a sql-statement string.

However, if there's some parameter in a sql-statement string from the toString() method, it is categorized in dynamic sql. In that case, do not access table directly but indirect from sql-procedure. Indirecting procedure is good for performance and security.

Returns

SQL storing record(s) to temporary table of a Procedure

Definition at line 20 of file ISQLEntity.cpp.

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

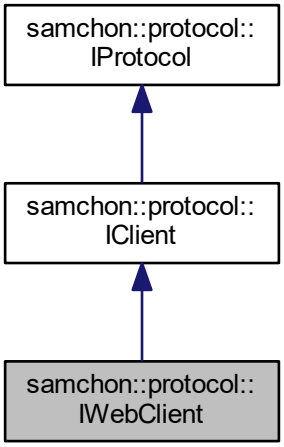
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ISQLEntity.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ISQLEntity.cpp

10.79 samchon::protocol::IWebClient Class Reference

An interface for a web-client.

```
#include <IWebClient.hpp>
```

Collaboration diagram for samchon::protocol::IWebClient:



Public Member Functions

- IWebClient ()
  Default Constructor.
- virtual void listen () override
  Listens message from a related system.
- virtual void sendData (std::shared_ptr< Invoke >) override
  Send Invoke message following web-socket protocol.

Additional Inherited Members

10.79.1 Detailed Description

An interface for a web-client.

IWebClient is a IClient following web-socket protocol.

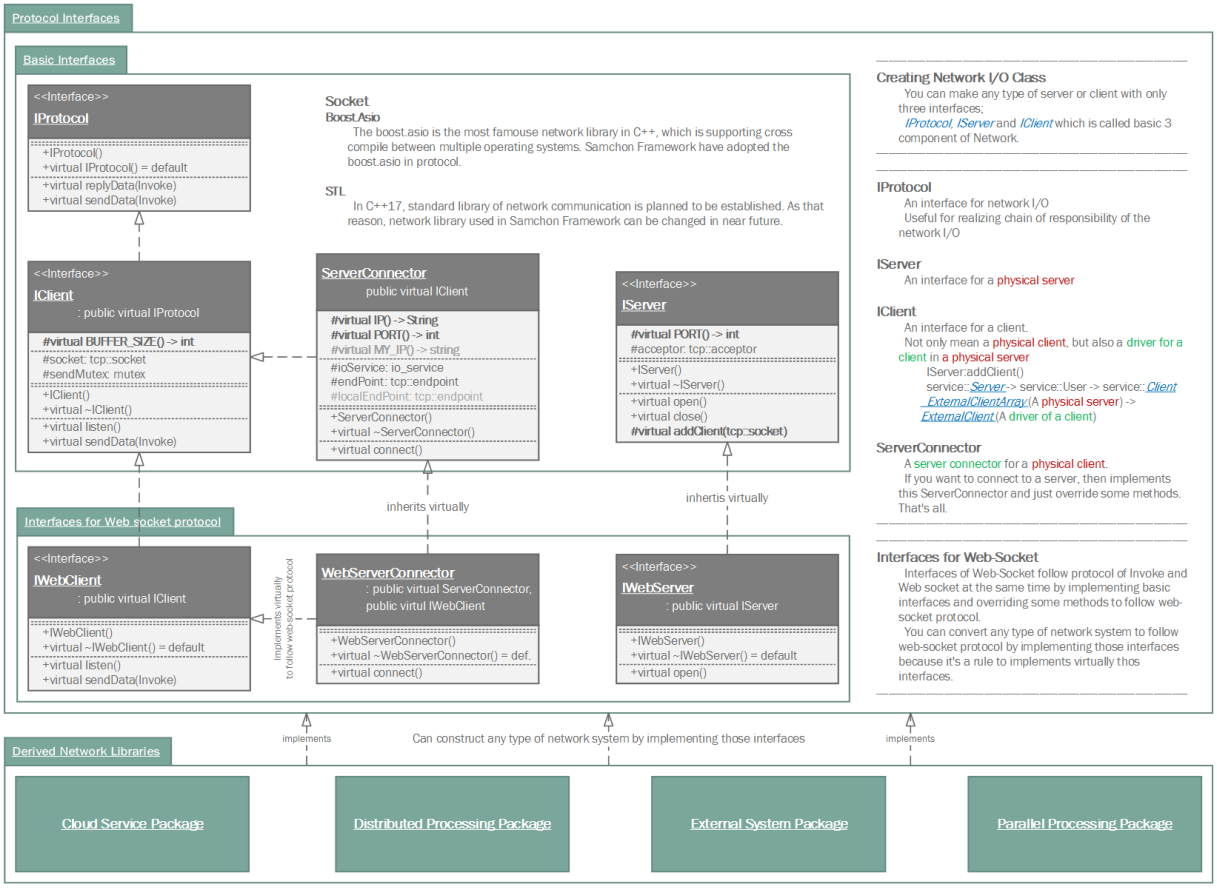
[Inherited]

IClient is an interface not only for a physical client, but also a driver for a client in a physical server.

- IServer:addClient()
  - service::Server -> service::User -> service::Client
  - ExternalClientArray (A physical server) -> ExternalClient (A driver of a client)

IClient is one of the basic 3 + 1 components that can make any type of network system in Samchon Framework with IProtocol and IServer. Looking around classes in Samchon Framework, you can see all classes related with client are implemented from the IClient.

IClient takes a role of real network I/O. IClient sends data to otherside network system from IProtocol::sendData(). IClient also gets data from otherside network system, generates standard Invoke message from parsing the replied data and shift the Invoke message to IProtocol::replyData().



Author

Jeongho Nam

Definition at line 19 of file IWebClient.hpp.

10.79.2 Constructor & Destructor Documentation

10.79.2.1 samchon::protocol::IWebClient::IWebClient ( )

Default Constructor.

10.79.3 Member Function Documentation

10.79.3.1 virtual void samchon::protocol::IWebClient::listen ( ) [override], [virtual]

Listens message from a related system.

Note

It holds (monopolies) a thread.

Reimplemented from samchon::protocol::IClient.

10.79.3.2 virtual void samchon::protocol::IWebClient::sendData (std::shared_ptr< Invoke >) [override], [virtual]

Send Invoke message following web-socket protocol.

Parameters

invoke	Invoke message to send associated network system.

Reimplemented from samchon::protocol::IClient.

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

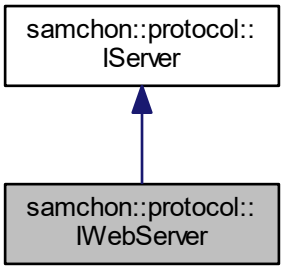
– D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IWebClient.hpp

10.80 samchon::protocol::IWebServer Class Reference

An interface for a physical server following web-socket.

`#include <IWebServer.hpp>`

Collaboration diagram for samchon::protocol::IWebServer:



Public Member Functions

– IWebServer ()

Default Constructor.

– virtual void open () override

Open the web-socket server.

Additional Inherited Members

10.80.1 Detailed Description

An interface for a physical server following web-socket.

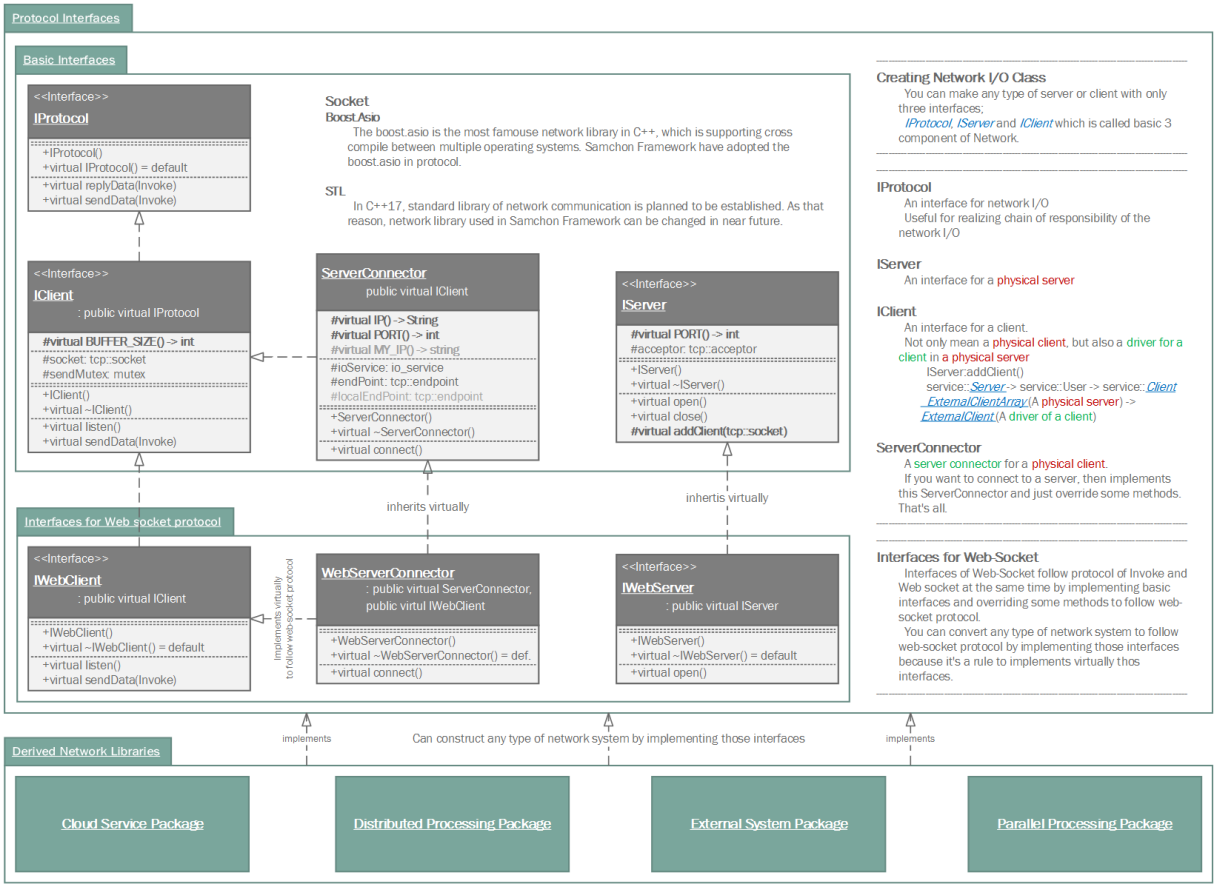
IWebServer is a IServer following web-socket protocol.

[Inherited]

IServer provides methods for opening a server.

IServer is one of the basic 3 + 1 components that can make any type of network system in Samchon Framework with IProtocol and IClient. Looking around classes in Samchon Framework, you can see all servers are implemented from the IServer.

When a client connects to the server, the abstract method IServer::addClient() is called with a new thread. If you want to accept only a client at a time, use OneToOneServer instead.



Example source

A simple chat server running on console

examples/console_chat_server/ChatServer.hpp

```
1 #pragma once
2 #include <samchon/protocol/IServer.hpp>
3 #include <samchon/protocol/IProtocol.hpp>
4
5 #include <set>
6 #include <samchon/library/RWMutex.hpp>
7
8 namespace samchon
9 {
10     namespace example
11     {
21         namespace console_chat_server
22         {
23             class ChatClient;
24
30             class ChatServer
31                 : public protocol::IServer,
32                 public protocol::IProtocol
33             {
34             private:
35                 typedef protocol::IServer super;
36
40                 std::set<ChatClient*> clientSet;
41
45                 library::RWMutex rwMutex;
46
47             public:
51                 ChatServer();
52                 virtual ~ChatServer() = default;
53
57                 void eraseClient(ChatClient *client);
58
59             protected:
63                 virtual auto PORT() const -> int override
64                 {
```

```
65                      return 37749;
66                  };
67
71                  virtual void addClient(protocol::Socket *socket) override;
72
73              public:
80                  virtual void replyData(std::shared_ptr<protocol::Invoke>) override;
81
88                  virtual void sendData(std::shared_ptr<protocol::Invoke>) override;
89              };
90          };
91      };
92 };
```

examples/console_chat_server/ChatServer.cpp

```
1 #include <samchon/examples/console_chat_server/ChatServer.hpp>
2 #include <samchon/examples/console_chat_server/ChatClient.hpp>
3
4 #include <thread>
5 #include <samchon/protocol/Invoke.hpp>
6
7 using namespace std;
8 using namespace samchon::library;
9 using namespace samchon::protocol;
10 using namespace samchon::example::console_chat_server;
11
12 ChatServer::ChatServer()
13     : super()
14 {
15 }
16 void ChatServer::addClient(Socket *socket)
17 {
18     ChatClient *client = new ChatClient(this, socket);
19
20     UniqueWriteLock uk(rwMutex);
21     clientSet.insert(client);
22
23     thread(&ChatClient::listen, client).detach();
24 }
25 void ChatServer::eraseClient(ChatClient *client)
26 {
27     UniqueWriteLock uk(rwMutex);
28     clientSet.erase(client);
29 }
30
31 void ChatServer::replyData(shared_ptr<Invoke> invoke)
32 {
33     sendData(invoke);
34 }
35 void ChatServer::sendData(shared_ptr<Invoke> invoke)
36 {
37     UniqueReadLock uk(rwMutex);
38
39     for(auto it = clientSet.begin(); it != clientSet.end(); it++)
40         (*it)->sendData(invoke);
41 }
```

examples/console_chat_server/ChatClient.hpp

```
1 #pragma once
2 #include <samchon/protocol/IClient.hpp>
3
4 namespace samchon
5 {
6     namespace example
7     {
8         namespace console_chat_server
9         {
10             class ChatServer;
11
21             class ChatClient
22                 : public protocol::IClient
23             {
24             private:
25                 typedef protocol::IClient super;
26
30                 ChatServer *server;
31
32             public:
39                 ChatClient(ChatServer*, protocol::Socket*);
40
45                 virtual ~ChatClient();
46
51                 virtual void replyData(std::shared_ptr<protocol::Invoke>) override;
52             };
```

```
        };
    };
};
```

examples/console_chat_server/ChatClient.cpp

```
#include <samchon/examples/console_chat_server/ChatClient.hpp>
#include <samchon/examples/console_chat_server/ChatServer.hpp>

#include <samchon/protocol/Invoke.hpp>

using namespace std;
using namespace samchon::library;
using namespace samchon::protocol;
using namespace samchon::example::console_chat_server;

ChatClient::ChatClient(ChatServer *server, Socket *socket)
    : super()
{
    this->server = server;
    this->socket = socket;
}
ChatClient::~ChatClient()
{
    server->eraseClient(this);
}

void ChatClient::replyData(shared_ptr<Invoke> invoke)
{
    server->replyData(invoke);
}
```

examples/console_chat_server/main.cpp

```
#include <iostream>
#include <samchon/examples/console_chat_server/ChatServer.hpp>

#ifdef _WIN64
#   ifdef _DEBUG
#       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "x64/Release/SamchonFramework.lib")
#   endif
#else
#   ifdef _DEBUG
#       pragma comment(lib, "Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "Release/SamchonFramework.lib")
#   endif
#endif

using namespace samchon::example::console_chat_server;

void main()
{
    ChatServer chatServer;
    chatServer.open();

    system("pause");
}
```

Author

Jeongho Nam

Definition at line 19 of file IWebServer.hpp.

10.80.2 Constructor & Destructor Documentation

10.80.2.1 samchon::protocol::IWebServer::IWebServer ( )

Default Constructor.

10.80.3 Member Function Documentation

10.80.3.1 virtual void samchon::protocol::IWebServer::open ( ) [override], [virtual]

Open the web-socket server.

Accepts clients and handshake with them following web-socket protocol.

Note

It monopolies a thread.

Reimplemented from samchon::protocol::IServer.

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

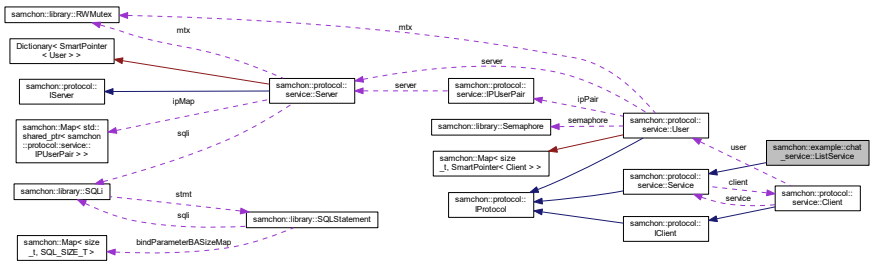
– D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/IWebServer.hpp

10.81 samchon::example::chat_service::ListService Class Reference

A list service of chat rooms.

`#include <ListService.hpp>`

Collaboration diagram for samchon::example::chat_service::ListService:



Public Member Functions

– ListService (protocol::service::Client *)

Construct from a client.

Private Member Functions

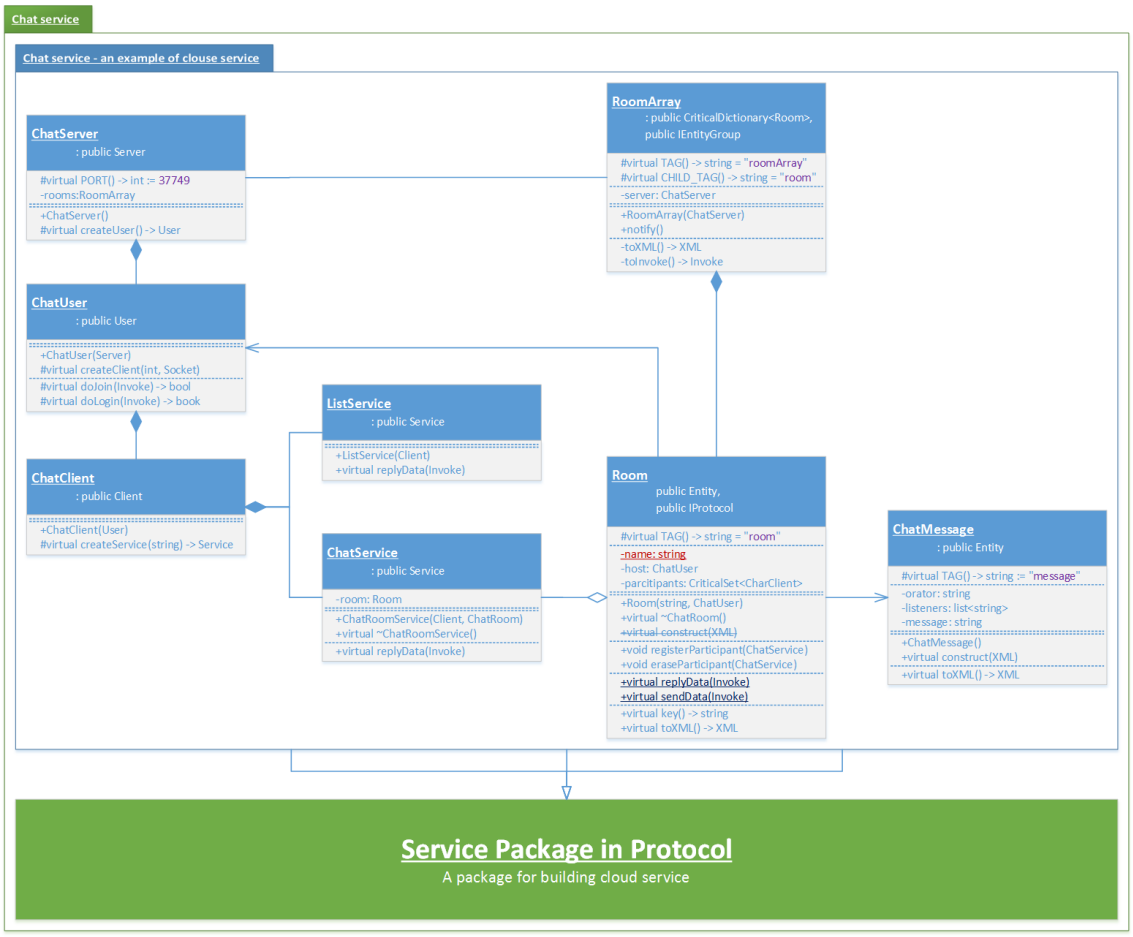
– void createRoom (const std::string &)

Create a chat room.

Additional Inherited Members

10.81.1 Detailed Description

A list service of chat rooms.



Author

Jeongho Nam

Definition at line 21 of file ListService.hpp.

10.81.2 Constructor & Destructor Documentation

10.81.2.1 samchon::example::chat_service::ListService::ListService (protocol::service::Client *)

Construct from a client.

10.81.3 Member Function Documentation

10.81.3.1 void samchon::example::chat_service::ListService::createRoom (const std::string &) [private]

Create a chat room.

Parameters

name	Name of the candidate room to create.

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

– D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/chat_service/ListService.hpp

10.82 samchon::Map< _Kty, _Ty, _Pr, _Alloc > Class Template Reference

Map.

```
#include <Map.hpp>
```

Public Member Functions

- auto has (const _Kty &key) const -> bool
  Whether have the item or not.
- auto get (const _Kty &key) -> _Ty &
  Get element.
- void set (const _Kty &key, const _Ty &val)
  Set element.
- auto pop (const _Kty &key) -> _Ty
  Pop item.

10.82.1 Detailed Description

template<typename _Kty, typename _Ty, typename _Pr = std::less<_Kty>, typename _Alloc = std::allocator<std::pair<const _Kty, _Ty>>>class samchon::Map< _Kty, _Ty, _Pr, _Alloc >

Map.

Map is a std::map some methods are modified

- Addicted methods
  - has(find(key) != end()) method is addicted
  - at method is addicted
  - set method is addicted
  - pop method is addicted
- Modified methods
  - at was deprecated (get is different with at)

[Inherited]

Maps are associative containers that store elements formed by a combination of a key value and a mapped value, following a specific order.

In a map, the key values are generally used to sort and uniquely identify the elements, while the mapped values store the content associated to this key. The types of key and mapped value may differ, and are grouped together in member type value_type, which is a pair type combining both:

- typedef pair<const Key, T> value_type;

Internally, the elements in a map are always sorted by its key following a specific strict weak ordering criterion indicated by its internal comparison object (of type Compare).

map containers are generally slower than unordered_map containers to access individual elements by their key, but they allow the direct iteration on subsets based on their order.

Maps are typically implemented as binary search trees.

The mapped values in a map can be accessed directly by their corresponding key using the bracket operator ((operator[]).

Maps are typically implemented as binary search trees.

Referenced comments of std::map

- http://www.cplusplus.com/reference/map/map/

Template Parameters

_Kty	

Type of the keys. Each element in a map is uniquely identified by its key value.

Aliased as member type map::key_type.

Template Parameters

_Ty	

Type of the mapped value. Each element in a map stores some data as its mapped value.

Aliased as member type map::mapped_type.

Template Parameters

_Pr	

A binary predicate that takes two element keys as arguments and returns a bool. The expression comp(a,b), where comp is an object of this type and a and b are key values, shall return true if a is considered to go before b in the strict weak ordering the function defines.

The map object uses this expression to determine both the order the elements follow in the container and whether two element keys are equivalent (by comparing them reflexively: they are equivalent if !comp(a,b) && !comp(b,a)). No two elements in a map container can have equivalent keys.

This can be a function pointer or a function object. This defaults to std::less<T>, which returns the same as applying the less-than operator (a<b).

Aliased as member type map::key_compare.

Template Parameters

_Alloc	

Type of the allocator object used to define the storage allocation model. By default, the allocator class template is used, which defines the simplest memory allocation model and is value-independent.

Aliased as member type map::allocator_type.

Author

Jeongho Nam

Definition at line 88 of file Map.hpp.

10.82.2 Member Function Documentation

10.82.2.1 template<typename _Kty, typename _Ty, typename _Pr = std::less<_Kty>, typename _Alloc = std::allocator<std::pair<const _Kty, _Ty>>> auto samchon::Map< _Kty, _Ty, _Pr, _Alloc >::has (const _Kty & *key*) const -> bool [inline]

Whether have the item or not.

Indicates whether a map has an item having the specified identifier.

Parameters

key	Key value of the element whose mapped value is accessed.

Returns

Whether the map has an item having the specified identifier

Definition at line 110 of file Map.hpp.

10.82.2.2 template<typename _Kty, typename _Ty, typename _Pr = std::less<_Kty>, typename _Alloc = std::allocator<std::pair<const _Kty, _Ty>>> auto samchon::Map< _Kty, _Ty, _Pr, _Alloc >::get (const _Kty & *key*) -> _Ty& [inline]

Get element.

Returns a reference to the mapped value of the element identified with key

Warning

get is different with std::map's at. get does not create object but throws excention if the matched key doesn't exist.

Parameters

key	Key value of the element whose mapped value is accessed.

Exceptions

exception	out of range

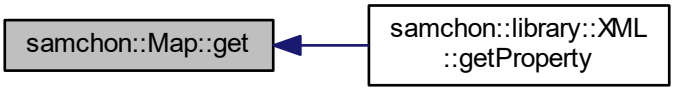
Returns

A reference object of the mapped value (_Ty)

Definition at line 129 of file Map.hpp.

Referenced by samchon::library::XML::getProperty().

Here is the caller graph for this function:



10.82.2.3 template<typename _Kty, typename _Ty, typename _Pr = std::less<_Kty>, typename _Alloc = std::allocator<std::pair<const _Kty, _Ty>>> void samchon::Map< _Kty, _Ty, _Pr, _Alloc >::set (const _Kty & *key*, const _Ty & *val*) [inline]

Set element.

Set an item as the specified identifier.

If the identifier is already in map, change value of the identifier.
If not, then insert the object with the identifier.

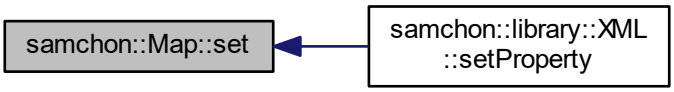
Parameters

key	Key value of the element whose mapped value is accessed.
val	Value, the item.

Definition at line 150 of file Map.hpp.

Referenced by samchon::library::XML::setProperty().

Here is the caller graph for this function:



10.82.2.4 template<typename _Kty, typename _Ty, typename _Pr = std::less<_Kty>, typename _Alloc = std::allocator<std::pair<const _Kty, _Ty>>> auto samchon::Map< _Kty, _Ty, _Pr, _Alloc >::pop (const _Kty & *key*) -> _Ty [inline]

Pop item.

Removes an item having specified key and returns the removed element.

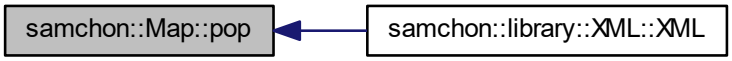
Returns

An item released by pop

Definition at line 165 of file Map.hpp.

Referenced by samchon::library::XML::XML().

Here is the caller graph for this function:



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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/Map.hpp

10.83 samchon::library::Math Class Reference

Utility of Math.

```
#include <Math.hpp>
```

Static Public Member Functions

- static auto random () -> double
  Get a random value.
- static auto degree_to_radian (double) -> double
  Convert degree to radian.
- static auto radian_to_degree (double) -> double

Convert radian to degree.

- template<typename _Cont , typename _Ty = _Cont::value_type>
  static auto minimum (const _Cont &container) -> IndexPair< _Ty >

  Calculate minimum value with its index.
- template<typename _Cont , typename _Ty = _Cont::value_type>
  static auto maximum (const _Cont &container) -> IndexPair< _Ty >

  Calculate maximum value with its index.
- template<typename _Cont , typename _Ty = _Cont::value_type>
  static auto mean (const _Cont &container) -> double

  Calculate average.
- template<typename _Cont , typename _Ty = _Cont::value_type>
  static auto median (const _Cont &container) -> double

  Get median value.
- template<typename _Cont , typename _Ty = _Cont::value_type>
  static auto mode (const _Cont &container) -> _Ty

  Find mode value.
- template<typename _Cont , typename _Ty = _Cont::value_type>
  static auto stdev_p (const _Cont &container) -> double

  Square root of a variance_p.
- template<typename _Cont , typename _Ty = _Cont::value_type>
  static auto stdev_s (const _Cont &container) -> double

  Square root of a variance_s.
- template<typename _Cont , typename _Ty = _Cont::value_type>
  static auto variance_p (const _Cont &container) -> double

  Calculate standard deviation for all population.
- template<typename _Cont , typename _Ty = _Cont::value_type>
  static auto variance_s (const _Cont &container) -> double

  Calculate variance for sample.

Static Public Attributes

- static const double E = exp(1.0)

  Exponent.
- static const double PI = 3.141592653589793

  mber Pi
- static const double LN2 = 1.0 / log2(E)

  $log_e 2$
- static const double LN10 = 1.0 / log10(E)

  $log_e 10$
- static const double LOG2E = log2(E)

  $log_2 e$
- static const double LOG10E = log10(E)

  $log_{10} e$
- static const double SQRT1_2 = sqrt(.5)

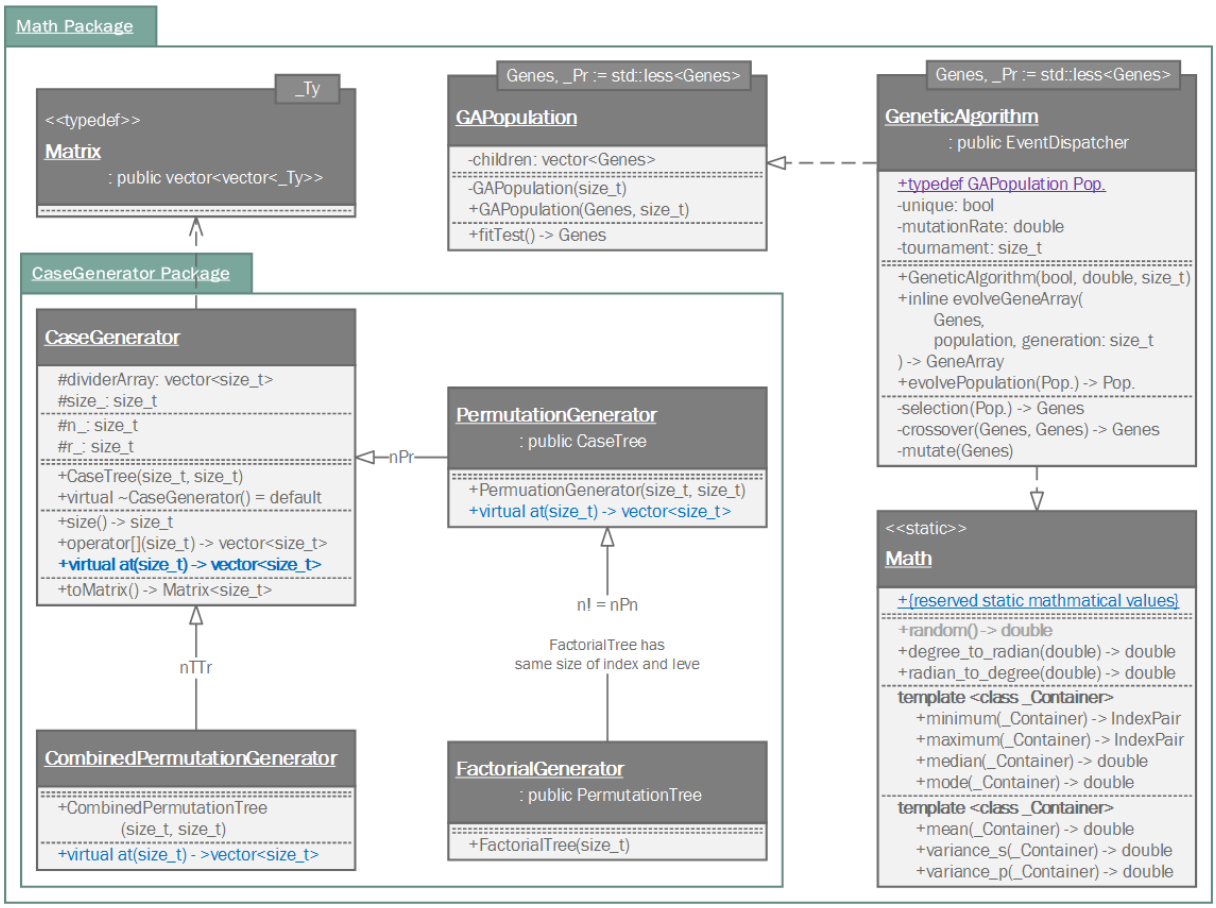
  Squart root of 0.5.
- static const double SQRT2 = sqrt(2.0)

  Squart root of 2.

10.83.1 Detailed Description

Utility of Math.

Provides some static values and mathmatical methods.



See also

samchon::library

Author

Jeongho Nam

Definition at line 23 of file Math.hpp.

10.83.2 Member Function Documentation

10.83.2.1 auto Math::random ( ) -> double [static]

Get a random value.

Returns a random value following uniform distribution

Returns

The random value

Definition at line 38 of file Math.cpp.

Referenced by samchon::library::GeneticAlgorithm< GeneArray, Compare >::crossover(), samchon::library::GeneticAlgorithm< GeneArray, Compare >::mutate(), and samchon::library::GeneticAlgorithm< GeneArray, Compare >::selection().

Here is the caller graph for this function:

10.83.2.2 auto Math::degree_to_radian (double *val*) -> double [static]

Convert degree to radian.

Definition at line 26 of file Math.cpp.

10.83.2.3 auto Math::radian_to_degree (double *val*) -> double [static]

Convert radian to degree.

Definition at line 30 of file Math.cpp.

10.83.2.4 template<typename _Cont , typename _Ty = _Cont::value_type> static auto samchon::library::Math::minimum (const _Cont & *container*) -> IndexPair<_Ty> [inline], [static]

Calculate minimum value with its index.

Template Parameters

_Cont	Type of the container
_Ty	

Value type of the container.

Do not modify the default template argument.

Parameters

container	Container to find its minimum value

Returns

Pair of minimum value and its index

Definition at line 143 of file Math.hpp.

10.83.2.5 template<typename _Cont , typename _Ty = _Cont::value_type> static auto samchon::library::Math::maximum (const _Cont & *container*) -> IndexPair<_Ty> [inline], [static]

Calculate maximum value with its index.

Template Parameters

_Cont	Type of the container
_Ty	

Value type of the container.

Do not modify the default template argument.

Parameters

container	Container to find its minimum value

Returns

Pair of maximum value and its index

Definition at line 174 of file Math.hpp.

10.83.2.6 template<typename _Cont , typename _Ty = _Cont::value_type> static auto samchon::library::Math::mean (const _Cont & *container*) -> double [inline], [static]

Calculate average.

Template Parameters

_Cont	Type of the container
_Ty	

Value type of the container.

Do not modify the default template argument.

Parameters

container	Container to find its mean

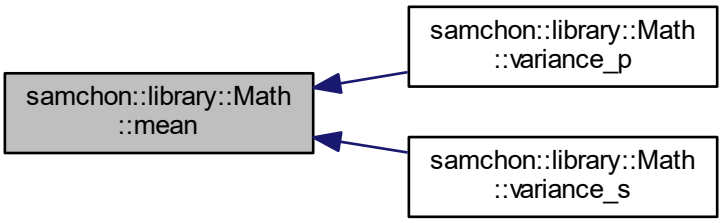
Returns

Average(mean) of the contaier

Definition at line 205 of file Math.hpp.

Referenced by variance_p(), and variance_s().

Here is the caller graph for this function:



10.83.2.7 template<typename _Cont , typename _Ty = _Cont::value_type> static auto samchon::library::Math::median (const _Cont & *container*) -> double [inline], [static]

Get median value.

Template Parameters

_Cont	Type of the container
_Ty	

Value type of the container.

Do not modify the default template argument.

Parameters

container	Container to find its median

Returns

Median value of the container

Definition at line 226 of file Math.hpp.

10.83.2.8 template<typename _Cont , typename _Ty = _Cont::value_type> static auto samchon::library::Math::mode (const _Cont & *container*) -> _Ty [inline], [static]

Find mode value.

Template Parameters

_Cont	Type of the container
_Ty	

Value type of the container.

Do not modify the default template argument.

Parameters

container	Container to find its mode

Returns

Most appeared value in the container

Definition at line 258 of file Math.hpp.

10.83.2.9 template<typename _Cont , typename _Ty = _Cont::value_type> static auto samchon::library::Math::stdev_p (const _Cont & *container*) -> double [inline], [static]

Square root of a variance_p.

$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Template Parameters

_Cont	Type of the container
_Ty	

Value type of the container.

Do not modify the default template argument.

Parameters

container	Container to find its standard deviation

Returns

Standard deviation of the container

Definition at line 282 of file Math.hpp.

10.83.2.10 template<typename _Cont , typename _Ty = _Cont::value_type> static auto samchon::library::Math::stdev_s (const _Cont & *container*) -> double [inline], [static]

Square root of a variance_s.

Template Parameters

_Cont	Type of the container
_Ty	

Value type of the container.

Do not modify the default template argument.

Parameters

container	Container to find its variance

Returns

Variance of the container

Definition at line 293 of file Math.hpp.

10.83.2.11 template<typename _Cont , typename _Ty = _Cont::value_type> static auto samchon::library::Math::variance_p (const _Cont & *container*) -> double [inline], [static]

Calculate standard deviation for all population.

Template Parameters

_Cont	Type of the container
_Ty	

Value type of the container.

Do not modify the default template argument.

Parameters

container	Container to find its standard deviation

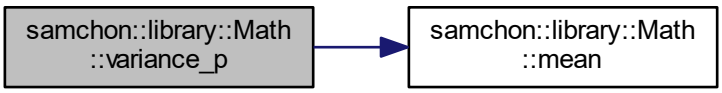
Returns

Standard deviation of the container

Definition at line 310 of file Math.hpp.

References mean().

Here is the call graph for this function:



10.83.2.12 template<typename _Cont , typename _Ty = _Cont::value_type> static auto samchon::library::Math::variance_s (const _Cont & *container*) -> double [inline], [static]

Calculate variance for sample.

Template Parameters

_Cont	Type of the container
_Ty	

Value type of the container.

Do not modify the default template argument.

Parameters

container	Container to find its variance

Returns

Variance of the container

Definition at line 333 of file Math.hpp.

References mean().

Here is the call graph for this function:

10.83.3 Member Data Documentation

10.83.3.1 const double Math::E = exp(1.0) [static]

Exponent.

A mathematical constant for the base of natural logarithms, expressed as e. The approximate value of e is 2.71828182845905.

Definition at line 36 of file Math.hpp.

10.83.3.2 const double Math::PI = 3.141592653589793 [static]

mber Pi

A mathematical constant for the ratio of the circumference of a circle to its diameter, expressed as pi, with a value of 3.141592653589793.

Definition at line 45 of file Math.hpp.

10.83.3.3 const double Math::LN2 = 1.0 / log2(E) [static]

$\log_e 2$

A mathematical constant for the natural logarithm of 2, expressed as $\log_e 2$, with an approximate value of 0.6931471805599453.

Definition at line 54 of file Math.hpp.

10.83.3.4 const double Math::LN10 = 1.0 / log10(E) [static]

$\log_e 10$

A mathematical constant for the natural logarithm of 10, expressed as $\log_e 10$, with an approximate value of 2.302585092994046.

Definition at line 63 of file Math.hpp.

10.83.3.5 const double Math::LOG2E = log2(E) [static]

$\log_2 e$

A mathematical constant for the base-2 logarithm of the constant e, expressed as $\log_2 e$, with an approximate value of 1.442695040888963387.

Definition at line 72 of file Math.hpp.

10.83.3.6 const double Math::LOG10E = log10(E) [static]

$\log_{10} e$

A mathematical constant for the base-10 logarithm of the constant e (Math.E), expressed as $\log_{10} e$, with an approximate value of 0.4342944819032518. The Math.log() method computes the natural logarithm of a number. Multiply the result of Math.log() by Math.LOG10E to obtain the base-10 logarithm.

Definition at line 83 of file Math.hpp.

10.83.3.7 const double Math::SQRT1_2 = sqrt(.5) [static]

Squart root of 0.5.

A mathematical constant for the square root of one-half, with an approximate value of 0.7071067811865476.

Definition at line 93 of file Math.hpp.

10.83.3.8 const double Math::SQRT2 = sqrt(2.0) [static]

Squart root of 2.

A mathematical constant for the square root of 2, with an approximate value of 1.4142135623730951.

Definition at line 102 of file Math.hpp.

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

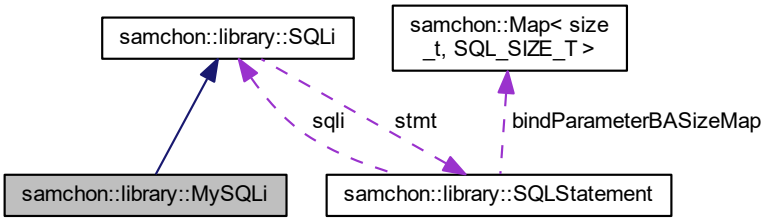
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/Math.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/Math.cpp

10.84 samchon::library::MySQLi Class Reference

A SQL interface for My-SQL.

```
#include <MySQLi.hpp>
```

Collaboration diagram for samchon::library::MySQLi:



Public Member Functions

- MySQLi (int port=3306)

  Construct from port.

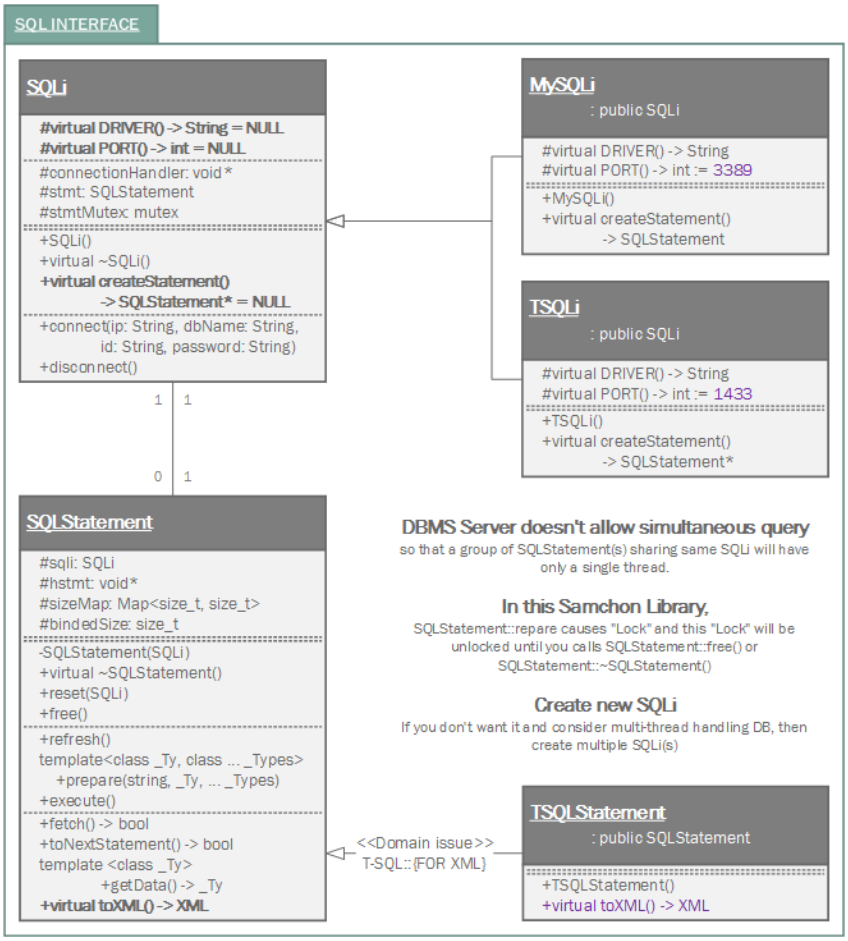
Additional Inherited Members

10.84.1 Detailed Description

A SQL interface for My-SQL.

SQLi is an abstract class managing connection (session) with a specific database and manages SQL statements are executed and results are returned within the context of a connection.

The reason why SQLStatement(s) are managed by SQLi is because of a principle of DBMS, DBMS system doesn't allow simultaneous query from a session (connection). Only a query (process0) is allowed at a time. If you try to simultaneous query from a SQLi, ODBC throws error. To avoid the error, an execute of query from SQLStatement will lock a mutex of SQLi to ensure exclusiveness.



Note

To ensure the exclusiveness, you've to make SQLStatement from SQLi. Do not make SQLStatement by yourself. call SQLi::createSQLStatement() instead. Even you make a derived class from SQLStatement, don't make its constructor to have public accessor.

Becuase execution of a sql statement causes lock on mutex in SQLi, you've to destruct the SQLStatement or call SQLStatement::free() method(). If you don't, the mutex will not be unlocked, thus you can't do anything by the SQLi.

See also

samchon::library

Author

Jeongho Nam

Definition at line 17 of file MySQLi.hpp.

10.84.2 Constructor & Destructor Documentation

10.84.2.1 samchon::library::MySQLi::MySQLi (int *port* = 3306)

Construct from port.

Parameters

port	Port number of My-SQL. default is 3306.

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

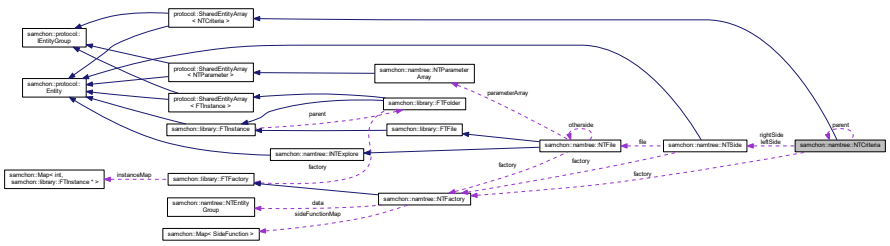
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/MySQLi.hpp

10.85 samchon::namtree::NTCriteria Class Reference

Criteria, a conditional expression with weight.

`#include <NTCriteria.hpp>`

Collaboration diagram for samchon::namtree::NTCriteria:



Public Member Functions

- virtual void construct (std::shared_ptr< library::XML >) override
  Construct data of the Entity from an XML object.
- virtual auto TAG () const -> std::string override
  A tag name when represented by XML.
- virtual auto CHILD_TAG () const -> std::string override
  A tag name of children.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override
  Get an XML object represents the EntityGroup.

Protected Member Functions

- virtual auto createChild (std::shared_ptr< library::XML >) -> NTCriteria *override
  Factory method of a child Entity.

10.85.1 Detailed Description

Criteria, a conditional expression with weight.

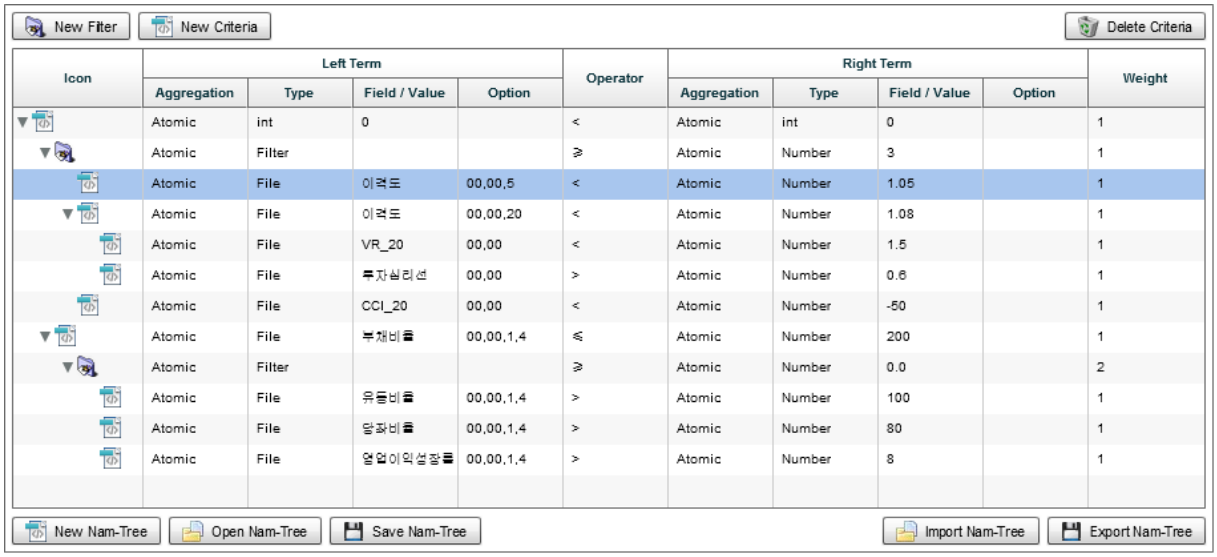
NTCriteria is an object realizing ANN(Artificial Neural Network).

You can make the ANN model having weight and bias.

1. A conditional expression with weight
   F(x) = (NTSide < NTSide ? 1: 0) X weight
   NTCrietria is made up for conditional expression.
   - When the expression is true, returns 1 * weight
   - When the expression is false, returns 0
2. Hierarchical relationship
   - In vertical relationship: multiply (X)

– In horizontal relationship: plus (+)

With that rule, you can make enormous conditions. I can sure there's not any condition can'be be expressed by the model.



Icon	Left Term				Operator	Right Term				Weight
	Aggregation	Type	Field / Value	Option		Aggregation	Type	Field / Value	Option	
	Atomic	int	0		<	Atomic	int	0		1
	Atomic	Filter			≥	Atomic	Number	3		1
	Atomic	File	이격도	00,00,5	<	Atomic	Number	1.05		1
	Atomic	File	이격도	00,00,20	<	Atomic	Number	1.08		1
	Atomic	File	VR_20	00,00	<	Atomic	Number	1.5		1
	Atomic	File	투자심리선	00,00	>	Atomic	Number	0.6		1
	Atomic	File	CCI_20	00,00	<	Atomic	Number	-50		1
	Atomic	File	부채비율	00,00,1,4	≤	Atomic	Number	200		1
	Atomic	Filter			≥	Atomic	Number	0.0		2
	Atomic	File	유동비율	00,00,1,4	>	Atomic	Number	100		1
	Atomic	File	당좌비율	00,00,1,4	>	Atomic	Number	80		1
	Atomic	File	영업이익설장률	00,00,1,4	>	Atomic	Number	8		1

3. Making bias

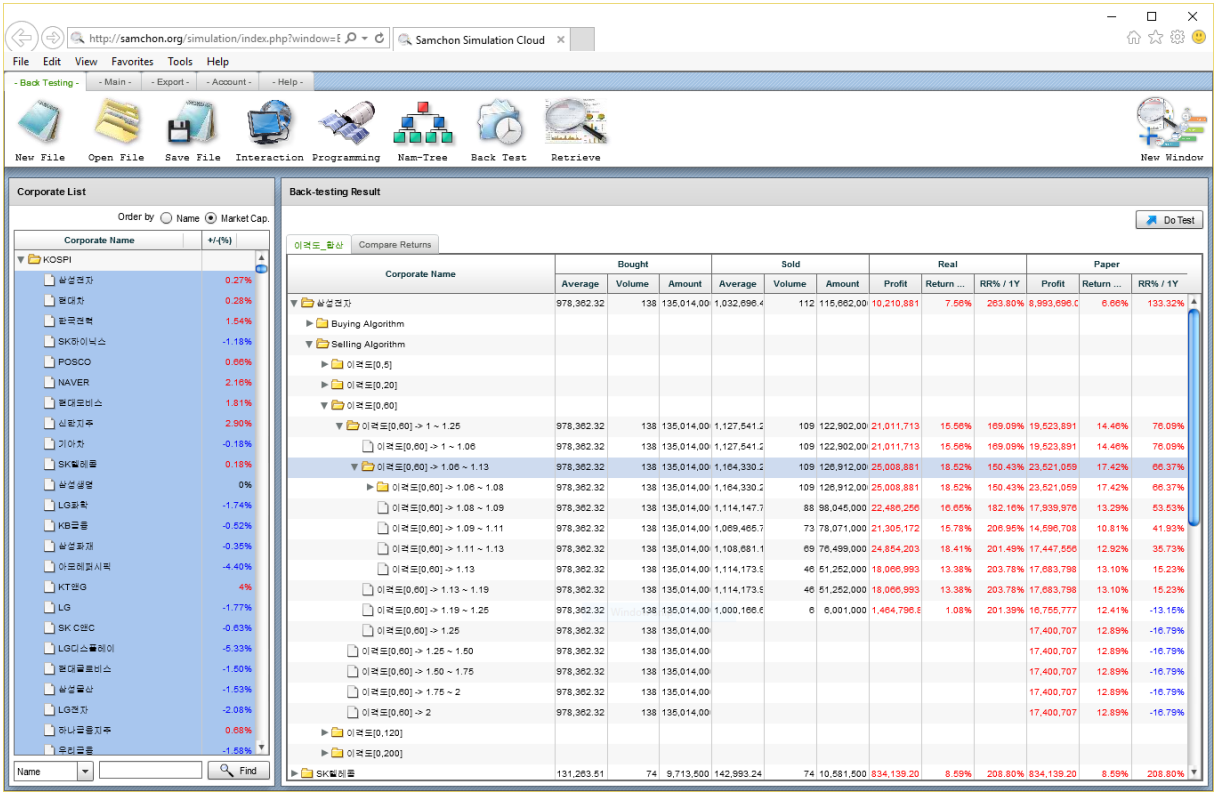
   Just make a NTCriteria returns only true.

   It's the bias returns only weight.

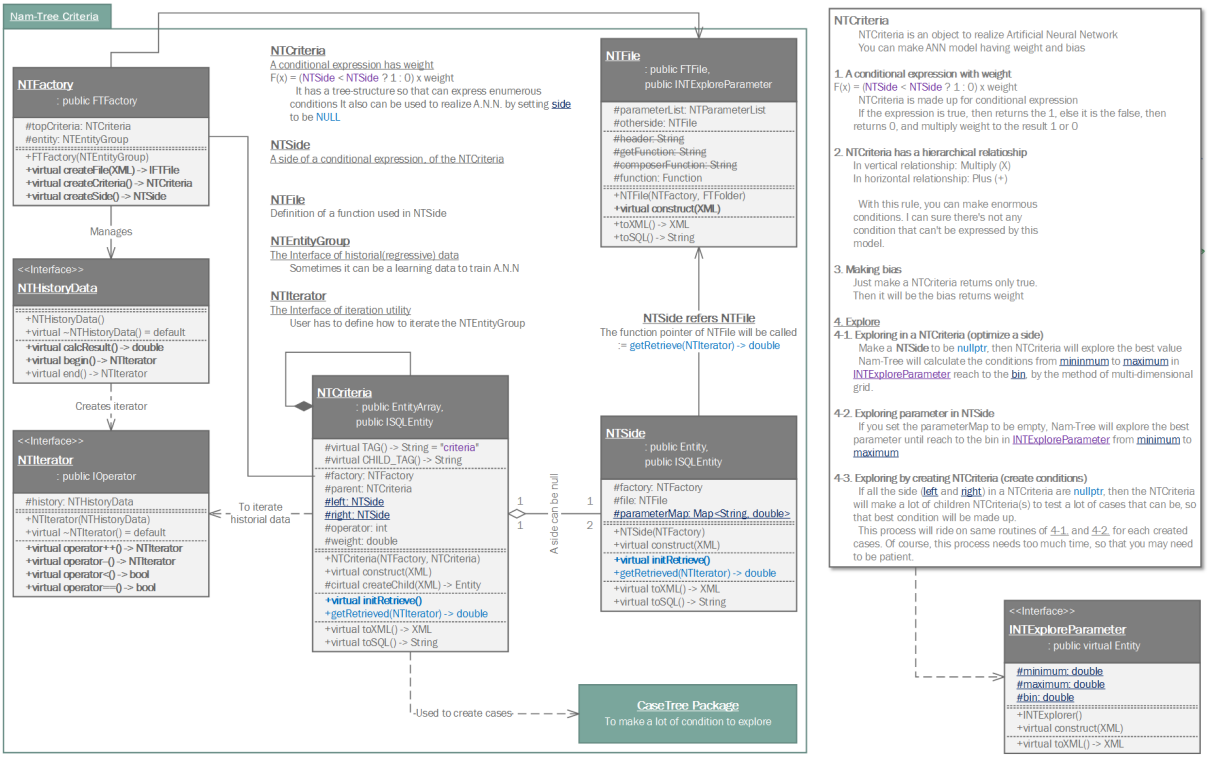
4. Explore

   (a) Exporing in a NTCriteria (optimize a side)
   Make a NTSide to be nullptr, then NTCriteria will explore the best value.
   Nam-Tree will calculate the condtions from minimum to maximum in INTExplore reach to the precision, by the method of multi-demensional grid.

   (b) Exploring parameter in NTSide
   If you set the parameterMNap to be empty, Nam-Tree will explore the best parameter until reach to the precision in INTExploreParameter from minimum to maximum.



(c) Exploring by creating NTCriteria
(Creates lots of hierarchical conditions and tests them)
If all the side (left and right) in a NTCriteria are nullptr, the NTCriteria will make a lot of children NTCriteria(s) to test lots of cases, so that derives the best condition set will be made up.
This process will ride on same routines of 4-1 and 4-2, for each created cases. Of course, this process needs too much time, so that you may need to be patient.



See also

samchon::namtree

Author

Jeongho Nam

Definition at line 77 of file NTCriteria.hpp.

10.85.2 Member Function Documentation

10.85.2.1 void NTCriteria::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

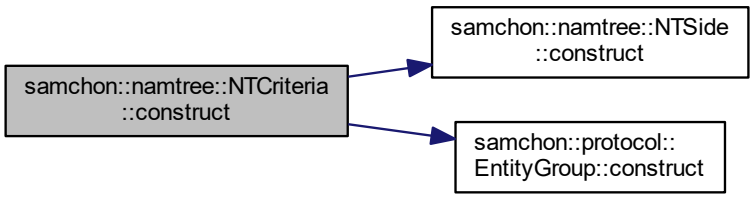
[Inherited]

Reimplemented from samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

Definition at line 34 of file NTCriteria.cpp.

References samchon::namtree::NTSide::construct(), and samchon::protocol::EntityGroup< _Container, _↩ ETy, _Ty >::construct().

Here is the call graph for this function:



10.85.2.2 auto NTCriteria::createChild (std::shared_ptr< library::XML >) -> NTCriteria *override [override], [protected], [virtual]

Factory method of a child Entity.

EntityGroup::createChild() is a factory method creating a new child Entity which is belonged to the Entity↩ Group. This method is called by EntityGroup::construct(). The children construction methods Entity↩ ::construct() will be called by abstract method of the EntityGroup::construct().

Returns

A new child Entity belongs to EntityGroup.

Implements samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

Definition at line 44 of file NTCriteria.cpp.

10.85.2.3 auto NTCriteria::TAG ( ) const -> std::string [override], [virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Definition at line 16 of file NTCriteria.cpp.

10.85.2.4 auto NTCriteria::CHILD_TAG ( ) const -> std::string [override], [virtual]

A tag name of children.

< TAG>
<CHILD_TAG />
<CHILD_TAG />
</TAG>

Implements samchon::protocol::IEntityGroup.

Definition at line 17 of file NTCriteria.cpp.

10.85.2.5 auto NTCriteria::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

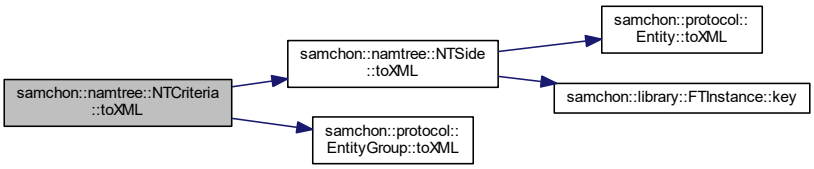
[Inherited]

Reimplemented from samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

Definition at line 70 of file NTCriteria.cpp.

References samchon::namtree::NTSide::toXML(), and samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::toXML().

Here is the call graph for this function:



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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTCriteria.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTCriteria.cpp

10.86 samchon::namtree::NTEntityGroup Class Reference

A historical, studying data.

`#include <NTEntityGroup.hpp>`

Public Member Functions

- NTEntityGroup ()

  Default Constructor.
- virtual auto begin () const -> std::shared_ptr< NTIterator >=0

  Return iterator to beginning.
- virtual auto end () const -> std::shared_ptr< NTIterator >=0

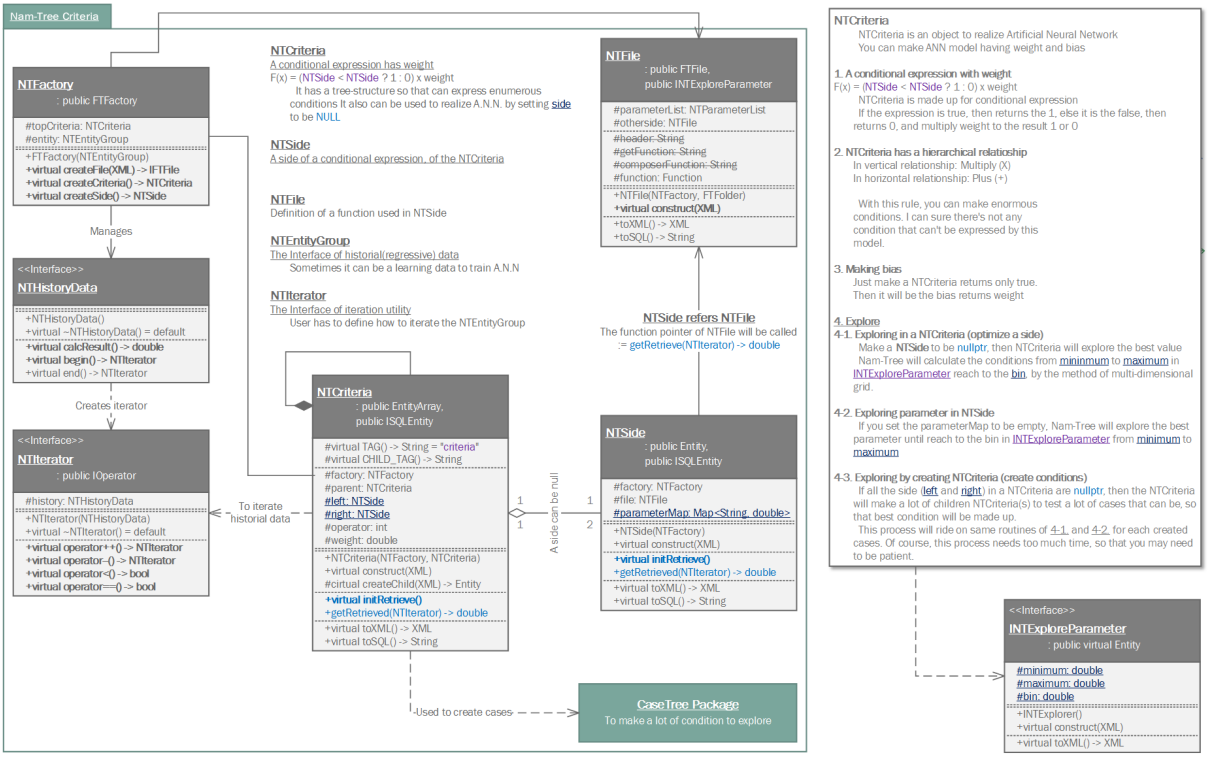
  Return iterator to end.

10.86.1 Detailed Description

A historical, studying data.

NTEntityGroup is an abstract class representing historical, regressive data

The NTEntitGroup is used to studying data evolving and training NTCriteria, representing Artificial neural network.



See also

samchon::namtree

Author

Jeongho Nam

Definition at line 26 of file NTEntityGroup.hpp.

10.86.2 Constructor & Destructor Documentation

10.86.2.1 NTEntityGroup::NTEntityGroup ( )

Default Constructor.

Definition at line 6 of file NTEntityGroup.cpp.

10.86.3 Member Function Documentation

10.86.3.1 virtual auto samchon::namtree::NTEntityGroup::begin ( ) const -> std::shared_ptr< NTIterator > [pure virtual]

Return iterator to beginning.

Returns an iterator referring the first element in the historical data container.

Note

If the container is empty, the returned iterator is same with end().

Returns

An iterator to the first element in the container.

The iterator containes the first element's pair; key and value.

10.86.3.2 virtual auto samchon::namtree::NTEntityGroup::end ( ) const -> std::shared_ptr< NTIterator > [pure virtual]

Return iterator to end.

Returns an iterator referring to the past-the-end element in the historical data container.

The past-the-end element is the theoretical element that would follow the last element in the NTEntityGroup container. It does not point to any element, and thus shall not be dereferenced.

Because the ranges used by functions of the NTEntityGroup do not include the element reference by their closing iterator, this function is often used in combination with NTEntityGroup::begin() to specify a range including all the elements in the container.

Note

Returned iterator from NTEntityGroup::end() does not refer any element. Trying to accessing element by the iterator will cause throwing exception (out of range).

If the container is empty, this function returns the same as NTEntityGroup::begin().

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

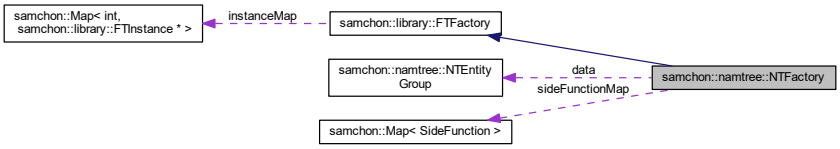
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTEntityGroup.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTEntityGroup.cpp

10.87 samchon::namtree::NTFactory Class Reference

A factory for Nam-Tree objects.

```
#include <NTFactory.hpp>
```

Collaboration diagram for samchon::namtree::NTFactory:



Public Member Functions

- NTFactory (NTEntityGroup ∗)

  Construct from historical(studying) data.
- virtual auto createCriteria (NTCriteria ∗, std::shared_ptr< library::XML >) -> NTCriteria ∗

  Factory method of a NTCriteria.
- virtual auto createSide (std::shared_ptr< library::XML >) -> NTSide ∗

  Factory method of a NTSide.

Protected Attributes

- NTEntityGroup ∗ data

  A entity gorup containing historical data.
- Dictionary< SideFunction > sideFunctionMap

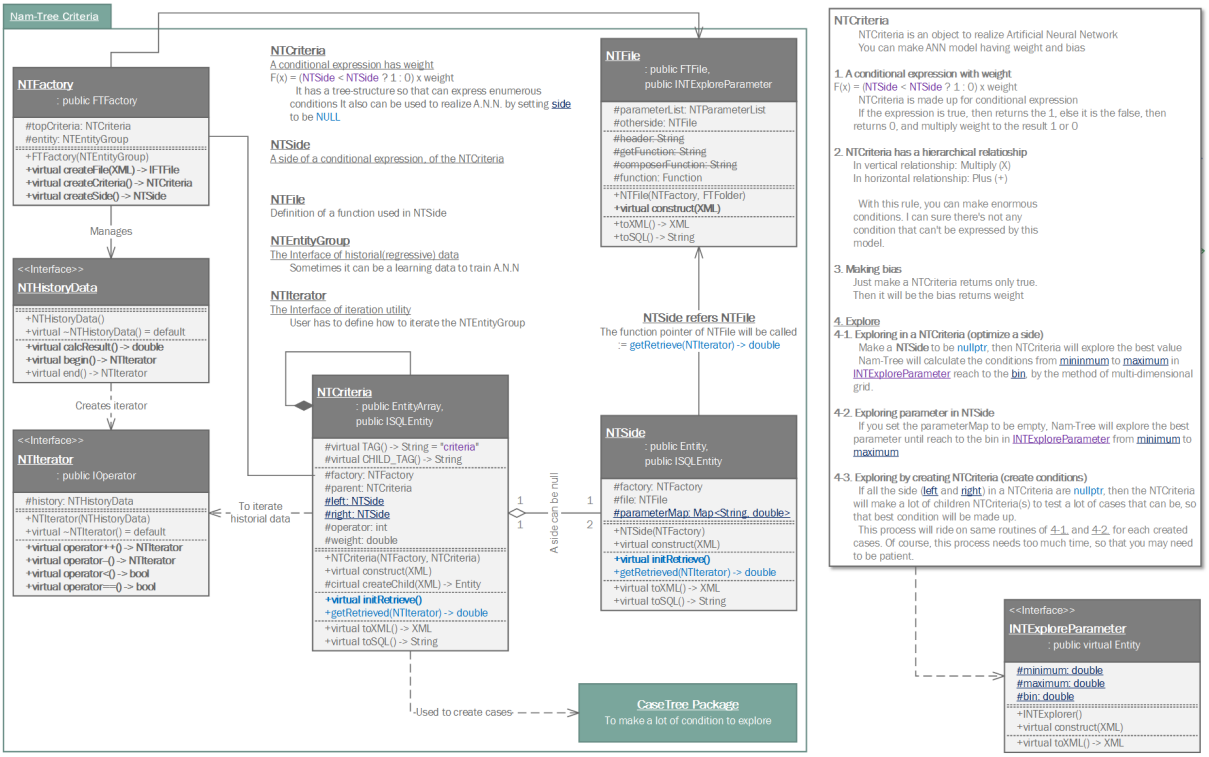
  A dictionary of function pointers.

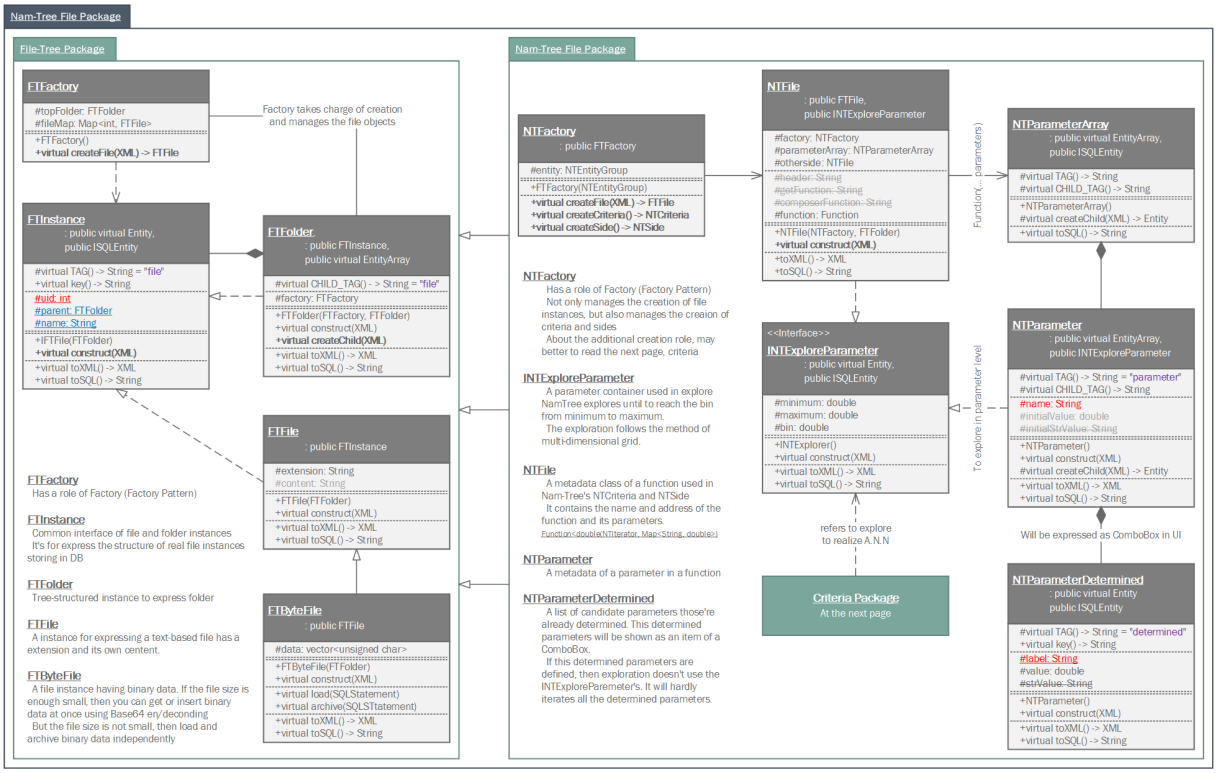
10.87.1 Detailed Description

A factory for Nam-Tree objects.

NTFactory class is a FTFactory and an abstract class creating and managing Nam-Tree objects. When overriding nam-tree module, do not override methods *createChild()* of file or criteria objects like NTFile, NT↩Criteria and NTSide. Only overrides method of creation in the NTFactory.

- Creator method of history data is not in NTFactory. It must be done by hand of user.





Note

Previous version of the Samchon Framework, NTFile class had a function script on the NTFile and Nam-Tree module compiled the script like Nam-Tree moudle of Flex standalone that is keeping the compiling method.

However, modern Samchon Framework's C++ NTFile class doesn't have a function script and does not compile. I don't know a way to compile a script in lots of operating systems, so I dropped the compiling method.

Until update, NTFactory has function pointers in a Dictionary and NTFile reference them. Users must put own methods into the Dictionary by their hands.

I will solve the problem sooon. Until next generation of Samchon Framework, put your own function pointers to a Dictionary in NTFactory by your hand please. Sorry for my unripe skill on programming.

See also

samchon::namtree

Author

Jeongho Nam

Definition at line 52 of file NTFactory.hpp.

10.87.2 Constructor & Destructor Documentation

10.87.2.1 NTFactory::NTFactory (NTEntityGroup * *data*)

Construct from historical(studying) data.

Parameters

entity	A group of history data.

Definition at line 13 of file NTFactory.cpp.

References data.

10.87.3 Member Function Documentation

10.87.3.1 auto NTFactory::createCriteria (NTCriteria * , std::shared_ptr< library::XML >) -> NTCriteria* [virtual]

Factory method of a NTCriteria.

Parameters

parent	A parent NTCriteria object xml An xml object for discriminating type.

Returns

A NTCrteria object representing a logical conditional.

Definition at line 26 of file NTFactory.cpp.

10.87.3.2 auto NTFactory::createSide (std::shared_ptr< library::XML >) -> NTSide* [virtual]

Factory method of a NTSide.

Parameters

xml	An xml oject for distriminating type.

Returns

A NTSide object which means a side of a logical condition.

Definition at line 30 of file NTFactory.cpp.

10.87.4 Member Data Documentation

10.87.4.1 NTEntityGroup* samchon::namtree::NTFactory::data [protected]

A entity gorup containing historical data.

Definition at line 64 of file NTFactory.hpp.

Referenced by NTFactory().

10.87.4.2 Dictionary<SideFunction> samchon::namtree::NTFactory::sideFunctionMap [protected]

A dictionary of function pointers.

Definition at line 69 of file NTFactory.hpp.

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

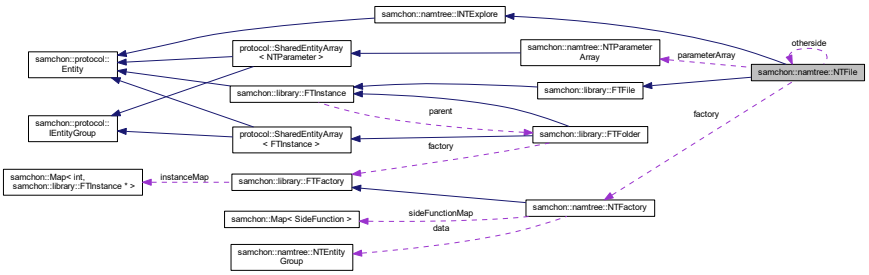
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTFactory.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTFactory.cpp

10.88 samchon::namtree::NTFile Class Reference

A file archiving metadata of a function.

`#include <NTFile.hpp>`

Collaboration diagram for samchon::namtree::NTFile:



Public Member Functions

- NTFile (NTFactory *, library::FTFolder *)

  Construct from factory and parent folder.
- virtual void construct (std::shared_ptr< library::XML >) override

  Construct data of the Entity from an XML object.
- auto getParameterArray () const -> NTParameterArray *

  Get metadata of parameters.
- auto getOtherside () const -> NTFile *

  Get otherside file.
- auto getFunction () const -> SideFunction

  Get function pointer.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the Entity.

Protected Attributes

- NTFactory * factory

  A factory and manager class for nam-tree.
- NTParameterArray * parameterArray

  Metadata of parameters.
- NTFile * otherside

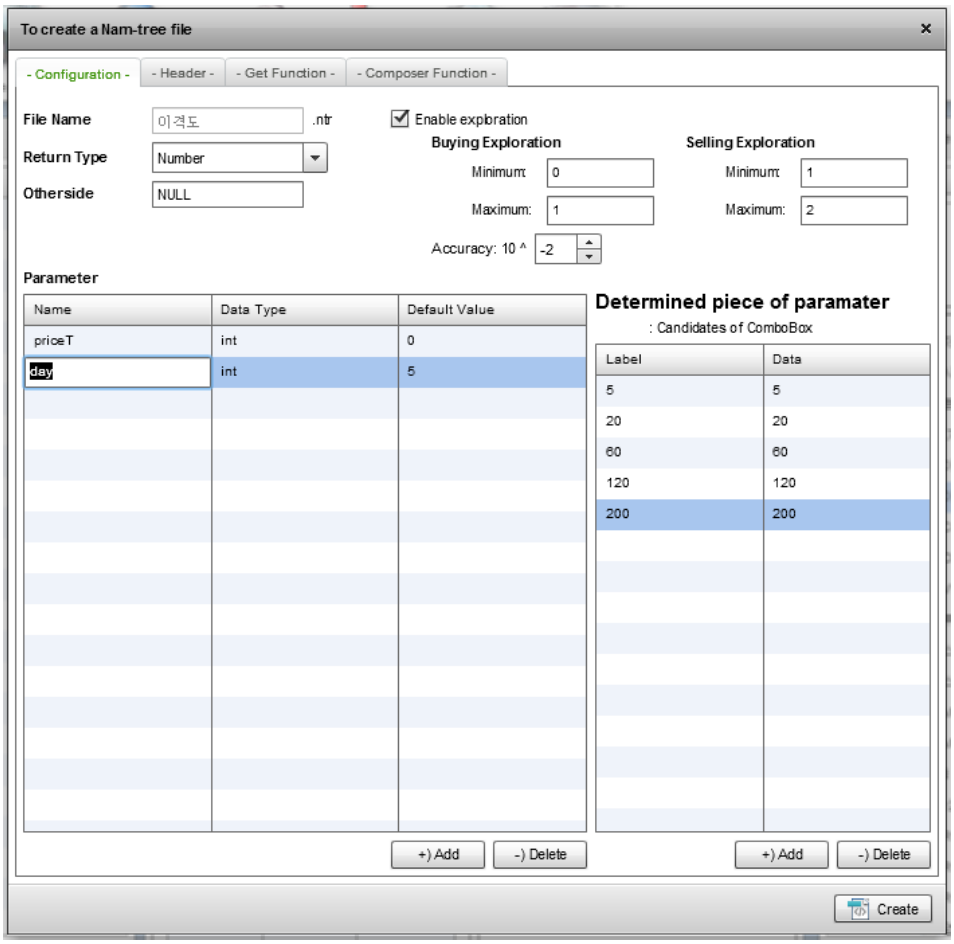
  A file of otherside.
- SideFunction function

  A function pointer.

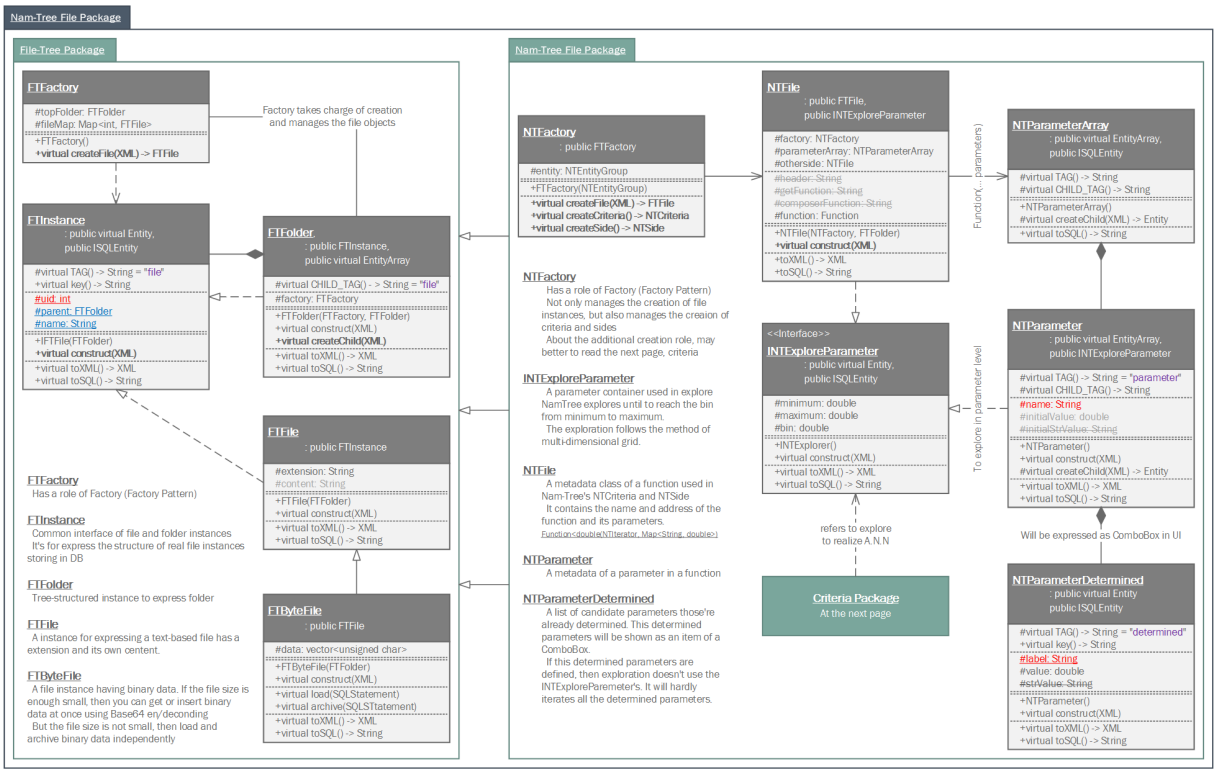
Additional Inherited Members

10.88.1 Detailed Description

A file archiving metadata of a function.



NTFile class is a FTFile archiving meta of a function which is used by a NTSide class. It may better to understand what NTFile is, if you consider the NTFile as a header of a function. Name, metadata of parameters and function pointer is archived on the NTFile.



Note

Previous version of the Samchon Framework, NTFile class had a function script on the NTFile and Nam-Tree module compiled the script like Nam-Tree moudle of Flex standalone that is keeping the compiling method.

However, modern Samchon Framework's C++ NTFile class doesn't have a function script and does not compile. I don't know a way to compile a script in lots of operating systems, so I dropped the compiling method. Until update, NTFactory has function pointers in a Dictionary and NTFile reference them. Users must put own methods into the Dictionary by their hands.

I will solve the problem sooon. Until next generation of Samchon Framework, put your own function pointers to a Dictionary in NTFactory by your hand please. Sorry for my unripe skill on programming.

See also

samchon::namtree
namtree::NTSide

Author

Jeongho Nam

Definition at line 50 of file NTFile.hpp.

10.88.2 Constructor & Destructor Documentation

10.88.2.1 NTFile::NTFile (NTFactory * *factory*, library::FTFolder * *parent*)

Construct from factory and parent folder.

Parameters

factory	A factory class creating nam-tree objects.
folder	A parent folder of the NTFile.

Definition at line 15 of file NTFile.cpp.

References factory, and otherside.

10.88.3 Member Function Documentation

10.88.3.1 void NTFile::construct (std::shared_ptr< library::XML >) [override],[virtual]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

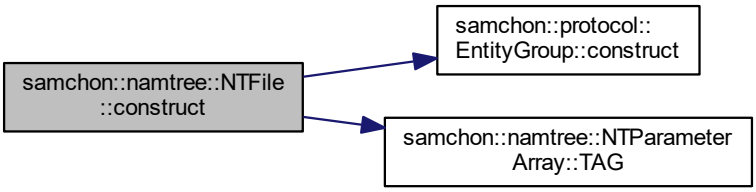
xml	An xml used to construct data of entity

Reimplemented from samchon::namtree::INTExplore.

Definition at line 23 of file NTFile.cpp.

References samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::construct(), parameterArray, and samchon::namtree::NTParameterArray::TAG().

Here is the call graph for this function:



10.88.3.2 auto NTFile::getParameterArray ( ) const -> NTParameterArray*

Get metadata of parameters.

Definition at line 30 of file NTFile.cpp.

References parameterArray.

10.88.3.3 auto NTFile::getOtherside ( ) const -> NTFile*

Get otherside file.

Definition at line 34 of file NTFile.cpp.

References otherside.

10.88.3.4 auto NTFile::getFunction ( ) const -> SideFunction

Get function pointer.

Definition at line 38 of file NTFile.cpp.

10.88.3.5 auto NTFile::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

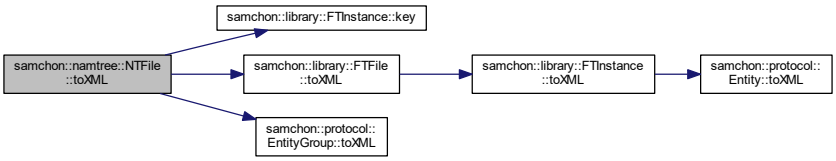
An XML object representing the Entity.

Reimplemented from samchon::library::FTFile.

Definition at line 43 of file NTFile.cpp.

References samchon::library::FTInstance::key(), otherside, parameterArray, samchon::library::FTFile::toXML(), and samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::toXML().

Here is the call graph for this function:



10.88.4 Member Data Documentation

10.88.4.1 NTFactory* samchon::namtree::NTFile::factory [protected]

A factory and manager class for nam-tree.

NTFile references NTFactory to find otherside file and function pointer.

Definition at line 65 of file NTFile.hpp.

Referenced by NTFile().

10.88.4.2 NTParameterArray* samchon::namtree::NTFile::parameterArray [protected]

Metadata of parameters.

Definition at line 70 of file NTFile.hpp.

Referenced by construct(), getParameterArray(), and toXML().

10.88.4.3 NTFile* samchon::namtree::NTFile::otherside [protected]

A file of otherside.

The variable otherside is a pair of file which is inserted to a criteria's opposite side.

When the otherside is specified and you determine the NTFile to be contained in a NTSide, the NTCriteria owning the NTSide will determine opposite side's NTSide to have the otherside NTFile forcibly.

Note

The determination will be automatic and unchangeble. Consider hardly when specifying an otherside.

Definition at line 86 of file NTFile.hpp.

Referenced by getOtherside(), NTFile(), and toXML().

10.88.4.4 SideFunction samchon::namtree::NTFile::function [protected]

A function pointer.

Definition at line 91 of file NTFile.hpp.

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

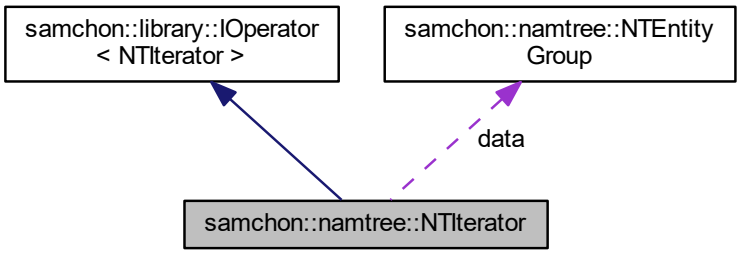
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTFile.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTFile.cpp

10.89 samchon::namtree::NTIterator Class Reference

Iterator of historical data.

`#include <NTIterator.hpp>`

Collaboration diagram for samchon::namtree::NTIterator:



Public Member Functions

- NTIterator (const NTEntityGroup *)
  Construt from historical data.
- virtual auto operator-- () const -> std::shared_ptr< NTIterator >=0
  Get iterator to previous element.
- virtual auto operator++ () const -> std::shared_ptr< NTIterator >=0
  Get iterator to next element.
- virtual auto operator== (const NTIterator &) const -> bool=0
  Whether an iterator is equal with the iterator.
- virtual auto operator< (const NTIterator &) const -> bool=0
  Whether the iterator is less than an iterator.

Protected Attributes

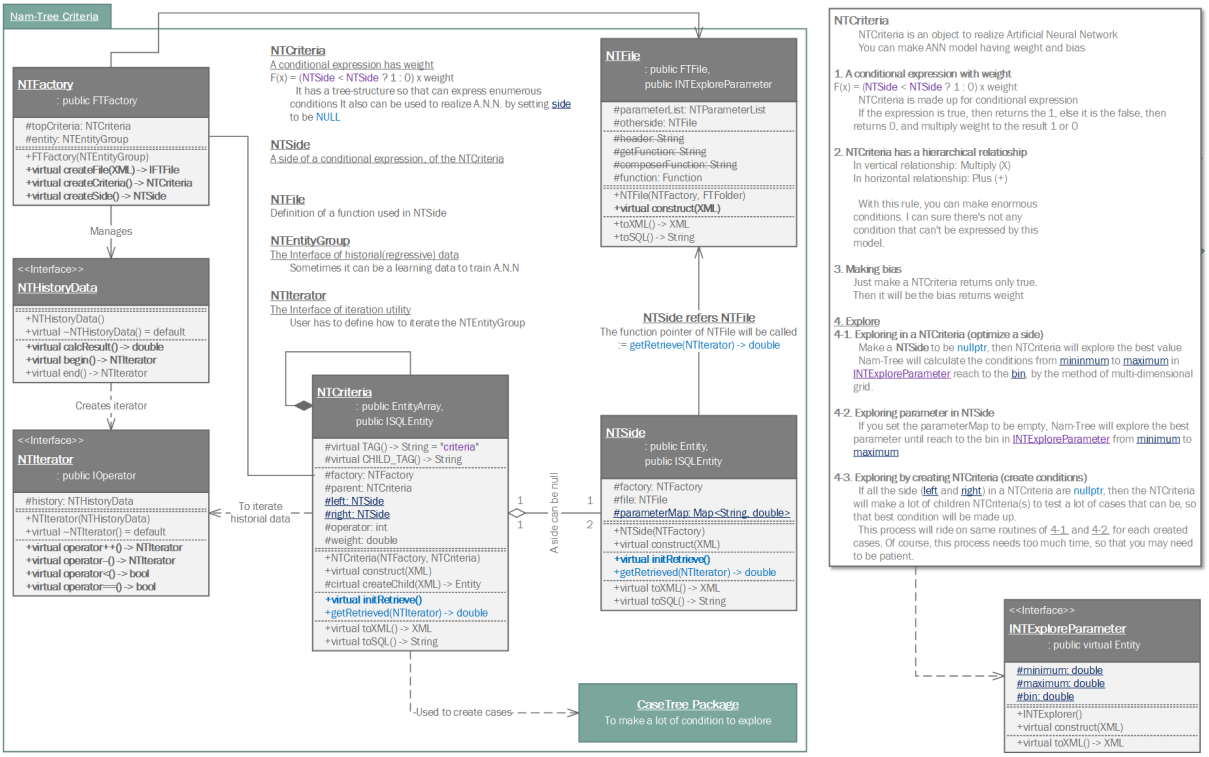
– const NTEntityGroup ∗ data

A historical data, source of the iterator.

10.89.1 Detailed Description

Iterator of historical data.

NTIterator is an abstract iterator of NTEntityGroup.



See also

samchon::namtree

Author

Jeongho Nam

Definition at line 26 of file NTIterator.hpp.

10.89.2 Constructor & Destructor Documentation

10.89.2.1 NTIterator::NTIterator (const NTEntityGroup ∗ *data*)

Construt from historical data.

Definition at line 6 of file NTIterator.cpp.

References data.

10.89.3 Member Function Documentation

10.89.3.1 virtual auto samchon::namtree::NTIterator::operator-- ( ) const -> std::shared_ptr< NTIterator > [pure virtual]

Get iterator to previous element.

If current iterator is the first item(equal with *begin()*), returns end().

Returns

An iterator of the previous item.

10.89.3.2 virtual auto samchon::namtree::NTIterator::operator++ ( ) const -> std::shared_ptr< NTIterator > [pure virtual]

Get iterator to next element.

If current iterator is the last item, returns end().

Returns

An iterator of the next item.

10.89.3.3 virtual auto samchon::namtree::NTIterator::operator== (const NTIterator &) const -> bool [pure virtual]

Whether an iterator is equal with the iterator.

Compare two iterators and returns whether they are equal or not.

Parameters

obj	An iterator to compare

Returns

Indicates whether equal or not.

10.89.3.4 virtual auto samchon::namtree::NTIterator::operator< (const NTIterator &) const -> bool [pure virtual]

Whether the iterator is less than an iterator.

Compare two iterators and returns whether this iterator is less.

Parameters

obj	An iterator to compare

Returns

Indicates whether less or not.

10.89.4 Member Data Documentation

10.89.4.1 const NTEntityGroup* samchon::namtree::NTIterator::data [protected]

A historical data, source of the iterator.

Definition at line 36 of file NTIterator.hpp.

Referenced by NTIterator().

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

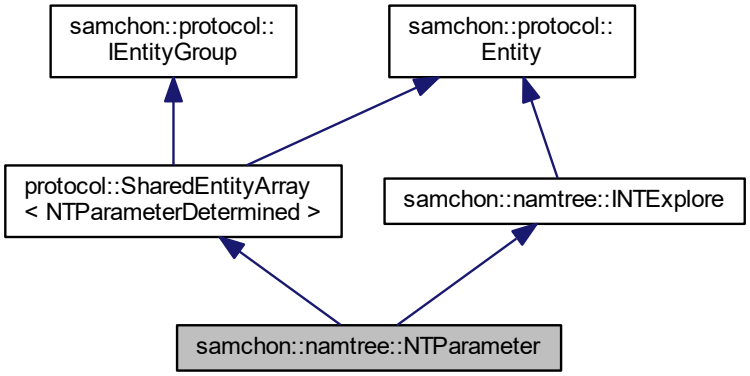
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTIterator.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTIterator.cpp

10.90 samchon::namtree::NTParameter Class Reference

A metadata of a parameter in a function.

`#include <NTParameter.hpp>`

Collaboration diagram for samchon::namtree::NTParameter:



Public Member Functions

- NTParameter ()
  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override
  Construct data of the Entity from an XML object.
- virtual auto key () const -> std::string override
  Get a key that can identify the Entity uniquely.
- auto getName () const -> std::string
  Get name.
- auto getInitialValue () const -> double
  Get initial, default vlaue.
- virtual auto TAG () const -> std::string override
  A tag name when represented by XML.
- virtual auto CHILD_TAG () const -> std::string override
  A tag name of children.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override
  Get an XML object represents the Entity.

Protected Member Functions

- virtual auto createChild (std::shared_ptr< library::XML >) -> NTParameterDetermined *override
  Factory method of a child Entity.

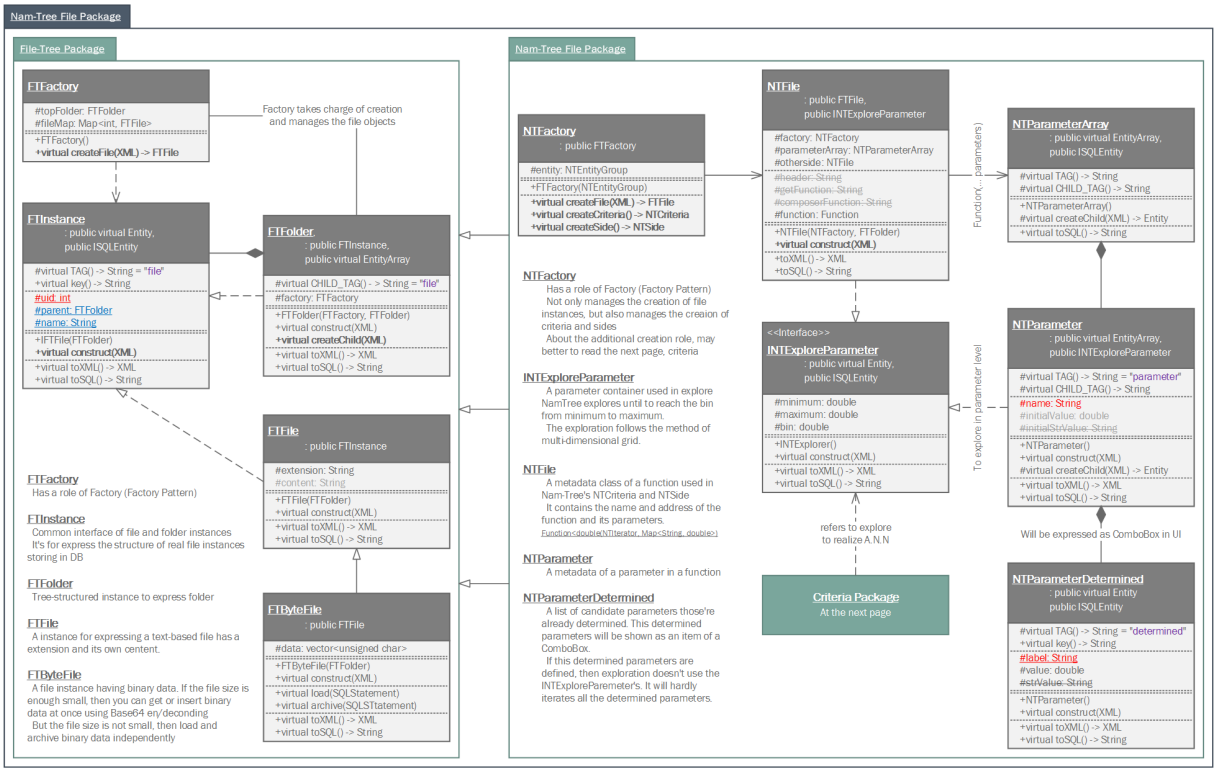
Protected Attributes

- std::string name
  Name of a paremeter.
- double initialValue
  An initial, default value.

10.90.1 Detailed Description

A metadata of a parameter in a function.

NTParameter is a class containing metadata of a parameter which is belonged to a class, metadata of a function, NTFile. Between the NTFile and NTParameter(s), there's a associate relationship.



See also

samchon::namtree

Author

Jeongho Nam

Definition at line 25 of file NTParameter.hpp.

10.90.2 Constructor & Destructor Documentation

10.90.2.1 NTParameter::NTParameter ( )

Default Constructor.

Definition at line 16 of file NTParameter.cpp.

References initialValue.

10.90.3 Member Function Documentation

10.90.3.1 void NTParameter::construct (std::shared_ptr< library::XML >) [override], [virtual]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

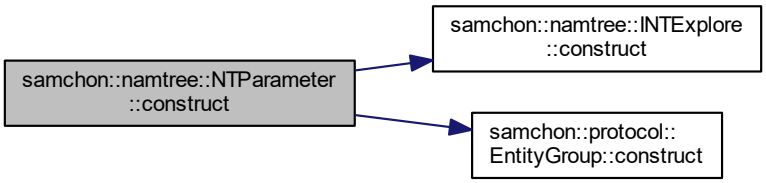
xml	An xml used to construct data of entity

Reimplemented from samchon::namtree::INTExplore.

Definition at line 21 of file NTParameter.cpp.

References samchon::namtree::INTExplore::construct(), samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::construct(), initialValue, and name.

Here is the call graph for this function:



10.90.3.2 auto NTParameter::createChild (std::shared_ptr< library::XML >) -> NTParameterDetermined *override [override], [protected], [virtual]

Factory method of a child Entity.

EntityGroup::createChild() is a factory method creating a new child Entity which is belonged to the Entity↩ Group. This method is called by EntityGroup::construct(). The children construction methods Entity↩ ::construct() will be called by abstract method of the EntityGroup::construct().

Returns

A new child Entity belongs to EntityGroup.

Implements samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

Definition at line 32 of file NTParameter.cpp.

10.90.3.3 auto NTParameter::key ( ) const -> std::string [override], [virtual]

Get a key that can identify the Entity uniquely.

If identifier of the Entity is not atomic value, returns a string represents the composite identifier. If identifier of the Entity is not string, converts the identifier to string and returns the string.

Returns

An identifier

Reimplemented from samchon::protocol::Entity.

Definition at line 37 of file NTParameter.cpp.

References name.

10.90.3.4 auto NTParameter::getName ( ) const -> std::string

Get name.

Definition at line 41 of file NTParameter.cpp.

References name.

10.90.3.5 auto NTParameter::getInitialValue ( ) const -> double

Get initial, default vlaue.

Definition at line 45 of file NTParameter.cpp.

References initialValue.

10.90.3.6 auto NTParameter::TAG ( ) const -> std::string [override], [virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Definition at line 13 of file NTParameter.cpp.

10.90.3.7 auto NTParameter::CHILD_TAG ( ) const -> std::string [override], [virtual]

A tag name of children.

< TAG>
<CHILD_TAG />
<CHILD_TAG />
</TAG>

Implements samchon::protocol::IEntityGroup.

Definition at line 14 of file NTParameter.cpp.

10.90.3.8 auto NTParameter::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

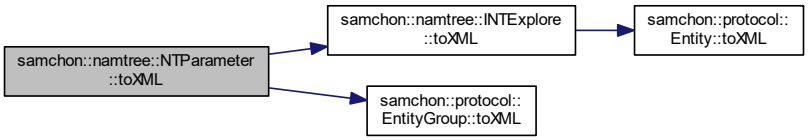
An XML object representing the Entity.

Reimplemented from samchon::namtree::INTExplore.

Definition at line 50 of file NTParameter.cpp.

References initialValue, name, samchon::namtree::INTExplore::toXML(), and samchon::protocol::Entity↩Group< _Container, _ETy, _Ty >::toXML().

Here is the call graph for this function:



10.90.4 Member Data Documentation

10.90.4.1 std::string samchon::namtree::NTParameter::name [protected]

Name of a paremeter.

The string label is identifier of a NTParameter. It must be unique.

Definition at line 37 of file NTParameter.hpp.

Referenced by construct(), getName(), key(), and toXML().

10.90.4.2 double samchon::namtree::NTParameter::initialValue [protected]

An initial, default value.

Definition at line 42 of file NTParameter.hpp.

Referenced by construct(), getInitialValue(), NTParameter(), and toXML().

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

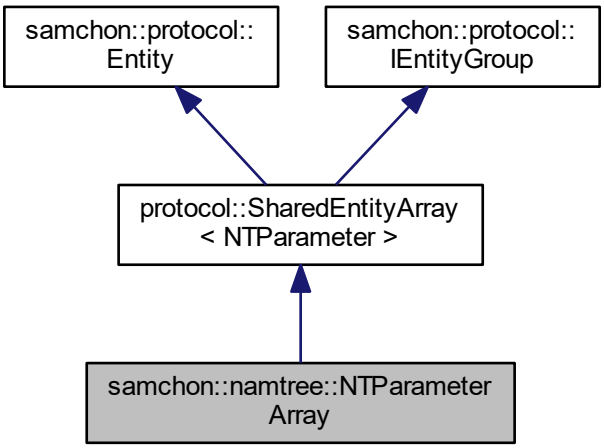
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTParameter.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTParameter.cpp

10.91 samchon::namtree::NTParameterArray Class Reference

An Array of NTParameter.

```
#include <NTParameterArray.hpp>
```

Collaboration diagram for samchon::namtree::NTParameterArray:



Public Member Functions

- virtual auto TAG () const -> std::string override

  A tag name when represented by XML.
- virtual auto CHILD_TAG () const -> std::string override

  A tag name of children.
- NTParameterArray ()

  Default Constructor.

Protected Member Functions

- virtual auto createChild (std::shared_ptr< library::XML >) -> NTParameter *override

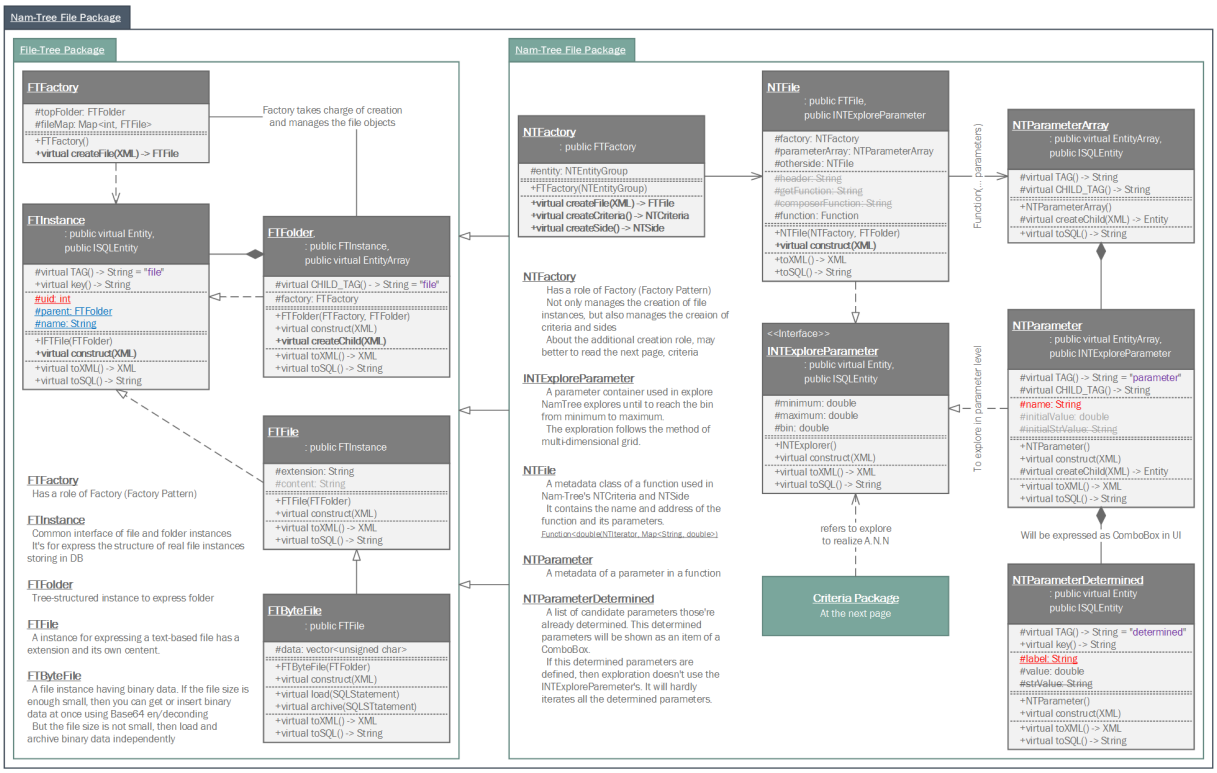
  Factory method of a child Entity.

10.91.1 Detailed Description

An Array of NTParameter.

NTParameterArray is an array class containing NTParameter objects.

The NTParameter is belonged to a NTFile class to realize associate relationship between NTFile and NT↩ Parameter(s).



See also

samchon::namtree

Author

Jeongho Nam

Definition at line 25 of file NTParameterArray.hpp.

10.91.2 Constructor & Destructor Documentation

10.91.2.1 NTParameterArray::NTParameterArray ( )

Default Constructor.

Definition at line 14 of file NTParameterArray.cpp.

10.91.3 Member Function Documentation

10.91.3.1 auto NTParameterArray::TAG ( ) const -> std::string [override], [virtual]

A tag name when represented by XML.

Returns

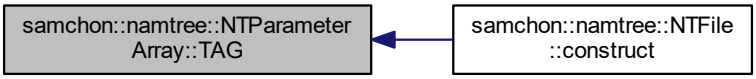
A tag name

Implements samchon::protocol::Entity.

Definition at line 11 of file NTParameterArray.cpp.

Referenced by samchon::namtree::NTFile::construct().

Here is the caller graph for this function:



10.91.3.2 auto NTParameterArray::CHILD_TAG ( ) const -> std::string [override], [virtual]

A tag name of children.

< TAG>
<CHILD_TAG />
<CHILD_TAG />
</TAG>

Implements samchon::protocol::IEntityGroup.

Definition at line 12 of file NTParameterArray.cpp.

10.91.3.3 auto NTParameterArray::createChild (std::shared_ptr< library::XML >) -> NTParameter *override [override], [protected], [virtual]

Factory method of a child Entity.

EntityGroup::createChild() is a factory method creating a new child Entity which is belonged to the Entity↩ Group. This method is called by EntityGroup::construct(). The children construction methods Entity↩ ::construct() will be called by abstract method of the EntityGroup::construct().

Returns

A new child Entity belongs to EntityGroup.

Implements samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

Definition at line 18 of file NTParameterArray.cpp.

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

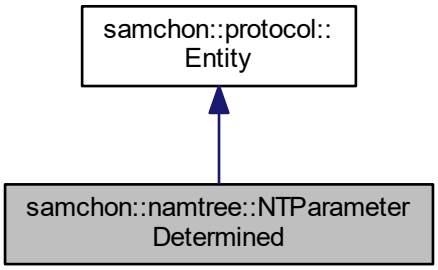
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTParameterArray.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTParameterArray.cpp

10.92 samchon::namtree::NTParameterDetermined Class Reference

A pre-determined value of a parameter.

```
#include <NTParameterDetermined.hpp>
```

Collaboration diagram for samchon::namtree::NTParameterDetermined:



Public Member Functions

- NTParameterDetermined ()

  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override

  Construct data of the Entity from an XML object.
- virtual auto key () const -> std::string override

  Get a key that can identify the Entity uniquely.
- auto getLabel () const -> std::string

  Get label.
- auto getValue () const -> double

  Get value.
- virtual auto TAG () const -> std::string override

  A tag name when represented by XML.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the Entity.

Protected Attributes

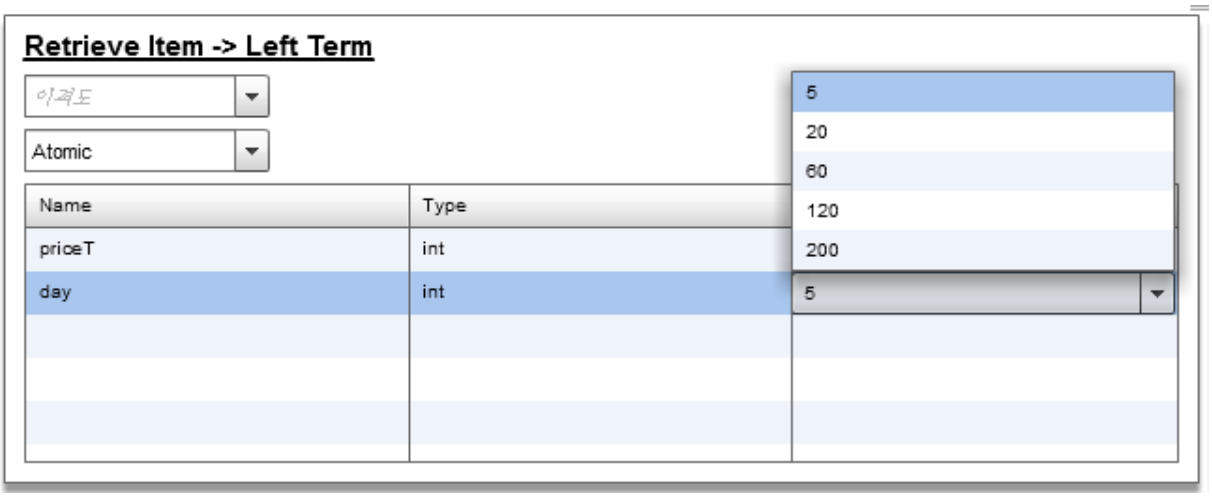
- std::string label

  A label will be printed in a combo-box.
- double value

  A value, pre-defined.

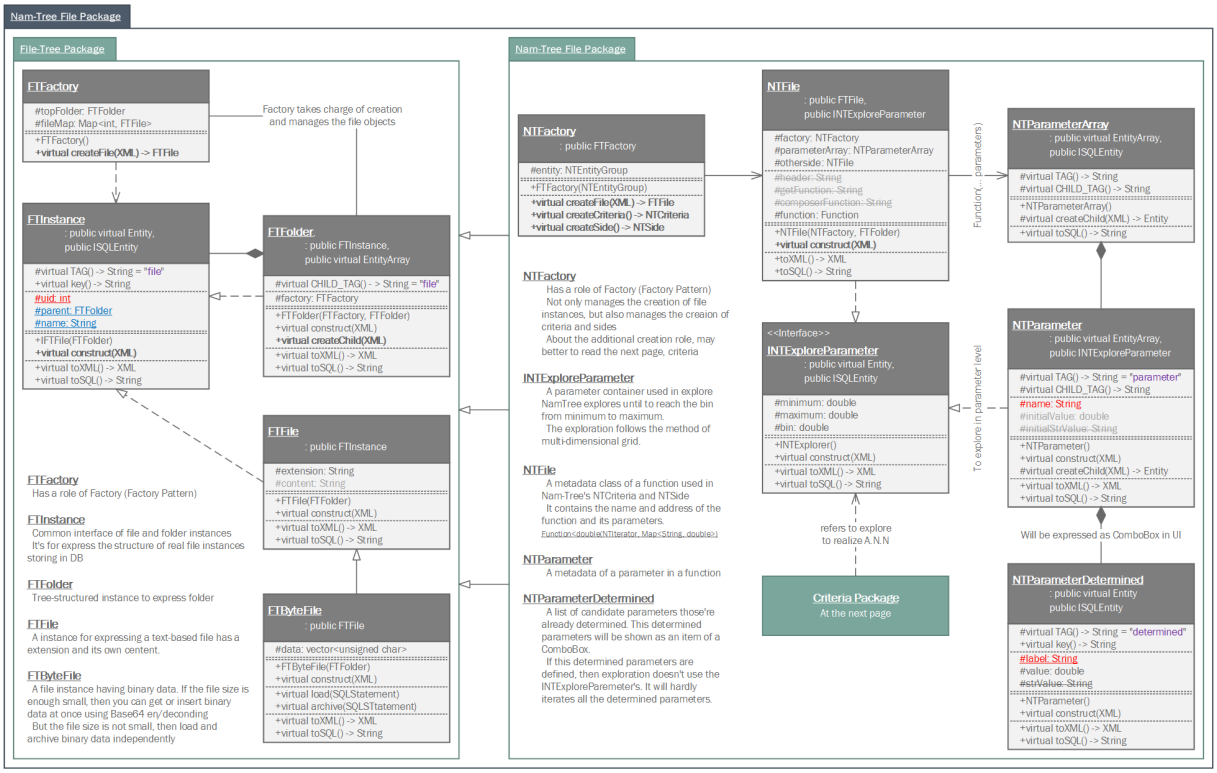
10.92.1 Detailed Description

A pre-determined value of a parameter.



NTParameterDetermined is a class belonged to a NTParameter object presents a pre-defined value. In UI module of Nam-Tree, the NTParameterDetermined is expressed by an item of a ComboBox.

When optimization of Nam-Tree, in a part of NTParameter, if the pre-determined value is defined optimizer does not explore with grid method from referencing INTExplore. Just selects a value between the pre-defineds. (One of NTParameterDetermined is selected in NTParameter).



See also

samchon::namtree

Author

Jeongho Nam

Definition at line 30 of file NTParameterDetermined.hpp.

10.92.2 Constructor & Destructor Documentation

10.92.2.1 NTParameterDetermined::NTParameterDetermined ( )

Default Constructor.

Definition at line 14 of file NTParameterDetermined.cpp.

10.92.3 Member Function Documentation

10.92.3.1 void NTParameterDetermined::construct (std::shared_ptr< library::XML >) [override], [virtual]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Implements samchon::protocol::Entity.

Definition at line 18 of file NTParameterDetermined.cpp.

References label, and value.

10.92.3.2 auto NTParameterDetermined::key ( ) const -> std::string [override], [virtual]

Get a key that can identify the Entity uniquely.

If identifier of the Entity is not atomic value, returns a string represents the composite identifier. If identifier of the Entity is not string, converts the identifier to string and returns the string.

Returns

An identifier

Reimplemented from samchon::protocol::Entity.

Definition at line 25 of file NTParameterDetermined.cpp.

References label.

10.92.3.3 auto NTParameterDetermined::getLabel ( ) const -> std::string

Get label.

Definition at line 29 of file NTParameterDetermined.cpp.

References label.

10.92.3.4 auto NTParameterDetermined::getValue ( ) const -> double

Get value.

Definition at line 33 of file NTParameterDetermined.cpp.

References value.

10.92.3.5 auto NTParameterDetermined::TAG ( ) const -> std::string [override], [virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Definition at line 12 of file NTParameterDetermined.cpp.

10.92.3.6 auto NTParameterDetermined::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

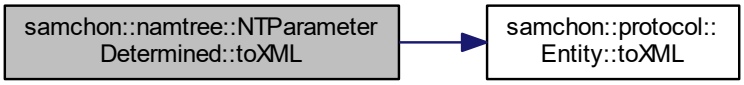
An XML object representing the Entity.

Reimplemented from samchon::protocol::Entity.

Definition at line 38 of file NTParameterDetermined.cpp.

References label, samchon::protocol::Entity::toXML(), and value.

Here is the call graph for this function:



10.92.4 Member Data Documentation

10.92.4.1 std::string samchon::namtree::NTParameterDetermined::label [protected]

A label will be printed in a combo-box.

The string label is identifier of a NTParameterDetermined. It must be unique.

Definition at line 41 of file NTParameterDetermined.hpp.

Referenced by construct(), getLabel(), key(), and toXML().

10.92.4.2 double samchon::namtree::NTParameterDetermined::value [protected]

A value, pre-defined.

Definition at line 46 of file NTParameterDetermined.hpp.

Referenced by construct(), getValue(), and toXML().

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

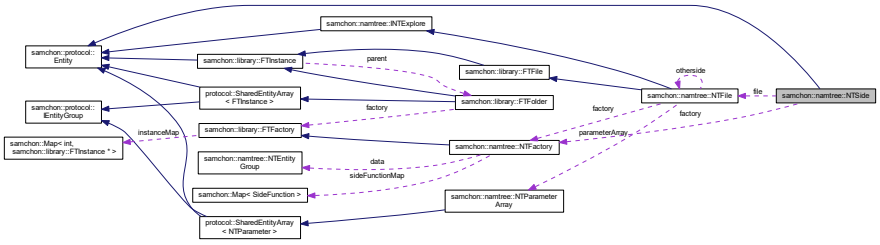
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTParameterDetermined.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTParameterDetermined.cpp

10.93 samchon::namtree::NTSide Class Reference

A side of a conditional expresson.

```
#include <NTSide.hpp>
```

Collaboration diagram for samchon::namtree::NTSide:



Public Member Functions

- virtual void construct (std::shared_ptr< library::XML >) override

  Construct data of the Entity from an XML object.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the Entity.

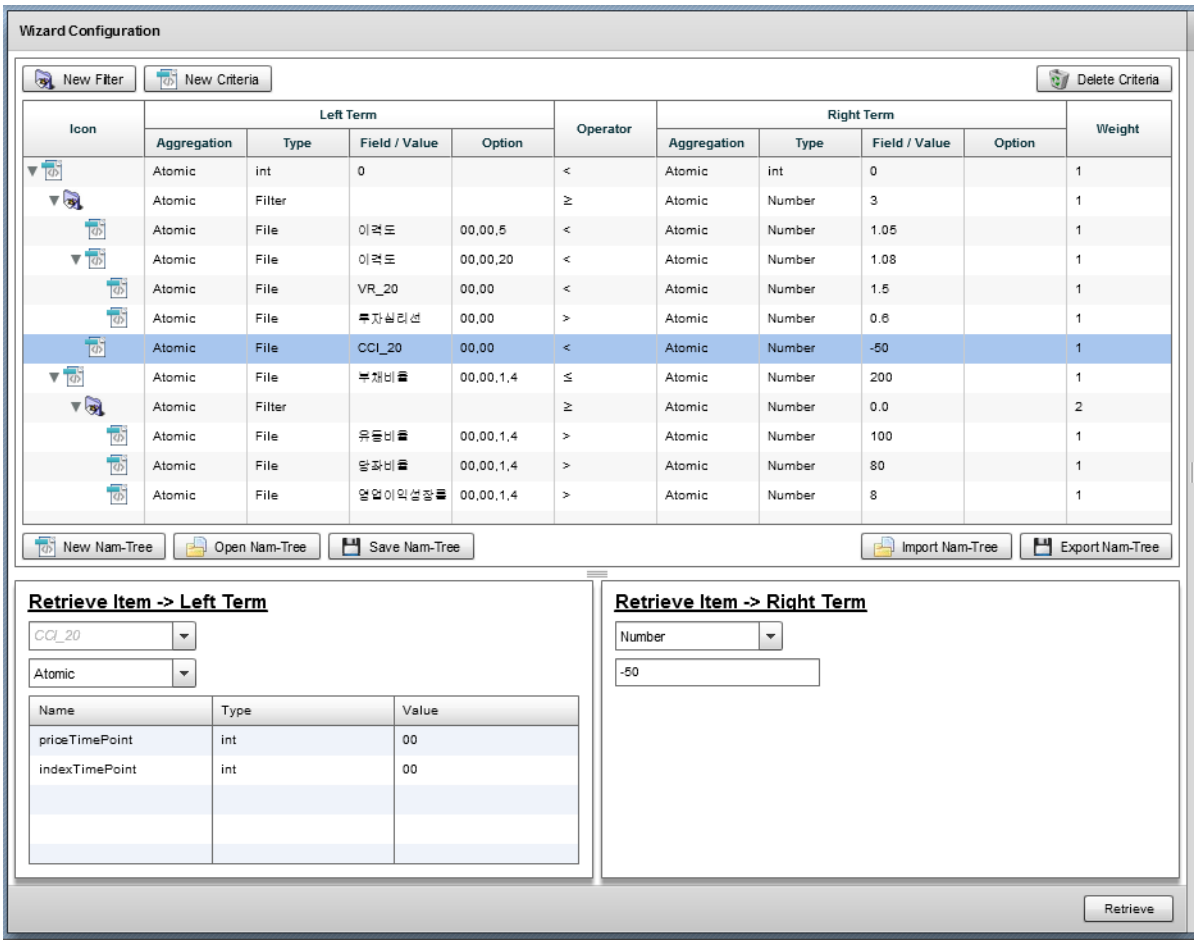
Protected Member Functions

- virtual auto TAG () const -> std::string override

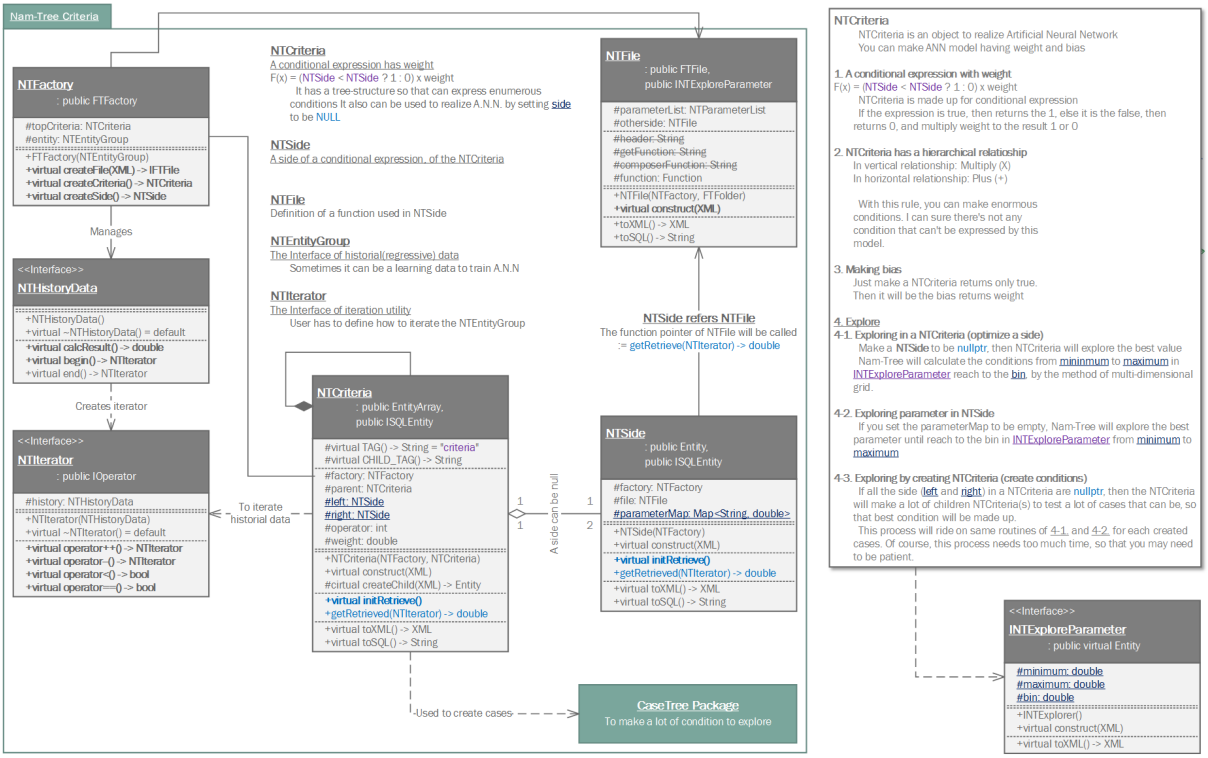
  A tag name when represented by XML.

10.93.1 Detailed Description

A side of a conditional expresson.



Two NTSide classes are belonged to a NTCriteria as left and right side. The NTSide class has a NTFile representing a function and parameters to adjust to the function.



See also

samchon::namtree

Author

Jeongho Nam

Definition at line 32 of file NTSide.hpp.

10.93.2 Member Function Documentation

10.93.2.1 auto NTSide::TAG ( ) const -> std::string [override], [protected], [virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Definition at line 17 of file NTSide.cpp.

10.93.2.2 void NTSide::construct (std::shared_ptr< library::XML >) [override], [virtual]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

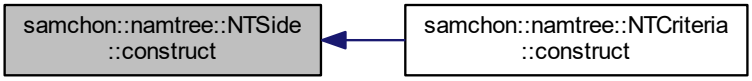
xml	An xml used to construct data of entity

Implements samchon::protocol::Entity.

Definition at line 25 of file NTSide.cpp.

Referenced by samchon::namtree::NTCriteria::construct().

Here is the caller graph for this function:



10.93.2.3 auto NTSide::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

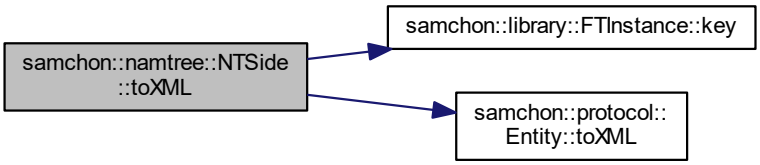
Reimplemented from samchon::protocol::Entity.

Definition at line 56 of file NTSide.cpp.

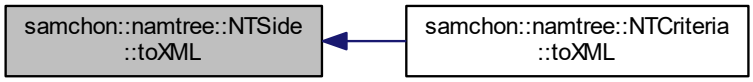
References samchon::library::FTInstance::key(), and samchon::protocol::Entity::toXML().

Referenced by samchon::namtree::NTCriteria::toXML().

Here is the call graph for this function:



Here is the caller graph for this function:



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

- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTSide.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/namtree/NTSide.cpp

10.94 samchon::example::packer::Packer Class Reference

A packer planning the best packaging.

```
#include <Packer.hpp>
```

Public Member Functions

- Packer (std::vector< Product > *, std::vector< Wrapper > *)

  Construct from products and wrapper.
- Packer (const Packer &)

  Copy Constructor.
- void optimize ()

  Find the best packaging method.
- auto calcPrice () const -> int

  Calculate price of the wrappers.
- auto toString () const -> std::string

  Return a string represents an packaging method.

Private Attributes

- std::vector< Product > * productArray

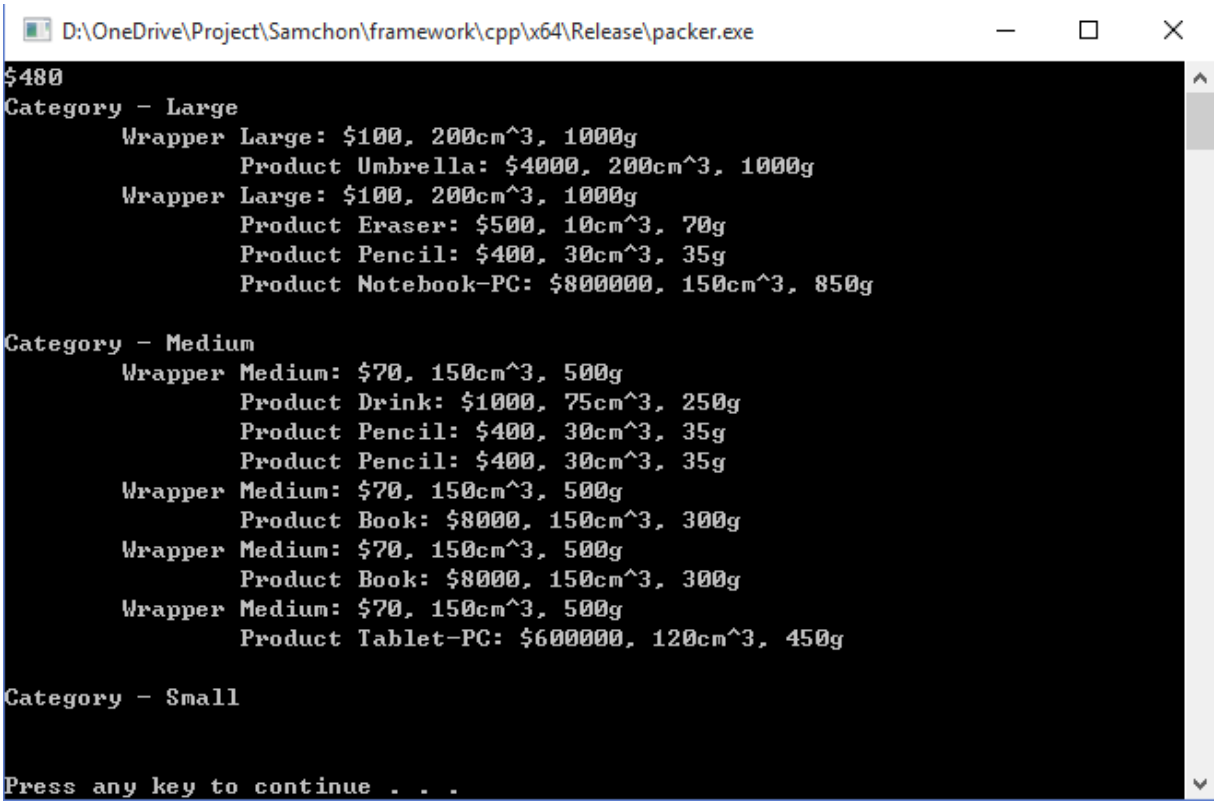
  Prodcut(s) to package in some Wrapper(s)
- std::vector< Wrapper > * wrapperArray

  Type of Wrapper(s) to be used for packaging.

10.94.1 Detailed Description

A packer planning the best packaging.

Retrieves the solution of packaging by combination permuation and factorial case.



Warning

Be careful about number of products and wrappers.

The time complexity of Packer overs O(m^n). Elapsed time of calculation increases enourmously. Do not use Packer if the digits of number of products or wrappers overs 2.

Author

Jeongho Nam

Definition at line 32 of file Packer.hpp.

10.94.2 Constructor & Destructor Documentation

10.94.2.1 Packer::Packer (std::vector< Product > * *productArray*, std::vector< Wrapper > * *wrapperArray*)

Construct from products and wrapper.

Parameters

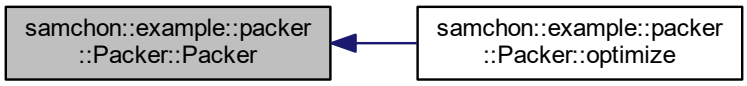
productArray	Product(s) to input some Wrapper
wrapperArray	Type of Wrapper(s) to be used

Definition at line 13 of file Packer.cpp.

References productArray, and wrapperArray.

Referenced by optimize().

Here is the caller graph for this function:



10.94.2.2 Packer::Packer (const Packer & *packer*)

Copy Constructor.

Copy constructor of Packer does not copy children items.

Only copies member variables(productArray, wrapperArray) of Packer's.

Definition at line 22 of file Packer.cpp.

10.94.3 Member Function Documentation

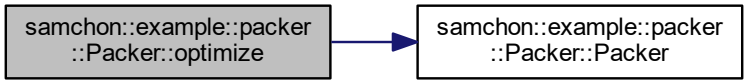
10.94.3.1 void Packer::optimize ( )

Find the best packaging method.

Definition at line 30 of file Packer.cpp.

References Packer(), productArray, and wrapperArray.

Here is the call graph for this function:



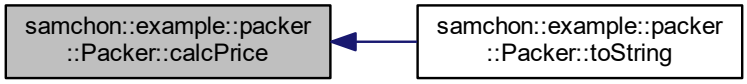
10.94.3.2 auto Packer::calcPrice ( ) const -> int

Calculate price of the wrappers.

Definition at line 73 of file Packer.cpp.

Referenced by toString().

Here is the caller graph for this function:



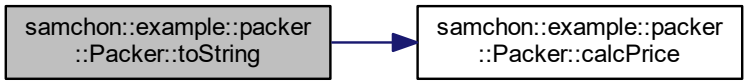
10.94.3.3 auto Packer::toString ( ) const -> std::string

Return a string represents an packaging method.

Definition at line 85 of file Packer.cpp.

References calcPrice().

Here is the call graph for this function:



10.94.4 Member Data Documentation

10.94.4.1 std::vector<Product>* samchon::example::packer::Packer::productArray [private]

Prodcut(s) to package in some Wrapper(s)

Definition at line 41 of file Packer.hpp.

Referenced by optimize(), and Packer().

10.94.4.2 std::vector<Wrapper>* samchon::example::packer::Packer::wrapperArray [private]

Type of Wrapper(s) to be used for packaging.

Definition at line 46 of file Packer.hpp.

Referenced by optimize(), and Packer().

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

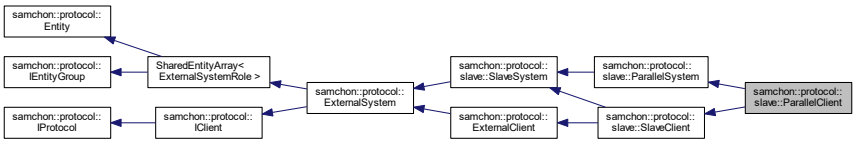
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/packer/Packer.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/packer/Packer.cpp

10.95 samchon::protocol::slave::ParallelClient Class Reference

A slave parallel client.

`#include <ParallelClient.hpp>`

Collaboration diagram for samchon::protocol::slave::ParallelClient:



Public Member Functions

- ParallelClient ()

  Default Constructor.

Additional Inherited Members

10.95.1 Detailed Description

A slave parallel client.

ParallelClient is a ParallelSystem class specialized in server driver.

[Inherited]

ParallelSystem is a class processing segmented Invoke messages from master.

[Inherited]

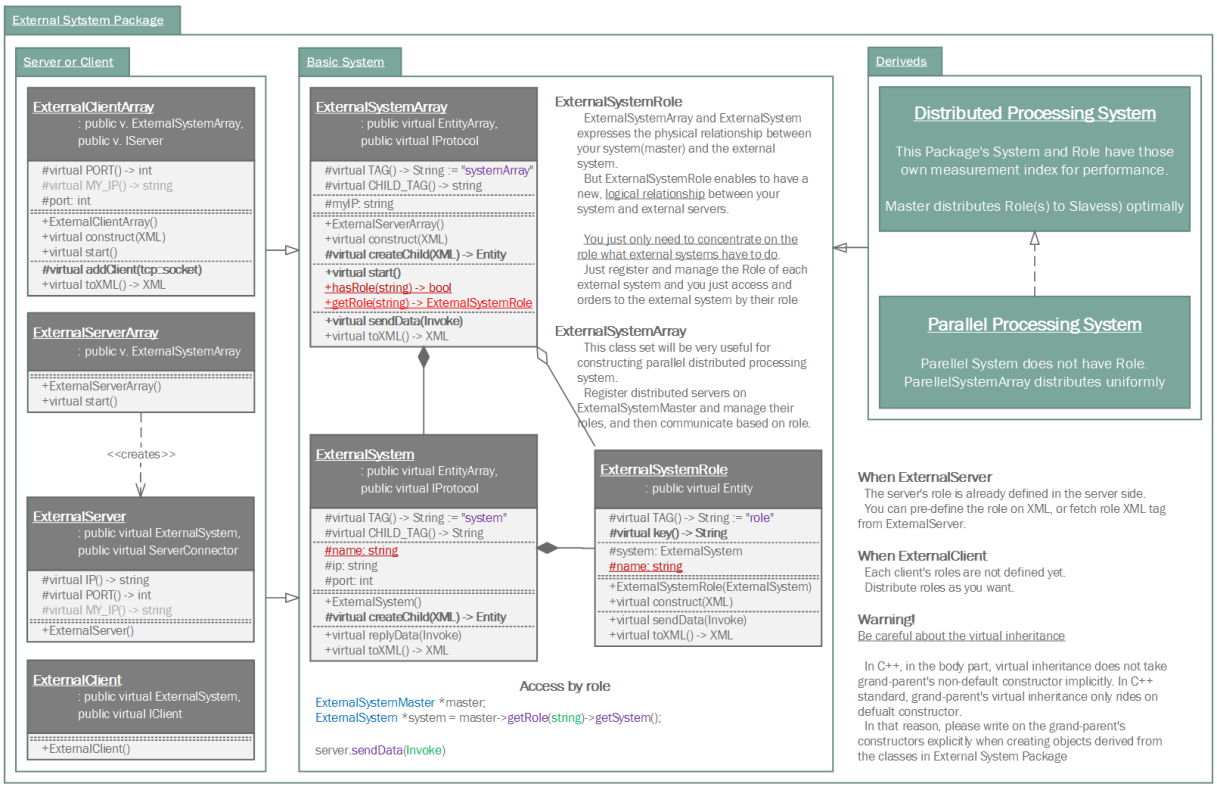
SlaveSystem, literally, means a slave system belongs to a maste system.

The SlaveSystem class is used in opposite side system of master::DistributedSystem and master::Parallel↩System and reports elapsed time of each commmand (by Invoke message) for estimation of its performance.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, ExternalSystem is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystem↩ Role.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an External↩ SystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an External↩ System. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 22 of file ParallelClient.hpp.

10.95.2 Constructor & Destructor Documentation

10.95.2.1 ParallelClient::ParallelClient ( )

Default Constructor.

Definition at line 8 of file ParallelClient_slave.cpp.

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

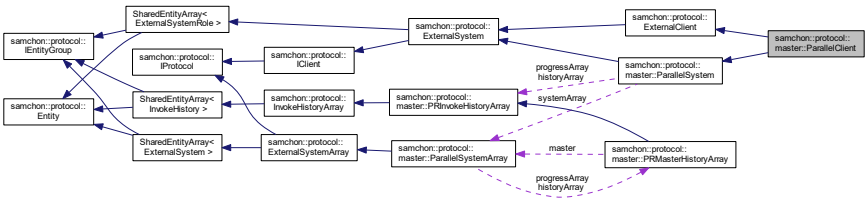
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/slave/ParallelClient.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/slave/ParallelClient_slave.cpp

10.96 samchon::protocol::master::ParallelClient Class Reference

A network driver for a parallel client.

```
#include <ParallelClient.hpp>
```

Collaboration diagram for samchon::protocol::master::ParallelClient:



Public Member Functions

- ParallelClient ()
  Default Constructor.

Additional Inherited Members

10.96.1 Detailed Description

A network driver for a parallel client.

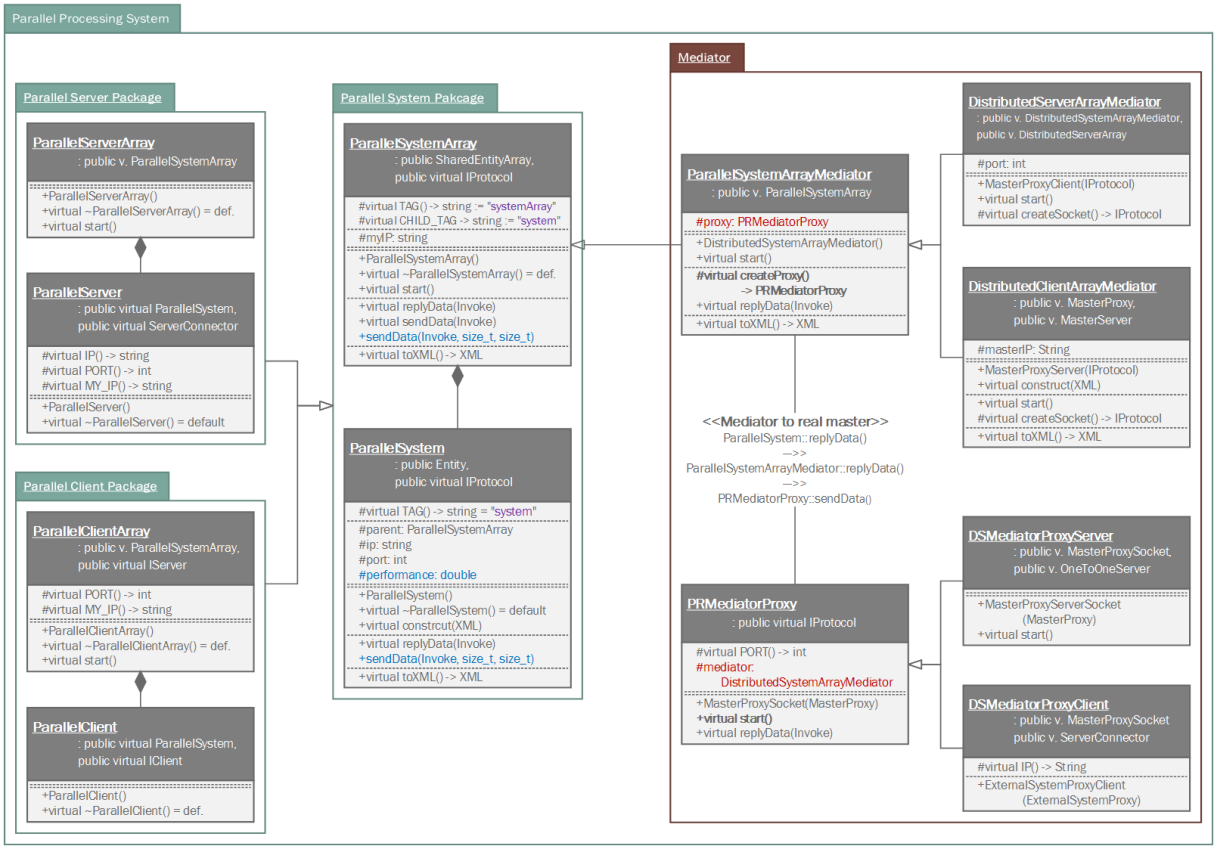
ParallelClient is an ParallelSystem specialized in client driver.

[Inherited]

ParallelSystem class is an ExternalSystem having performance index and history logs of handled Invoke messages with those elapsed times. Those performance index and history logs are used to determine how much segmentation to be allocated.

Each segmentation in requested process (Invoke message) is equivalent. Thus, role of ParallelSystem objects in a ParallelSystemArray are almost same and does not need to specify ExternalSystemRole on ParallelSystem.

- Not a matter to specifying ExternalSystemRole objects to each ParallelSystem. In that case, Invoke messages having segmentation size will be processed by ParallelSystem's own logic and Invoke messages without segmentation size will be handled by ordinary logic of ExternalSystem's own.



[Inherited]

Definition at line 22 of file ParallelClient.hpp.

10.96.2 Constructor & Destructor Documentation

10.96.2.1 samchon::protocol::master::ParallelClient::ParallelClient ( )

Default Constructor.

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

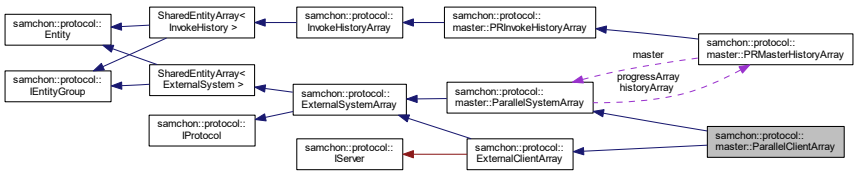
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelClient.hpp

10.97 samchon::protocol::master::ParallelClientArray Class Reference

An array of ParallelClient(s).

```
#include <ParallelClientArray.hpp>
```

Collaboration diagram for samchon::protocol::master::ParallelClientArray:



Public Member Functions

– ParallelClientArray ()

Default Constructor.

Additional Inherited Members

10.97.1 Detailed Description

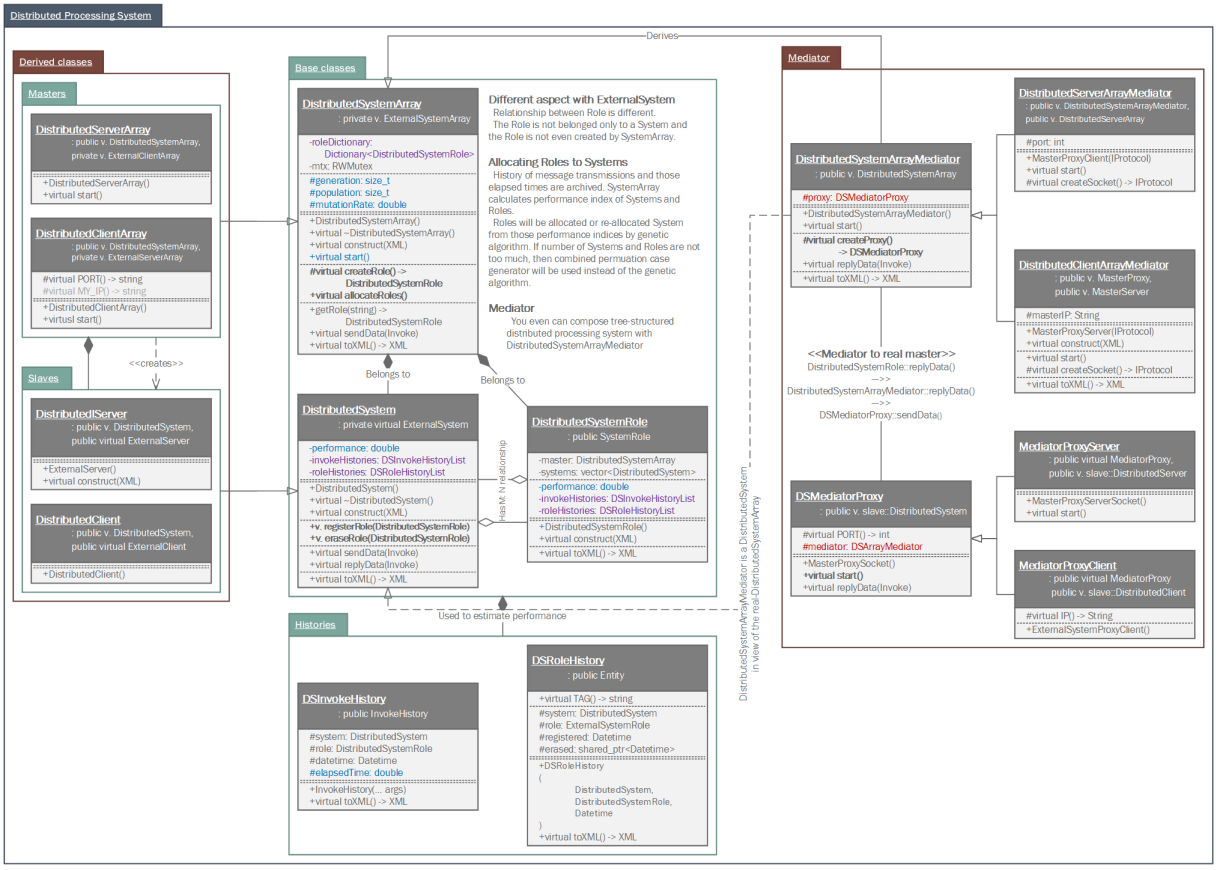
An array of ParallelClient(s).

ParallelClientArray is an ParallelSystemArray specialized in clients' driver.

[Inherited]

DistributedSystemArray class is an ExternalSystemArray and an abstract class containing and managing distributed system drivers within framework of master. The DistributedSystemArray class allocates DistributedSystemRole objects to DistributedSystem objects for optimization.

Unlike the ExternalSystemRole class which is dependent on an ExternalSystem, DistributedSystemRole is not dependent on a DistributedSystem and even can be allocated to multiple DistributedSystem objects. Allocation of DistributedSystemRole(s) to DistributedSystem(s) is determined by estimation of performance index pursuing best distribution. If number of DistributedSystem and DistributedSystemRole objects is not a lot, optimize by combined permuation algorithm. Else the objects are a lot, the allocation will be optimized by genetic algorithm.



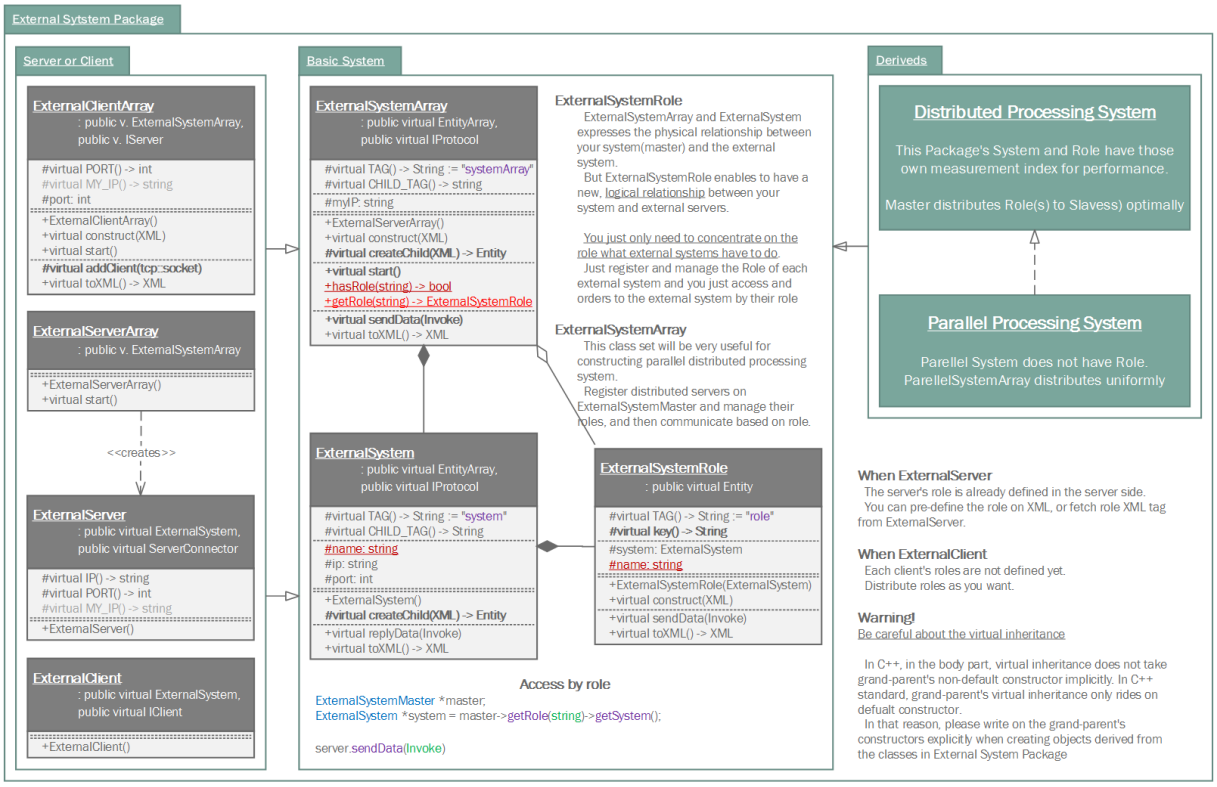
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

- ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 22 of file ParallelClientArray.hpp.

10.97.2 Constructor & Destructor Documentation

10.97.2.1 ParallelClientArray::ParallelClientArray ( )

Default Constructor.

Definition at line 8 of file ParallelClientArray.cpp.

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

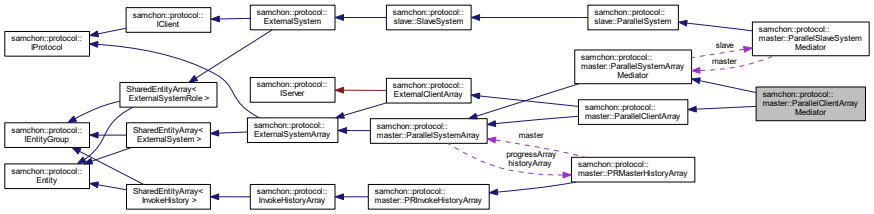
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelClientArray.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelClientArray.cpp

10.98 samchon::protocol::master::ParallelClientArrayMediator Class Reference

A mediator of parallel system between master and slaves.

`#include <ParallelClientArrayMediator.hpp>`

Collaboration diagram for samchon::protocol::master::ParallelClientArrayMediator:



Public Member Functions

- ParallelClientArrayMediator ()

  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override

  Construct data of the Entity from an XML object.
- virtual void start () override

  Start interaction.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the EntityGroup.

Additional Inherited Members

10.98.1 Detailed Description

A mediator of parallel system between master and slaves.

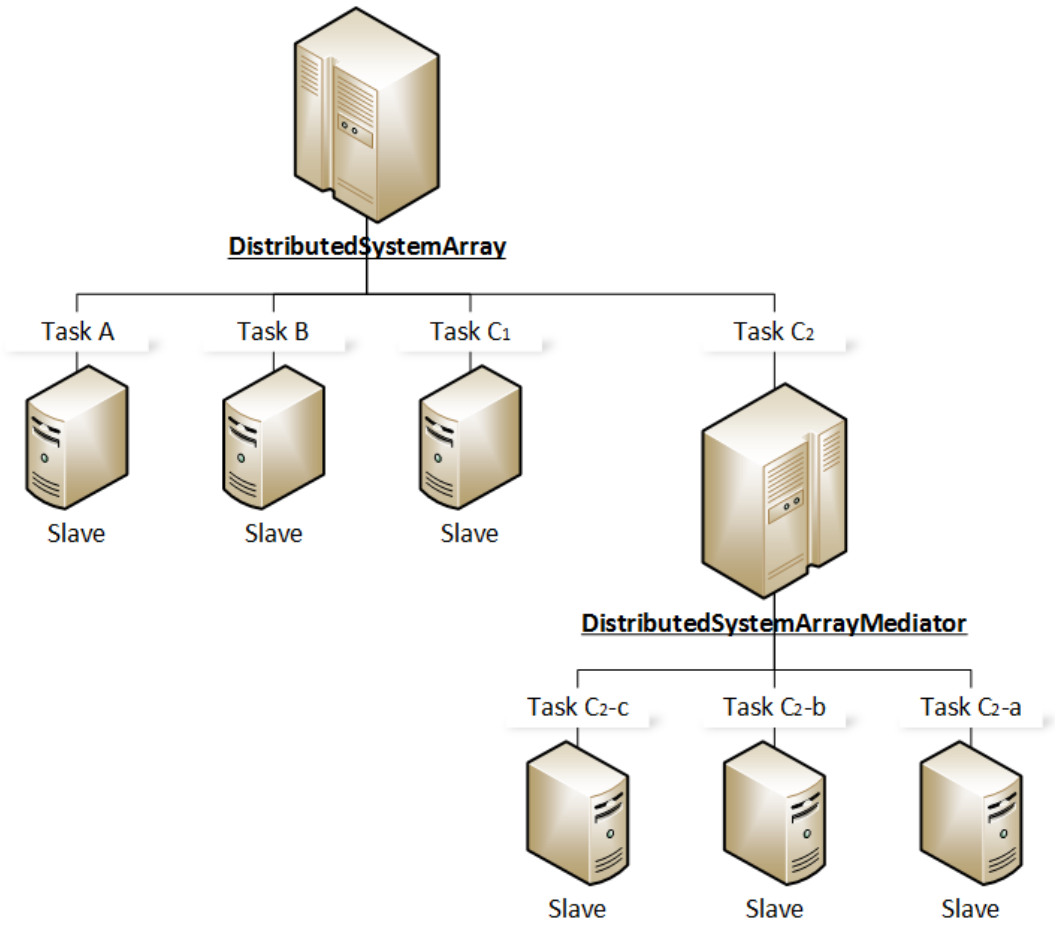
ParallelClientArrayMediator is a ParallelSystemArrayMediator specialized in clients' driver. ParallelClient↩ ArrayMediator is a client for its master and also manages children clients' slaves.

[Inherited]

ParallelSystemArrayMediator class is used to realizing a mediator system of tree-structured parallel processing system. ParallelSystemArrayMediator intermediates between origin-master(ParallelSystemArray) and its slaves(slave::ParallelSystem).

ParallelSystemArrayMediator is a master(ParallelSystemArray) within the framework of slaves systems (slave::ParallelSystem objects) and also can be a slave system (slave::ParallelSystem) within framework of its master(ParallelSystemArray) by having ParallelSlaveSystemMediator.

Todo Sends PRMasterHistory to its origin master.



[Inherited]

ParallelSystemArray is an ExternalSystemArray and an abstract class containing. and managing parallel system drvers within framework of master. The ParallelSystemArray class allocates segmented processes (Invoke message containing segmentation size) following by each system's performance index.

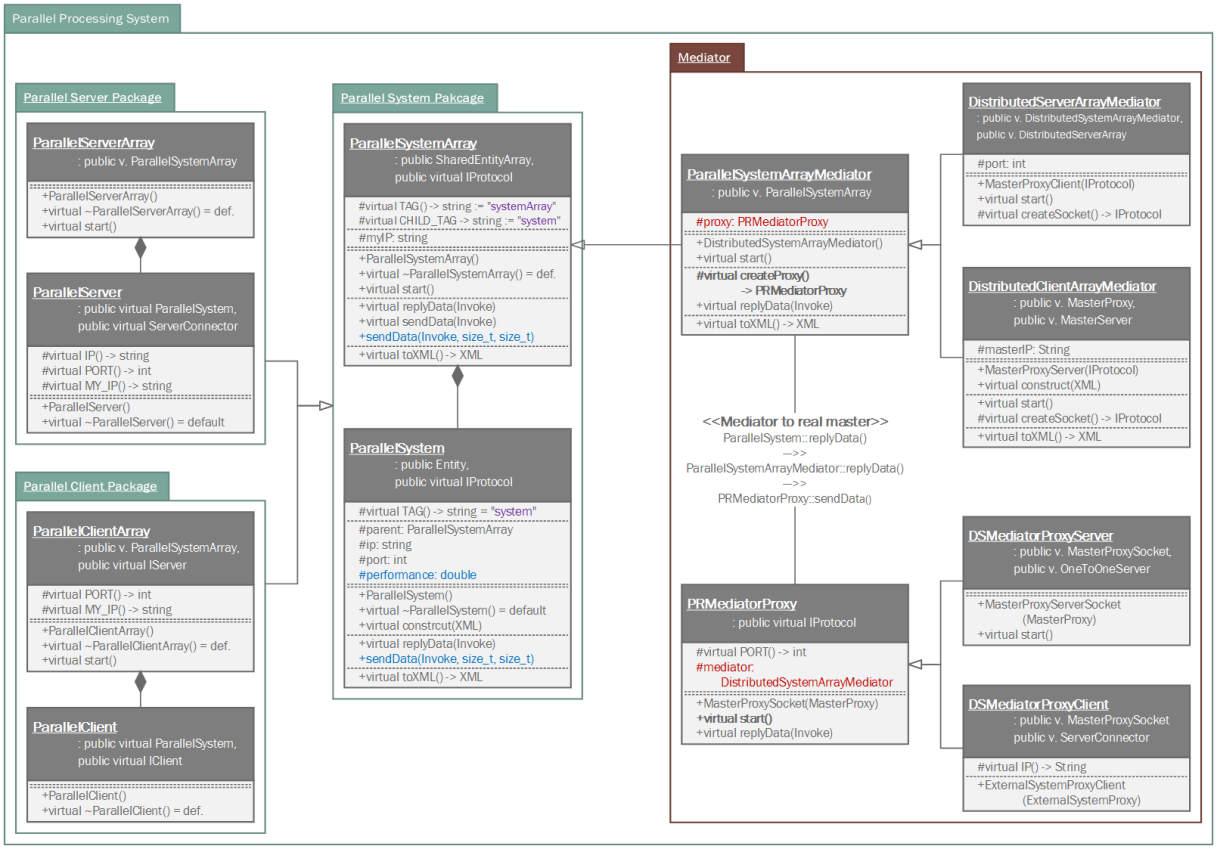
Unlike DistributedSystemArray, ParallelSystemArray does not have such complicate relationships and logics. All segmentation sent by sendData(Invoke, size_t) requires uniform performance. ParallelSystemArray just calculates each system's performance index by elapsing time such easily.

Of course, DistributedSystemArray determines which size of segmentation allocation will be suitable for each system by the performance index

.

Each segmentation in requested process (Invoke message) is equivalent. Thus, role of ParallelSystem objects in a ParallelSystemArray are almost same and does not need to specify ExternalSystemRole on each ParallelSystem.

- Not a matter to specifying ExternalSystemRole objects to each ParallelSystem. In that case, Invoke messages having segmentation size will be processed by ParallelSystemArray's own logic and Invoke messages without segmentation size will be handled by ordinary logic of ExternalSystemArray's own.



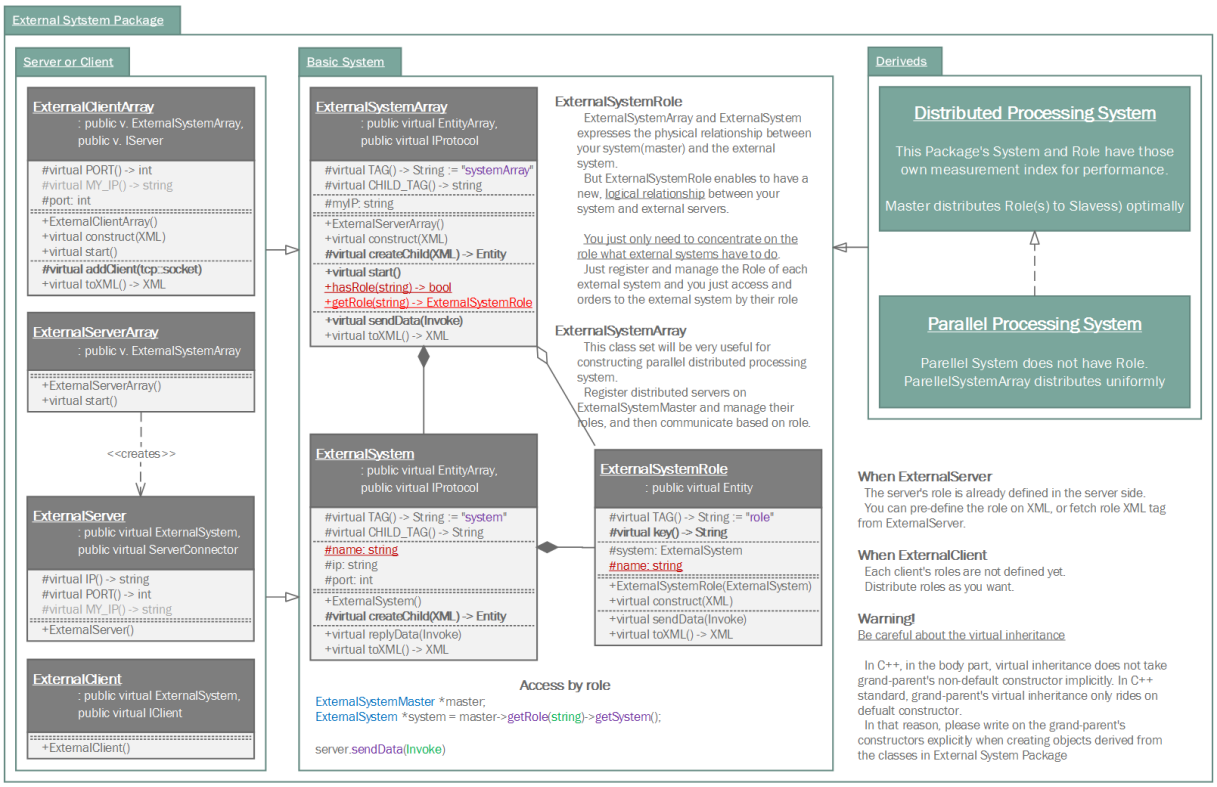
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

- ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 24 of file ParallelClientArrayMediator.hpp.

10.98.2 Constructor & Destructor Documentation

10.98.2.1 ParallelClientArrayMediator::ParallelClientArrayMediator ( )

Default Constructor.

Definition at line 17 of file ParallelClientArrayMediator.cpp.

References samchon::protocol::master::ParallelSystemArrayMediator::slave.

10.98.3 Member Function Documentation

10.98.3.1 void ParallelClientArrayMediator::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

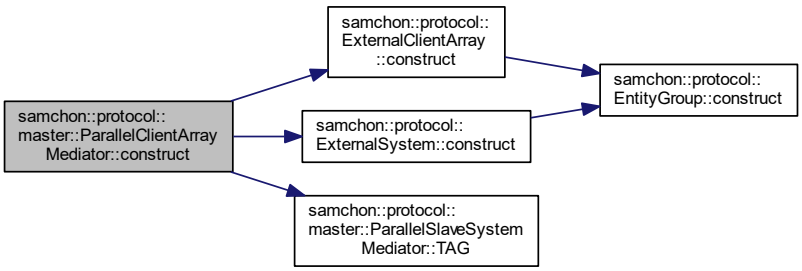
xml	An xml used to construct data of entity

Reimplemented from samchon::protocol::master::ParallelSystemArrayMediator.

Definition at line 23 of file ParallelClientArrayMediator.cpp.

References samchon::protocol::ExternalClientArray::construct(), samchon::protocol::ExternalSystem↩ ::construct(), samchon::protocol::master::ParallelSystemArrayMediator::slave, and samchon::protocol↩ ::master::ParallelSlaveSystemMediator::TAG().

Here is the call graph for this function:



10.98.3.2 void ParallelClientArrayMediator::start ( ) [override], [virtual]

Start interaction.

An abstract method starting interaction with external systems.

If external systems are servers, starts connection to them, else clients, opens a server and accepts the external systems. You can addict your own procudures of starting drivers, but if you directly override method of abstract ExternalSystemArray, be careful about virtual inheritance.

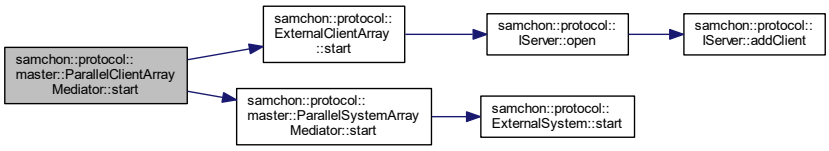
- DistributedClientArray::start() calls DistributedSystemArray::start() and ExternalClientArray:start()

Reimplemented from samchon::protocol::master::ParallelSystemArrayMediator.

Definition at line 34 of file ParallelClientArrayMediator.cpp.

References samchon::protocol::ExternalClientArray::start(), and samchon::protocol::master::Parallel↩ SystemArrayMediator::start().

Here is the call graph for this function:



10.98.3.3 auto ParallelClientArrayMediator::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

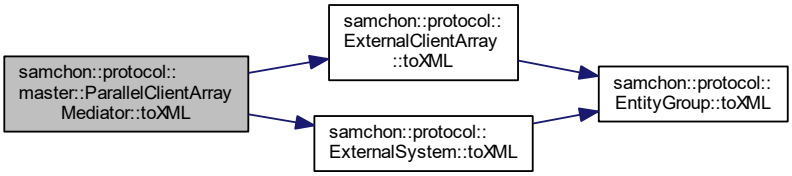
An XML object representing the Entity.

Reimplemented from samchon::protocol::master::ParallelSystemArrayMediator.

Definition at line 63 of file ParallelClientArrayMediator.cpp.

References samchon::protocol::master::ParallelSystemArrayMediator::slave, samchon::protocol::External↩ClientArray::toXML(), and samchon::protocol::ExternalSystem::toXML().

Here is the call graph for this function:



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

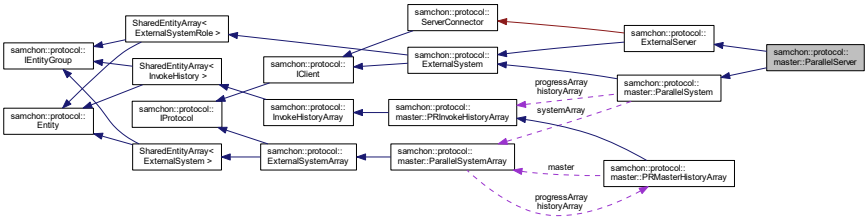
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelClientArrayMediator.↩hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelClientArrayMediator.↩cpp

10.99 samchon::protocol::master::ParallelServer Class Reference

A network driver for a parallel server.

```
#include <ParallelServer.hpp>
```

Collaboration diagram for samchon::protocol::master::ParallelServer:



Public Member Functions

- ParallelServer ()

  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override

  Construct data of the Entity from an XML object.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override

  Get an XML object represents the EntityGroup.

Additional Inherited Members

10.99.1 Detailed Description

A network driver for a parallel server.

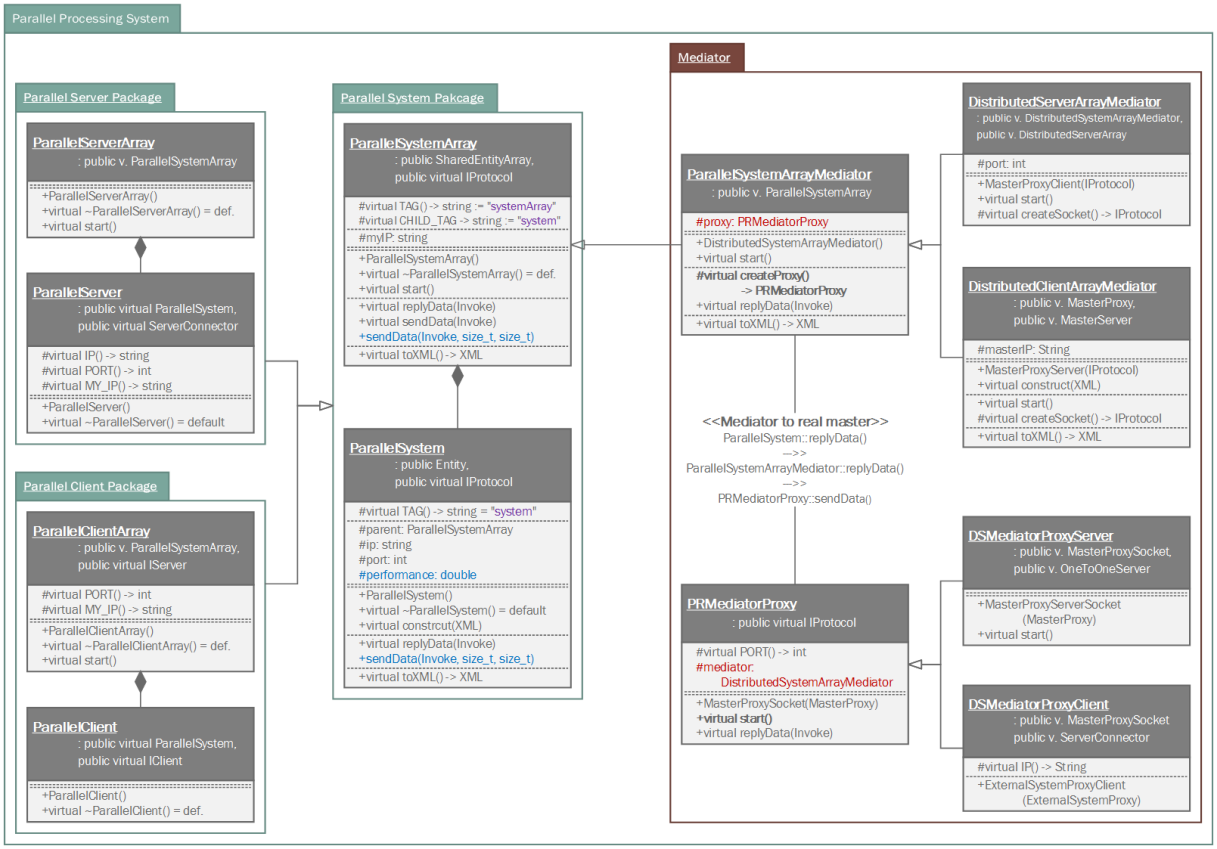
ParallelServer is an ParallelSystem specialized in server driver.

[Inherited]

ParallelSystem class is an ExternalSystem having performance index and history logs of handled Invoke messages with those elapsed times. Those performance index and history logs are used to determine how much segmentation to be allocated.

Each segmentation in requested process (Invoke message) is equivalent. Thus, role of ParallelSystem objects in a ParallelSystemArray are almost same and does not need to specify ExternalSystemRole on ParallelSystem.

- Not a matter to specifying ExternalSystemRole objects to each ParallelSystem. In that case, Invoke messages having segmentation size will be processed by ParallelSystem's own logic and Invoke messages without segmentation size will be handled by ordinary logic of ExternalSystem's own.



[Inherited]

Definition at line 22 of file ParallelServer.hpp.

10.99.2 Constructor & Destructor Documentation

10.99.2.1 samchon::protocol::master::ParallelServer::ParallelServer ( )

Default Constructor.

10.99.3 Member Function Documentation

10.99.3.1 virtual void samchon::protocol::master::ParallelServer::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Reimplemented from samchon::protocol::ExternalServer.

10.99.3.2 virtual auto samchon::protocol::master::ParallelServer::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::ExternalServer.

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

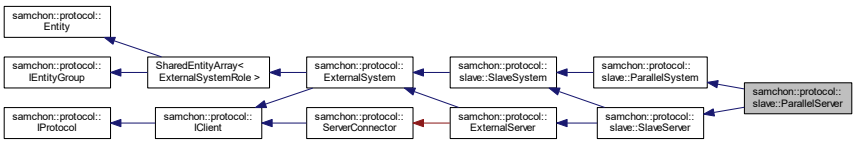
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelServer.hpp

10.100 samchon::protocol::slave::ParallelServer Class Reference

A slave parallel server.

`#include <ParallelServer.hpp>`

Collaboration diagram for samchon::protocol::slave::ParallelServer:



Public Member Functions

- ParallelServer ()
  Default Constructor.

Additional Inherited Members

10.100.1 Detailed Description

A slave parallel server.

ParallelSystem is a ParallelSystem class specialized in server driver.

[Inherited]

ParallelSystem is a class processing segmented Invoke messages from master.

[Inherited]

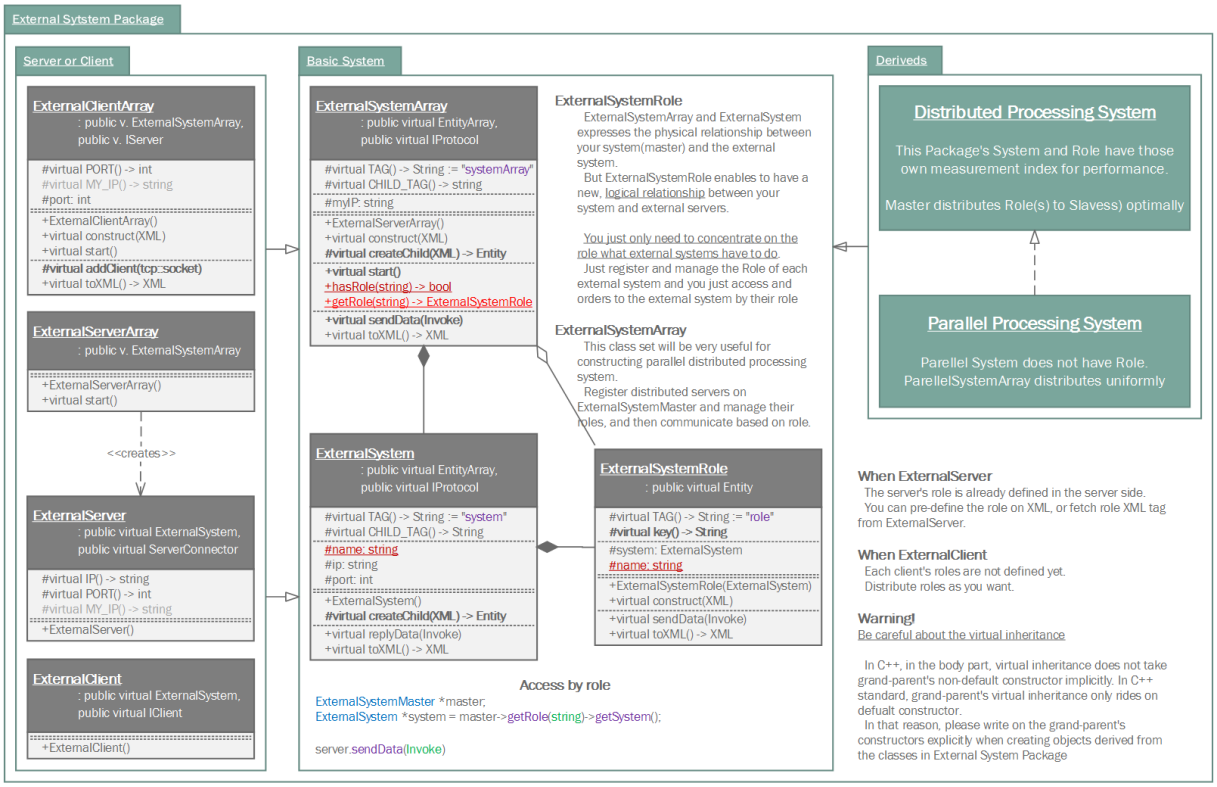
SlaveSystem, literally, means a slave system belongs to a maste system.

The SlaveSystem class is used in opposite side system of master::DistributedSystem and master::Parallel↩ System and reports elapsed time of each commmand (by Invoke message) for estimation of its performance.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, ExternalSystem is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystemRole.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an ExternalSystem. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 22 of file ParallelServer.hpp.

10.100.2 Constructor & Destructor Documentation

10.100.2.1 ParallelServer::ParallelServer ( )

Default Constructor.

Definition at line 8 of file ParallelServer_slave.cpp.

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

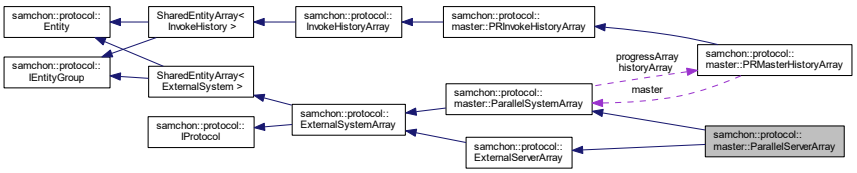
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/slave/ParallelServer.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/slave/ParallelServer_slave.cpp

10.101 samchon::protocol::master::ParallelServerArray Class Reference

An array of ParallelServer(s).

```
#include <ParallelServerArray.hpp>
```

Collaboration diagram for samchon::protocol::master::ParallelServerArray:



Public Member Functions

- ParallelServerArray ()
  Default Constructor.

Additional Inherited Members

10.101.1 Detailed Description

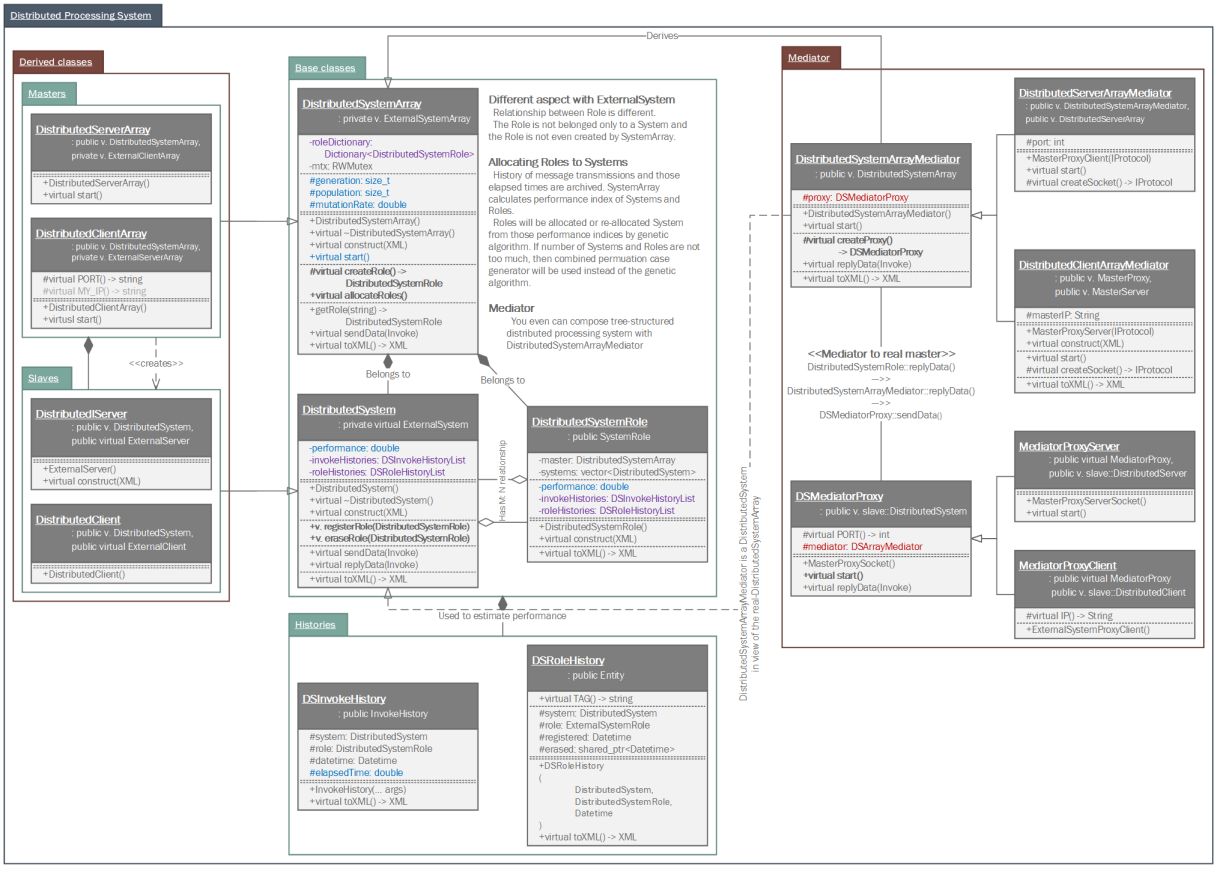
An array of ParallelServer(s).

ParallelServerArray is an ParallelSystemArray specialized in servers' driver.

[Inherited]

DistributedSystemArray class is an ExternalSystemArray and an abstract class containing and managing distributed system drivers within framework of master. The DistributedSystemArray class allocates DistributedSystemRole objects to DistributedSystem objects for optimization.

Unlike the ExternalSystemRole class which is dependent on an ExternalSystem, DistributedSystemRole is not dependent on a DistributedSystem and even can be allocated to multiple DistributedSystem objects. Allocation of DistributedSystemRole(s) to DistributedSystem(s) is determined by estimation of performance index pursuing best distribution. If number of DistributedSystem and DistributedSystemRole objects is not a lot, optimize by combined permuation algorithm. Else the objects are a lot, the allocation will be optimized by genetic algorithm.



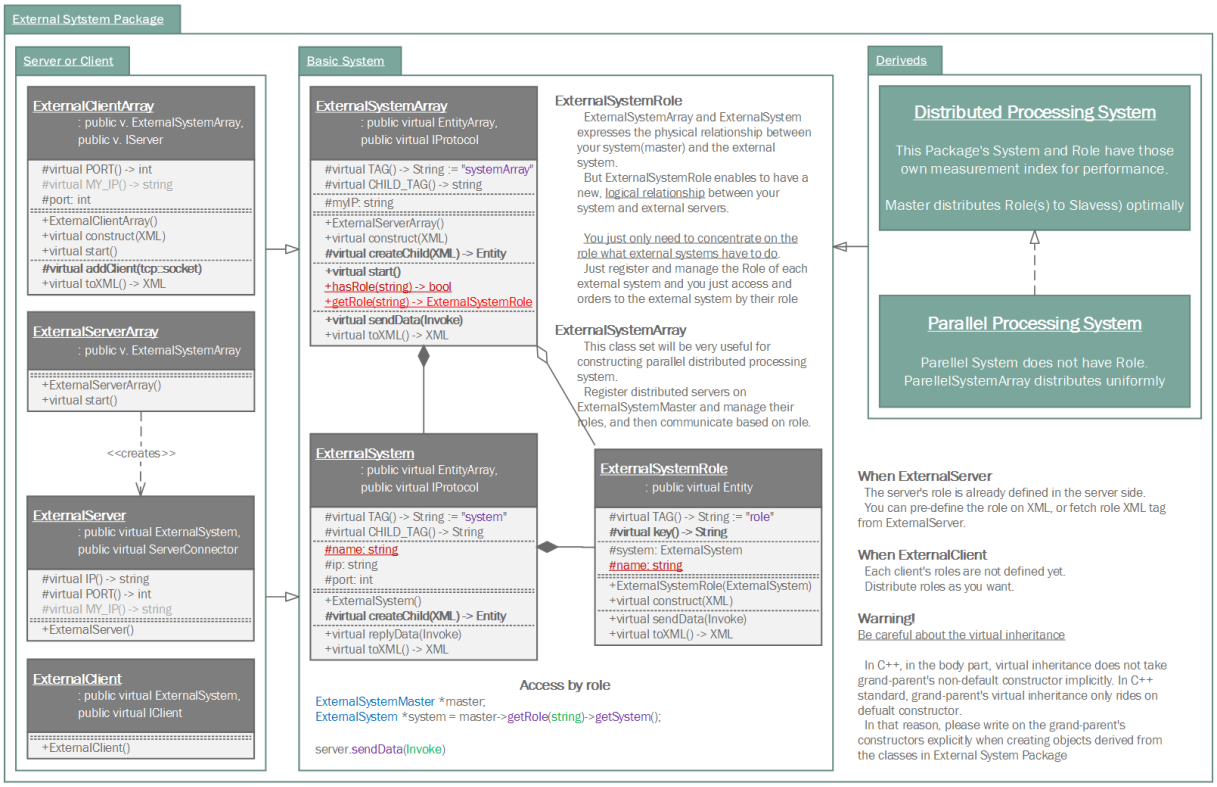
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

- ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 22 of file ParallelServerArray.hpp.

10.101.2 Constructor & Destructor Documentation

10.101.2.1 ParallelServerArray::ParallelServerArray ( )

Default Constructor.

Definition at line 8 of file ParallelServerArray.cpp.

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

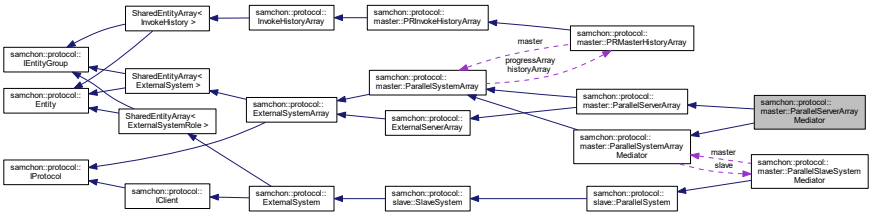
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelServerArray.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelServerArray.cpp

10.102 samchon::protocol::master::ParallelServerArrayMediator Class Reference

A mediator of parallel system between master and slaves.

```
#include <ParallelServerArrayMediator.hpp>
```

Collaboration diagram for samchon::protocol::master::ParallelServerArrayMediator:



Public Member Functions

- ParallelServerArrayMediator ()

  Default Constructor.

- virtual void start () override

  Start interaction.

Additional Inherited Members

10.102.1 Detailed Description

A mediator of parallel system between master and slaves.

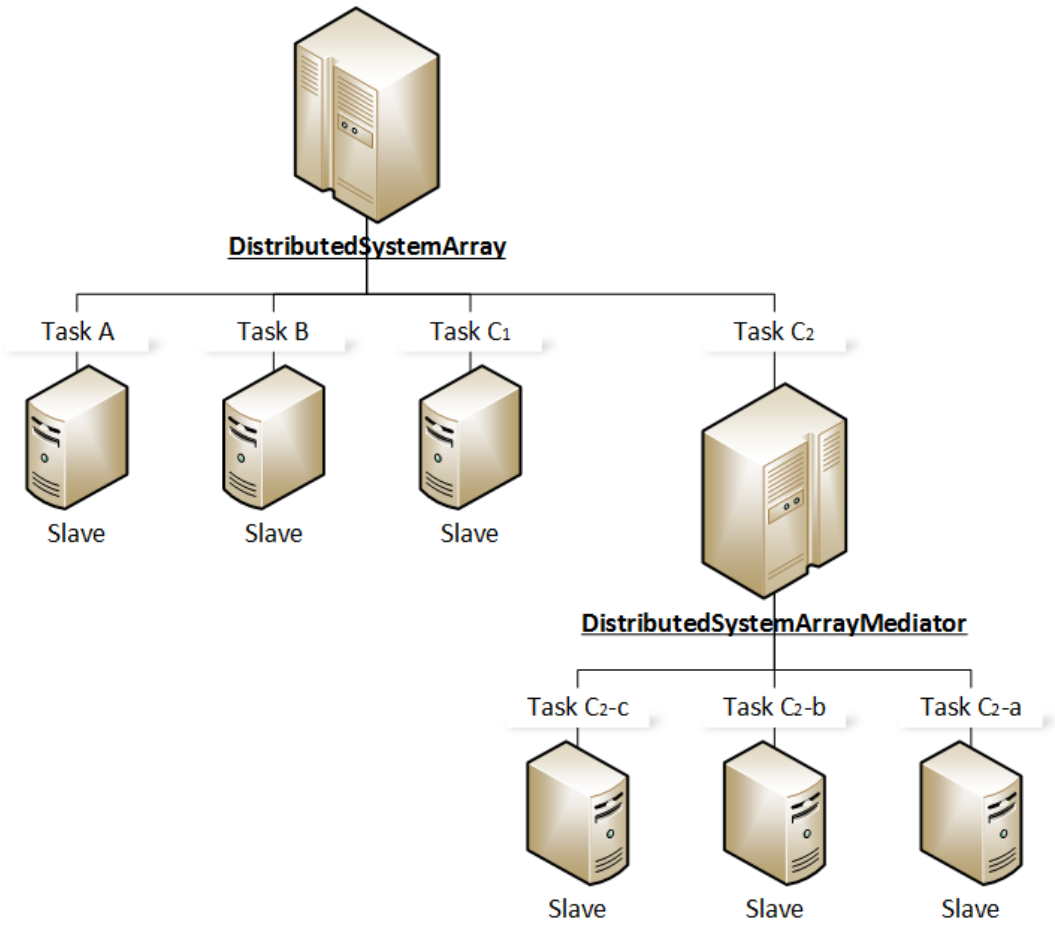
ParallelServerArrayMediator is a ParallelSystemArrayMediator specialized in servers' driver. ParallelServer↩ ArrayMediator is a server for its master and also manages children servers' slaves.

[Inherited]

ParallelSystemArrayMediator class is used to realizing a mediator system of tree-structured parallel processing system. ParallelSystemArrayMediator intermediates between origin-master(ParallelSystemArray) and its slaves(slave::ParallelSystem).

ParallelSystemArrayMediator is a master(ParallelSystemArray) within the framework of slaves systems (slave::ParallelSystem objects) and also can be a slave system (slave::ParallelSystem) within framework of its master(ParallelSystemArray) by having ParallelSlaveSystemMediator.

Todo Sends PRMasterHistory to its origin master.



[Inherited]

ParallelSystemArray is an ExternalSystemArray and an abstract class containing. and managing parallel system drvers within framework of master. The ParallelSystemArray class allocates segmented processes (Invoke message containing segmentation size) following by each system's performance index.

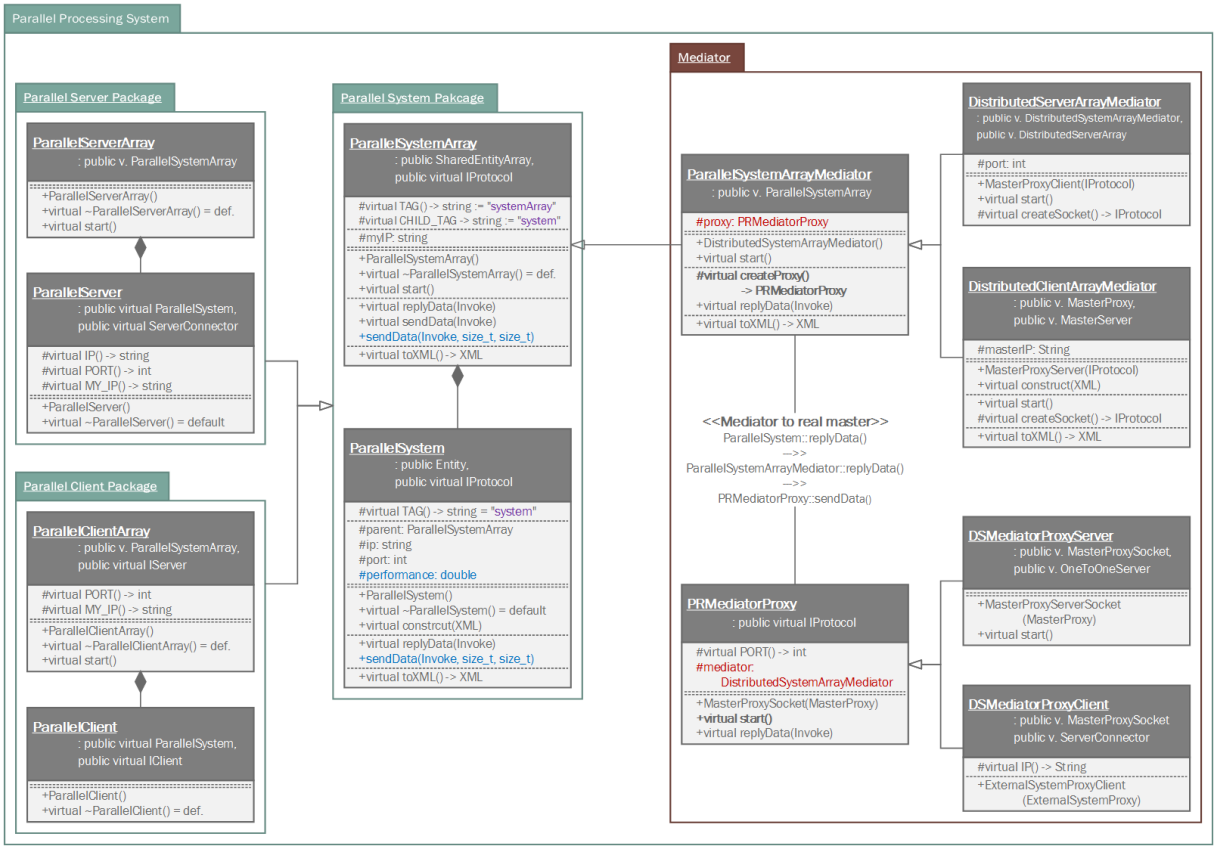
Unlike DistributedSystemArray, ParallelSystemArray does not have such complicate relationships and logics. All segmentation sent by sendData(Invoke, size_t) requires uniform performance. ParallelSystemArray just calculates each system's performance index by elapsing time such easily.

Of course, DistributedSystemArray determines which size of segmentation allocation will be suitable for each system by the performance index

.

Each segmentation in requested process (Invoke message) is equivalent. Thus, role of ParallelSystem objects in a ParallelSystemArray are almost same and does not need to specify ExternalSystemRole on each ParallelSystem.

- Not a matter to specifying ExternalSystemRole objects to each ParallelSystem. In that case, Invoke messages having segmentation size will be processed by ParallelSystemArray's own logic and Invoke messages without segmentation size will be handled by ordinary logic of ExternalSystemArray's own.



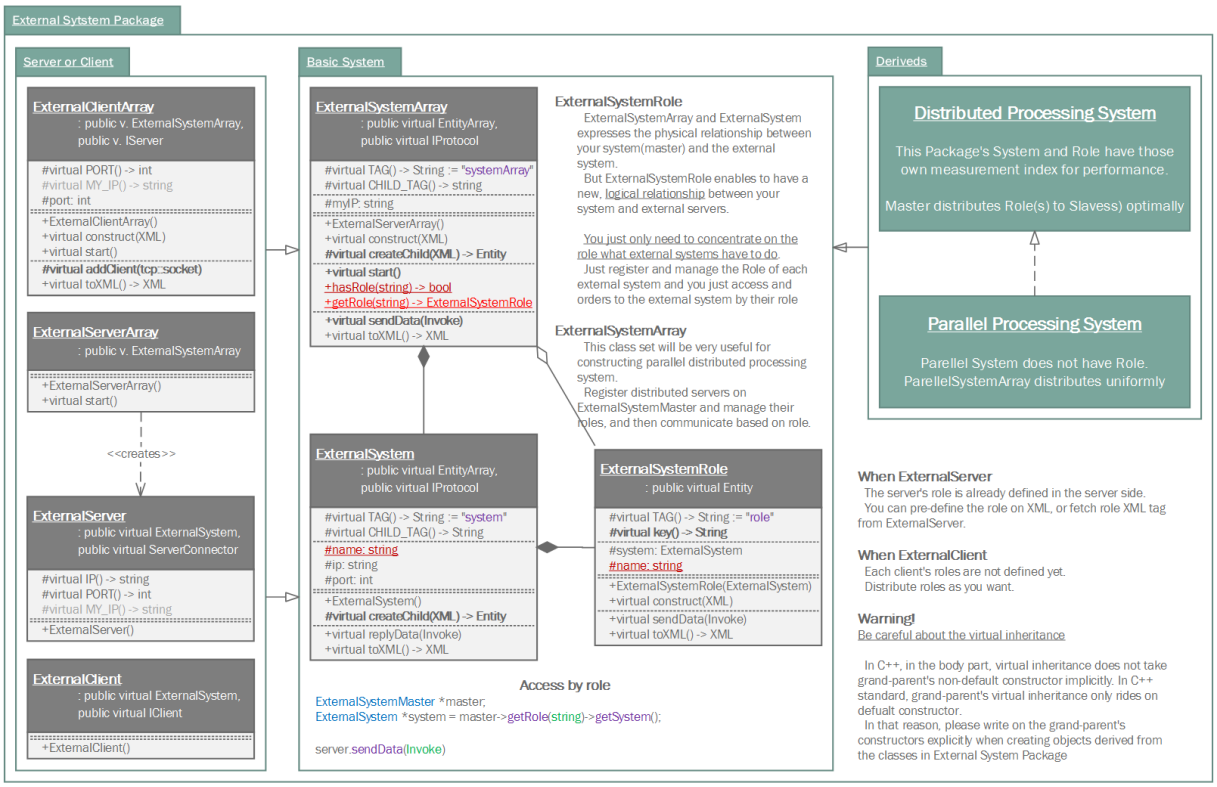
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

- ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 24 of file ParallelServerArrayMediator.hpp.

10.102.2 Constructor & Destructor Documentation

10.102.2.1 ParallelServerArrayMediator::ParallelServerArrayMediator ( )

Default Constructor.

Definition at line 13 of file ParallelServerArrayMediator.cpp.

References samchon::protocol::master::ParallelSystemArrayMediator::slave.

10.102.3 Member Function Documentation

10.102.3.1 void ParallelServerArrayMediator::start ( ) [override], [virtual]

Start interaction.

An abstract method starting interaction with external systems.

If external systems are servers, starts connection to them, else clients, opens a server and accepts the external systems. You can addict your own procudures of starting drivers, but if you directly override method of abstract ExternalSystemArray, be careful about virtual inheritance.

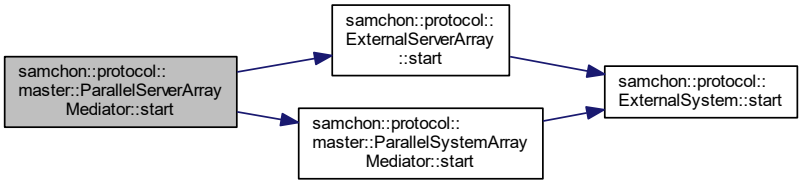
- DistributedClientArray::start() calls DistributedSystemArray::start() and ExternalClientArray:start()

Reimplemented from samchon::protocol::master::ParallelSystemArrayMediator.

Definition at line 19 of file ParallelServerArrayMediator.cpp.

References samchon::protocol::ExternalServerArray::start(), and samchon::protocol::master::Parallel↩ SystemArrayMediator::start().

Here is the call graph for this function:



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

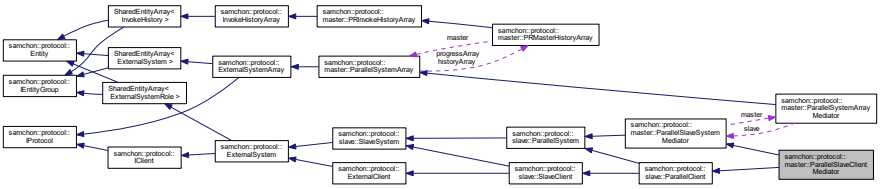
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelServerArrayMediator.↩ hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelServerArrayMediator.↩ cpp

10.103 samchon::protocol::master::ParallelSlaveClientMediator Class Reference

A parallel slave client for mediation.

`#include <ParallelSlaveClientMediator.hpp>`

Collaboration diagram for samchon::protocol::master::ParallelSlaveClientMediator:



Public Member Functions

- ParallelSlaveClientMediator ()
  Default Constructor.

Additional Inherited Members

10.103.1 Detailed Description

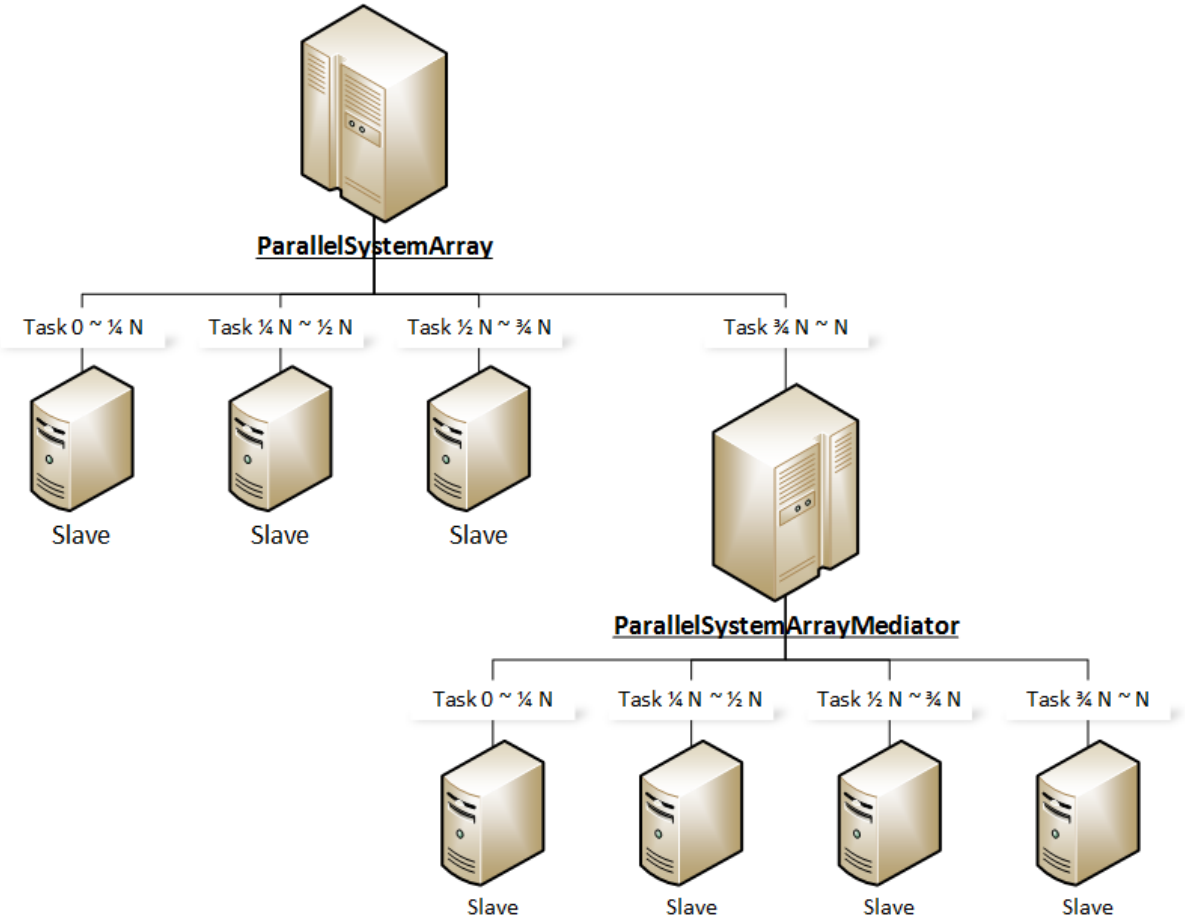
A parallel slave client for mediation.

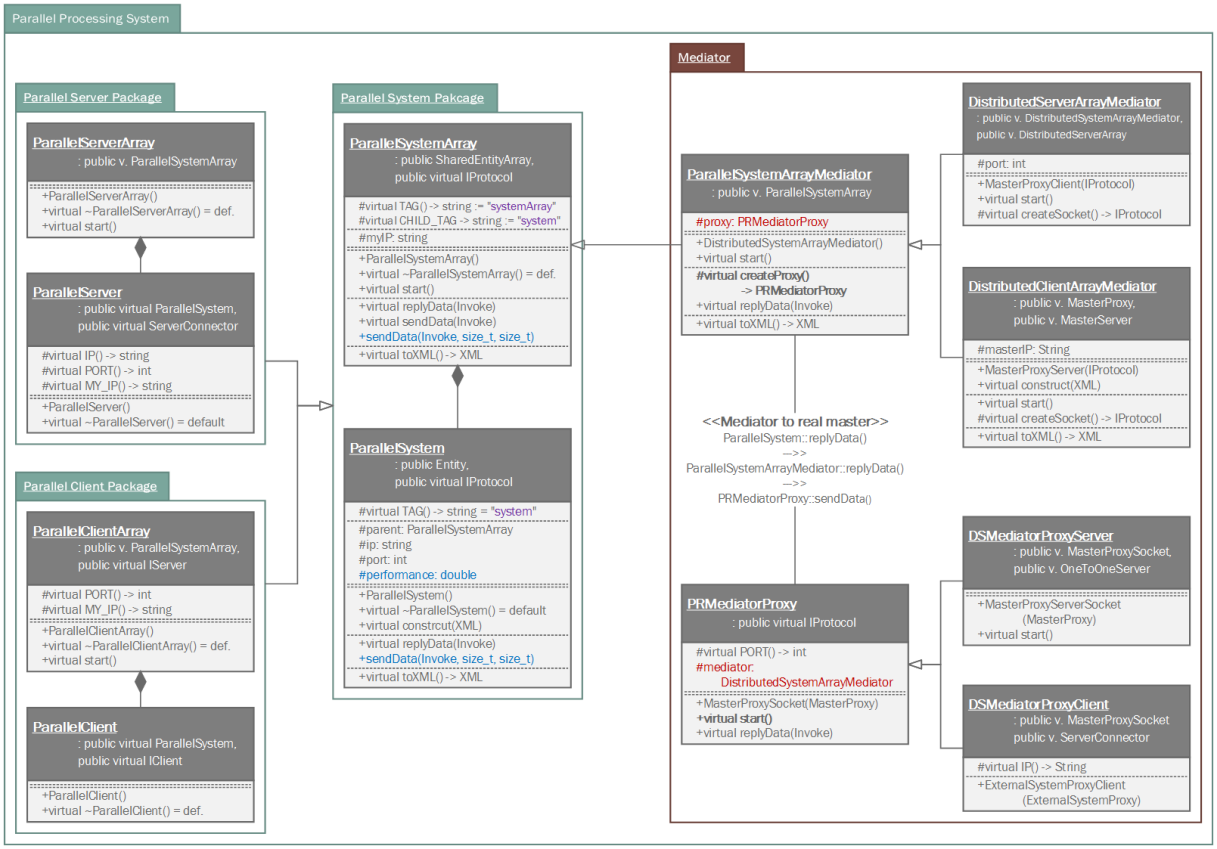
ParallelSlaveClientMediator is a ParallelSlaveSystemMediator class specialized in client driver.

[Inherited]

ParallelSlaveSystemMediator is an abstract class derived from slave::ParallelSystem for realizing tree-structured distriuted processins system. ParallelSlaveSystemMediator acts a slave system's role for its master, on the above level.

ParallelSystemArrayMediator and ParallelSlaveSystemMediator classes are paired to act a mediator system role by acting their own role, master system role for below slaves and slave system role for above master system.





[Inherited]

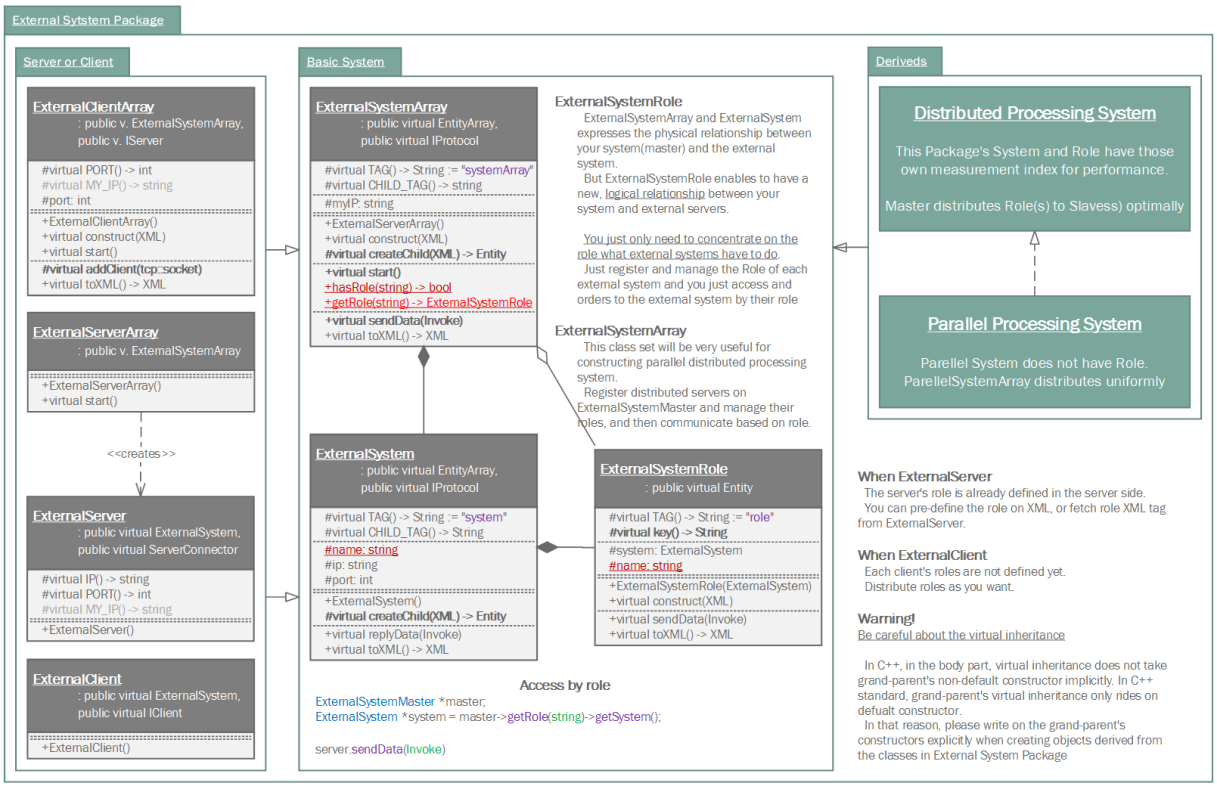
SlaveSystem, literally, means a slave system belongs to a maste system.

The SlaveSystem class is used in opposite side system of master::DistributedSystem and master::Parallel↩System and reports elapsed time of each commmand (by Invoke message) for estimation of its performance.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, ExternalSystem is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystem↩ Role.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an External↩ SystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an External↩ System. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 23 of file ParallelSlaveClientMediator.hpp.

10.103.2 Constructor & Destructor Documentation

10.103.2.1 ParallelSlaveClientMediator::ParallelSlaveClientMediator ( )

Default Constructor.

Definition at line 8 of file ParallelSlaveClientMediator.cpp.

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

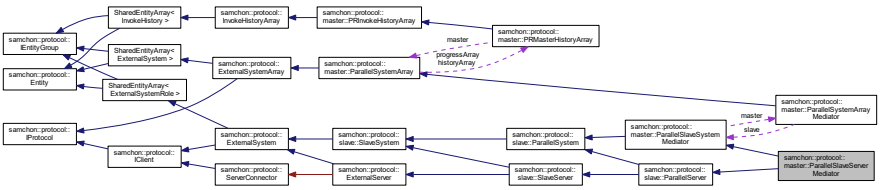
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelSlaveClientMediator.↩ hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelSlaveClientMediator.↩ cpp

10.104 samchon::protocol::master::ParallelSlaveServerMediator Class Reference

A parallel slave server for mediation.

```
#include <ParallelSlaveServerMediator.hpp>
```

Collaboration diagram for samchon::protocol::master::ParallelSlaveServerMediator:



Public Member Functions

- ParallelSlaveServerMediator ()

  Default Constructor.

Additional Inherited Members

10.104.1 Detailed Description

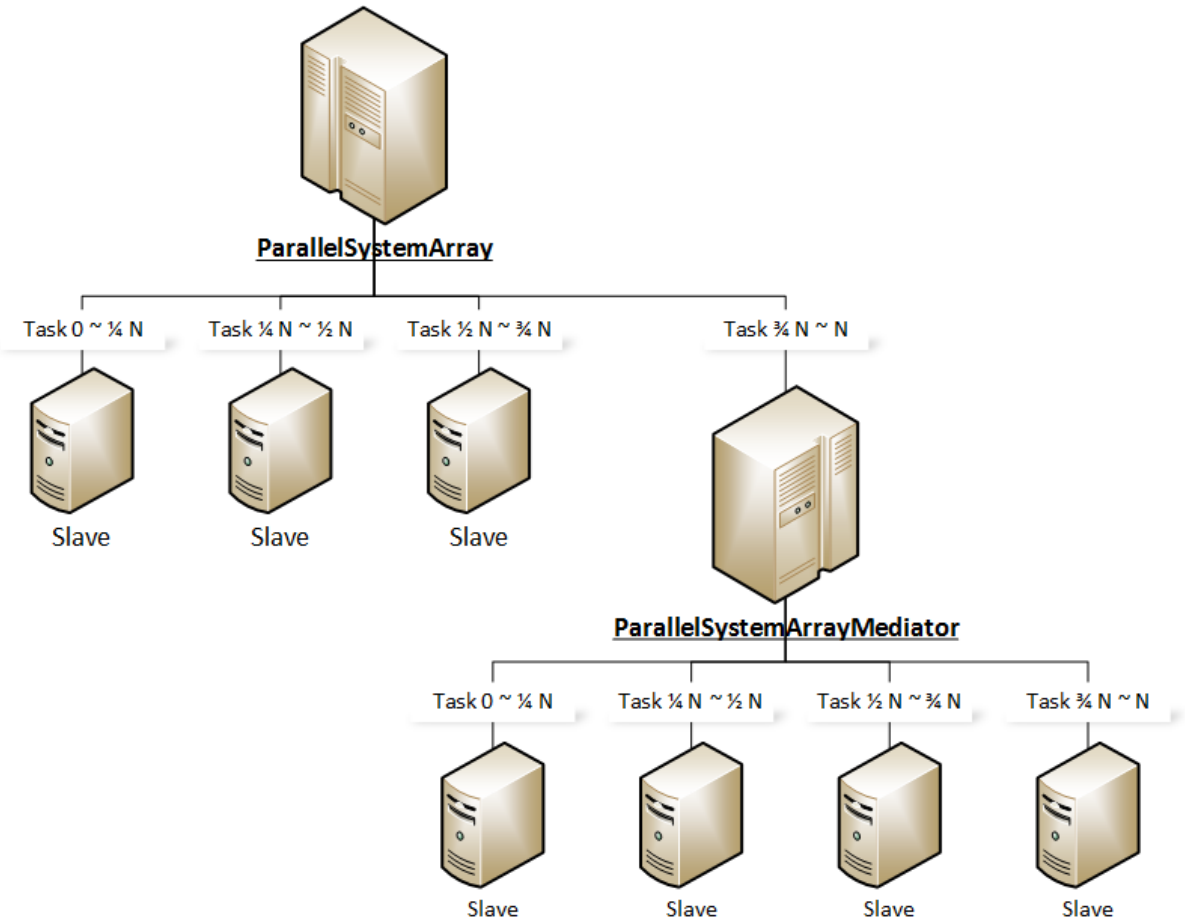
A parallel slave server for mediation.

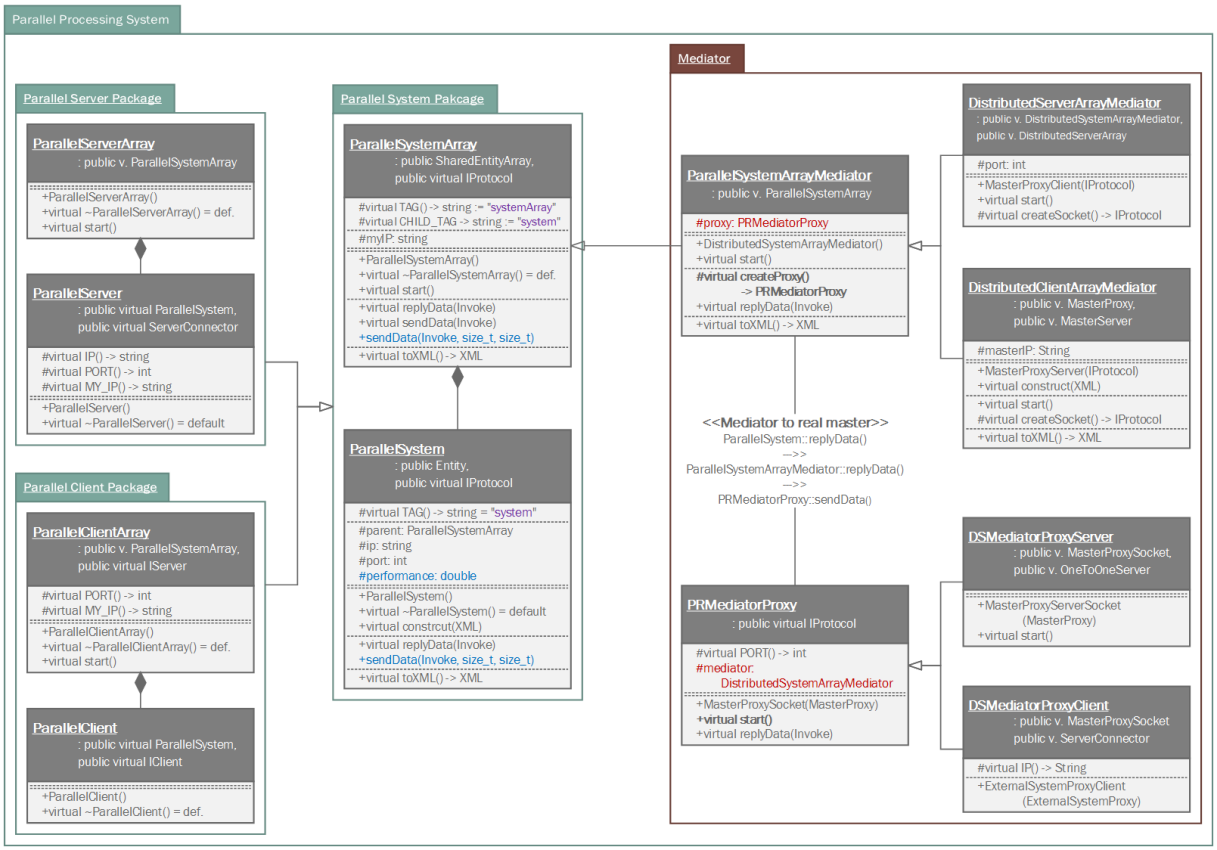
ParallelSlaveServerMediator is a ParallelSlaveSystemMediator class specialized in server driver.

[Inherited]

ParallelSlaveSystemMediator is an abstract class derived from slave::ParallelSystem for realizing tree-structured distriuted processins system. ParallelSlaveSystemMediator acts a slave system's role for its master, on the above level.

ParallelSystemArrayMediator and ParallelSlaveSystemMediator classes are paired to act a mediator system role by acting their own role, master system role for below slaves and slave system role for above master system.


ParallelSystemArray
Task 0 ~ ¼ N
Task ¼ N ~ ½ N
Task ½ N ~ ¾ N
Task ¾ N ~ N
Slave
Slave
Slave
ParallelSystemArrayMediator
Task 0 ~ ¼ N
Task ¼ N ~ ½ N
Task ½ N ~ ¾ N
Task ¾ N ~ N
Slave
Slave
Slave
Slave


Parallel Processing System
Parallel Server Package
ParallelServerArray
: public v. ParallelSystemArray
+ParallelServerArray()
+virtual ~ParallelServerArray() = def.
+virtual start()
ParallelServer
: public virtual ParallelSystem,
public virtual ServerConnector
#virtual IP() -> string
#virtual PORT() -> int
#virtual MY_IP() -> string
+ParallelServer()
+virtual ~ParallelServer() = default
Parallel Client Package
ParallelClientArray
: public v. ParallelSystemArray,
public virtual IServer
#virtual PORT() -> int
#virtual MY_IP() -> string
+ParallelClientArray()
+virtual ~ParallelClientArray() = def.
+virtual start()
ParallelClient
: public virtual ParallelSystem,
public virtual IClient
+ParallelClient()
+virtual ~ParallelClient() = def.
Parallel System Pakcage
ParallelSystemArray
: public SharedEntityArray,
public virtual IProtocol
#virtual TAG() -> string := "systemArray"
#virtual CHILD_TAG -> string := "system"
#myIP: string
+ParallelSystemArray()
+virtual ~ParallelSystemArray() = def.
+virtual start()
+virtual replyData(Invoke)
+virtual sendData(Invoke)
+sendData(Invoke, size_t, size_t)
+virtual toXML() -> XML
ParallelSystem
: public Entity,
public virtual IProtocol
#virtual TAG() -> string = "system"
#parent: ParallelSystemArray
#ip: string
#port: int
#performance: double
+ParallelSystem()
+virtual ~ParallelSystem() = default
+virtual constrcut(XML)
+virtual replyData(Invoke)
+sendData(Invoke, size_t, size_t)
+virtual toXML() -> XML
Mediator
ParallelSystemArrayMediator
: public v. ParallelSystemArray
#proxy: PRMediatorProxy
+DistributedSystemArrayMediator()
+virtual start()
#virtual createProxy()
-> PRMediatorProxy
+virtual replyData(Invoke)
+virtual toXML() -> XML
DistributedServerArrayMediator
: public v. DistributedSystemArrayMediator,
public v. DistributedServerArray
#port: int
+MasterProxyClient(IProtocol)
+virtual start()
#virtual createSocket() -> IProtocol
DistributedClientArrayMediator
: public v. MasterProxy,
public v. MasterServer
#masterIP: String
+MasterProxyServer(IProtocol)
+virtual construct(XML)
+virtual start()
#virtual createSocket() -> IProtocol
+virtual toXML() -> XML
<<Mediator to real master>>
ParallelSystem::replyData()
-->>
ParallelSystemArrayMediator::replyData()
-->>
PRMediatorProxy::sendData()
PRMediatorProxy
: public virtual IProtocol
#virtual PORT() -> int
#mediator:
DistributedSystemArrayMediator
+MasterProxySocket(MasterProxy)
+virtual start()
+virtual replyData(Invoke)
DSMediatorProxyServer
: public v. MasterProxySocket,
public v. OneToOneServer
+MasterProxyServerSocket
(MasterProxy)
+virtual start()
DSMediatorProxyClient
: public v. MasterProxySocket
public v. ServerConnector
#virtual IP() -> String
+ExternalSystemProxyClient
(ExternalSystemProxy)

[Inherited]

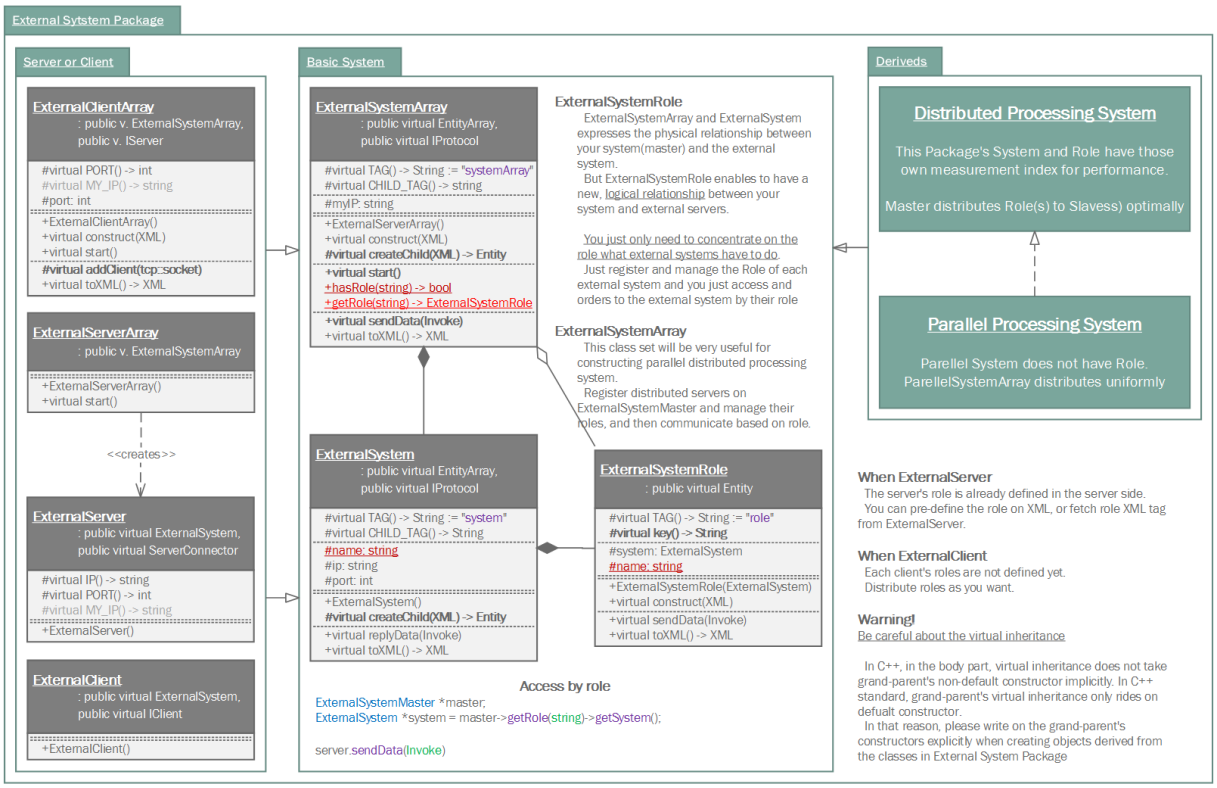
SlaveSystem, literally, means a slave system belongs to a maste system.

The SlaveSystem class is used in opposite side system of master::DistributedSystem and master::Parallel↩System and reports elapsed time of each commmand (by Invoke message) for estimation of its performance.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, ExternalSystem is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystem↩Role.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an External↩SystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an External↩System. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 23 of file ParallelSlaveServerMediator.hpp.

10.104.2 Constructor & Destructor Documentation

10.104.2.1 ParallelSlaveServerMediator::ParallelSlaveServerMediator ( )

Default Constructor.

Definition at line 8 of file ParallelSlaveServerMediator.cpp.

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

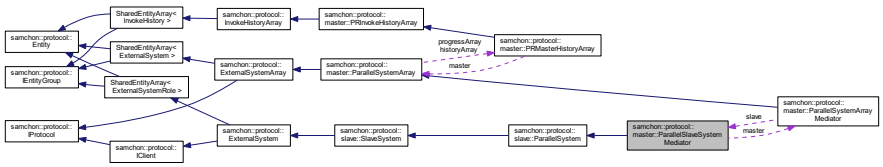
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelSlaveServerMediator.↩ hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelSlaveServerMediator.↩ cpp

10.105 samchon::protocol::master::ParallelSlaveSystemMediator Class Reference

A parallel slave system for mediation.

```
#include <ParallelSlaveSystemMediator.hpp>
```

Collaboration diagram for samchon::protocol::master::ParallelSlaveSystemMediator:



Public Member Functions

- ParallelSlaveSystemMediator ()

  Default Constructor.
- virtual void replyPieceData (std::shared_ptr< Invoke >, size_t, size_t) override

  Handle requested processes which are segmented.
- virtual auto TAG () const -> std::string override

  A tag name when represented by XML.

Private Attributes

- ParallelSystemArrayMediator * master

  A master system for mediation.

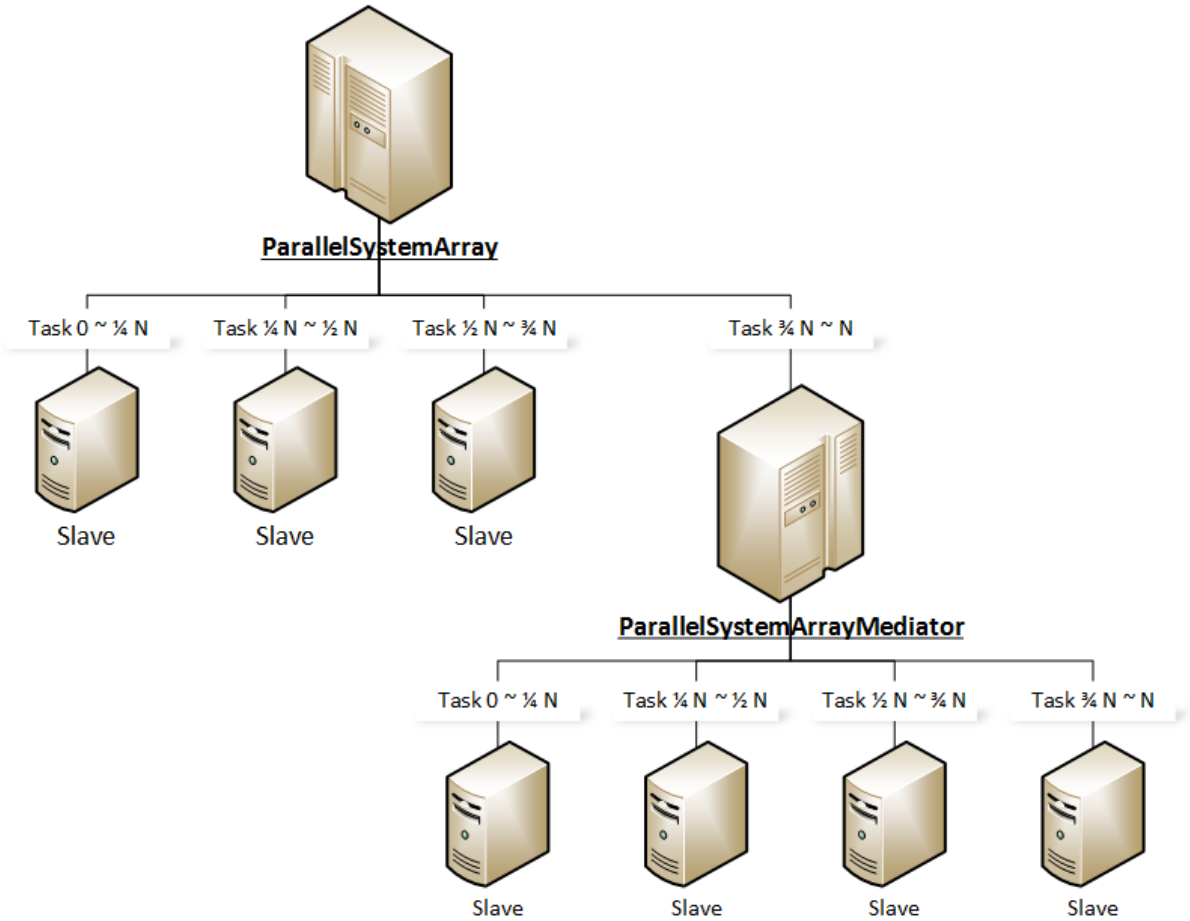
Additional Inherited Members

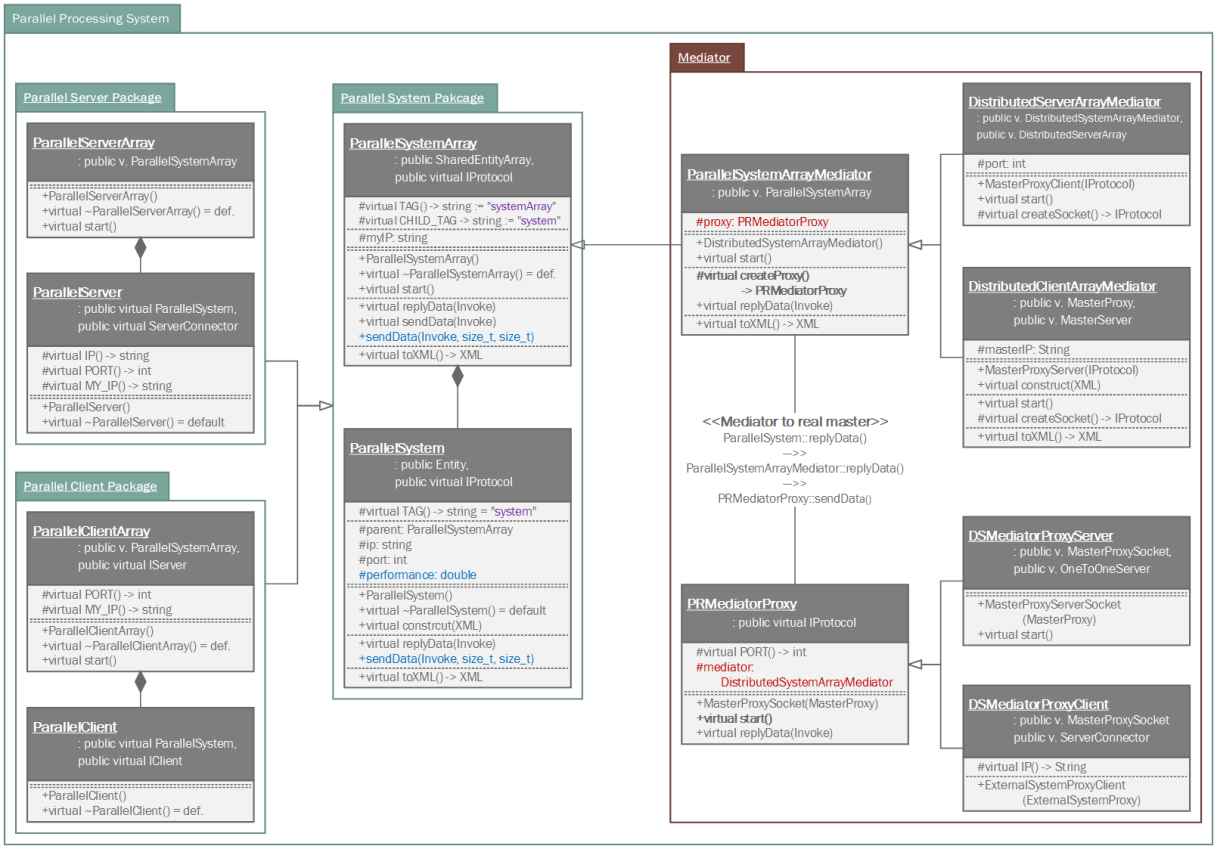
10.105.1 Detailed Description

A parallel slave system for mediation.

ParallelSlaveSystemMediator is an abstract class derived from slave::ParallelSystem for realizing tree-structured distriuted processins system. ParallelSlaveSystemMediator acts a slave system's role for its master, on the above level.

ParallelSystemArrayMediator and ParallelSlaveSystemMediator classes are paired to act a mediator system role by acting their own role, master system role for below slaves and slave system role for above master system.





[Inherited]

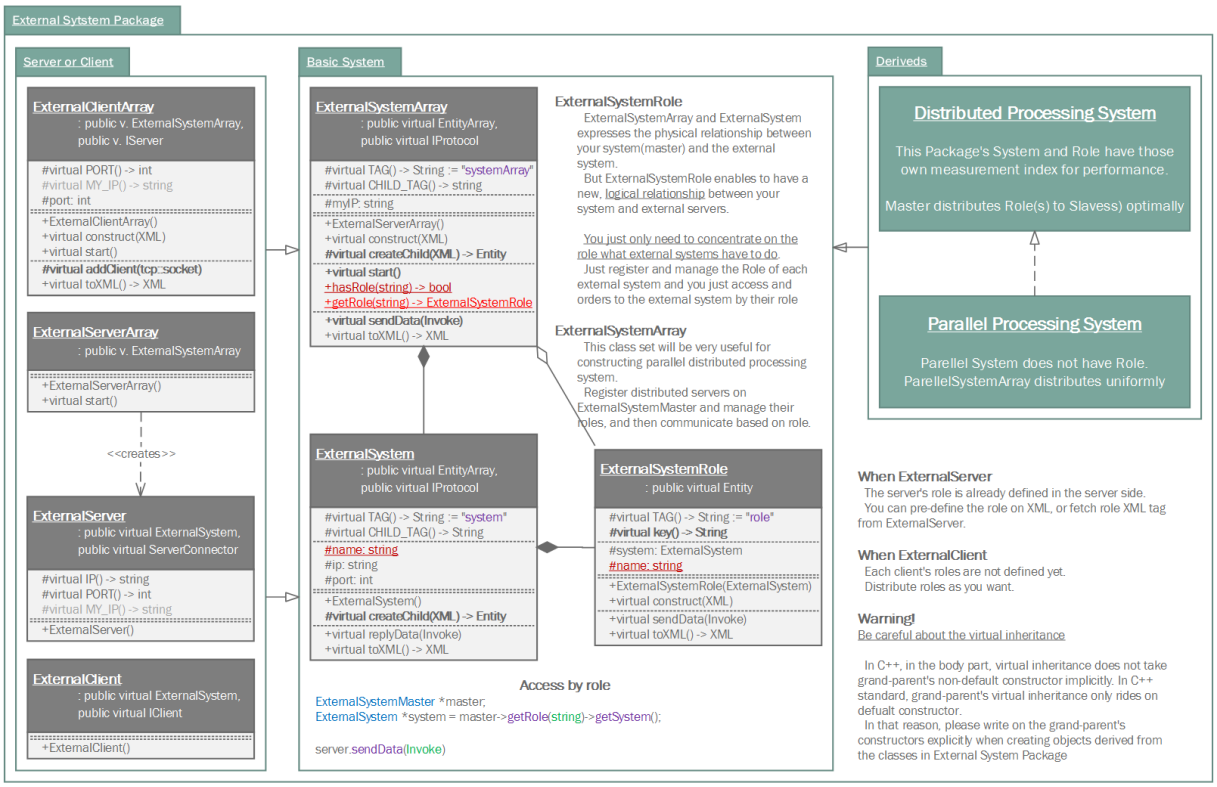
SlaveSystem, literally, means a slave system belongs to a maste system.

The SlaveSystem class is used in opposite side system of master::DistributedSystem and master::Parallel↩ System and reports elapsed time of each commmand (by Invoke message) for estimation of its performance.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, ExternalSystem is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystem↩ Role.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an External↩ SystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an External↩ System. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 35 of file ParallelSlaveSystemMediator.hpp.

10.105.2 Constructor & Destructor Documentation

10.105.2.1 ParallelSlaveSystemMediator::ParallelSlaveSystemMediator ( )

Default Constructor.

Definition at line 12 of file ParallelSlaveSystemMediator.cpp.

References master.

10.105.3 Member Function Documentation

10.105.3.1 void ParallelSlaveSystemMediator::replyPieceData (std::shared_ptr< Invoke > , size_t , size_t) [override],[virtual]

Handle requested processes which are segmented.

Parameters

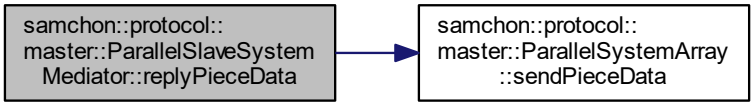
invoke	An invoke message containing segmented process requestments.
startIndex	Starting index number of segmentation.
size	Size of segmentation.

Implements samchon::protocol::slave::ParallelSystem.

Definition at line 17 of file ParallelSlaveSystemMediator.cpp.

References master, and samchon::protocol::master::ParallelSystemArray::sendPieceData().

Here is the call graph for this function:



10.105.3.2 auto ParallelSlaveSystemMediator::TAG ( ) const -> std::string [override], [virtual]

A tag name when represented by XML.

Returns

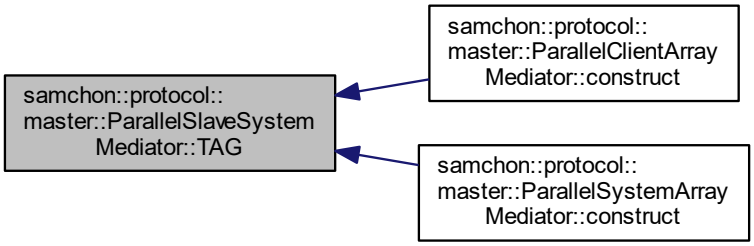
A tag name

Reimplemented from samchon::protocol::ExternalSystem.

Definition at line 25 of file ParallelSlaveSystemMediator.cpp.

Referenced by samchon::protocol::master::ParallelClientArrayMediator::construct(), and samchon↩ ::protocol::master::ParallelSystemArrayMediator::construct().

Here is the caller graph for this function:



10.105.4 Member Data Documentation

10.105.4.1 ParallelSystemArrayMediator* samchon::protocol::master::ParallelSlaveSystemMediator::master [private]

A master system for mediation.

Definition at line 46 of file ParallelSlaveSystemMediator.hpp.

Referenced by ParallelSlaveSystemMediator(), and replyPieceData().

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

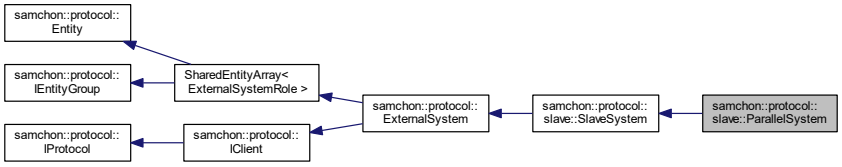
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelSlaveSystemMediator.↩ hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelSlaveSystemMediator.↩ cpp

10.106 samchon::protocol::slave::ParallelSystem Class Reference

A slave parallel system.

`#include <ParallelSystem.hpp>`

Collaboration diagram for samchon::protocol::slave::ParallelSystem:



Public Member Functions

- ParallelSystem ()
  Default Constructor.
- virtual void replyPieceData (std::shared_ptr< Invoke >, size_t, size_t)=0
  Handle requested processes which are segmented.

Protected Member Functions

- virtual auto createChild (std::shared_ptr< library::XML >) -> ExternalSystemRole *override
  Factory method of a child Entity.
- virtual void _replyData (std::shared_ptr< Invoke >) override
  Pre-processor for segmented processes and reporting elapsed time.

Additional Inherited Members

10.106.1 Detailed Description

A slave parallel system.

ParallelSystem is a class processing segmented Invoke messages from master.

[Inherited]

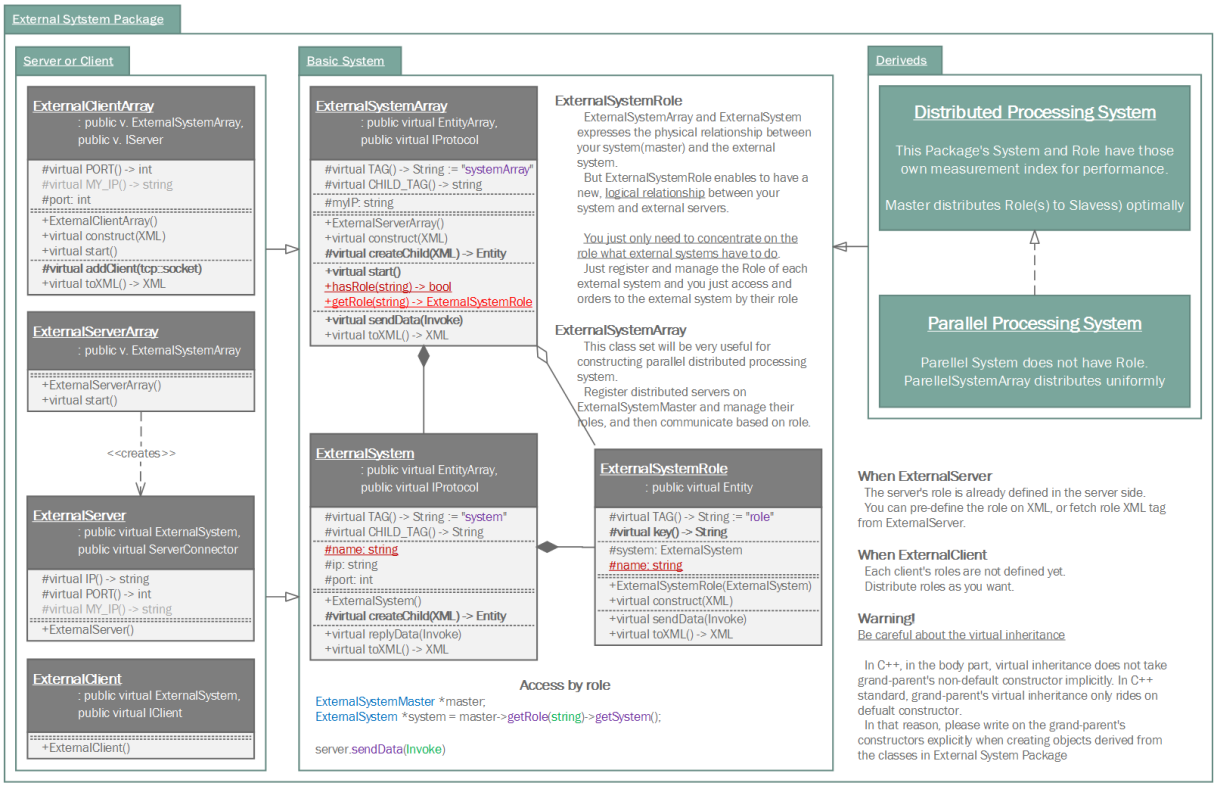
SlaveSystem, literally, means a slave system belongs to a maste system.

The SlaveSystem class is used in opposite side system of master::DistributedSystem and master::Parallel↩ System and reports elapsed time of each commmand (by Invoke message) for estimation of its performance.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, ExternalSystem is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystem↩ Role.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an External↩ SystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an External↩ System. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 21 of file ParallelSystem.hpp.

10.106.2 Constructor & Destructor Documentation

10.106.2.1 ParallelSystem::ParallelSystem ( )

Default Constructor.

Definition at line 12 of file ParallelSystem_slave.cpp.

10.106.3 Member Function Documentation

10.106.3.1 virtual auto samchon::protocol::slave::ParallelSystem::createChild (std::shared_ptr< library::XML >) -> ExternalSystemRole *override [override], [protected], [virtual]

Factory method of a child Entity.

EntityGroup::createChild() is a factory method creating a new child Entity which is belonged to the EntityGroup. This method is called by EntityGroup::construct(). The children construction methods Entity::construct() will be called by abstract method of the EntityGroup::construct().

Returns

A new child Entity belongs to EntityGroup.

Implements samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

10.106.3.2 virtual void samchon::protocol::slave::ParallelSystem::_replyData (std::shared_ptr< Invoke >) [override], [protected], [virtual]

Pre-processor for segmented processes and reporting elapsed time.

[Inherited]

You can realize a pre-processing about the replied Invoke message by overriding the IClient::_replyData().

Archiving a history log, filtering and truncating, let handling the message with a new thread, calculating performance by estimating elapsed time about the Invoke message, and so on. Anything you can define pre-processing works before IProtocol::replyData().

- Archive history log -> cloud service
- Estimating performance -> distributed and parallel processing system

Parameters

invoke	An invoke message to be pre-processed and shifted to IProtocol::replyData().

Reimplemented from samchon::protocol::slave::SlaveSystem.

10.106.3.3 virtual void samchon::protocol::slave::ParallelSystem::replyPieceData (std::shared_ptr< Invoke >, size_t, size_t) [pure virtual]

Handle requested processes which are segmented.

Parameters

invoke	An invoke message containing segmented process requestments.
startIndex	Starting index number of segmentation.

size	Size of segmentation.

Implemented in samchon::protocol::master::ParallelSlaveSystemMediator.

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

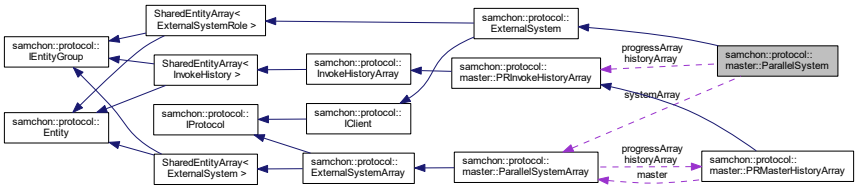
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/slave/ParallelSystem.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/slave/ParallelSystem_slave.cpp

10.107 samchon::protocol::master::ParallelSystem Class Reference

A network driver for a parallel system.

`#include <ParallelSystem.hpp>`

Collaboration diagram for samchon::protocol::master::ParallelSystem:



Public Member Functions

- ParallelSystem ()
  Default Constructor.
- virtual void construct (std::shared_ptr< library::XML >) override
  Construct data of the Entity from an XML object.
- virtual auto toXML () const -> std::shared_ptr< library::XML > override
  Get an XML object represents the EntityGroup.

Protected Member Functions

- virtual auto createChild (std::shared_ptr< library::XML >) -> ExternalSystemRole *override
  Factory method of a child Entity.
- virtual void _replyData (std::shared_ptr< Invoke >) override
  A method for pre-processing replied Invoke message.

Private Member Functions

- void sendPieceData (PRMasterHistory *, std::shared_ptr< Invoke >, size_t, size_t)
  Send a message with segmentation index.

Private Attributes

- ParallelSystemArray * systemArray
  A master that the system is belonged to.
- PRInvokeHistoryArray * historyArray
  A list of history log for reported Invoke messages.
- PRInvokeHistoryArray * progressArray
  A list of invoke messages on progress.
- double performance
  A performance index.

Additional Inherited Members

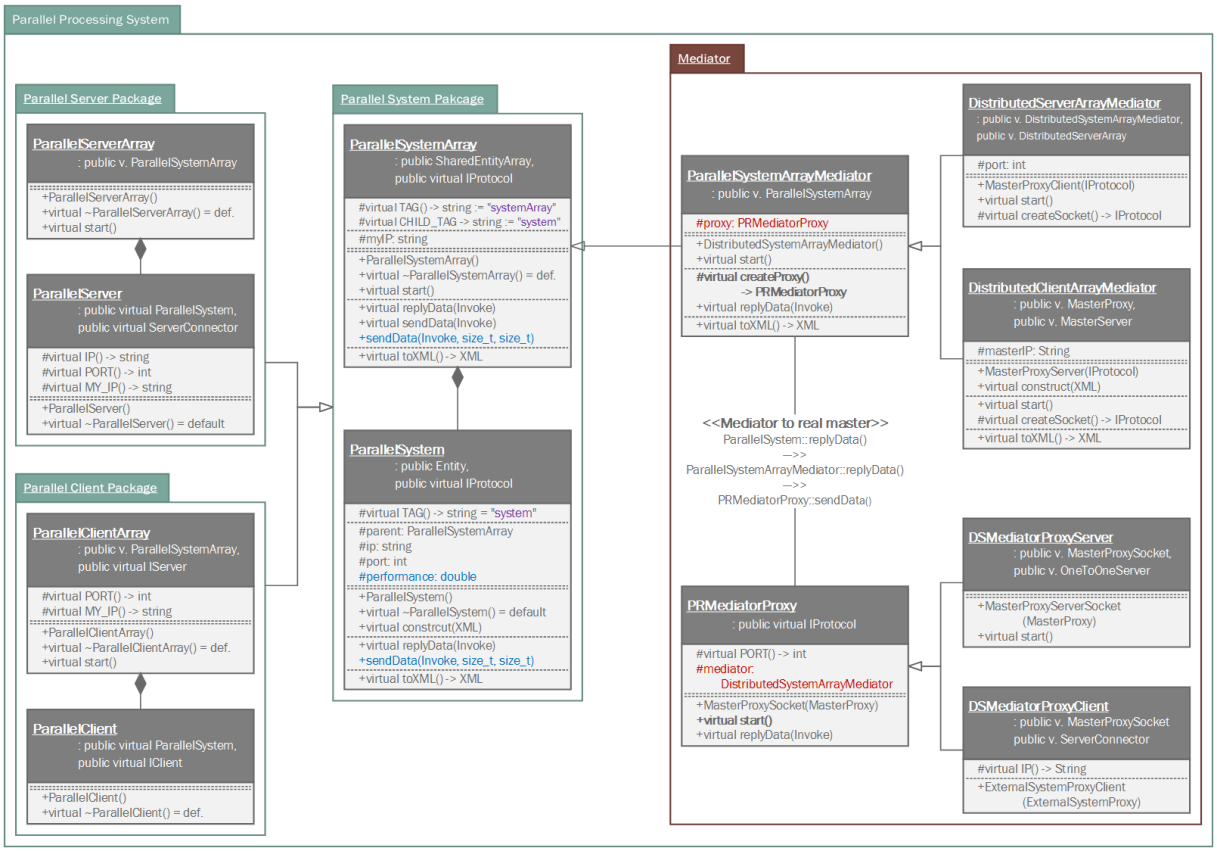
10.107.1 Detailed Description

A network driver for a parallel system.

ParallelSystem class is an ExternalSystem having performance index and history logs of handled Invoke messages with those elapsed times. Those performance index and history logs are used to determine how much segmentation to be allocated.

Each segmentation in requested process (Invoke message) is equivalent. Thus, role of ParallelSystem objects in a ParallelSystemArray are almost same and does not need to specify ExternalSystemRole on ParallelSystem.

- Not a matter to specifying ExternalSystemRole objects to each ParallelSystem. In that case, Invoke messages having segmentation size will be processed by ParallelSystem's own logic and Invoke messages without segmentation size will be handled by ordinary logic of ExternalSystem's own.



[Inherited]

Definition at line 40 of file ParallelSystem.hpp.

10.107.2 Constructor & Destructor Documentation

10.107.2.1 ParallelSystem::ParallelSystem ( )

Default Constructor.

Definition at line 17 of file ParallelSystem.cpp.

References historyArray, performance, and systemArray.

10.107.3 Member Function Documentation

10.107.3.1 void ParallelSystem::construct (std::shared_ptr< library::XML > *xml*) [override], [virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

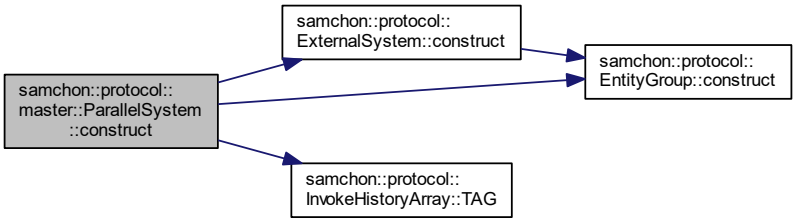
Reimplemented from samchon::protocol::ExternalSystem.

Reimplemented in samchon::protocol::master::ParallelServer.

Definition at line 31 of file ParallelSystem.cpp.

References samchon::protocol::ExternalSystem::construct(), samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::construct(), historyArray, performance, and samchon::protocol::InvokeHistoryArray::TAG().

Here is the call graph for this function:



10.107.3.2 virtual auto samchon::protocol::master::ParallelSystem::createChild (std::shared_ptr< library::XML >) -> ExternalSystemRole *override [override], [protected], [virtual]

Factory method of a child Entity.

EntityGroup::createChild() is a factory method creating a new child Entity which is belonged to the EntityGroup. This method is called by EntityGroup::construct(). The children construction methods Entity::construct() will be called by abstract method of the EntityGroup::construct().

Returns

A new child Entity belongs to EntityGroup.

Implements samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

10.107.3.3 virtual void samchon::protocol::master::ParallelSystem::_replyData (std::shared_ptr< Invoke >) [override], [protected], [virtual]

A method for pre-processing replied Invoke message.

You can realize a pre-processing about the replied Invoke message by overriding the IClient::_replyData().

Archiving a history log, filtering and truncating, let handling the message with a new thread, calculating performance by estimating elapsed time about the Invoke message, and so on. Anything you can define pre-processing works before IProtocol::replyData().

- Archive history log -> cloud service
- Estimating performance -> distributed and parallel processing system

Parameters

invoke	An invoke message to be pre-processed and shifted to IProtocol::replyData().

Reimplemented from samchon::protocol::IClient.

10.107.3.4 void ParallelSystem::sendPieceData (PRMasterHistory *, std::shared_ptr< Invoke >, size_t, size_t) [private]

Send a message with segmentation index.

Parameters

invoke	An invoke message requesting a process.
startIndex	Starting index number of segmentation.
size	Size of segmentation.

Definition at line 73 of file ParallelSystem.cpp.

10.107.3.5 auto ParallelSystem::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

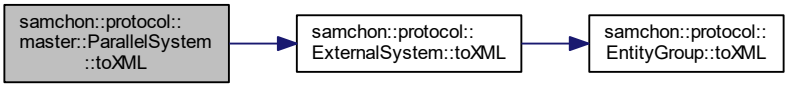
Reimplemented from samchon::protocol::ExternalSystem.

Reimplemented in samchon::protocol::master::ParallelServer.

Definition at line 97 of file ParallelSystem.cpp.

References performance, and samchon::protocol::ExternalSystem::toXML().

Here is the call graph for this function:



10.107.4 Member Data Documentation

10.107.4.1 ParallelSystemArray* samchon::protocol::master::ParallelSystem::systemArray [private]

A master that the system is belonged to.

Definition at line 52 of file ParallelSystem.hpp.

Referenced by ParallelSystem().

10.107.4.2 PRInvokeHistoryArray* samchon::protocol::master::ParallelSystem::historyArray [private]

A list of history log for reported Invoke messages.

Definition at line 57 of file ParallelSystem.hpp.

Referenced by construct(), and ParallelSystem().

10.107.4.3 PRInvokeHistoryArray* samchon::protocol::master::ParallelSystem::progressArray [private]

A list of invoke messages on progress.

Definition at line 62 of file ParallelSystem.hpp.

10.107.4.4 double samchon::protocol::master::ParallelSystem::performance [private]

A performance index.

A performance index z is calculated by normalization calculating reverse number of

whole parallel system's average elapsed times and its z value between the normal distribution.

- X = (

  If a parallel system has no history of handling Invoke message, then set the performance index to 0 as default. The performance index will be re-calcuated whenever segmented process has sent (ParallelSystemArray::sendData(Invoke, size_t)).

  Definition at line 78 of file ParallelSystem.hpp.

  Referenced by construct(), samchon::protocol::master::ParallelSystemArray::notifyEnd(), ParallelSystem(), and toXML().

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

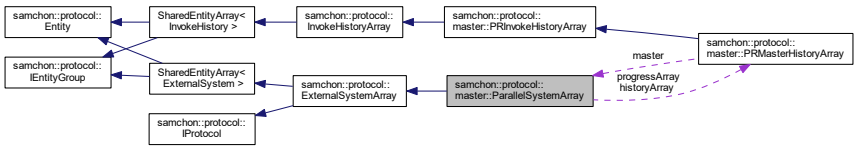
  * D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelSystem.hpp
  * D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelSystem.cpp

10.108 samchon::protocol::master::ParallelSystemArray Class Reference

An array of parallel system drivers.

`#include <ParallelSystemArray.hpp>`

Collaboration diagram for samchon::protocol::master::ParallelSystemArray:



Public Member Functions

- ParallelSystemArray ()
  Default Constructor.
- virtual void sendSegmentData (std::shared_ptr< Invoke >, size_t)
  Send a message with segmentation size.
- virtual void sendPieceData (std::shared_ptr< Invoke >, size_t, size_t)
  Send a message with piece index and size.

Protected Member Functions

- virtual void notifyEnd (PRMasterHistory *)
  Notify end of a master history.

Private Attributes

- std::atomic< size_t > uid
  Sequence of uid allocating for history.
- PRMasterHistoryArray * historyArray
  An array of history have occured.
- PRMasterHistoryArray * progressArray
  An array of history on progress.

10.108.1 Detailed Description

An array of parallel system drivers.

ParallelSystemArray is an ExternalSystemArray and an abstract class containing. and managing parallel system drvers within framework of master. The ParallelSystemArray class allocates segmented processes (Invoke message containing segmentation size) following by each system's performance index.

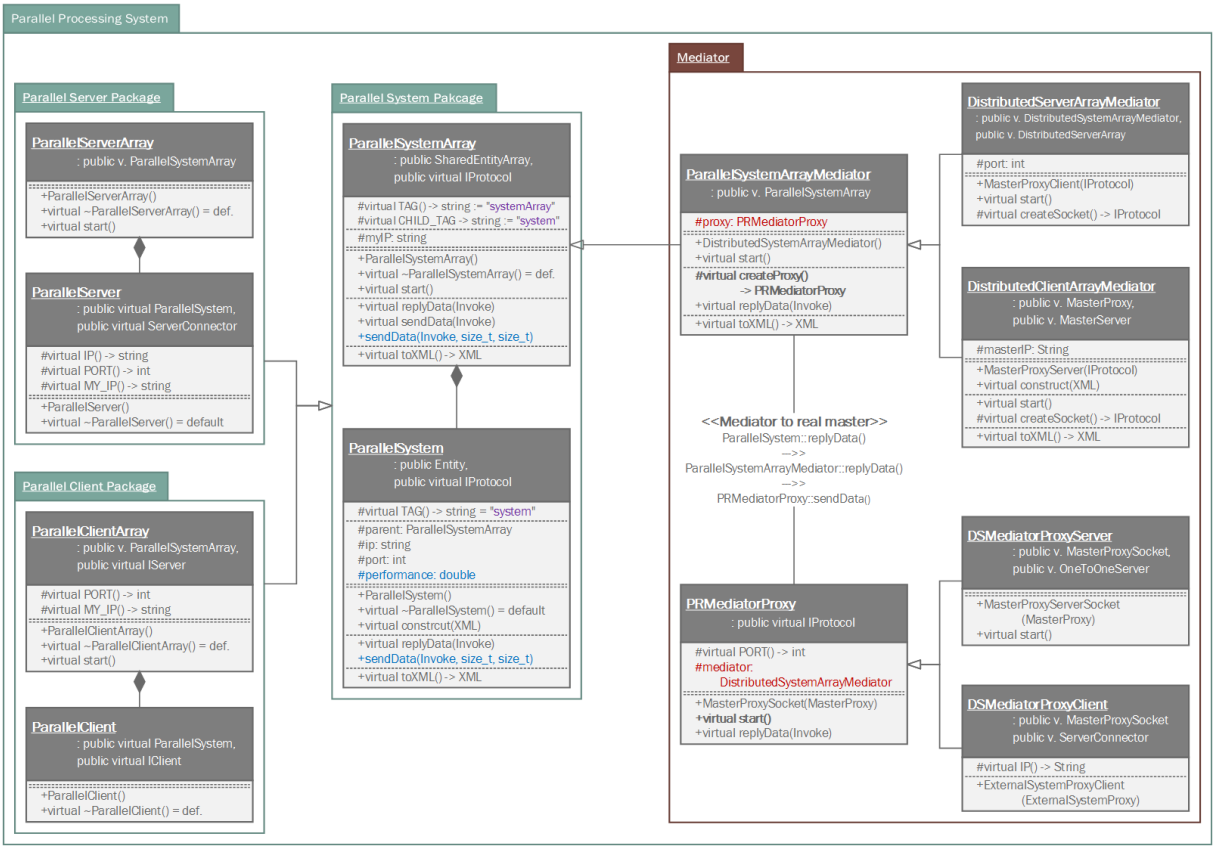
Unlike DistributedSystemArray, ParallelSystemArray does not have such complicate relationships and logics. All segmentation sent by sendData(Invoke, size_t) requires uniform performance. ParallelSystemArray just calculates each system's performance index by elapsing time such easily.

Of course, DistributedSystemArray determines which size of segmentation allocation will be suitable for each system by the performance index

.

Each segmentation in requested process (Invoke message) is equivalent. Thus, role of ParallelSystem objects in a ParallelSystemArray are almost same and does not need to specify ExternalSystemRole on each ParallelSystem.

- Not a matter to specifying ExternalSystemRole objects to each ParallelSystem. In that case, Invoke messages having segmentation size will be processed by ParallelSystemArray's own logic and Invoke messages without segmentation size will be handled by ordinary logic of ExternalSystemArray's own.



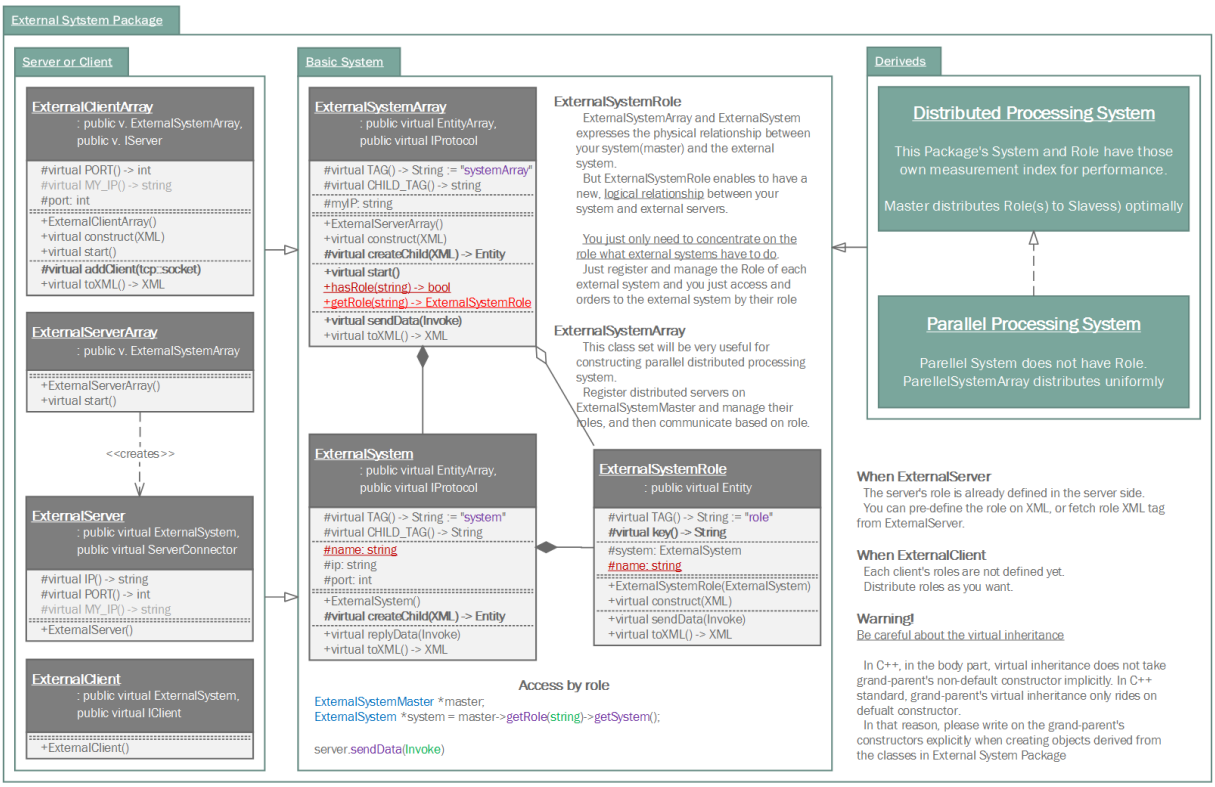
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an External↩SystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which External↩System.

* ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 50 of file ParallelSystemArray.hpp.

10.108.2 Constructor & Destructor Documentation

10.108.2.1 ParallelSystemArray::ParallelSystemArray ( )

Default Constructor.

Definition at line 18 of file ParallelSystemArray.cpp.

10.108.3 Member Function Documentation

10.108.3.1 void ParallelSystemArray::sendSegmentData (std::shared_ptr< Invoke > , size_t) [virtual]

Send a message with segmentation size.

Parameters

invoke	An invoke message requesting a process.
size	Size of segmentation.

Definition at line 37 of file ParallelSystemArray.cpp.

10.108.3.2 void ParallelSystemArray::sendPieceData (std::shared_ptr< Invoke > , size_t , size_t) [virtual]

Send a message with piece index and size.

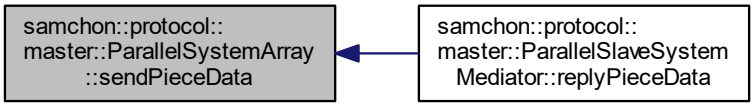
Parameters

invoke	An invoke message requesting a process.
index	Starting index number of segmentation.
size	Size of segmentation.

Definition at line 41 of file ParallelSystemArray.cpp.

Referenced by samchon::protocol::master::ParallelSlaveSystemMediator::replyPieceData().

Here is the caller graph for this function:



10.108.3.3 void ParallelSystemArray::notifyEnd (PRMasterHistory * *masterHistory*) [protected], [virtual]

Notify end of a master history.

Estimates performance of each system by a master history containing their elapsed times.

The estimation is done by normalizing performances of children systems. The estimation of performance is divided by averagae elapsed time of each segmentation.

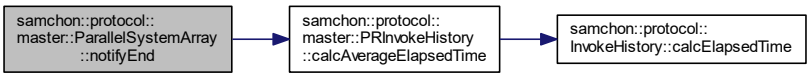
Reimplemented in samchon::protocol::master::ParallelSystemArrayMediator.

Definition at line 78 of file ParallelSystemArray.cpp.

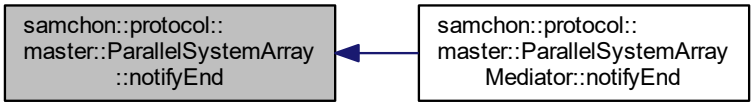
References samchon::protocol::master::PRInvokeHistory::calcAverageElapsedTime(), samchon::protocol::master::PRMasterHistory::historyArray, samchon::protocol::master::ParallelSystem::performance, and samchon::protocol::master::PRInvokeHistory::system.

Referenced by samchon::protocol::master::ParallelSystemArrayMediator::notifyEnd().

Here is the call graph for this function:



Here is the caller graph for this function:



10.108.4 Member Data Documentation

10.108.4.1 std::atomic<size_t> samchon::protocol::master::ParallelSystemArray::uid [private]

Sequence of uid allocating for history.
Definition at line 62 of file ParallelSystemArray.hpp.

10.108.4.2 PRMasterHistoryArray* samchon::protocol::master::ParallelSystemArray::historyArray [private]

An array of history have occured.
Definition at line 67 of file ParallelSystemArray.hpp.

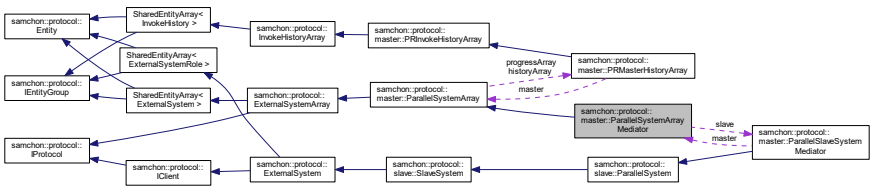
10.108.4.3 PRMasterHistoryArray* samchon::protocol::master::ParallelSystemArray::progressArray [private]

An array of history on progress.
Definition at line 72 of file ParallelSystemArray.hpp.
The documentation for this class was generated from the following files:

* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelSystemArray.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelSystemArray.cpp

10.109 samchon::protocol::master::ParallelSystemArrayMediator Class Reference

A mediator of parallel system between master and slaves.
`#include <ParallelSystemArrayMediator.hpp>`
Collaboration diagram for samchon::protocol::master::ParallelSystemArrayMediator:



Public Member Functions

* ParallelSystemArrayMediator ()
  Default Constructor.
* virtual void construct (std::shared_ptr< library::XML >) override
  Construct data of the Entity from an XML object.
* virtual void start () override
  Start interaction.
* virtual auto toXML () const -> std::shared_ptr< library::XML > override
  Get an XML object represents the EntityGroup.

Protected Member Functions

* virtual void notifyEnd (PRMasterHistory *)
  Notify end of a master history.

Protected Attributes

* ParallelSlaveSystemMediator * slave
  A slave system for mediation.

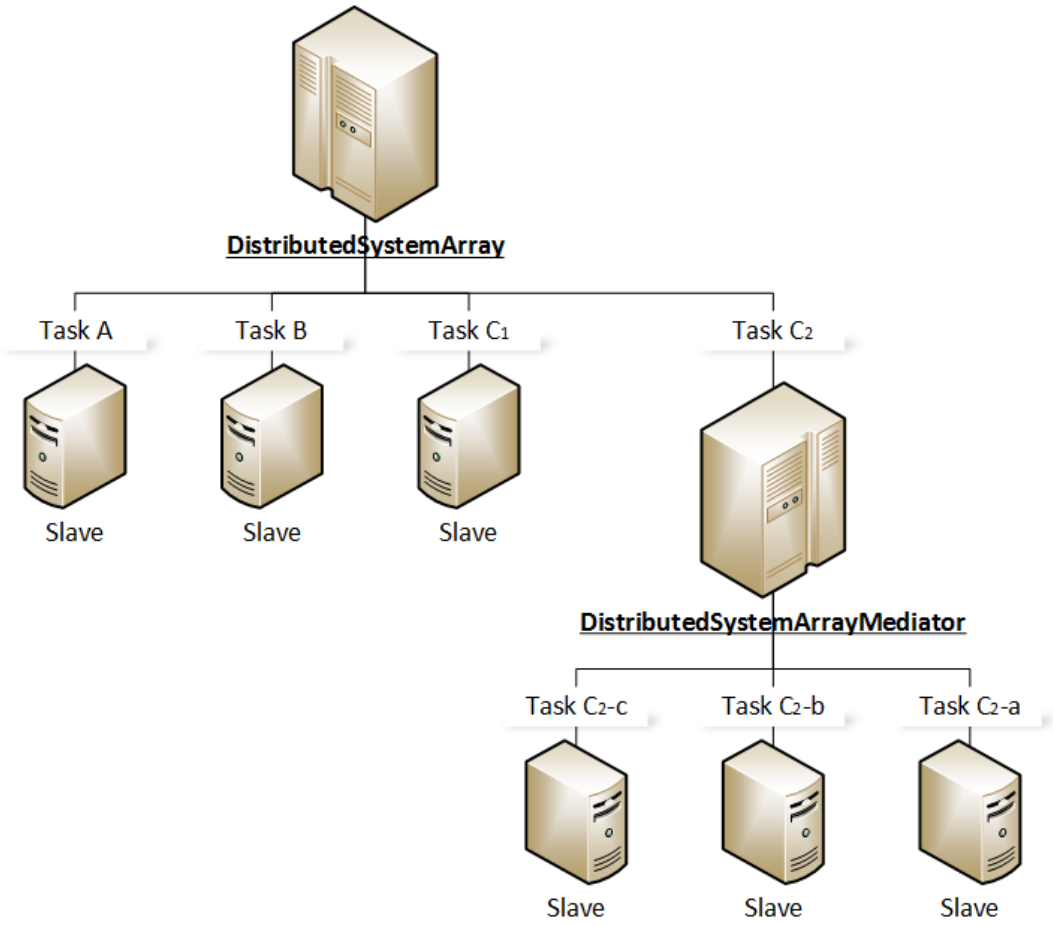
10.109.1 Detailed Description

A mediator of parallel system between master and slaves.

ParallelSystemArrayMediator class is used to realizing a mediator system of tree-structured parallel processing system. ParallelSystemArrayMediator intermediates between origin-master(ParallelSystem↩Array) and its slaves(slave::ParallelSystem).

ParallelSystemArrayMediator is a master(ParallelSystemArray) within the framework of slaves systems (slave::ParallelSystem objects) and also can be a slave system (slave::ParallelSystem) within framework of its master(ParallelSystemArray) by having ParallelSlaveSystemMediator.

Todo Sends PRMasterHistory to its origin master.



[Inherited]

ParallelSystemArray is an ExternalSystemArray and an abstract class containing. and managing parallel system drvers within framework of master. The ParallelSystemArray class allocates segmented processes (Invoke message containing segmentation size) following by each system's performance index.

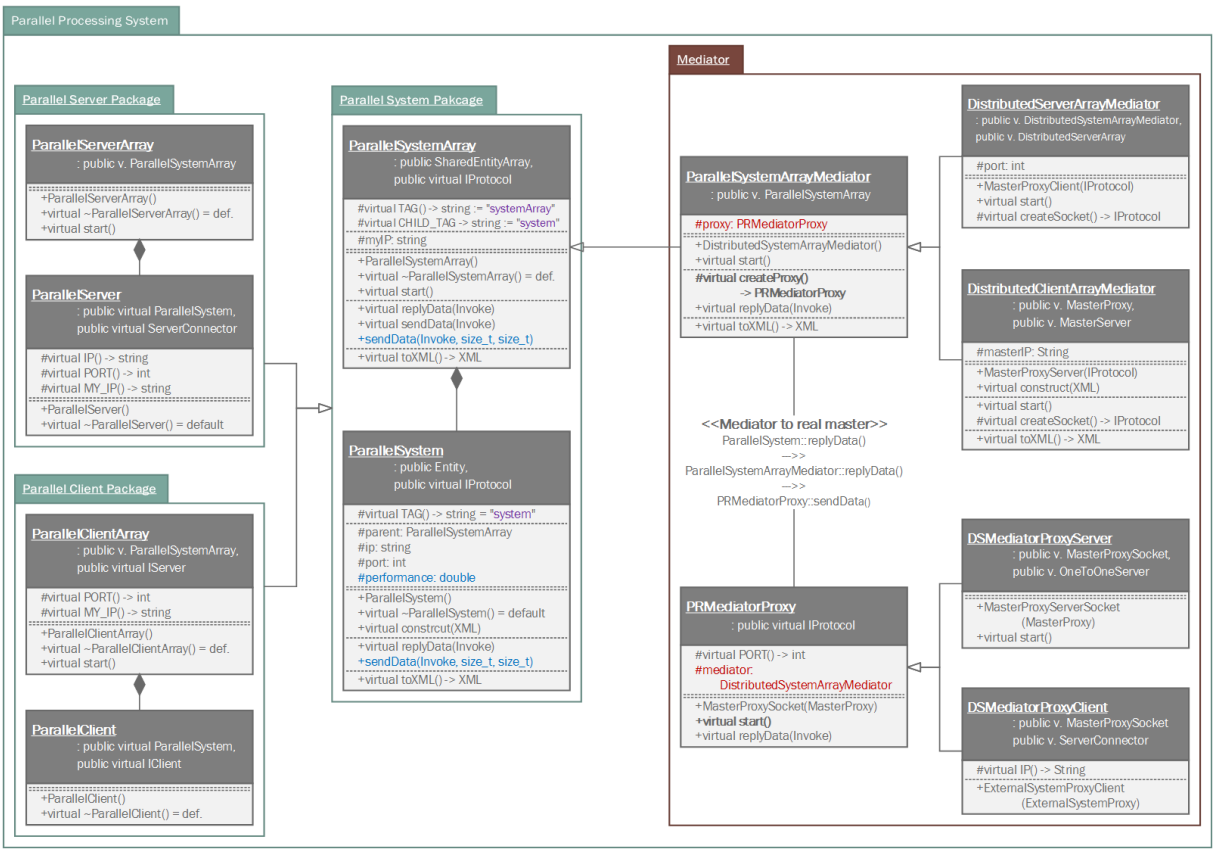
Unlike DistributedSystemArray, ParallelSystemArray does not have such complicate relationships and logics. All segmentation sent by sendData(Invoke, size_t) requires uniform performance. Parallel↩SystemArray just calculates each system's performance index by elapsing time such easily.

Of course, DistributedSystemArray determines which size of segmentation allocation will be suitable for each system by the performance index

.

Each segmentation in requested process (Invoke message) is equivalent. Thus, role of ParallelSystem objects in a ParallelSystemArray are almost same and does not need to specify ExternalSystemRole on each ParallelSystem.

* Not a matter to specifying ExternalSystemRole objects to each ParallelSystem. In that case, Invoke messages having segmentation size will be processed by ParallelSystemArray's own logic and Invoke messages without segmentation size will be handled by ordinary logic of ExternalSystem↩ Array's own.



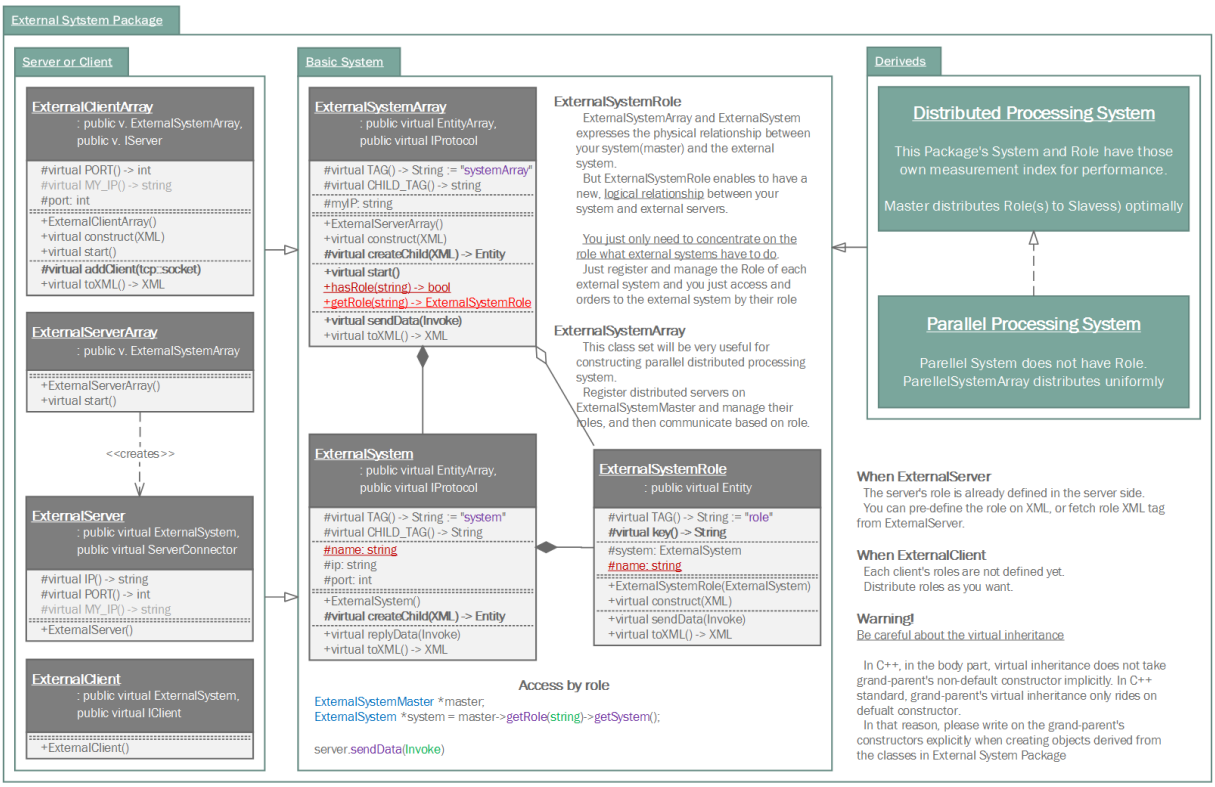
[Inherited]

ExternalSystemArray is an abstract class containing and managing external system drivers.

Also, ExternalSystemArray can access to ExternalSystemRole(s) directly. With the method, you can use an ExternalSystemRole as "logical proxy" of an ExternalSystem. Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an External↩ SystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, the ExternalSystemRole acted a role of proxy.

It's called as "Proxy pattern". With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which External↩ System.

* ExternalSystemArray::getRole("something")->sendData(invoke);



See also

samchon::protocol

Author

Jeongho Nam

Definition at line 36 of file ParallelSystemArrayMediator.hpp.

10.109.2 Constructor & Destructor Documentation

10.109.2.1 ParallelSystemArrayMediator::ParallelSystemArrayMediator ( )

Default Constructor.

Definition at line 19 of file ParallelSystemArrayMediator.cpp.

References slave.

10.109.3 Member Function Documentation

10.109.3.1 void ParallelSystemArrayMediator::construct (std::shared_ptr< library::XML > *xml*) [override],[virtual]

Construct data of the Entity from an XML object.

Constructs the EntityGroup's own member variables only from the input XML object.

Do not consider about constructing children Entity objects' data in EntityGroup::construct(). Those children Entity objects' data will constructed by their own construct() method. Even insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Constructs only data of EntityGroup's own.

[Inherited]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

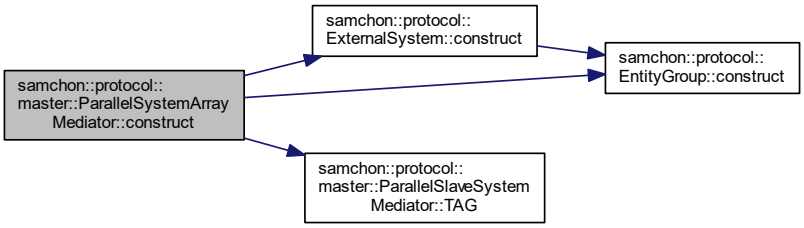
Reimplemented from samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

Reimplemented in samchon::protocol::master::ParallelClientArrayMediator.

Definition at line 29 of file ParallelSystemArrayMediator.cpp.

References samchon::protocol::ExternalSystem::construct(), samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::construct(), slave, and samchon::protocol::master::ParallelSlaveSystemMediator::TAG().

Here is the call graph for this function:



10.109.3.2 void ParallelSystemArrayMediator::start ( ) [override], [virtual]

Start interaction.

An abstract method starting interaction with external systems.

If external systems are servers, starts connection to them, else clients, opens a server and accepts the external systems. You can addict your own procudures of starting drivers, but if you directly override method of abstract ExternalSystemArray, be careful about virtual inheritance.

* DistributedClientArray::start() calls DistributedSystemArray::start() and ExternalClientArray:start()

Implements samchon::protocol::ExternalSystemArray.

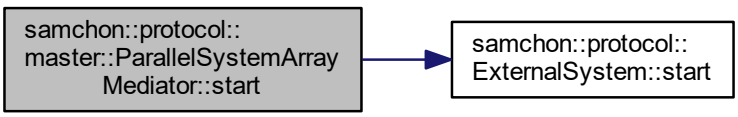
Reimplemented in samchon::protocol::master::ParallelClientArrayMediator, and samchon::protocol::master::ParallelServerArrayMediator.

Definition at line 43 of file ParallelSystemArrayMediator.cpp.

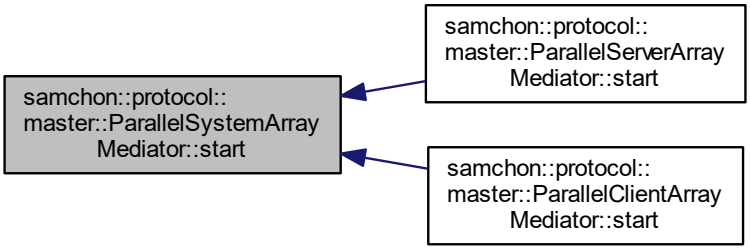
References slave, and samchon::protocol::ExternalSystem::start().

Referenced by samchon::protocol::master::ParallelServerArrayMediator::start(), and samchon::protocol::master::ParallelClientArrayMediator::start().

Here is the call graph for this function:



Here is the caller graph for this function:



10.109.3.3 void ParallelSystemArrayMediator::notifyEnd (PRMasterHistory * *masterHistory*) [protected],[virtual]

Notify end of a master history.

Estimates performance of each system by a master history containing their elapsed times.

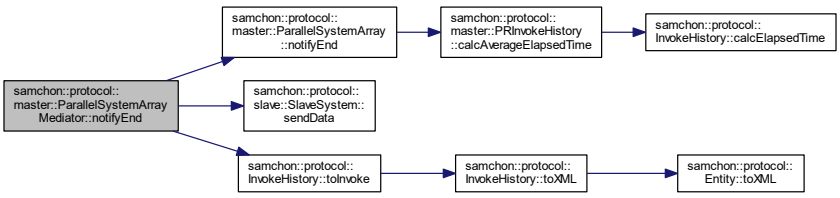
The estimation is done by normalizing performances of children systems. The estimation of performance is divided by averagae elapsed time of each segmentation.

Reimplemented from samchon::protocol::master::ParallelSystemArray.

Definition at line 59 of file ParallelSystemArrayMediator.cpp.

References samchon::protocol::master::ParallelSystemArray::notifyEnd(), samchon::protocol::slave::↩ SlaveSystem::sendData(), slave, and samchon::protocol::InvokeHistory::toInvoke().

Here is the call graph for this function:



10.109.3.4 auto ParallelSystemArrayMediator::toXML ( ) const -> std::shared_ptr<library::XML> [override],[virtual]

Get an XML object represents the EntityGroup.

Archives the EntityGroup's own member variables only to the returned XML object.

Do not consider about archiving children Entity objects' data in EntityGroup::toXML(). Those children Entity objects will converted to XML object by their own toXML() method. The insertion of XML objects representing children are done by abstract method of EntityGroup::toXML().

Archives only data of EntityGroup's own.

[Inherited]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

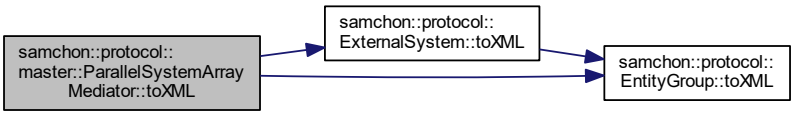
Reimplemented from samchon::protocol::EntityGroup< _Container, _ETy, _Ty >.

Reimplemented in samchon::protocol::master::ParallelClientArrayMediator.

Definition at line 71 of file ParallelSystemArrayMediator.cpp.

References slave, samchon::protocol::ExternalSystem::toXML(), and samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::toXML().

Here is the call graph for this function:



10.109.4 Member Data Documentation

10.109.4.1 ParallelSlaveSystemMediator* samchon::protocol::master::ParallelSystemArrayMediator::slave [protected]

A slave system for mediation.

Definition at line 46 of file ParallelSystemArrayMediator.hpp.

Referenced by samchon::protocol::master::ParallelClientArrayMediator::construct(), construct(), notifyEnd(), samchon::protocol::master::ParallelClientArrayMediator::ParallelClientArrayMediator(), samchon::protocol::master::ParallelServerArrayMediator::ParallelServerArrayMediator(), ParallelSystemArrayMediator(), start(), samchon::protocol::master::ParallelClientArrayMediator::toXML(), and toXML().

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

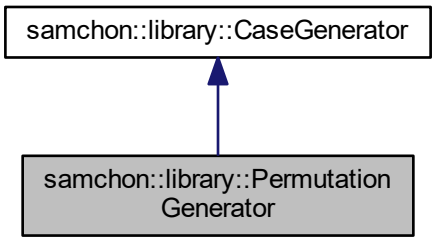
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelSystemArrayMediator.hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/ParallelSystemArrayMediator.cpp

10.110 samchon::library::PermutationGenerator Class Reference

A permutation case generator.

```
#include <PermutationGenerator.hpp>
```

Collaboration diagram for samchon::library::PermutationGenerator:



Public Member Functions

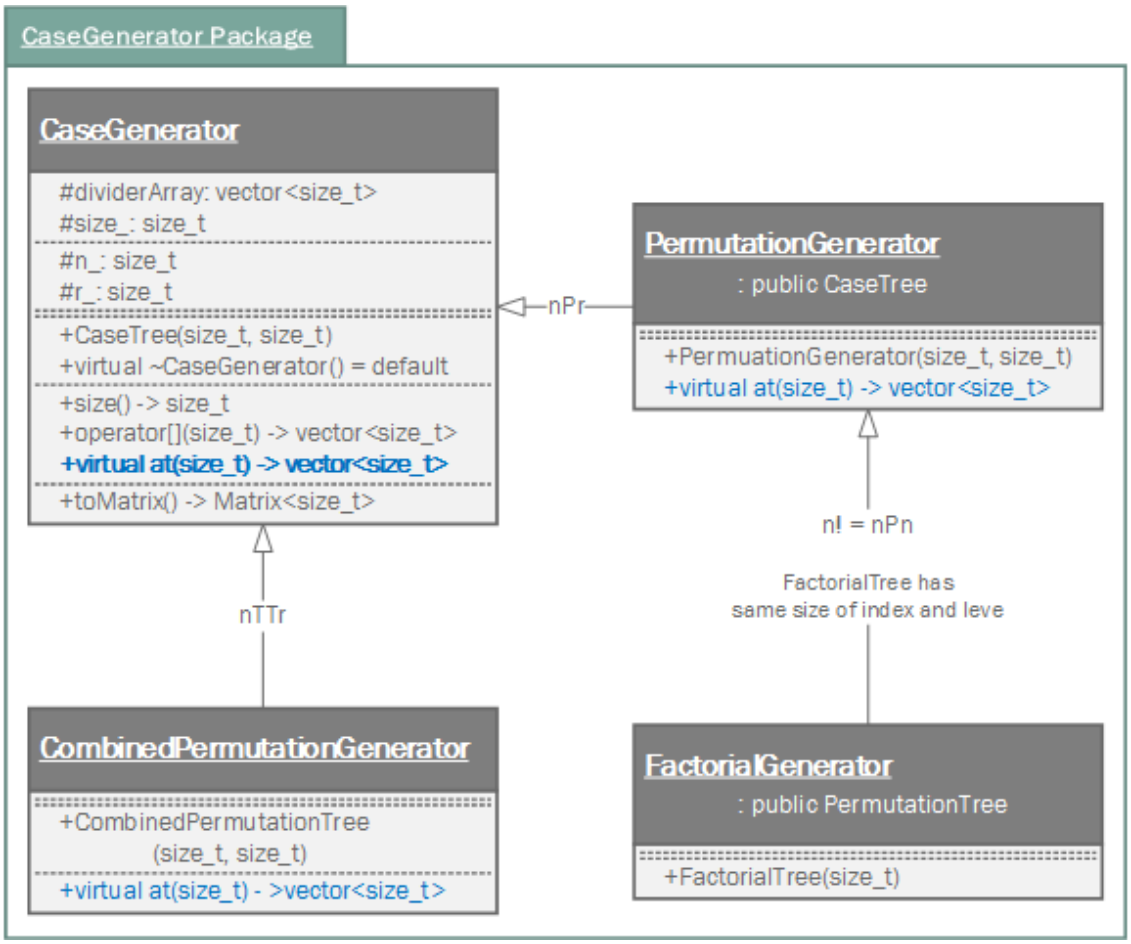
* PermutationGenerator (size_t n, size_t r)
  Construct from size of N and R.
* virtual auto operator[ ] (size_t) const -> std::vector< size_t > override
  Get x'th case.

Additional Inherited Members

10.110.1 Detailed Description

A permutation case generator.

nPr



See also

samchon::library

Author

Jeongho Nam

Definition at line 21 of file PermutationGenerator.hpp.

10.110.2 Constructor & Destructor Documentation

10.110.2.1 PermutationGenerator::PermutationGenerator (size_t *n*, size_t *r*)

Construct from size of N and R.

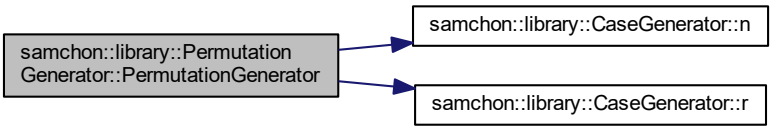
Parameters

n	Size of candidates
r	Size of elements of each case

Definition at line 6 of file PermutationGenerator.cpp.

References samchon::library::CaseGenerator::n(), samchon::library::CaseGenerator::r(), and samchon↩ ::library::CaseGenerator::size_.

Here is the call graph for this function:



10.110.3 Member Function Documentation

10.110.3.1 auto PermutationGenerator::operator[] (size_t) const -> std::vector< size_t > [override], [virtual]

Get x'th case.

Returns

The row of the x'th in combined permuation case

Implements samchon::library::CaseGenerator.

Definition at line 17 of file PermutationGenerator.cpp.

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

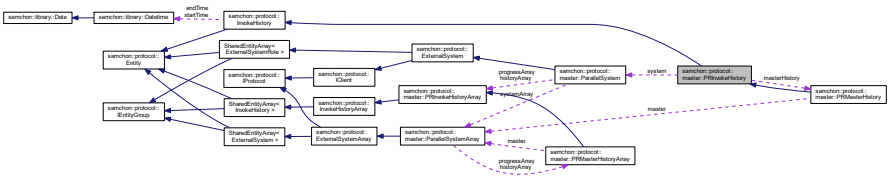
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/PermutationGenerator.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/PermutationGenerator.cpp

10.111 samchon::protocol::master::PRInvokeHistory Class Reference

A reported history of an Invoke message.

`#include <PRInvokeHistory.hpp>`

Collaboration diagram for samchon::protocol::master::PRInvokeHistory:



Public Member Functions

* PRInvokeHistory ()
  Default Constructor.
* PRInvokeHistory (PRMasterHistory *, ParallelSystem *, std::shared_ptr< Invoke >)
  Default Constructor.
* virtual void construct (std::shared_ptr< library::XML >) override
  Construct data of the Entity from an XML object.
* virtual void notifyEnd () override
  Notify end of the process.
* auto getIndex () const -> size_t
  Get index.
* auto getSize () const -> size_t
  Get size.
* auto calcAverageElapsedTime () const -> double
  Calculate average of elapsed time for each segmentation.
* virtual auto toXML () const -> std::shared_ptr< library::XML > override
  Get an XML object represents the Entity.

Public Attributes

- PRMasterHistory * masterHistory
  A history of master.
- ParallelSystem * system
  A system the history is belonged to.
- size_t index
  Start index.
- size_t size
  The Size.

Additional Inherited Members

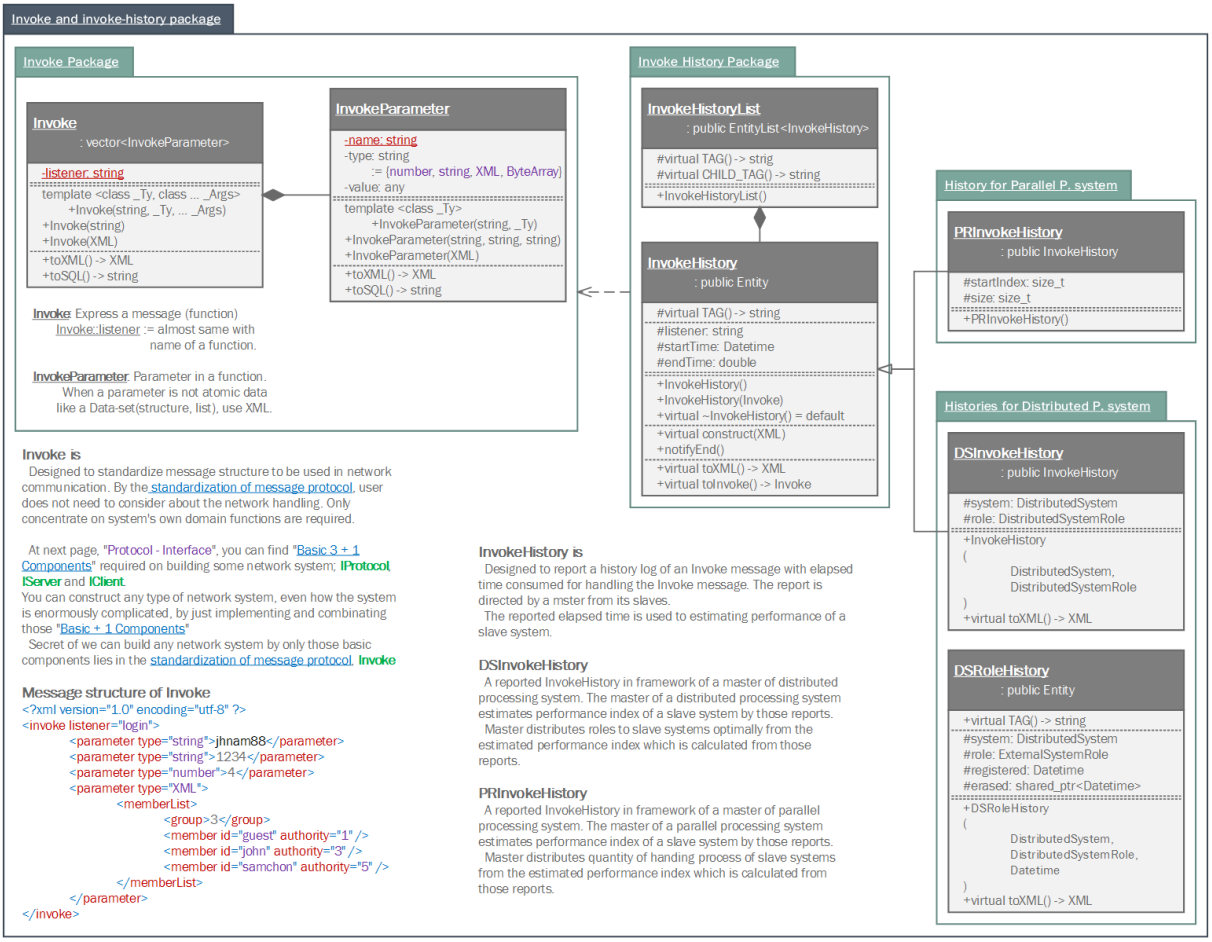
10.111.1 Detailed Description

A reported history of an Invoke message.

[Inherited]

InvokeHistory is a class for reporting history log of an Invoke message with elapsed time from a slave to its master.

With the elapsed time, consumed time for a process of handling the Invoke message, InvokeHistory is reported to the master. The master utilizies the elapsed time to estimating performances of each slave system. With the estimated performan index, master retrives the optimal solution of distributing processes.



See also

samchon::protocol
samchon::protocol::master
samchon::protocol::slave

Author

Jeongho Nam

Definition at line 23 of file PRInvokeHistory.hpp.

10.111.2 Constructor & Destructor Documentation

10.111.2.1 PRInvokeHistory::PRInvokeHistory ( )

Default Constructor.

Definition at line 17 of file PRInvokeHistory.cpp.

10.111.2.2 samchon::protocol::master::PRInvokeHistory::PRInvokeHistory (PRMasterHistory *, ParallelSystem *, std::shared_ptr< Invoke >)

Default Constructor.

Parameters

masterHistory	A master history that the segment history is belonged to.
system	A system that the history is belonged to.
invoke	An invoke message that the history must represent.

10.111.3 Member Function Documentation

10.111.3.1 void PRInvokeHistory::construct (std::shared_ptr< library::XML >) [override], [virtual]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

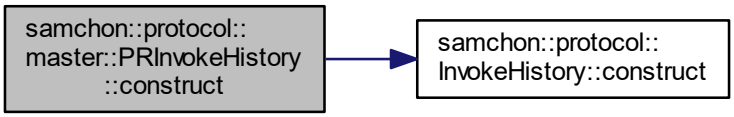
xml	An xml used to construct data of entity

Reimplemented from samchon::protocol::InvokeHistory.

Definition at line 30 of file PRInvokeHistory.cpp.

References samchon::protocol::InvokeHistory::construct(), and index.

Here is the call graph for this function:



10.111.3.2 void PRInvokeHistory::notifyEnd ( ) [override],[virtual]

Notify end of the process.

Notifies end of a process handling the matched Invoke message to InvokeHistory.

InvokeHistory archives the end datetime and calculates elapsed time as nanoseconds.

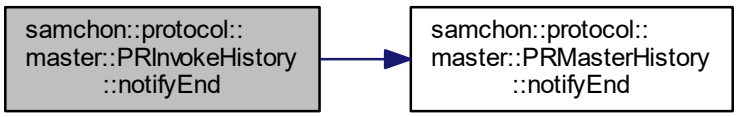
Reimplemented from samchon::protocol::InvokeHistory.

Reimplemented in samchon::protocol::master::PRMasterHistory.

Definition at line 38 of file PRInvokeHistory.cpp.

References samchon::protocol::master::PRMasterHistory::notifyEnd().

Here is the call graph for this function:



10.111.3.3 auto PRInvokeHistory::getIndex ( ) const -> size_t [inline]

Get index.

Definition at line 46 of file PRInvokeHistory.cpp.

References index.

10.111.3.4 auto PRInvokeHistory::getSize ( ) const -> size_t

Get size.

Definition at line 50 of file PRInvokeHistory.cpp.

References size.

10.111.3.5 auto PRInvokeHistory::calcAverageElapsedTime ( ) const -> double

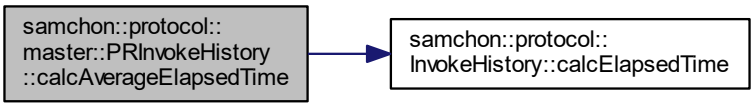
Calculate average of elapsed time for each segmentation.

Definition at line 55 of file PRInvokeHistory.cpp.

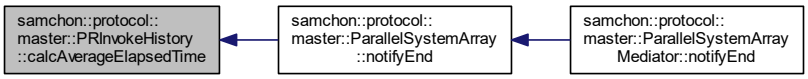
References samchon::protocol::InvokeHistory::calcElapsedTime().

Referenced by samchon::protocol::master::ParallelSystemArray::notifyEnd().

Here is the call graph for this function:



Here is the caller graph for this function:



10.111.3.6 auto PRInvokeHistory::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

An XML object representing the Entity.

Reimplemented from samchon::protocol::InvokeHistory.

Definition at line 63 of file PRInvokeHistory.cpp.

References index, and samchon::protocol::InvokeHistory::toXML().

Here is the call graph for this function:

10.111.4 Member Data Documentation

10.111.4.1 PRMasterHistory* samchon::protocol::master::PRInvokeHistory::masterHistory

A history of master.

Definition at line 36 of file PRInvokeHistory.hpp.

10.111.4.2 ParallelSystem* samchon::protocol::master::PRInvokeHistory::system

A system the history is belonged to.

Definition at line 41 of file PRInvokeHistory.hpp.

Referenced by samchon::protocol::master::ParallelSystemArray::notifyEnd().

10.111.4.3 size_t samchon::protocol::master::PRInvokeHistory::index

Start index.

Definition at line 46 of file PRInvokeHistory.hpp.

Referenced by construct(), getIndex(), and toXML().

10.111.4.4 size_t samchon::protocol::master::PRInvokeHistory::size

The Size.

Definition at line 51 of file PRInvokeHistory.hpp.

Referenced by getSize().

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

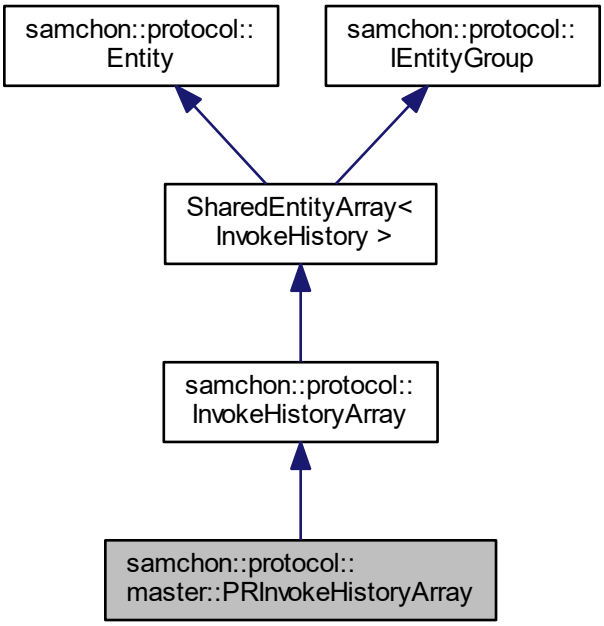
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/PRInvokeHistory.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/PRInvokeHistory.cpp

10.112 samchon::protocol::master::PRInvokeHistoryArray Class Reference

An array of PRInvokeHistory.

`#include <PRInvokeHistoryArray.hpp>`

Collaboration diagram for samchon::protocol::master::PRInvokeHistoryArray:



Public Member Functions

* PRInvokeHistoryArray ()
  Default Constructor.

Protected Member Functions

* virtual auto createChild (std::shared_ptr< library::XML >) -> InvokeHistory *override
  Factory method of a child Entity.

10.112.1 Detailed Description

An array of PRInvokeHistory.

Author

Jeongho Nam

Definition at line 18 of file PRInvokeHistoryArray.hpp.

10.112.2 Constructor & Destructor Documentation

10.112.2.1 PRInvokeHistoryArray::PRInvokeHistoryArray ( )

Default Constructor.

Definition at line 8 of file PRInvokeHistoryArray.cpp.

10.112.3 Member Function Documentation

10.112.3.1 auto PRInvokeHistoryArray::createChild (std::shared_ptr< library::XML >) -> InvokeHistory *override [override], [protected], [virtual]

Factory method of a child Entity.

EntityGroup::createChild() is a factory method creating a new child Entity which is belonged to the EntityGroup. This method is called by EntityGroup::construct(). The children construction methods Entity::construct() will be called by abstract method of the EntityGroup::construct().

Returns

A new child Entity belongs to EntityGroup.

Reimplemented from samchon::protocol::InvokeHistoryArray.

Definition at line 12 of file PRInvokeHistoryArray.cpp.

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

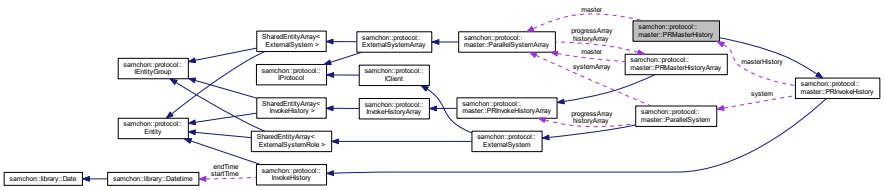
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/PRInvokeHistoryArray.↩ hpp
- D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/PRInvokeHistoryArray.↩ cpp

10.113 samchon::protocol::master::PRMasterHistory Class Reference

A history log of an Invoke message on a master.

`#include <PRMasterHistory.hpp>`

Collaboration diagram for samchon::protocol::master::PRMasterHistory:



Public Member Functions

- PRMasterHistory (PRMasterHistoryArray *, std::shared_ptr< Invoke >, size_t, size_t)
  Construct from master and invoke message.
- virtual void notifyEnd () override
  Notify end of the process.

Protected Attributes

* ParallelSystemArray * master
  A master the history is belonged to.
* library::CriticalVector< PRInvokeHistory * > historyArray
  An array of histories which are generated in each system.
* std::atomic< size_t > completed
  Completed count.

Additional Inherited Members

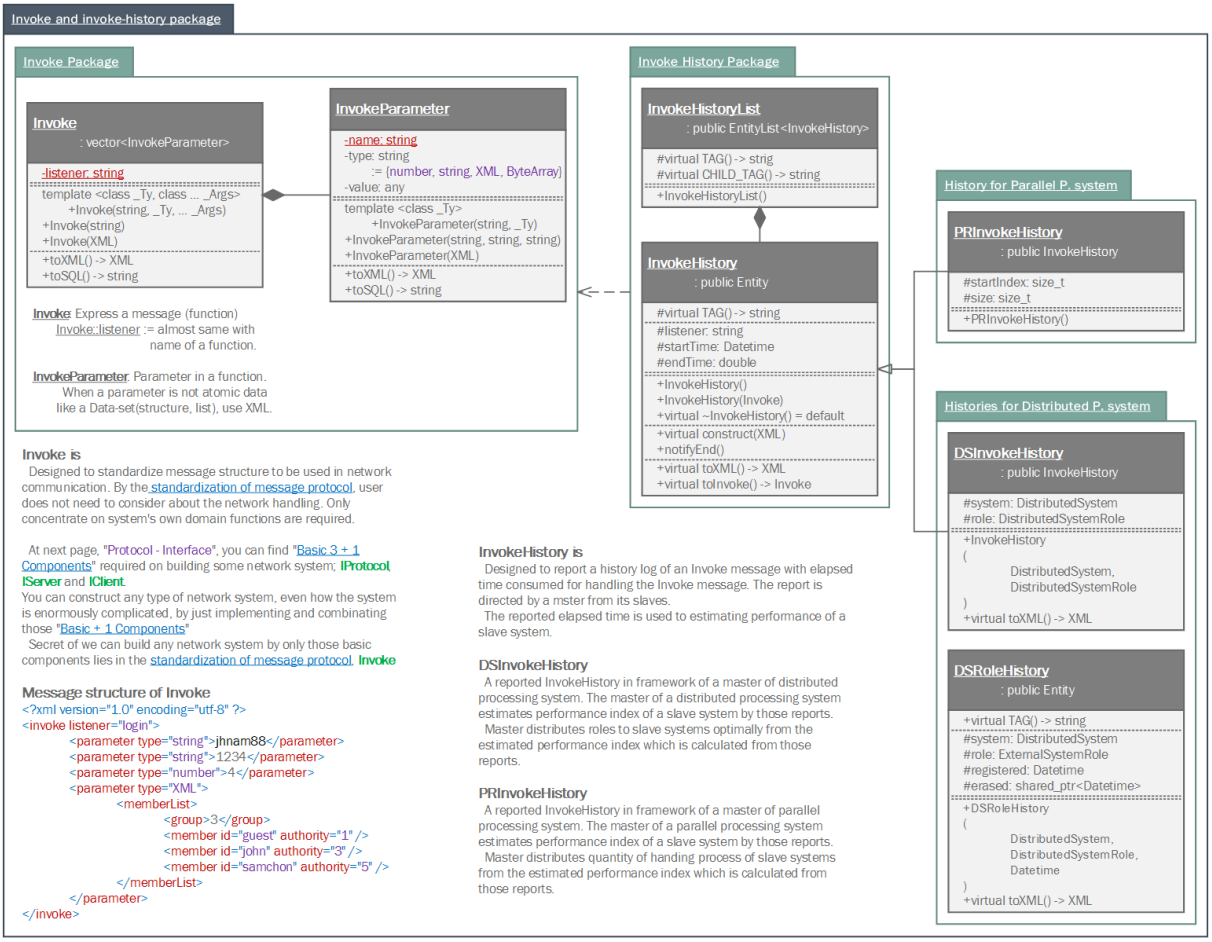
10.113.1 Detailed Description

A history log of an Invoke message on a master.

[Inherited]

InvokeHistory is a class for reporting history log of an Invoke message with elapsed time from a slave to its master.

With the elapsed time, consumed time for a process of handling the Invoke message, InvokeHistory is reported to the master. The master utilizies the elapsed time to estimating performances of each slave system. With the estimated performan index, master retrives the optimal solution of distributing processes.



See also

samchon::protocol
samchon::protocol::master
samchon::protocol::slave

Author

Jeongho Nam
Jeongho Nam

Definition at line 28 of file PRMasterHistory.hpp.

10.113.2 Constructor & Destructor Documentation

10.113.2.1 PRMasterHistory::PRMasterHistory (PRMasterHistoryArray *, std::shared_ptr< Invoke >, size_t, size_t)

Construct from master and invoke message.

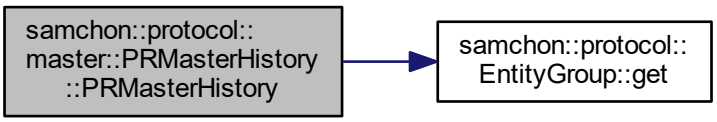
Parameters

master	An array of master history.
invoke	An invoke message to send.

Definition at line 16 of file PRMasterHistory.cpp.

References samchon::protocol::EntityGroup< _Container, _ETy, _Ty >::get(), and samchon::protocol::master::PRMasterHistoryArray::master.

Here is the call graph for this function:



10.113.3 Member Function Documentation

10.113.3.1 void PRMasterHistory::notifyEnd ( ) [override],[virtual]

Notify end of the process.

Notifies end of a process handling the matched Invoke message to InvokeHistory.

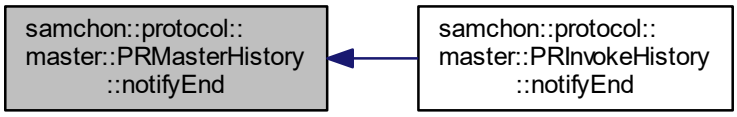
InvokeHistory archives the end datetime and calculates elapsed time as nanoseconds.

Reimplemented from samchon::protocol::master::PRInvokeHistory.

Definition at line 32 of file PRMasterHistory.cpp.

Referenced by samchon::protocol::master::PRInvokeHistory::notifyEnd().

Here is the caller graph for this function:



10.113.4 Member Data Documentation

10.113.4.1 ParallelSystemArray* samchon::protocol::master::PRMasterHistory::master [protected]

A master the history is belonged to.

Definition at line 39 of file PRMasterHistory.hpp.

10.113.4.2 library::CriticalVector<PRInvokeHistory*> samchon::protocol::master::PRMasterHistory↩ ::historyArray [protected]

An array of histories which are generated in each system.

Definition at line 44 of file PRMasterHistory.hpp.

Referenced by samchon::protocol::master::ParallelSystemArray::notifyEnd().

10.113.4.3 std::atomic<size_t> samchon::protocol::master::PRMasterHistory::completed [protected]

Completed count.

Definition at line 49 of file PRMasterHistory.hpp.

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

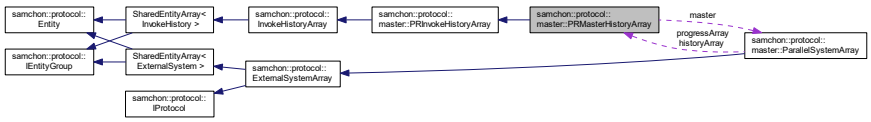
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/PRMasterHistory.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/PRMasterHistory.cpp

10.114 samchon::protocol::master::PRMasterHistoryArray Class Reference

An array of invoke histories of master.

`#include <PRMasterHistoryArray.hpp>`

Collaboration diagram for samchon::protocol::master::PRMasterHistoryArray:



Public Member Functions

* PRMasterHistoryArray (ParallelSystemArray *)
  Construct from a master.

Protected Attributes

* ParallelSystemArray * master
  A master the array of histories is belonged to.

Additional Inherited Members

10.114.1 Detailed Description

An array of invoke histories of master.

Author

Jeongho Nam

Definition at line 22 of file PRMasterHistoryArray.hpp.

10.114.2 Constructor & Destructor Documentation

10.114.2.1 PRMasterHistoryArray::PRMasterHistoryArray (ParallelSystemArray * *master*)

Construct from a master.

Definition at line 11 of file PRMasterHistoryArray.cpp.

References master.

10.114.3 Member Data Documentation

10.114.3.1 ParallelSystemArray* samchon::protocol::master::PRMasterHistoryArray::master [protected]

A master the array of histories is belonged to.

Definition at line 33 of file PRMasterHistoryArray.hpp.

Referenced by samchon::protocol::master::PRMasterHistory::PRMasterHistory(), and PRMaster↩HistoryArray().

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

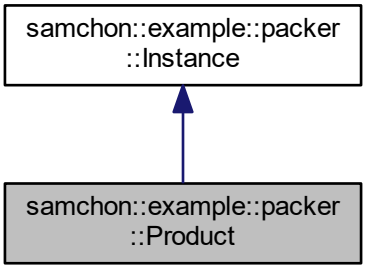
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/PRMasterHistoryArray.↩hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/master/PRMasterHistoryArray.↩cpp

10.115 samchon::example::packer::Product Class Reference

A product, merchandise.

`#include <Product.hpp>`

Collaboration diagram for samchon::example::packer::Product:



Public Member Functions

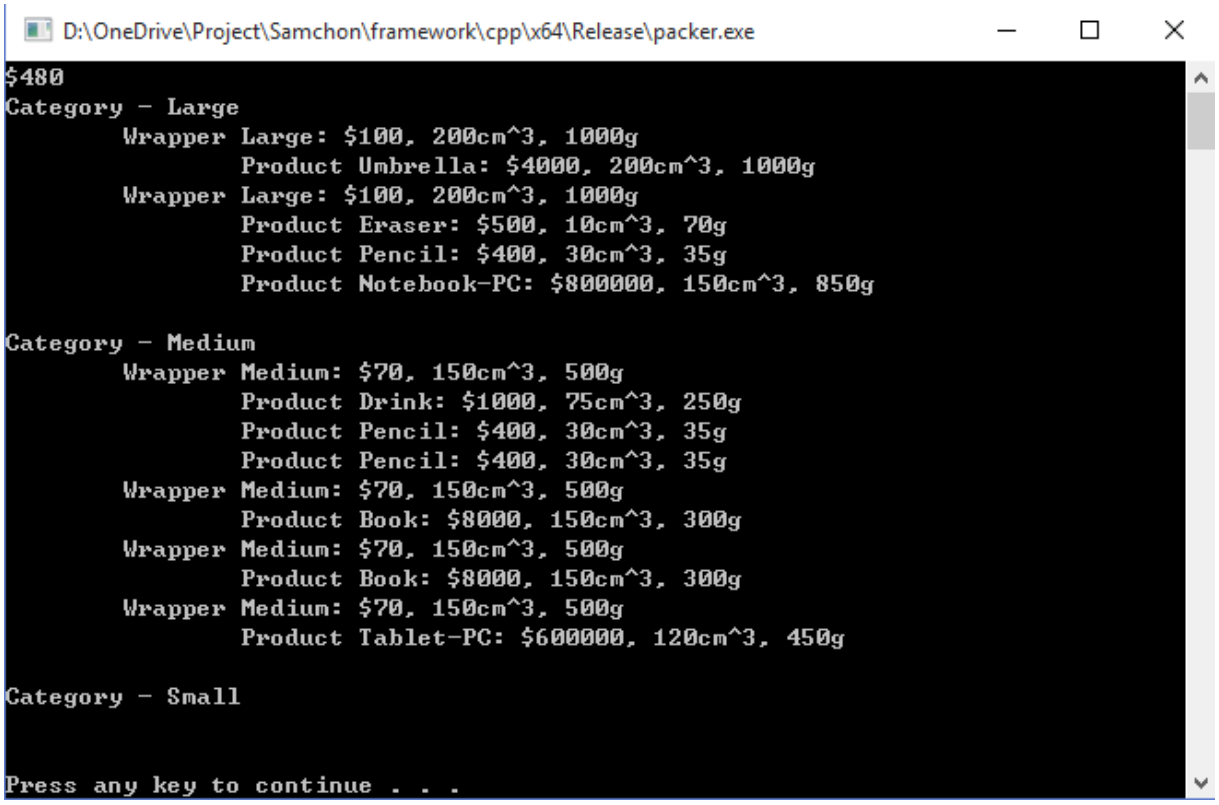
* Product (const std::string &name, int price, int volume, int weight)
  Construct from arguments.
* virtual auto toString () const -> std::string override
  Return a string represents the Instance.

Additional Inherited Members

10.115.1 Detailed Description

A product, merchandise.

A merchandise with its price, volume and weight.



Author

Jeongho Nam

Definition at line 21 of file Product.hpp.

10.115.2 Constructor & Destructor Documentation

10.115.2.1 Product::Product (const std::string & *name*, int *price*, int *volume*, int *weight*)

Construct from arguments.

Parameters

name	Name of a product can be an identifier
price	Price of the product
volume	Volume of the product
weight	Weight of the product

Definition at line 7 of file Product.cpp.

10.115.3 Member Function Documentation

10.115.3.1 auto Product::toString ( ) const -> std::string [override], [virtual]

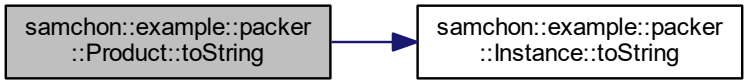
Return a string represents the Instance.

Reimplemented from samchon::example::packer::Instance.

Definition at line 11 of file Product.cpp.

References samchon::example::packer::Instance::toString().

Here is the call graph for this function:



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

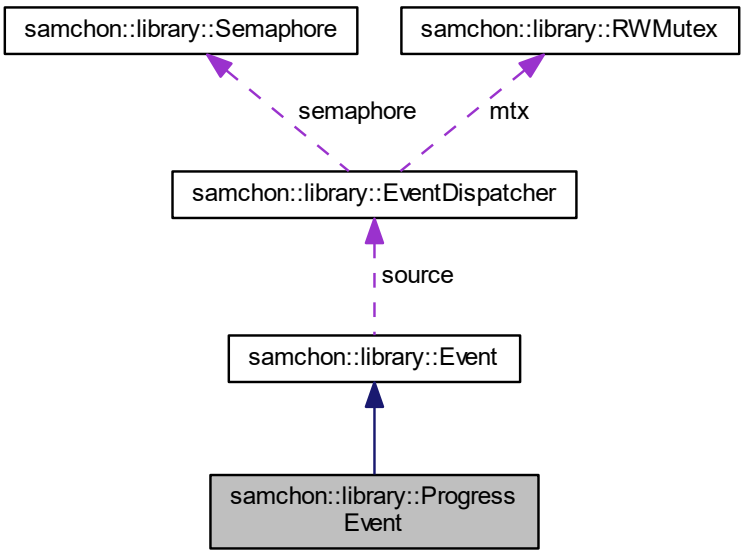
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/packer/Product.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/packer/Product.cpp

10.116 samchon::library::ProgressEvent Class Reference

Event representing a progress.

`#include <ProgressEvent.hpp>`

Collaboration diagram for samchon::library::ProgressEvent:



Public Member Functions

* ProgressEvent (EventDispatcher *, size_t, size_t)
  Construct from source and progress.
* auto getNumerator () const -> size_t
  Get numerator, number of current progress.
* auto getDenominator () const -> size_t
  Get denominator, number of total progress.
* auto getPercent () const -> double
  Get percentage of the progress.

Protected Attributes

* size_t numerator
  The number of current progress.
* size_t denominator
  The number of total progress.

10.116.1 Detailed Description

Event representing a progress.
[Inherited]

Definition at line 17 of file ProgressEvent.hpp.

10.116.2 Constructor & Destructor Documentation

10.116.2.1 ProgressEvent::ProgressEvent (EventDispatcher * *source*, size_t *numerator*, size_t *denominator*)

Construct from source and progress.

Parameters

source	Source of the event; who made the event
numerator	The number of current progress
denominator	The number of total progress

Exceptions

exception	denominator is zero

Definition at line 4 of file ProgressEvent.cpp.
References denominator, and numerator.

10.116.3 Member Function Documentation

10.116.3.1 auto ProgressEvent::getNumerator ( ) const -> size_t

Get numerator, number of current progress.
Definition at line 11 of file ProgressEvent.cpp.
References numerator.

10.116.3.2 auto ProgressEvent::getDenominator ( ) const -> size_t

Get denominator, number of total progress.
Definition at line 15 of file ProgressEvent.cpp.
References denominator.

10.116.3.3 auto ProgressEvent::getPercent ( ) const -> double

Get percentage of the progress.
Definition at line 19 of file ProgressEvent.cpp.
References denominator, and numerator.

10.116.4 Member Data Documentation

10.116.4.1 size_t samchon::library::ProgressEvent::numerator [protected]

The number of current progress.
Definition at line 34 of file ProgressEvent.hpp.
Referenced by getNumerator(), getPercent(), and ProgressEvent().

10.116.4.2 size_t samchon::library::ProgressEvent::denominator [protected]

The number of total progress.

Warning

denominator cannot be zero

Definition at line 40 of file ProgressEvent.hpp.

Referenced by getDenominator(), getPercent(), and ProgressEvent().

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

* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/ProgressEvent.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/ProgressEvent.cpp

10.117 samchon::library::RWMutex Class Reference

rw_mutex

```
#include <RWMutex.hpp>
```

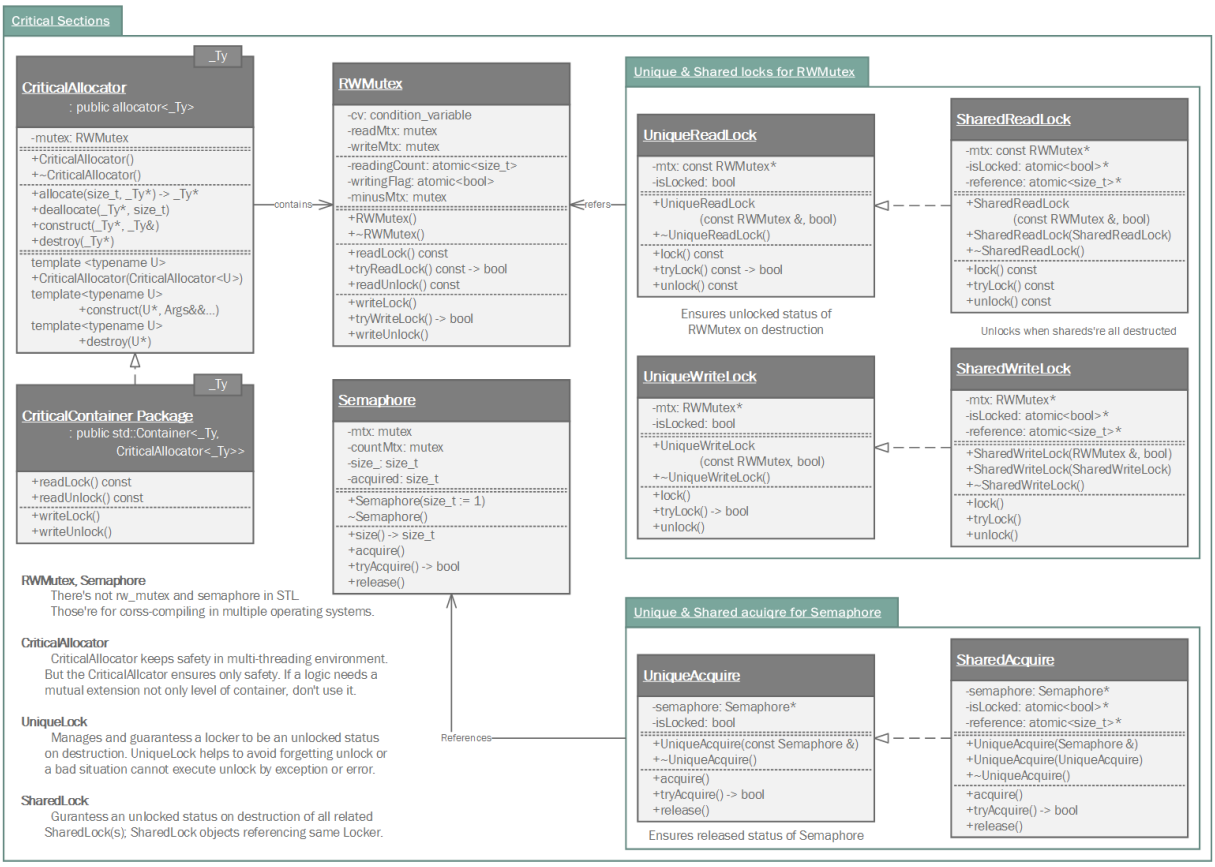
Public Member Functions

* RWMutex ()
  Default Constructor.
* void readLock () const
  Lock on read.
* void readUnlock () const
  Unlock of read.
* void writeLock ()
  Lock on writing.
* void writeUnlock ()
  Unlock on writing.

10.117.1 Detailed Description

rw_mutex

A mutex divided into reading and writing.



Note

Of course, rw_mutex is already defined in linux C. But it is dependent on the linux OS, so that cannot be compiled in Window having the rw_mutex. There's not a class like rw_mutex in STL yet. It's the reason why RWMutex is provided.
As that reason, if STL supports the rw_mutex in near future, the RWMutex can be deprecated.

See also

samchon::library

Author

Jeongho Nam

Definition at line 33 of file RWMutex.hpp.

10.117.2 Constructor & Destructor Documentation

10.117.2.1 RWMutex::RWMutex ( )

Default Constructor.
Definition at line 10 of file RWMutex.cpp.

10.117.3 Member Function Documentation

10.117.3.1 void RWMutex::readLock ( ) const

Lock on read.
Increases a reading count.
When write_lock is on a progress, wait until write_unlock to be called.

* Reading can be done by multiple sections.
* Reading can't be done when writing.

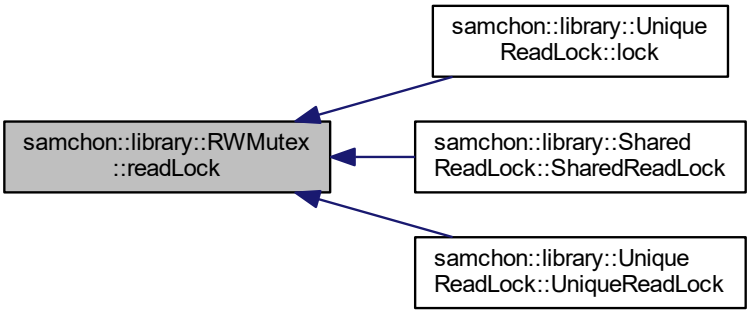
Warning

You've to call read_unlock when the reading work is terminated.

Definition at line 31 of file RWMutex.cpp.

Referenced by samchon::library::UniqueReadLock::lock(), samchon::library::SharedReadLock::↩ SharedReadLock(), and samchon::library::UniqueReadLock::UniqueReadLock().

Here is the caller graph for this function:



10.117.3.2 void RWMutex::readUnlock ( ) const

Unlock of read.

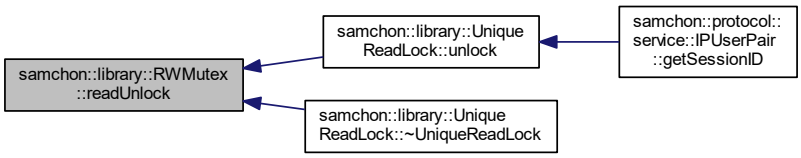
Decreases a reading count.

When write_lock had done after read_lock, it continues by read_unlock if the reading count was 1 (read_unlock makes the count to be zero).

Definition at line 39 of file RWMutex.cpp.

Referenced by samchon::library::UniqueReadLock::unlock(), and samchon::library::UniqueReadLock↩ ::∼UniqueReadLock().

Here is the caller graph for this function:



10.117.3.3 void RWMutex::writeLock ( )

Lock on writing.

Changes writing flag to true.

If another write_lock or read_lock is on a progress, wait until them to be unlocked.

* Writing can be done by only a section at once.
* Writing can't be done when reading.

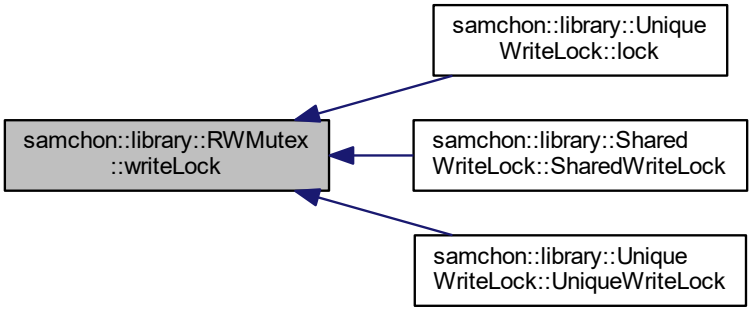
Note

You've to call write_unlock when writing work was terminated.

Definition at line 53 of file RWMutex.cpp.

Referenced by samchon::library::UniqueWriteLock::lock(), samchon::library::SharedWriteLock::↩ SharedWriteLock(), and samchon::library::UniqueWriteLock::UniqueWriteLock().

Here is the caller graph for this function:



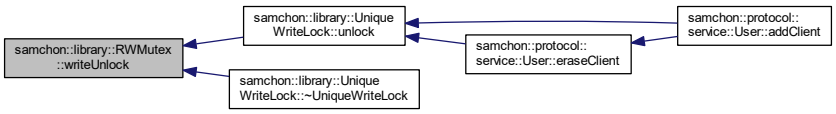
10.117.3.4 void RWMutex::writeUnlock ( )

Unlock on writing.

Definition at line 61 of file RWMutex.cpp.

Referenced by samchon::library::UniqueWriteLock::unlock(), and samchon::library::UniqueWriteLock↩ ::~UniqueWriteLock().

Here is the caller graph for this function:



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

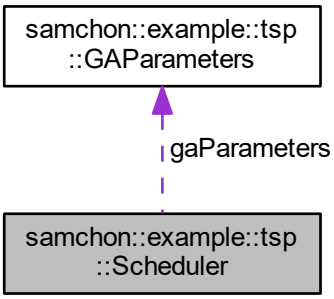
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/RWMutex.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/RWMutex.cpp

10.118 samchon::example::tsp::Scheduler Class Reference

A scheduler.

```
#include <Scheduler.hpp>
```

Collaboration diagram for samchon::example::tsp::Scheduler:



Public Member Functions

* Scheduler (std::shared_ptr< Travel >, const struct GAParameters &)
  Construct from points and parameter of genetic algorithm.
* auto optimize () -> std::shared_ptr< Travel >
  Derive optimized schedule.

Private Attributes

* std::shared_ptr< Travel > travel
  A travel to optimize or optimized.
* struct GAParameters gaParameters
  Parameters for Genetic-Algorithm.

10.118.1 Detailed Description

A scheduler.

Scheduler deducts an optimal schdule(Travel) by genetic algorithm.

Author

Jeongho Nam

Definition at line 38 of file Scheduler.hpp.

10.118.2 Constructor & Destructor Documentation

10.118.2.1 Scheduler::Scheduler (std::shared_ptr< Travel > , const struct GAParameters &)

Construct from points and parameter of genetic algorithm.

Definition at line 10 of file Scheduler.cpp.

10.118.3 Member Function Documentation

10.118.3.1 auto Scheduler::optimize ( ) -> std::shared_ptr<Travel>

Derive optimized schedule.

Returns

An optimized travel schedule.

Definition at line 15 of file Scheduler.cpp.

References samchon::library::GeneticAlgorithm< GeneArray, Compare >::evolveGeneArray().

Here is the call graph for this function:

10.118.4 Member Data Documentation

10.118.4.1 std::shared_ptr<Travel> samchon::example::tsp::Scheduler::travel [private]

A travel to optimize or optimized.

Definition at line 44 of file Scheduler.hpp.

10.118.4.2 struct GAParameters samchon::example::tsp::Scheduler::gaParameters [private]

Parameters for Genetic-Algorithm.

Definition at line 49 of file Scheduler.hpp.

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

* D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/tsp/Scheduler.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/tsp/Scheduler.cpp

10.119 samchon::library::Semaphore Class Reference

A semaphore.

`#include <Semaphore.hpp>`

Public Member Functions

* Semaphore (size_t=2)
* void setSize (size_t)
  Set size.
* auto size () const -> size_t
  Get size.
* auto acquiredSize () const -> size_t
  Get acquired size.
* void acquire ()
  Acquire admission.
* auto tryAcquire () -> bool
  Try to acquire admission.
* void release ()
  Release an admission.

Private Attributes

* size_t size_
  The size.
* size_t acquired
  Acquired count.
* std::mutex * mtx
  Locker.

10.119.1 Detailed Description

A semaphore.

In computer science, particularly in operating systems, a semaphore is a variable or abstract data type that is used for controlling access, by multiple processes, to a common resource in a concurrent system such as a multiprogramming operating system.

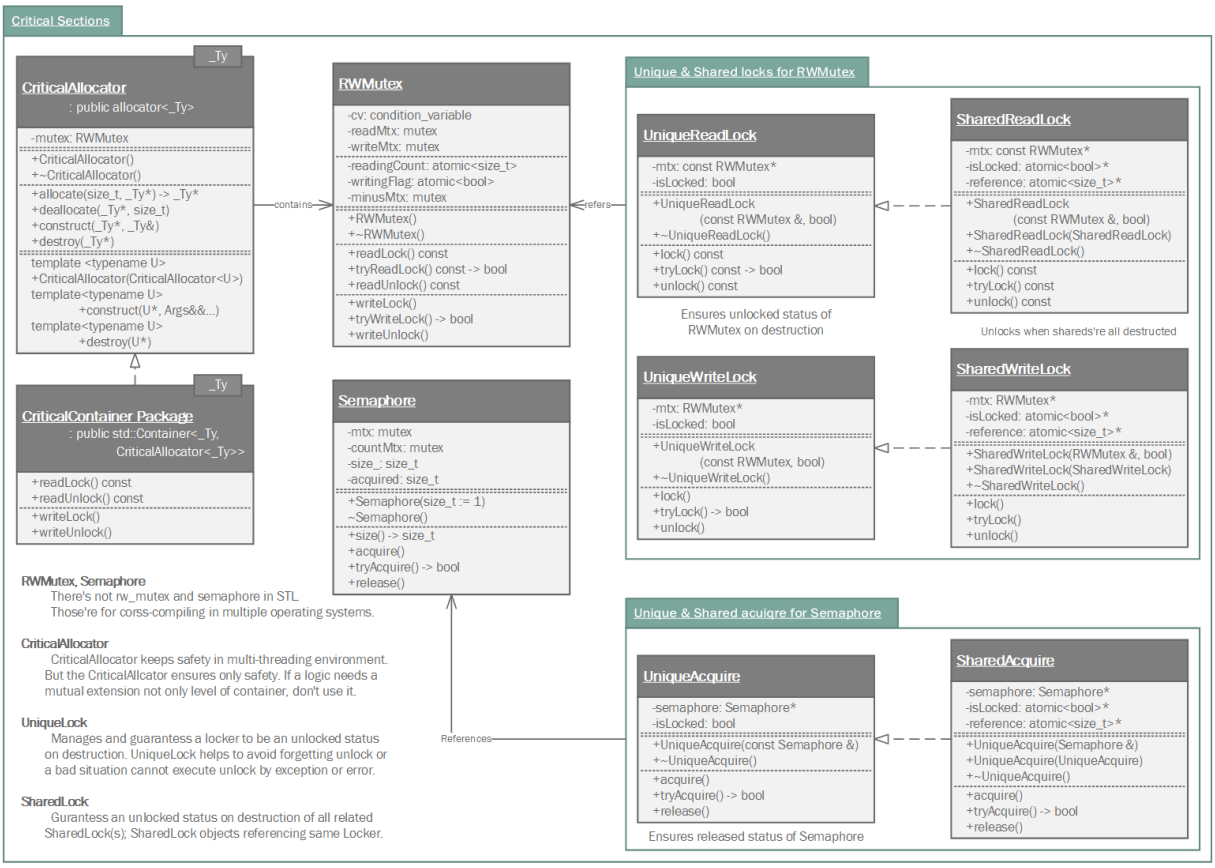
A trivial semaphore is a plain variable that is changed (for example, incremented or decremented, or toggled) depending on programmer-defined conditions. The variable is then used as a condition to control access to some system resource.

A useful way to think of a semaphore as used in the real-world systems is as a record of how many units of a particular resource are available, coupled with operations to safely (i.e., without race conditions) adjust that record as units are required or become free, and, if necessary, wait until a unit of the resource becomes available. Semaphores are a useful tool in the prevention of race conditions; however, their use is by no means a guarantee that a program is free from these problems. Semaphores which allow an arbitrary resource count are called counting semaphores, while semaphores which are restricted to the values 0 and 1 (or locked/unlocked, unavailable/available) are called binary semaphores

* Categorized in binary semaphore

Referenced Wediapedia

* https://en.wikipedia.org/wiki/Semaphore_(programming)



Note

Of course, semaphore is already defined in linux C and MFC in Window. But it is dependent on each operating system, so that cannot be compiled in another OS with those semaphores. There's not a class like semaphore in STL yet. It's the reason why Semaphore is provided.

As that reason, if STL supports the semaphore in near future, the Semaphore can be deprecated.

See also

samchon::library

Author

Jeongho Nam

Definition at line 51 of file Semaphore.hpp.

10.119.2 Constructor & Destructor Documentation

10.119.2.1 Semaphore::Semaphore (size_t *size* = 2)

Constructor.

Parameters

size	The size of the semaphore to permit

Definition at line 12 of file Semaphore.cpp.

10.119.3 Member Function Documentation

10.119.3.1 void Semaphore::setSize (size_t *val*)

Set size.

Set permitted size of the semaphore.

Definition at line 26 of file Semaphore.cpp.

10.119.3.2 auto Semaphore::size ( ) const -> size_t

Get size.

Returns size which means the permitted count of the semaphore

Returns

The size of semaphore

Definition at line 36 of file Semaphore.cpp.

10.119.3.3 auto Semaphore::acquiredSize ( ) const -> size_t

Get acquired size.

Definition at line 40 of file Semaphore.cpp.

10.119.3.4 void Semaphore::acquire ( )

Acquire admission.

Acquires an admission and increases count of admission by 1.

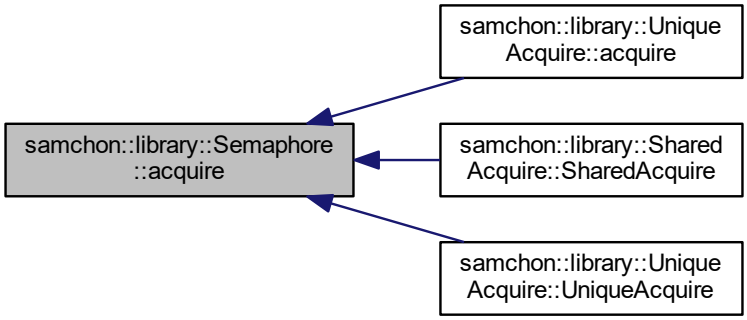
If the count is over permitted size, wait until other admissions to be released.

* Lock on mutex

Definition at line 50 of file Semaphore.cpp.

Referenced by samchon::library::UniqueAcquire::acquire(), samchon::library::SharedAcquire::↩ SharedAcquire(), and samchon::library::UniqueAcquire::UniqueAcquire().

Here is the caller graph for this function:



10.119.3.5 auto Semaphore::tryAcquire ( ) -> bool

Try to acquire admission.

If admission count is below the permitted size, acquire admission and increase the count by 1 and return true which means succeded to get admission.

* If the count is matched to the permitted size, lock the mutex

Else, do not acquire admission and return false which means failed to get admmission.

Returns

Whether succeded to acquire an admission or not

Definition at line 60 of file Semaphore.cpp.

10.119.3.6 void Semaphore::release ( )

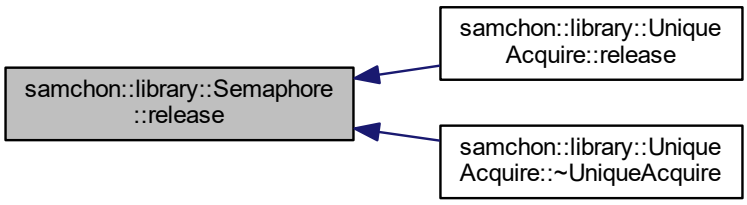
Release an admission.

Releases an admission what you've acquired. If the admission count was over the limited size, unlock the mutex.

Definition at line 75 of file Semaphore.cpp.

Referenced by samchon::library::UniqueAcquire::release(), and samchon::library::UniqueAcquire::~↩ UniqueAcquire().

Here is the caller graph for this function:



10.119.4 Member Data Documentation

10.119.4.1 size_t samchon::library::Semaphore::size_ [private]

The size.

Permitted size of the semaphore

Definition at line 58 of file Semaphore.hpp.

10.119.4.2 size_t samchon::library::Semaphore::acquired [private]

Acquired count.

Definition at line 66 of file Semaphore.hpp.

10.119.4.3 std::mutex* samchon::library::Semaphore::mtx [private]

Locker.

Manages lock and unlock of the semaphore

Definition at line 73 of file Semaphore.hpp.

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

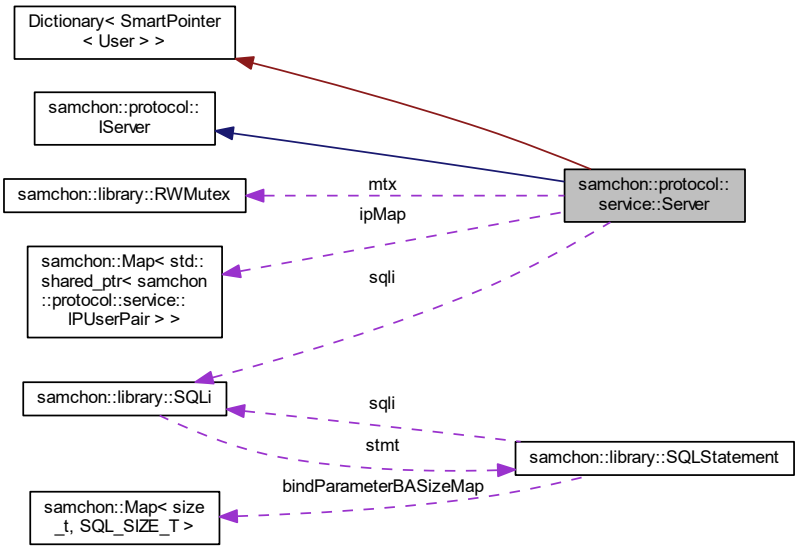
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/Semaphore.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/Semaphore.cpp

10.120 samchon::protocol::service::Server Class Reference

A server for (cloud) service.

`#include <Server.hpp>`

Collaboration diagram for samchon::protocol::service::Server:



Public Member Functions

* Server ()
  Default Constructor.
* auto size () const -> size_t

Size of User(s).
* auto begin () const -> const_iterator
  A const iterator of begin.
* auto end () const -> const_iterator
  A const iterator of end.

Protected Member Functions

* virtual auto NAME () const -> std::string=0
  A name can be an identifier of a cloud server.
* virtual auto createUser () -> User *=0
  Factory method of User.
* virtual void addClient (Socket *) override
  Handling connection of a client.

Protected Attributes

* library::SQLi * sqli
  SQLi for archiving log.

Private Attributes

* Dictionary< std::shared_ptr< IPUserPair > > ipMap
  Dictionary of issuer of session ID of each ip.
* size_t sequence
  Sequence for issuing session ID.

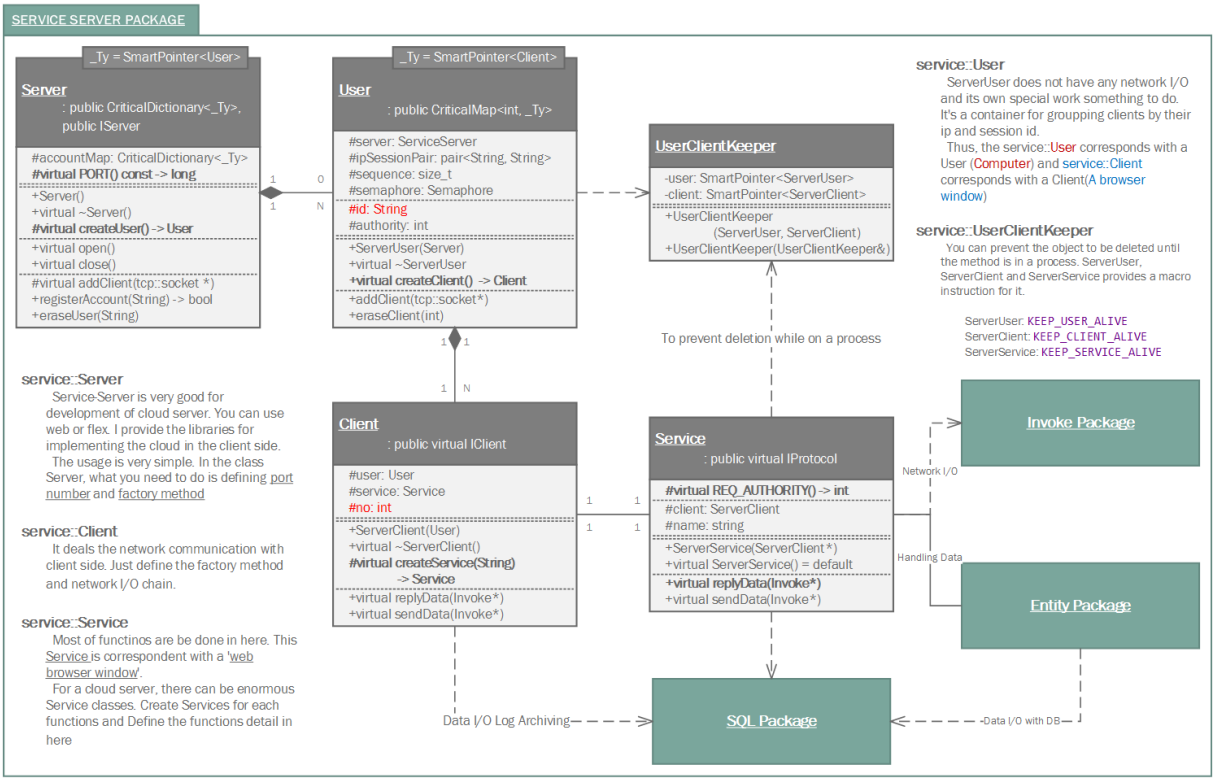
Additional Inherited Members

10.120.1 Detailed Description

A server for (cloud) service.

A server class representing the real physical server of a cloud.

The Server is very good for development of a cloud server. You can use web or flex as (physical) client. The usage is very simple. In the class Server, what you need to do are overriding getter of port number and factory method creating an User.



Note

Override those methods.

* Server::NAME()
* Server::PORT()
* Server::createUser()

See also

samchon::protocol
samchon::protocol::service

Author

Jeongho Nam

Definition at line 47 of file Server.hpp.

10.120.2 Constructor & Destructor Documentation

10.120.2.1 Server::Server ( )

Default Constructor.

Definition at line 31 of file Server.cpp.

References sequence, and sqli.

10.120.3 Member Function Documentation

10.120.3.1 virtual auto samchon::protocol::service::Server::NAME ( ) const -> std::string [protected], [pure virtual]

A name can be an identifier of a cloud server.

10.120.3.2 auto Server::size ( ) const -> size_t

Size of User(s).

Definition at line 53 of file Server.cpp.

10.120.3.3 auto Server::begin ( ) const -> const_iterator

A const iterator of begin.

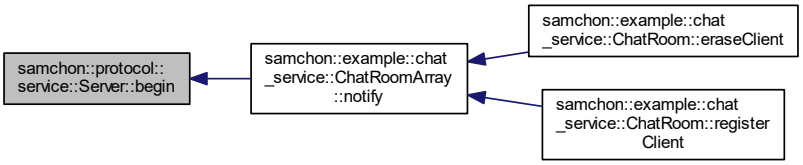
Note

Be careful about concurrency

Definition at line 57 of file Server.cpp.

Referenced by samchon::example::chat_service::ChatRoomArray::notify().

Here is the caller graph for this function:



10.120.3.4 auto Server::end ( ) const -> const_iterator

A const iterator of end.

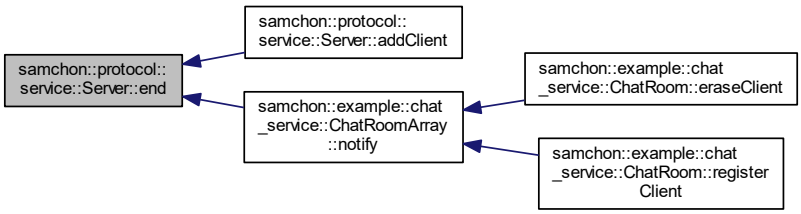
Note

Be careful about concurrency

Definition at line 61 of file Server.cpp.

Referenced by addClient(), and samchon::example::chat_service::ChatRoomArray::notify().

Here is the caller graph for this function:



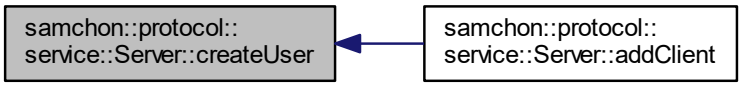
10.120.3.5 virtual auto samchon::protocol::service::Server::createUser ( ) -> User * [protected], [pure virtual]

Factory method of User.

Implemented in samchon::example::chat_service::ChatServer.

Referenced by addClient().

Here is the caller graph for this function:



10.120.3.6 void Server::addClient (Socket * *socket*) [override], [protected], [virtual]

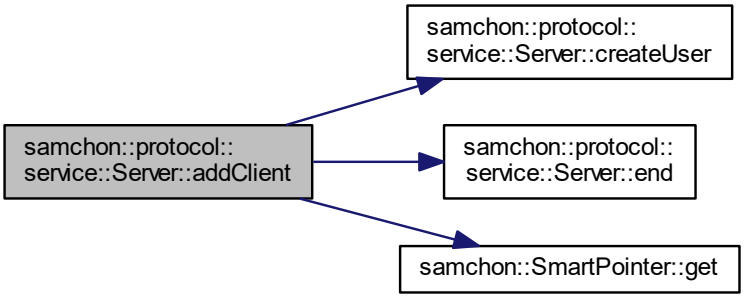
Handling connection of a client.

Implements samchon::protocol::IServer.

Definition at line 69 of file Server.cpp.

References createUser(), end(), samchon::SmartPointer< _Ty >::get(), ipMap, and sequence.

Here is the call graph for this function:



10.120.4 Member Data Documentation

10.120.4.1 library::SQLi* samchon::protocol::service::Server::sqli [protected]

SQLi for archiving log.

Definition at line 66 of file Server.hpp.

Referenced by Server().

10.120.4.2 Dictionary<std::shared_ptr<IPUserPair> > samchon::protocol::service::Server::ipMap [private]

Dictionary of issuer of session ID of each ip.

Definition at line 74 of file Server.hpp.

Referenced by addClient().

10.120.4.3 size_t samchon::protocol::service::Server::sequence [private]

Sequence for issuing session ID.

Definition at line 79 of file Server.hpp.

Referenced by addClient(), and Server().

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

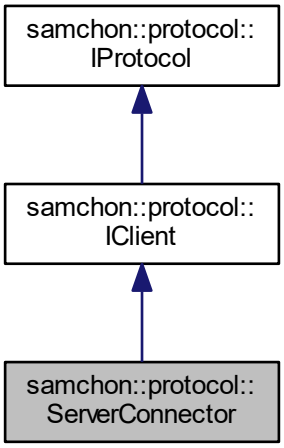
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/service/Server.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/service/Server.cpp

10.121 samchon::protocol::ServerConnector Class Reference

A server connector for a physical client.

`#include <ServerConnector.hpp>`

Collaboration diagram for samchon::protocol::ServerConnector:



Public Member Functions

* ServerConnector ()
  Default Constructor.
* virtual auto getIP () const -> std::string=0
  Destinatio IP.
* virtual auto getPort () const -> int=0
  Destination port.
* virtual auto getMyIP () const -> std::string
  (optional) My IP, if you want to bind
* virtual void connect ()
  Connect to a server.

Protected Attributes

* boost::asio::io_service * ioService
  An io_service of Boost.Asio's own.
* EndPoint * endPoint
  An endpoint directing a server.
* EndPoint * localEndPoint
  (Optional) An local endpoint of the client (my system)

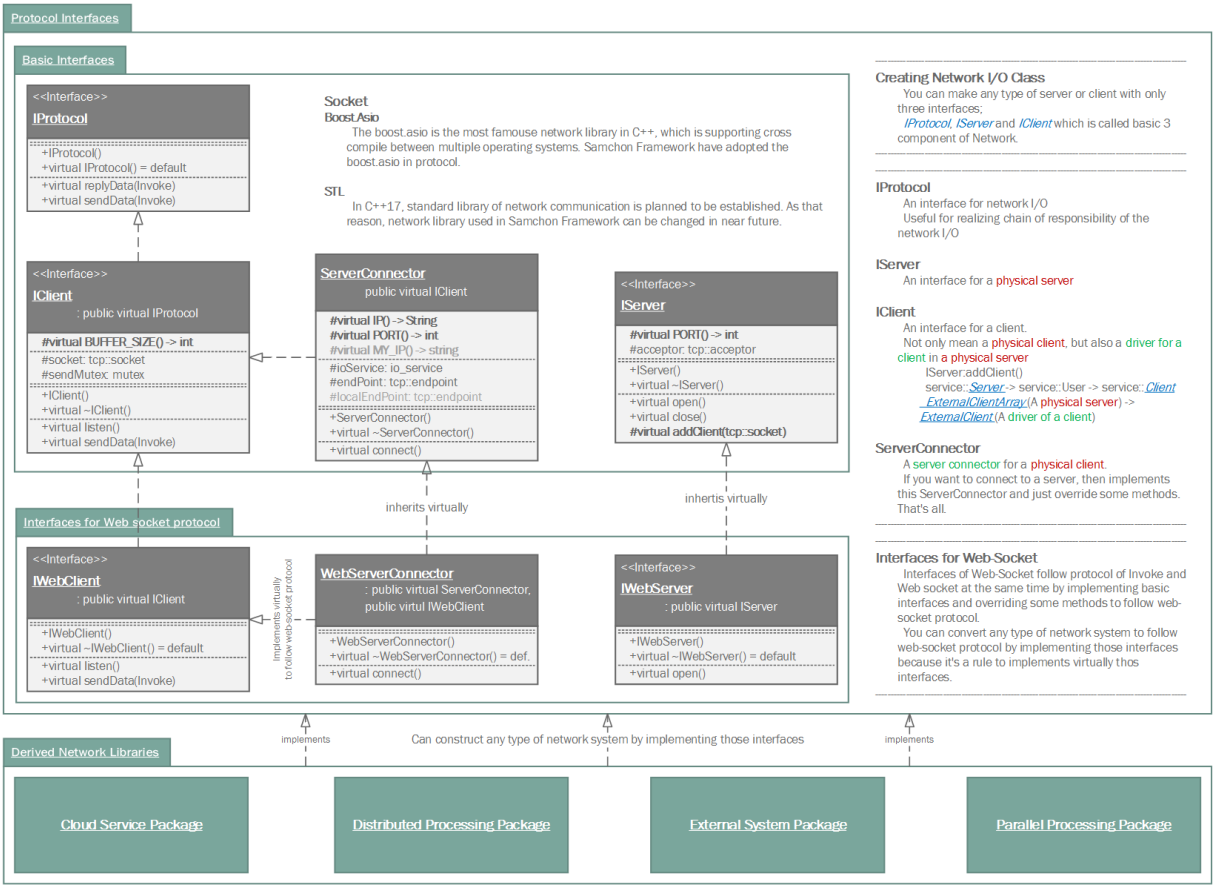
Additional Inherited Members

10.121.1 Detailed Description

A server connector for a physical client.

ServerConnector is a class for a physical client connecting a server. If you want to connect to a server, then implements this ServerConnector and just override some methods like getIP(), getPort() and replyData(). That's all.

In Samchon Framework, package protocol, There are basic 3 + 1 components that can make any type of network system in Samchon Framework. The basic 3 components are IProtocol, IServer and IClient. The last, surplus one is the ServerConnector. Looking around classes in Samchon Framework, especially module master and slave which are designed for realizing distributed processing systems and parallel processing systems, physical client classes are all derived from this ServerConnector.



Author

Jeongho Nam

Definition at line 32 of file ServerConnector.hpp.

10.121.2 Constructor & Destructor Documentation

10.121.2.1 ServerConnector::ServerConnector ( )

Default Constructor.

Definition at line 14 of file ServerConnector.cpp.

References endPoint, ioService, and localEndPoint.

10.121.3 Member Function Documentation

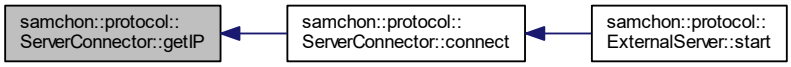
10.121.3.1 virtual auto samchon::protocol::ServerConnector::getIP ( ) const -> std::string [pure virtual]

Destinatio IP.

Implemented in samchon::protocol::ExternalServer.

Referenced by connect().

Here is the caller graph for this function:



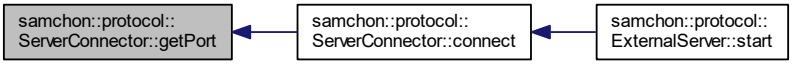
10.121.3.2 virtual auto samchon::protocol::ServerConnector::getPort ( ) const -> int [pure virtual]

Destination port.

Implemented in samchon::protocol::ExternalServer.

Referenced by connect().

Here is the caller graph for this function:



10.121.3.3 auto ServerConnector::getMyIP ( ) const -> std::string [virtual]

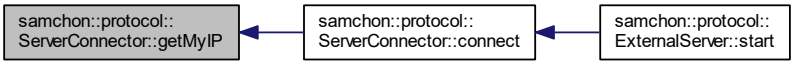
(optional) My IP, if you want to bind

Reimplemented in samchon::protocol::ExternalServer.

Definition at line 28 of file ServerConnector.cpp.

Referenced by connect().

Here is the caller graph for this function:



10.121.3.4 void ServerConnector::connect ( ) [virtual]

Connect to a server.

Connects to a server with configured ip address and port number. After connection, listens data replied from the server (IClient::listen()).

Note

It monopolies a thread

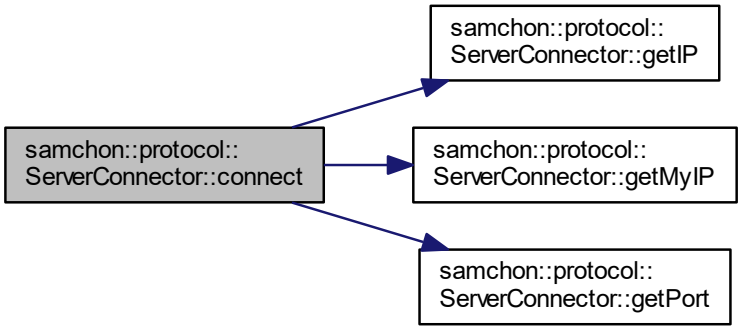
Reimplemented in samchon::protocol::WebServerConnector.

Definition at line 33 of file ServerConnector.cpp.

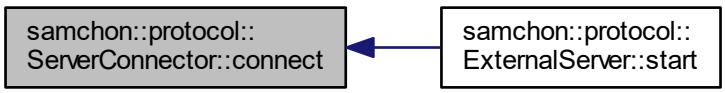
References endPoint, getIP(), getMyIP(), getPort(), ioService, localEndPoint, and samchon::protocol↩ ::IClient::socket.

Referenced by samchon::protocol::ExternalServer::start().

Here is the call graph for this function:



Here is the caller graph for this function:



10.121.4 Member Data Documentation

10.121.4.1 boost::asio::io_service* samchon::protocol::ServerConnector::ioService [protected]

An io_service of Boost.Asio's own.

Definition at line 41 of file ServerConnector.hpp.

Referenced by connect(), and ServerConnector().

10.121.4.2 EndPoint* samchon::protocol::ServerConnector::endPoint [protected]

An endpoint directing a server.

Definition at line 46 of file ServerConnector.hpp.

Referenced by connect(), and ServerConnector().

10.121.4.3 EndPoint* samchon::protocol::ServerConnector::localEndPoint [protected]

(Optional) An local endpoint of the client (my system)

An local endpoint fof my system. It binds my ip to custom. It acts when my_ip is configured. If the my_ip is not configured, does not act.

Definition at line 55 of file ServerConnector.hpp.

Referenced by connect(), and ServerConnector().

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

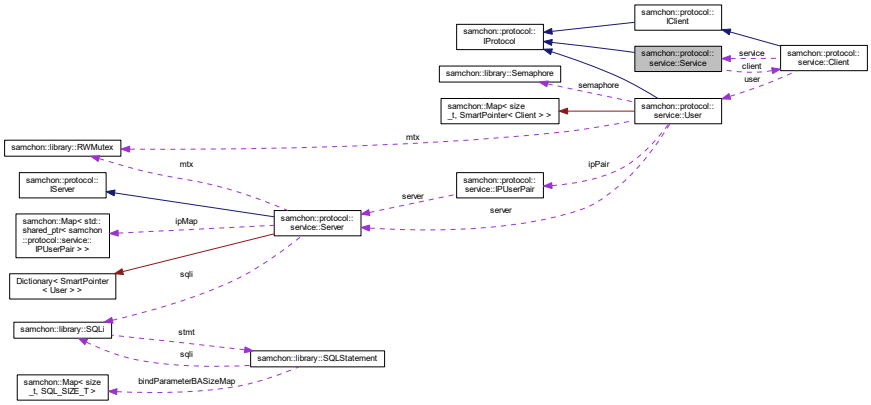
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ServerConnector.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/ServerConnector.cpp

10.122 samchon::protocol::service::Service Class Reference

A service.

`#include <Service.hpp>`

Collaboration diagram for samchon::protocol::service::Service:



Public Member Functions

* Service (Client *)
  Construct from Client.
* auto getClient () const -> Client *
  Get Client.
* virtual void sendData (std::shared_ptr< Invoke >)
  Shift the responsibility of sending an Invoke message to Client.

Protected Member Functions

* virtual auto REQUIRE_AUTHORITY () const -> int=NULL
  Required authority to access the service.

Private Attributes

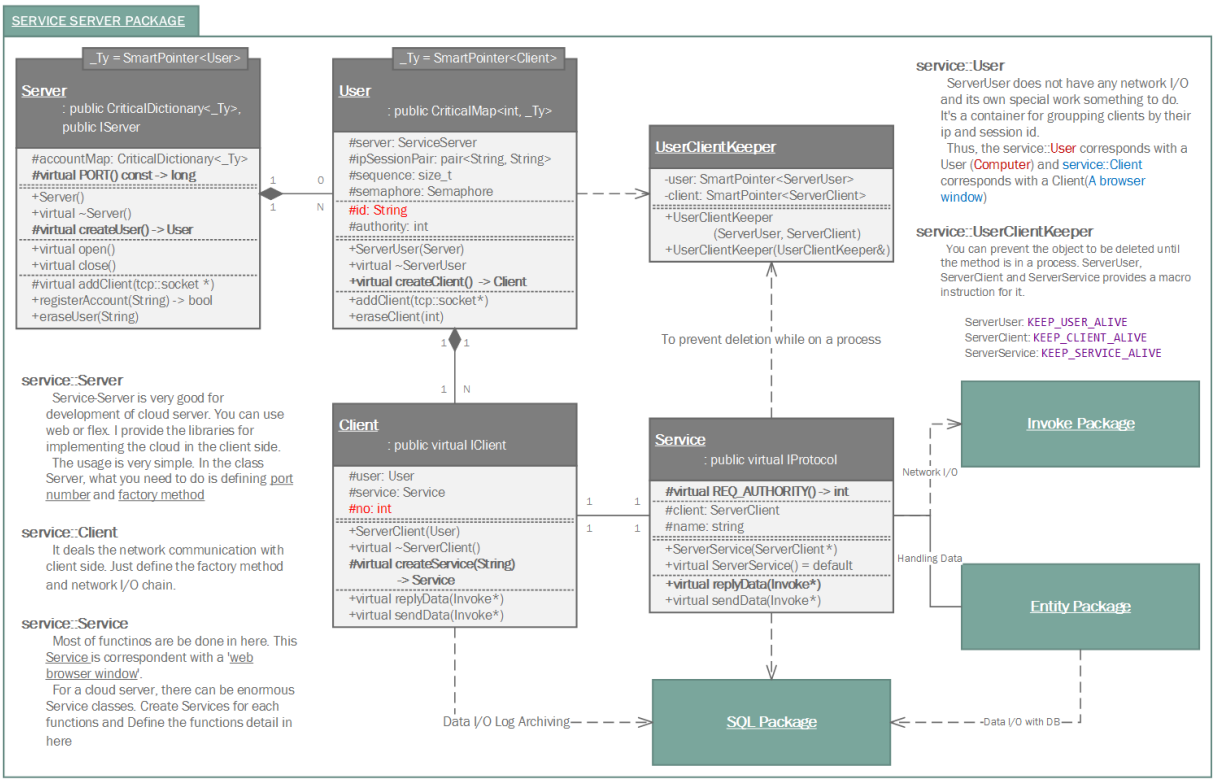
* Client * client
  Client object Service is belonged to.
* std::string name
  A name represents a Service.

10.122.1 Detailed Description

A service.

A service represents an unique functions of a web page

Most of functinos are realized in here. The Service is correspondent with a 'a service of a web page'. For a cloud Server, there can be enormous Service classes. Create Services for each functions and Define the functions detail in here.



See also

samchon::protocol
samchon::protocol::service

Author

Jeongho Nam

Definition at line 44 of file Service.hpp.

10.122.2 Constructor & Destructor Documentation

10.122.2.1 Service::Service (Client * *client*)

Construct from Client.

Parameters

client	Client object Service is belonged to

Definition at line 18 of file Service.cpp.

References client.

10.122.3 Member Function Documentation

10.122.3.1 virtual auto samchon::protocol::service::Service::REQUIRE_AUTHORITY ( ) const -> int=NULL [protected],[virtual]

Required authority to access the service.

Referenced by samchon::protocol::service::Client::constructService().

Here is the caller graph for this function:

10.122.3.2 auto Service::getClient ( ) const -> Client*

Get Client.

Definition at line 32 of file Service.cpp.

References client.

10.122.3.3 void Service::sendData (std::shared_ptr< Invoke >) [virtual]

Shift the responsibility of sending an Invoke message to Client.

Parameters

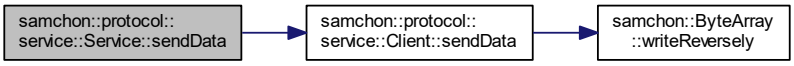
in	An Invoke message to be sent to the (physical) client

Reimplemented from samchon::protocol::IProtocol.

Definition at line 40 of file Service.cpp.

References client, and samchon::protocol::service::Client::sendData().

Here is the call graph for this function:



10.122.4 Member Data Documentation

10.122.4.1 Client* samchon::protocol::service::Service::client [private]

Client object Service is belonged to.

Definition at line 62 of file Service.hpp.

Referenced by getClient(), sendData(), and Service().

10.122.4.2 std::string samchon::protocol::service::Service::name [private]

A name represents a Service.

Definition at line 67 of file Service.hpp.

Referenced by samchon::protocol::service::Client::constructService().

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

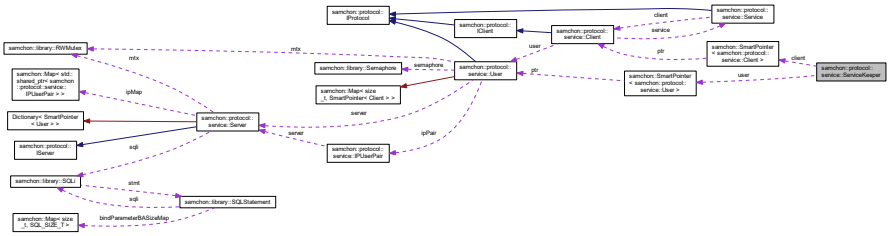
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/service/Service.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/service/Service.cpp

10.123 samchon::protocol::service::ServiceKeeper Class Reference

A keeper blocking destrunction of User and Client.

```
#include <ServiceKeeper.hpp>
```

Collaboration diagram for samchon::protocol::service::ServiceKeeper:



Public Member Functions

* ServiceKeeper (const User *user, const Client *client)
  Construct from User and Client.
* ServiceKeeper (const ServiceKeeper &)
  Copy Constructor.
* ServiceKeeper (ServiceKeeper &&)
  Movie Constructor.

Private Attributes

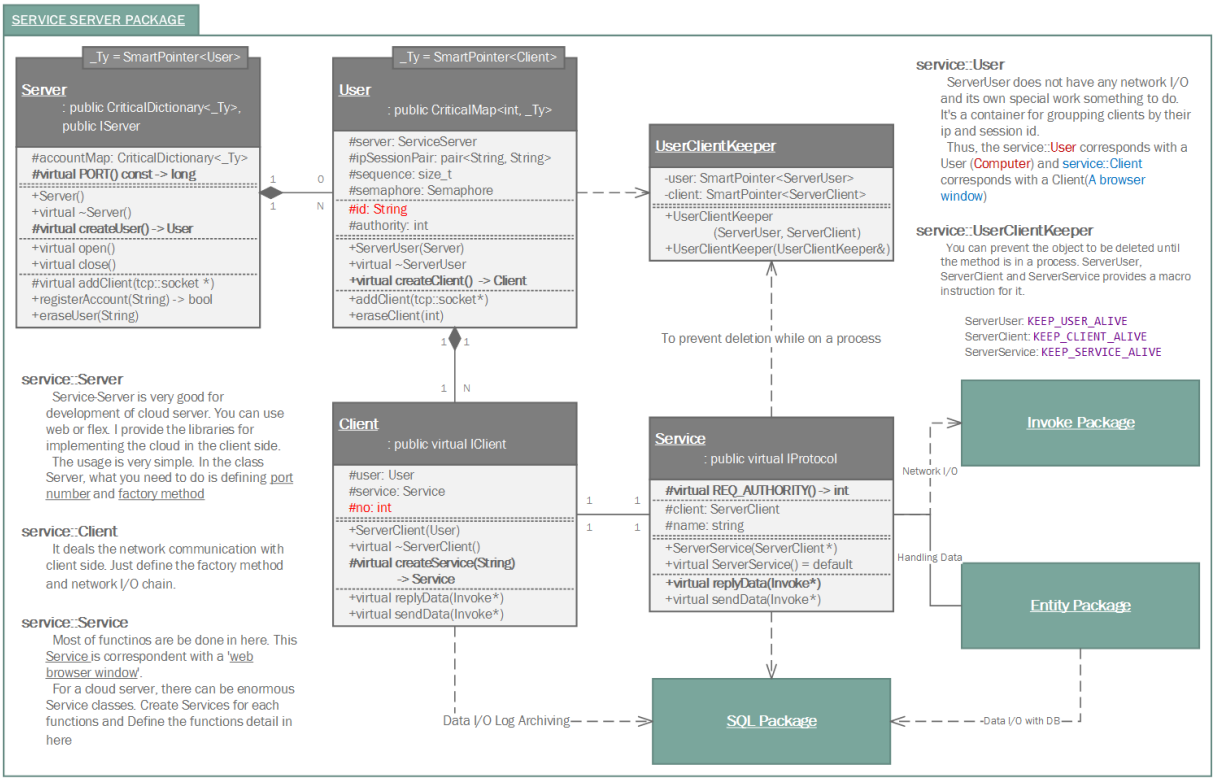
* SmartPointer< User > user
  A shared pionter of an User.
* SmartPointer< Client > client
  A shared pointer of a Client.

10.123.1 Detailed Description

A keeper blocking destrunction of User and Client.

Keeps User, Client and Service objects until termination of a stack.

You can prevent the object to be destructed until the method is in a process. User, Client and Service provides a macro instruction for it.



Note

Defined macros

- KEEP_USER_ALIVE
- KEEP_CLIENT_ALIVE
- KEEP_SERVICE_ALIVE

See also

samchon::protocol
samchon::protocol::service

Author

Jeongho Nam

Definition at line 35 of file ServiceKeeper.hpp.

10.123.2 Constructor & Destructor Documentation

10.123.2.1 ServiceKeeper::ServiceKeeper (const User * *user*, const Client * *client*)

Construct from User and Client.

user An User to keep client A Client to keep

Definition at line 8 of file ServiceKeeper.cpp.

10.123.2.2 ServiceKeeper::ServiceKeeper (const ServiceKeeper & *pair*)

Copy Constructor.

Definition at line 13 of file ServiceKeeper.cpp.

References client, and user.

10.123.2.3 ServiceKeeper::ServiceKeeper (ServiceKeeper && *pair*)

Movie Constructor.

Definition at line 18 of file ServiceKeeper.cpp.

References client, and user.

10.123.3 Member Data Documentation

10.123.3.1 SmartPointer<User> samchon::protocol::service::ServiceKeeper::user [private]

A shared pionter of an User.

Definition at line 41 of file ServiceKeeper.hpp.

Referenced by ServiceKeeper().

10.123.3.2 SmartPointer<Client> samchon::protocol::service::ServiceKeeper::client [private]

A shared pointer of a Client.

Definition at line 46 of file ServiceKeeper.hpp.

Referenced by ServiceKeeper().

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

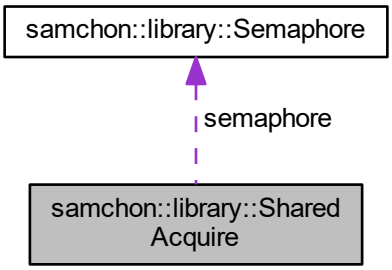
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/service/ServiceKeeper.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/service/ServiceKeeper.cpp

10.124 samchon::library::SharedAcquire Class Reference

Shared acquire from a Semaphore.

`#include <SharedAcquire.hpp>`

Collaboration diagram for samchon::library::SharedAcquire:



Public Member Functions

* SharedAcquire (Semaphore &, bool=true)
  Construct from Semaphore.
* SharedAcquire (const SharedAcquire &)
  Copy Constructor.
* SharedAcquire (SharedAcquire &&)
  Move Constructor.
* ~SharedAcquire ()
  Default Destructor.
* void acquire ()
  Acquire admission.
* void release ()
  Release an admission.

Private Attributes

* Semaphore * semaphore

Managed Semaphore.

* std::atomic< size_t > * reference
  Referencing count sharing same Semaphore.
* std::atomic< bool > * isLocked
  Whether the mutex was locked by SharedAcquire.

10.124.1 Detailed Description

Shared acquire from a Semaphore.

A SharedAcquire is an object manages a Semaphore with shared ownership in both states.

* acquired
* released

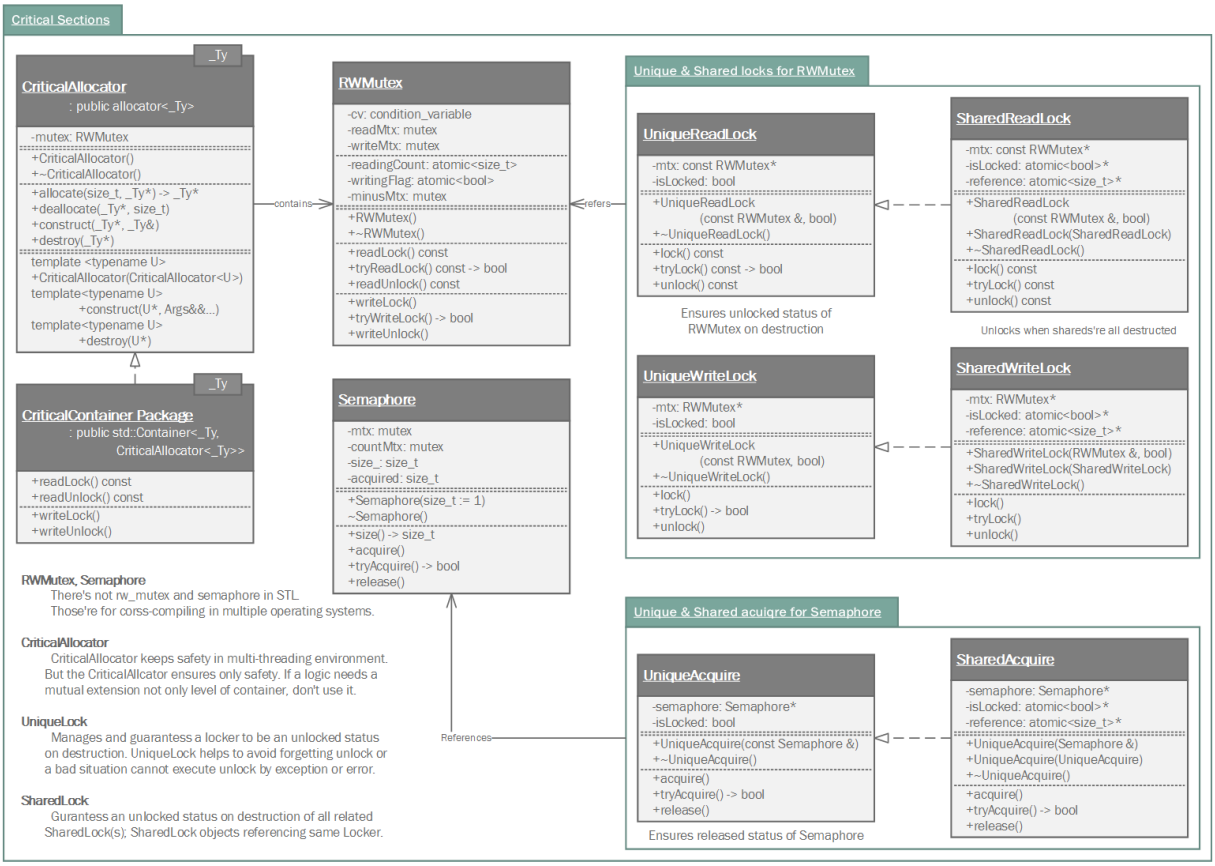
On default construction, the object acquires a Semaphore object, for whose acquiring and releasing operations becomes responsible. When copy constructions, responsibilities of acquiring and releasing Semaphore are shared with copied with those SharedAcquire objects.

The class shared_lock is a general-purpose shared mutex ownership wrapper allowing deferred locking, timed locking and transfer of lock ownership. Locking a shared_lock locks the associated shared mutex in shared mode (to lock it in exclusive mode, std::unique_lock can be used)

This class guarantees a released status on destruction of all shared objects (even if not called explicitly). Therefore it is especially useful as an object with automatic duration, as it guarantees the Semaphore object is properly released in case an exception is thrown.

Referenced comments of std::unique_lock

* http://www.cplusplus.com/reference/mutex/unique_lock/



Note

Though, that the SharedAcquire object does not manage the lifetime of the Semaphore object in any way: the duration of the Semaphore object shall extend at least until the destruction of the Shared↩ Acquire that manages it.

See also

library::UniqueAcquire
samchon::library

Author

Jeongho Nam

Definition at line 52 of file SharedAcquire.hpp.

10.124.2 Constructor & Destructor Documentation

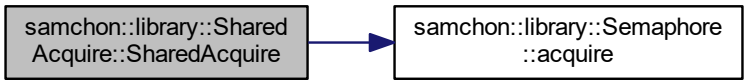
10.124.2.1 SharedAcquire::SharedAcquire (Semaphore & *semaphore*, bool *doLock* = `true`)

Construct from Semaphore.

Parameters

semaphore	Semaphore to manage
doLock	Whether to lock directly or not

Definition at line 12 of file SharedAcquire.cpp.
References samchon::library::Semaphore::acquire().
Here is the call graph for this function:



10.124.2.2 SharedAcquire::SharedAcquire (const SharedAcquire & *obj*)

Copy Constructor.
Definition at line 22 of file SharedAcquire.cpp.
References isLocked, reference, and semaphore.

10.124.2.3 SharedAcquire::SharedAcquire (SharedAcquire && *obj*)

Move Constructor.
Definition at line 30 of file SharedAcquire.cpp.

10.124.2.4 SharedAcquire::~SharedAcquire ( )

Default Destructor.
Definition at line 43 of file SharedAcquire.cpp.

10.124.3 Member Function Documentation

10.124.3.1 void SharedAcquire::acquire ( )

Acquire admission.
Acquires an admission and increases count of admission by 1.
If the count is over permitted size, wait until other admissions to be released.

* Lock on mutex

Definition at line 58 of file SharedAcquire.cpp.

10.124.3.2 void SharedAcquire::release ( )

Release an admission.

Releases an admission what you've acquired. If the admission count was over the limited size, unlock the mutex.

Definition at line 66 of file SharedAcquire.cpp.

10.124.4 Member Data Documentation

10.124.4.1 Semaphore* samchon::library::SharedAcquire::semaphore [private]

Managed Semaphore.

Definition at line 58 of file SharedAcquire.hpp.

Referenced by SharedAcquire().

10.124.4.2 std::atomic<size_t>* samchon::library::SharedAcquire::reference [private]

Referencing count sharing same Semaphore.

Definition at line 63 of file SharedAcquire.hpp.

Referenced by SharedAcquire().

10.124.4.3 std::atomic<bool>* samchon::library::SharedAcquire::isLocked [private]

Whether the mutex was locked by SharedAcquire.

Definition at line 68 of file SharedAcquire.hpp.

Referenced by SharedAcquire().

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

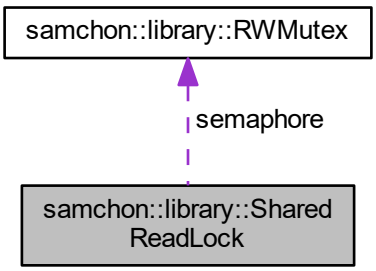
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/SharedAcquire.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/SharedAcquire.cpp

10.125 samchon::library::SharedReadLock Class Reference

Shared lock from a RWMutex.

`#include <SharedReadLock.hpp>`

Collaboration diagram for samchon::library::SharedReadLock:



Public Member Functions

* SharedReadLock (const RWMutex &, bool=true)
  Construct from RWMutex.
* SharedReadLock (const SharedReadLock &)
  Copy Constructor.
* SharedReadLock (SharedReadLock &&)
  Move Constructor.
* ~SharedReadLock ()
  Default Destructor.
* void lock () const
  Lock on read.
* void unlock () const
  Unlock of read.

Private Attributes

* const RWMutex * semaphore
  Managed RWMutex.
* std::atomic< size_t > * reference
  Referencing count sharing same RWMutex.
* std::atomic< bool > * isLocked
  Whether the mutex was locked by SharedReadLock.

10.125.1 Detailed Description

Shared lock from a RWMutex.

A SharedReadLock is an object manages a RWMutex with shared ownership in both states.

* locked
* unlocked

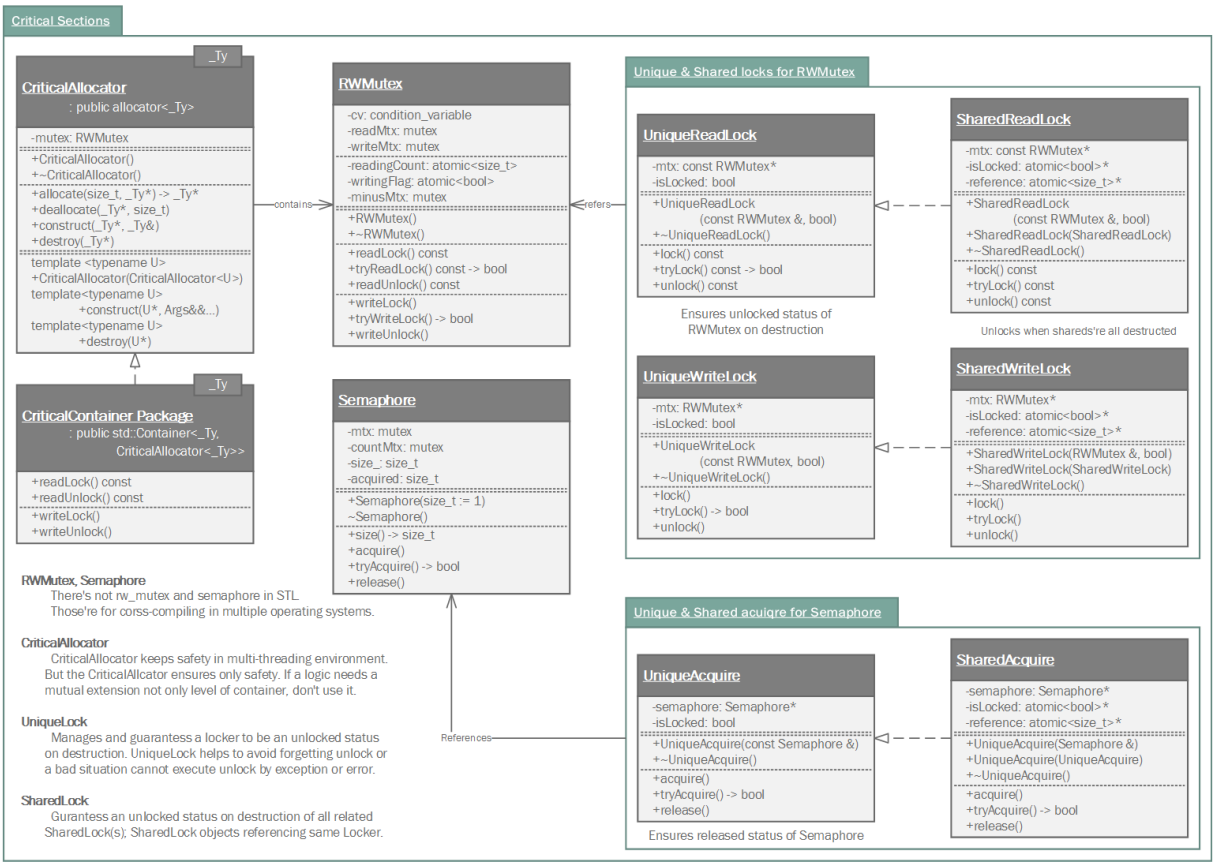
On default construction, the object acquires a RWMutex object, for whose locking and unlocking operations becomes responsible. When copy constructions, responsibilities of locking and unlocking RWMutex are shared with copied with those SharedReadLock objects.

The class shared_lock is a general-purpose shared mutex ownership wrapper allowing deferred locking, timed locking and transfer of lock ownership. Locking a shared_lock locks the associated shared mutex in shared mode (to lock it in exclusive mode, std::unique_lock can be used)

This class guarantees a unlocked status on destruction of all shared objects (even if not called explicitly). Therefore it is especially useful as an object with automatic duration, as it guarantees the RWMutex object is properly unlocked in case an exception is thrown.

Referenced comments of std::unique_lock

* http://www.cplusplus.com/reference/mutex/unique_lock/



Note

Though, that the SharedReadLock object does not manage the lifetime of the RWMutex object in any way: the duration of the RWMutex object shall extend at least until the destruction of the SharedRead↩Lock that manages it.

See also

library::UniqueAcquire
samchon::library

Author

Jeongho Nam

Definition at line 52 of file SharedReadLock.hpp.

10.125.2 Constructor & Destructor Documentation

10.125.2.1 SharedReadLock::SharedReadLock (const RWMutex & *semaphore*, bool *doLock* = `true`)

Construct from RWMutex.

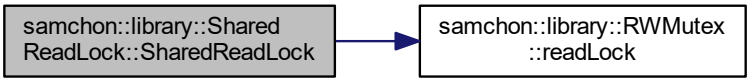
Parameters

semaphore	RWMutex to manage
doLock	Whether to lock directly or not

Definition at line 12 of file SharedReadLock.cpp.

References samchon::library::RWMutex::readLock().

Here is the call graph for this function:



10.125.2.2 SharedReadLock::SharedReadLock (const SharedReadLock & *obj*)

Copy Constructor.
Definition at line 22 of file SharedReadLock.cpp.
References isLocked, reference, and semaphore.

10.125.2.3 SharedReadLock::SharedReadLock (SharedReadLock && *obj*)

Move Constructor.
Definition at line 30 of file SharedReadLock.cpp.

10.125.2.4 SharedReadLock::~SharedReadLock ( )

Default Destructor.
Definition at line 43 of file SharedReadLock.cpp.

10.125.3 Member Function Documentation

10.125.3.1 void SharedReadLock::lock ( ) const

Lock on read.
Increases a reading count.
When write_lock is on a progress, wait until write_unlock to be called.

* Reading can be done by multiple sections.
* Reading can't be done when writing.

Warning

You've to call read_unlock when the reading work is terminated.

Definition at line 58 of file SharedReadLock.cpp.

10.125.3.2 void SharedReadLock::unlock ( ) const

Unlock of read.
Decreases a reading count.
When write_lock had done after read_lock, it continues by read_unlock if the reading count was 1 (read_unlock makes the count to be zero).
Definition at line 66 of file SharedReadLock.cpp.

10.125.4 Member Data Documentation

10.125.4.1 const RWMutex* samchon::library::SharedReadLock::semaphore [private]

Managed RWMutex.
Definition at line 58 of file SharedReadLock.hpp.
Referenced by SharedReadLock().

10.125.4.2 std::atomic<size_t>* samchon::library::SharedReadLock::reference [private]

Referencing count sharing same RWMutex.

Definition at line 63 of file SharedReadLock.hpp.

Referenced by SharedReadLock().

10.125.4.3 std::atomic<bool>* samchon::library::SharedReadLock::isLocked [private]

Whether the mutex was locked by SharedReadLock.

Definition at line 68 of file SharedReadLock.hpp.

Referenced by SharedReadLock().

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

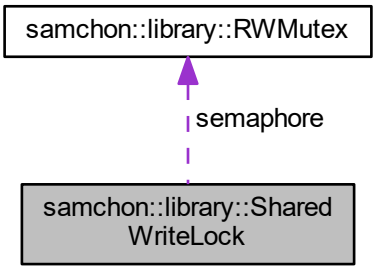
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/SharedReadLock.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/SharedReadLock.cpp

10.126 samchon::library::SharedWriteLock Class Reference

Shared lock from a RWMutex.

`#include <SharedWriteLock.hpp>`

Collaboration diagram for samchon::library::SharedWriteLock:



Public Member Functions

* SharedWriteLock (RWMutex &, bool=true)
  Construct from RWMutex.
* SharedWriteLock (const SharedWriteLock &)
  Copy Constructor.
* SharedWriteLock (SharedWriteLock &&)
  Move Constructor.
* ~SharedWriteLock ()
  Default Destructor.
* void lock ()
  Lock on writing.
* void unlock ()
  Unlock on writing.

Private Attributes

* RWMutex * semaphore
  Managed RWMutex.
* std::atomic< size_t > * reference

Referencing count sharing same RWMutex.

* std::atomic< bool > * isLocked

  Whether the mutex was locked by SharedWriteLock.

10.126.1 Detailed Description

Shared lock from a RWMutex.

A SharedWriteLock is an object manages a RWMutex with shared ownership in both states.

* locked
* unlocked

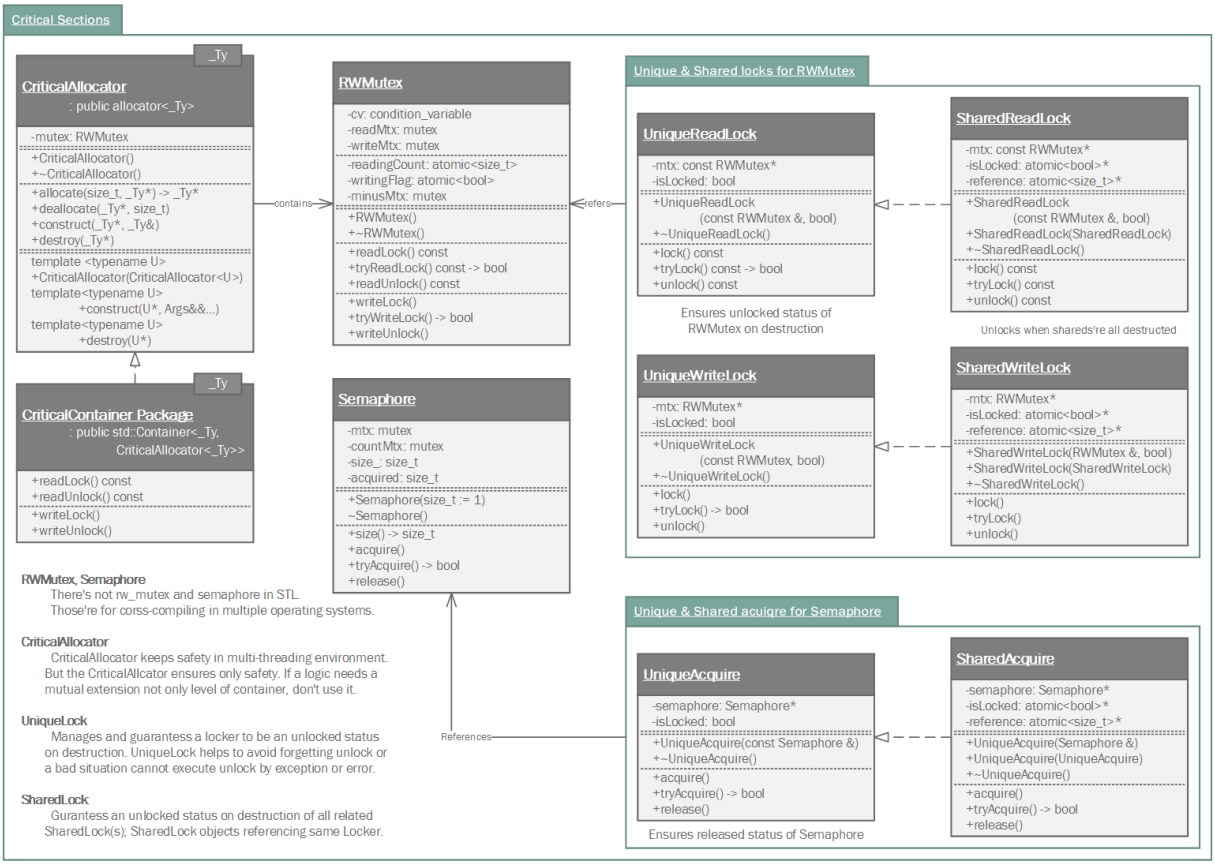
On default construction, the object acquires a RWMutex object, for whose locking and unlocking operations becomes responsible. When copy constructions, responsibilities of locking and unlocking RWMutex are shared with copied with those SharedWriteLock objects.

The class shared_lock is a general-purpose shared mutex ownership wrapper allowing deferred locking, timed locking and transfer of lock ownership. Locking a shared_lock locks the associated shared mutex in shared mode (to lock it in exclusive mode, std::unique_lock can be used)

This class guarantees a unlocked status on destruction of all shared objects (even if not called explicitly). Therefore it is especially useful as an object with automatic duration, as it guarantees the RWMutex object is properly unlocked in case an exception is thrown.

Referenced comments of std::unique_lock

* http://www.cplusplus.com/reference/mutex/unique_lock/



Note

Though, that the SharedWriteLock object does not manage the lifetime of the RWMutex object in any way: the duration of the RWMutex object shall extend at least until the destruction of the SharedWriteLock that manages it.

See also

library::UniqueAcquire
samchon::library

Author

Jeongho Nam

Definition at line 52 of file SharedWriteLock.hpp.

10.126.2 Constructor & Destructor Documentation

10.126.2.1 SharedWriteLock::SharedWriteLock (RWMutex & *semaphore*, bool *doLock* = `true`)

Construct from RWMutex.

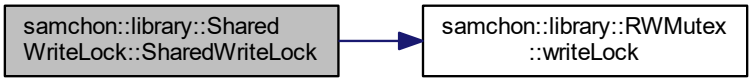
Parameters

semaphore	RWMutex to manage
doLock	Whether to lock directly or not

Definition at line 12 of file SharedWriteLock.cpp.

References samchon::library::RWMutex::writeLock().

Here is the call graph for this function:



10.126.2.2 SharedWriteLock::SharedWriteLock (const SharedWriteLock & *obj*)

Copy Constructor.

Definition at line 22 of file SharedWriteLock.cpp.

References isLocked, reference, and semaphore.

10.126.2.3 SharedWriteLock::SharedWriteLock (SharedWriteLock && *obj*)

Move Constructor.

Definition at line 30 of file SharedWriteLock.cpp.

10.126.2.4 SharedWriteLock::~SharedWriteLock ( )

Default Destructor.

Definition at line 43 of file SharedWriteLock.cpp.

10.126.3 Member Function Documentation

10.126.3.1 void SharedWriteLock::lock ( )

Lock on writing.

Changes writing flag to true.

If another write_lock or read_lock is on a progress, wait until them to be unlocked.

* Writing can be done by only a section at once.
* Writing can't be done when reading.

Note

You've to call write_unlock when writing work was terminated.

Definition at line 58 of file SharedWriteLock.cpp.

10.126.3.2 void SharedWriteLock::unlock ( )

Unlock on writing.

Definition at line 66 of file SharedWriteLock.cpp.

10.126.4 Member Data Documentation

10.126.4.1 RWMutex* samchon::library::SharedWriteLock::semaphore [private]

Managed RWMutex.

Definition at line 58 of file SharedWriteLock.hpp.

Referenced by SharedWriteLock().

10.126.4.2 std::atomic<size_t>* samchon::library::SharedWriteLock::reference [private]

Referencing count sharing same RWMutex.

Definition at line 63 of file SharedWriteLock.hpp.

Referenced by SharedWriteLock().

10.126.4.3 std::atomic<bool>* samchon::library::SharedWriteLock::isLocked [private]

Whether the mutex was locked by SharedWriteLock.

Definition at line 68 of file SharedWriteLock.hpp.

Referenced by SharedWriteLock().

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

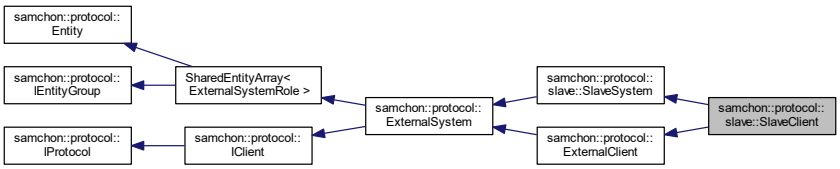
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/SharedWriteLock.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/SharedWriteLock.cpp

10.127 samchon::protocol::slave::SlaveClient Class Reference

A slave client.

`#include <SlaveClient.hpp>`

Collaboration diagram for samchon::protocol::slave::SlaveClient:



Public Member Functions

* SlaveClient ()
  Default Constructor.

Additional Inherited Members

10.127.1 Detailed Description

A slave client.

SlaveClient is a SlaveSystem specialized in client driver.

[Inherited]

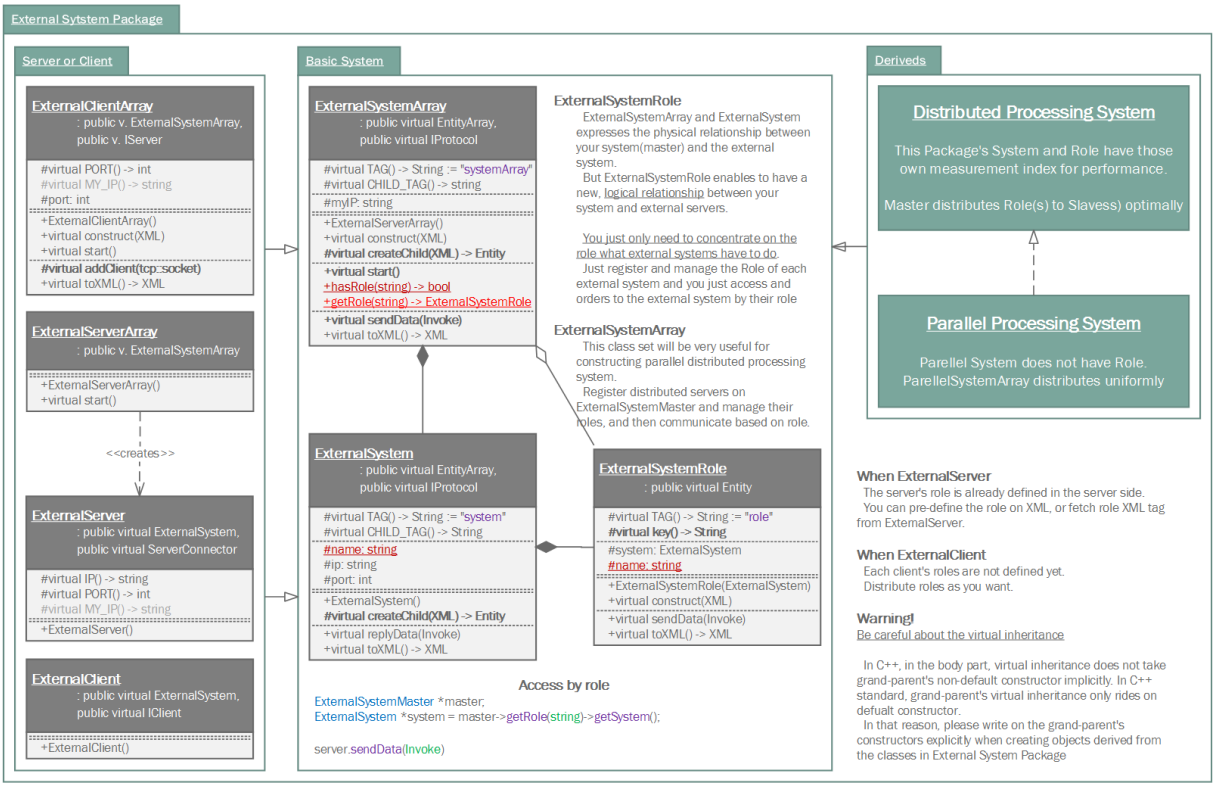
SlaveSystem, literally, means a slave system belongs to a maste system.

The SlaveSystem class is used in opposite side system of master::DistributedSystem and master::↩ ParallelSystem and reports elapsed time of each commmand (by Invoke message) for estimation of its performance.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, ExternalSystem is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystemRole.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem,

and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an ExternalSystem. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 22 of file SlaveClient.hpp.

10.127.2 Constructor & Destructor Documentation

10.127.2.1 SlaveClient::SlaveClient ( )

Default Constructor.

Definition at line 8 of file SlaveClient.cpp.

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

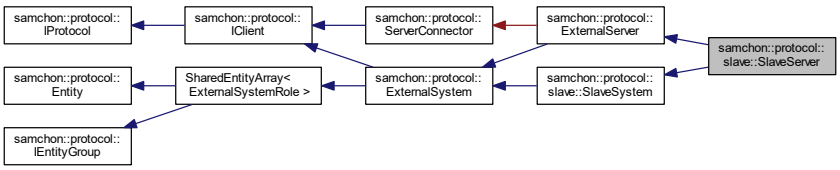
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/slave/SlaveClient.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/slave/SlaveClient.cpp

10.128 samchon::protocol::slave::SlaveServer Class Reference

A slave server.

`#include <SlaveServer.hpp>`

Collaboration diagram for samchon::protocol::slave::SlaveServer:



Public Member Functions

* SlaveServer ()
  Default Constructor.

Additional Inherited Members

10.128.1 Detailed Description

A slave server.

SlaveServer is a SlaveSystem specialized in server driver.

[Inherited]

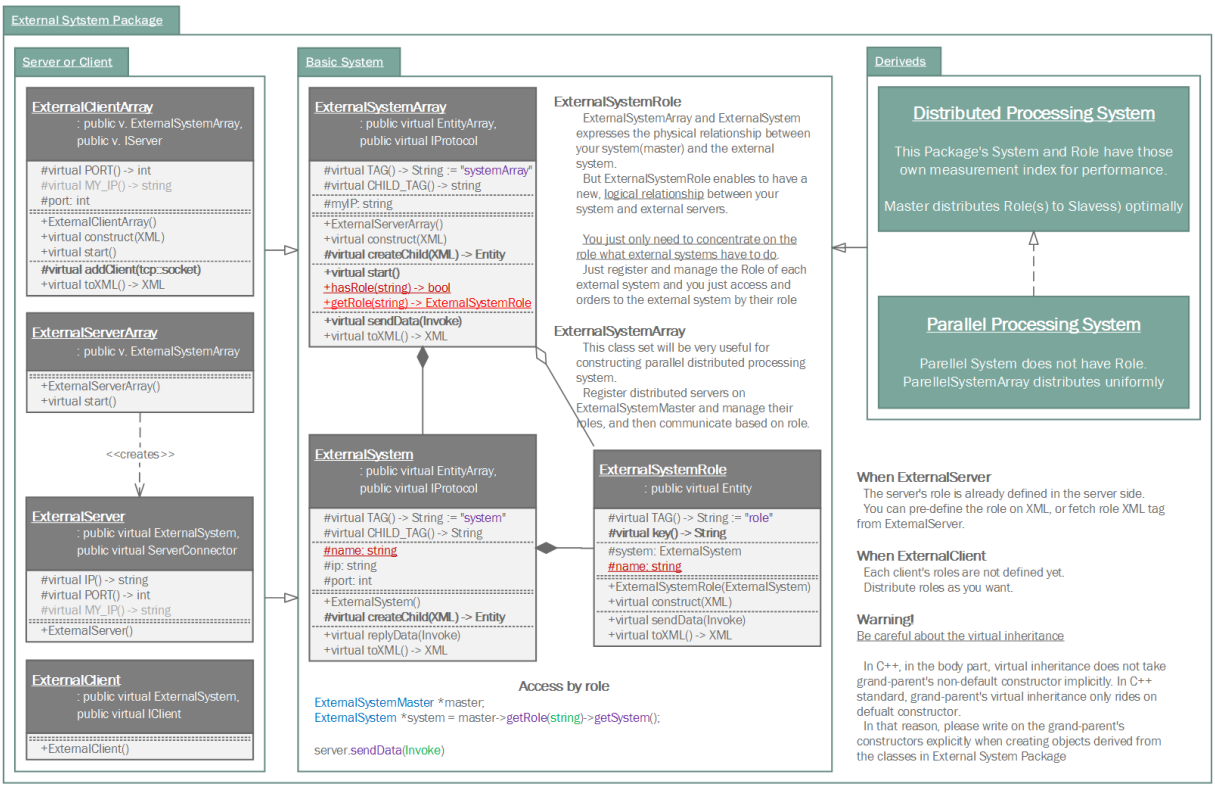
SlaveSystem, literally, means a slave system belongs to a maste system.

The SlaveSystem class is used in opposite side system of master::DistributedSystem and master::↩ ParallelSystem and reports elapsed time of each commmand (by Invoke message) for estimation of its performance.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, ExternalSystem is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystemRole.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an ExternalSystem. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 22 of file SlaveServer.hpp.

10.128.2 Constructor & Destructor Documentation

10.128.2.1 SlaveServer::SlaveServer ( )

Default Constructor.

Definition at line 8 of file SlaveServer.cpp.

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

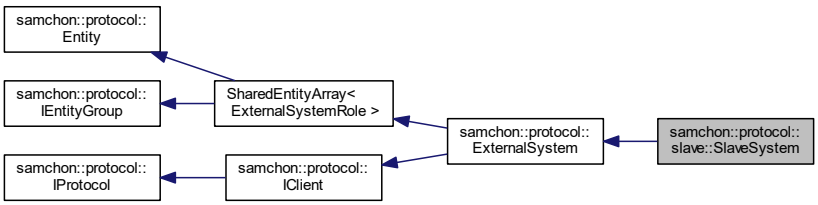
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/slave/SlaveServer.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/slave/SlaveServer.cpp

10.129 samchon::protocol::slave::SlaveSystem Class Reference

A slave system.

`#include <SlaveSystem.hpp>`

Collaboration diagram for samchon::protocol::slave::SlaveSystem:



Public Member Functions

* SlaveSystem ()
  Default Constructor.
* virtual void sendData (std::shared_ptr< Invoke >) override
  Sends message to a related system.

Protected Member Functions

* virtual void _replyData (std::shared_ptr< Invoke >) override
  Pre-processor for reporting elapsed time.

Additional Inherited Members

10.129.1 Detailed Description

A slave system.

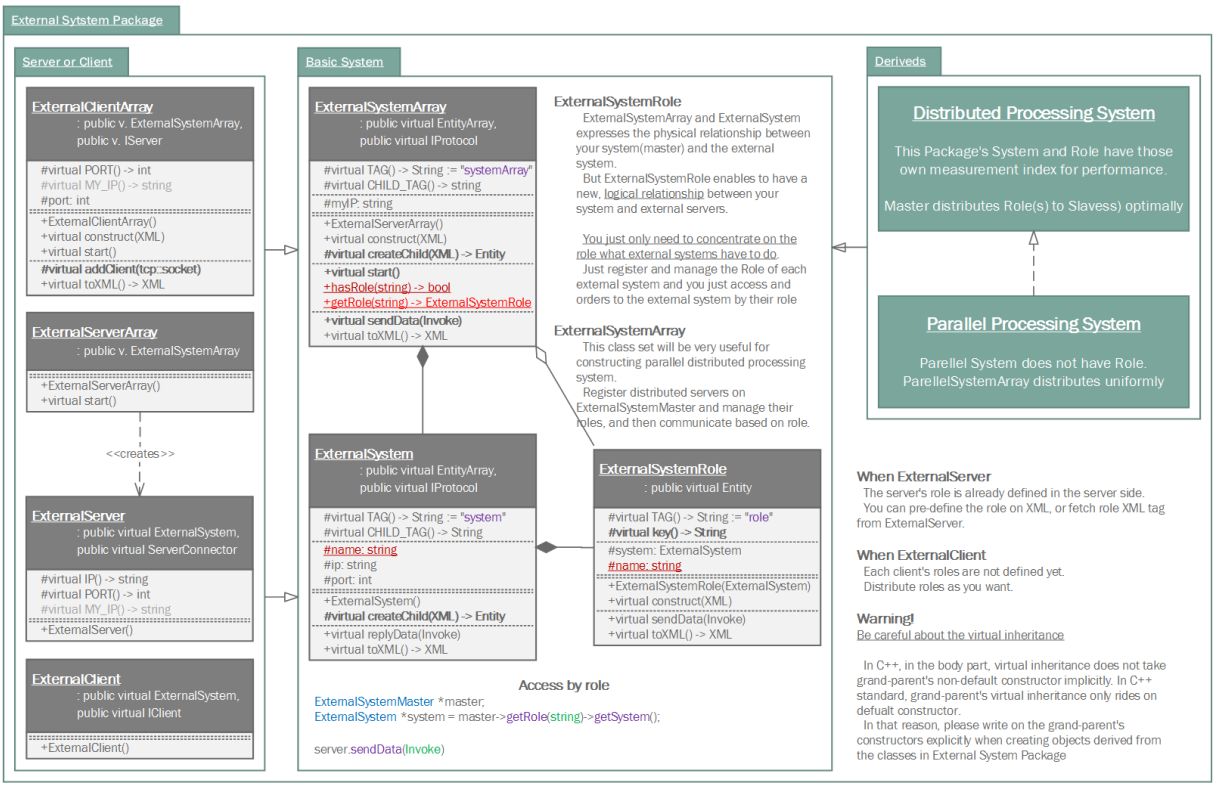
SlaveSystem, literally, means a slave system belongs to a maste system.

The SlaveSystem class is used in opposite side system of master::DistributedSystem and master::↩ ParallelSystem and reports elapsed time of each commmand (by Invoke message) for estimation of its performance.

[Inherited]

ExternalSystem is a boundary class interacting with an external system by network communication. Also, ExternalSystem is an abstract class that a network role, which one is server and which one is client, is not determined yet.

The ExternalSystem has ExternalSystemRole(s) groupped methods, handling Invoke message interacting with the external system, by subject or unit of a moudle. The ExternalSystemRole is categorized in a 'control'.



Note

The ExternalSystem class takes a role of interaction with external system in network level. However, within a framework of Samchon Framework, a boundary class like the ExternalSystem is not such important. You can find some evidence in a relationship between ExternalSystemArray, ExternalSystem and ExternalSystemRole.

Of course, the ExternalSystemRole is belonged to an ExternalSystem. However, if you access an ExternalSystemRole from an ExternalSystemArray directly, not passing by a belonged ExternalSystem, and send an Invoke message even you're not knowing which ExternalSystem is related in, it's called "Proxy pattern".

Like the explanation of "Proxy pattern", you can utilize an ExternalSystemRole as a proxy of an ExternalSystem. With the pattern, you can only concentrate on ExternalSystemRole itself, what to do with Invoke message, irrespective of the ExternalSystemRole is belonged to which ExternalSystem.

See also

samchon::protocol

Author

Jeongho Nam

Definition at line 25 of file SlaveSystem.hpp.

10.129.2 Constructor & Destructor Documentation

10.129.2.1 SlaveSystem::SlaveSystem ( )

Default Constructor.

Definition at line 12 of file SlaveSystem.cpp.

10.129.3 Member Function Documentation

10.129.3.1 void SlaveSystem::sendData (std::shared_ptr< Invoke >) [override], [virtual]

Sends message to a related system.

Parameters

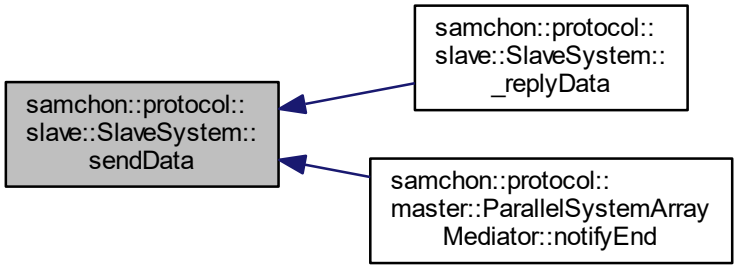
invoke	Invoke message to send

Reimplemented from samchon::protocol::IClient.

Definition at line 37 of file SlaveSystem.cpp.

Referenced by _replyData(), and samchon::protocol::master::ParallelSystemArrayMediator::notifyEnd().

Here is the caller graph for this function:



10.129.3.2 void SlaveSystem::_replyData (std::shared_ptr< Invoke >) [override], [protected], [virtual]

Pre-processor for reporting elapsed time.

[Inherited]

You can realize a pre-processing about the replied Invoke message by overriding the IClient::_reply↩Data().

Archiving a history log, filtering and truncating, let handling the message with a new thread, calculating performance by estimating elapsed time about the Invoke message, and so on. Anything you can define pre-processing works before IProtocol::replyData().

* Archive history log -> cloud service
* Estimating performance -> distributed and parallel processing system

Parameters

invoke	An invoke message to be pre-processed and shifted to IProtocol::replyData().

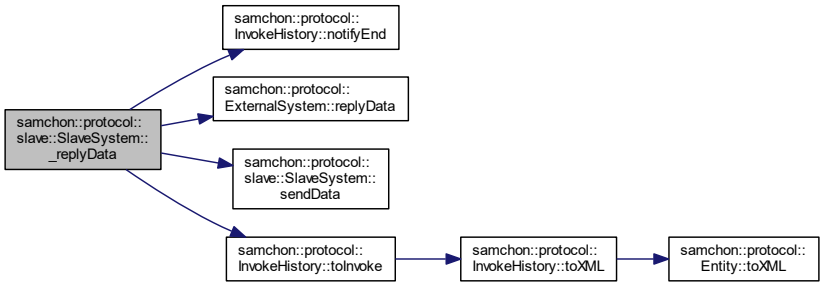
Reimplemented from samchon::protocol::IClient.

Reimplemented in samchon::protocol::slave::ParallelSystem.

Definition at line 16 of file SlaveSystem.cpp.

References samchon::protocol::InvokeHistory::notifyEnd(), samchon::protocol::ExternalSystem↩::replyData(), sendData(), and samchon::protocol::InvokeHistory::toInvoke().

Here is the call graph for this function:



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

* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/slave/SlaveSystem.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/slave/SlaveSystem.cpp

10.130 samchon::SmartPointer< _Ty > Class Template Reference

Global shared pointer

.

```
#include <SmartPointer.hpp>
```

Public Member Functions

* SmartPointer ()
  Default Constructor.
* SmartPointer (const _Ty *ptr)
  Constrct from pointer.
* SmartPointer (const SmartPointer &smartPointer)
  Copy Constructor.
* SmartPointer (SmartPointer &&smartPointer)
  Move constructor.
* ~SmartPointer ()
  Destroy SmartPointer.
* void reset (const _Ty *ptr)
  Reset pointer.
* auto get () const -> _Ty *
  Get pointer.
* auto operator-> () const -> _Ty *
  Dereference object membr.
* auto operator* () const -> _Ty &
  Dereference object.

Private Attributes

* _Ty * ptr
  A pointer managed by SmartPointer.

Static Private Attributes

* static std::map< _Ty *, size_t > useCountMap
  Map of use count of each pointer.
* static std::mutex mtx
  Mutex assigned to useCountMap.

10.130.1 Detailed Description

template<typename _Ty>class samchon::SmartPointer< _Ty >

Global shared pointer

.

SmartPointer is a global shared pointer based on map, map referencing.

Manages the storage of a pointer, providing a limited garbage-collection facility, possibly sharing that management with other objects.

Referenced comments of std::allocator.

* http://www.cplusplus.com/reference/memory/shared_ptr/

Template Parameters

The	type of managed object

Author

Jeongho Nam

Definition at line 22 of file SmartPointer.hpp.

10.130.2 Constructor & Destructor Documentation

10.130.2.1 template<typename _Ty> samchon::SmartPointer< _Ty >::SmartPointer ( ) [inline]

Default Constructor.

The object is empty (owns no pointer, use count of zero)

Definition at line 49 of file SmartPointer.hpp.

10.130.2.2 template<typename _Ty> samchon::SmartPointer< _Ty >::SmartPointer (const _Ty * *ptr*) [inline], [explicit]

Constrct from pointer.

The object owns ptr, setting the use count to add 1

Parameters

ptr	The pointer to own

Definition at line 60 of file SmartPointer.hpp.

10.130.2.3 template<typename _Ty> samchon::SmartPointer< _Ty >::SmartPointer (const SmartPointer< _Ty > & *smartPointer*) [inline]

Copy Constructor.

The object shares ownership of smartPointer's asset and increases the use count

Parameters

smartPointer	The object to copy

Definition at line 73 of file SmartPointer.hpp.

10.130.2.4 template<typename _Ty> samchon::SmartPointer< _Ty >::SmartPointer (SmartPointer< _Ty > && *smartPointer*) [inline]

Move constructor.

The object acquires the content managed by smartPointer

The ceding object becomes empty and there's no change on use count

Parameters

smartPointer	The object to move

Definition at line 84 of file SmartPointer.hpp.

10.130.2.5 template<typename _Ty> samchon::SmartPointer< _Ty >::~SmartPointer ( ) [inline]

Destroy SmartPointer.

Destroys the object. But, before, it may produce the following side effects depending on the use_count of member

- If use_count is greater than 1: The use count is decreased by 1.
- If use_count is 1 (i.e., the object is the unique owner of the managed pointer): the object pointed by its owned pointer is deleted.
- If use_count is zero (i.e., the object is empty), this destructor has no side effects.

Definition at line 105 of file SmartPointer.hpp.

10.130.3 Member Function Documentation

10.130.3.1 template<typename _Ty> void samchon::SmartPointer< _Ty >::reset (const _Ty * *ptr*) [inline]

Reset pointer.

Reset pointer to manage and shrink use count of previous pointer.

Additionally, a call to this function has the same side effects as if SmartPointer's destructor was called before its value changed (including the deletion of the managed object if this SmartPointer was unique).

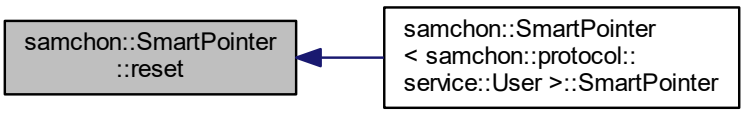
Parameters

Pointer	whose ownership is taken over by the object. Unlike std::shared_ptr, ptr being managed by another SmartPointer does not cause any problem

Definition at line 128 of file SmartPointer.hpp.

Referenced by samchon::SmartPointer< samchon::protocol::service::User >::SmartPointer().

Here is the caller graph for this function:



10.130.3.2 template<typename _Ty> auto samchon::SmartPointer< _Ty >::get ( ) const -> _Ty* [inline]

Get pointer.

Returns the stored pointer.

The stored pointer points to the object the shared_ptr object dereferences to, which is generally the same as its owned pointer.

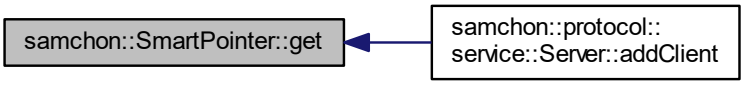
Returns

The stored pointer

Definition at line 154 of file SmartPointer.hpp.

Referenced by samchon::protocol::service::Server::addClient().

Here is the caller graph for this function:



10.130.3.3 template<typename _Ty> auto samchon::SmartPointer< _Ty >::operator-> ( ) const -> _Ty* [inline]

Dereference object membr.

Returns a pointer to the object pointed by the stored pointer in order to access one of its members. This member function shall not be called if the stored pointer is a null pointer.

It returns the same value as get().

Returns

A pointer to be managed by SmartPointer

Definition at line 170 of file SmartPointer.hpp.

10.130.3.4 template<typename _Ty> auto samchon::SmartPointer< _Ty >::operator* ( ) const -> _Ty& [inline]

Dereference object.

Returns a reference to the object pointerd by pointer. It is equivalent to: *get()

Returns

A reference to the object pointed

Definition at line 184 of file SmartPointer.hpp.

10.130.4 Member Data Documentation

10.130.4.1 template<typename _Ty> std::map< _Ty *, size_t > samchon::SmartPointer< _Ty >::useCountMap [static], [private]

Map of use count of each pointer.

Definition at line 28 of file SmartPointer.hpp.

10.130.4.2 template<typename _Ty> std::mutex samchon::SmartPointer< _Ty >::mtx [static], [private]

Mutex assigned to useCountMap.

Definition at line 33 of file SmartPointer.hpp.

10.130.4.3 template<typename _Ty> _Ty* samchon::SmartPointer< _Ty >::ptr [private]

A pointer managed by SmartPointer.

Definition at line 39 of file SmartPointer.hpp.

Referenced by samchon::SmartPointer< samchon::protocol::service::User >::get().

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

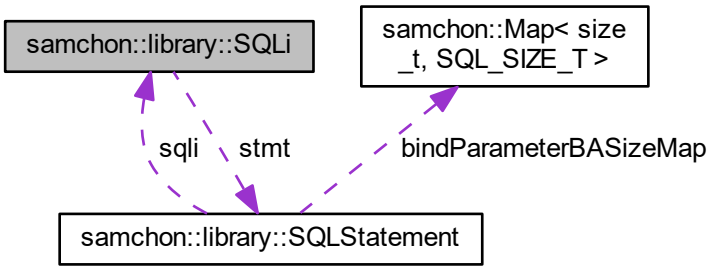
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/SmartPointer.hpp

10.131 samchon::library::SQLi Class Reference

A SQL interface; DBMS connector.

`#include <SQLi.hpp>`

Collaboration diagram for samchon::library::SQLi:



Public Member Functions

* SQLi (const std::string &driver, int port)
  Construct from driver name and port.
* virtual void connect (const std::string &ip, const std::string &db, const std::string &id, const std::string &pwd)
  Connect to the DBMS
  .
* virtual void disconnect ()
  Disconnect from DBMS.
* virtual auto createStatement () -> std::shared_ptr< SQLStatement >
  Factory method for creating SQLStatement.

Protected Attributes

* std::string driver
  Driver; name of DBMS used for connection.
* int port
  Port number of DBMS.

Private Attributes

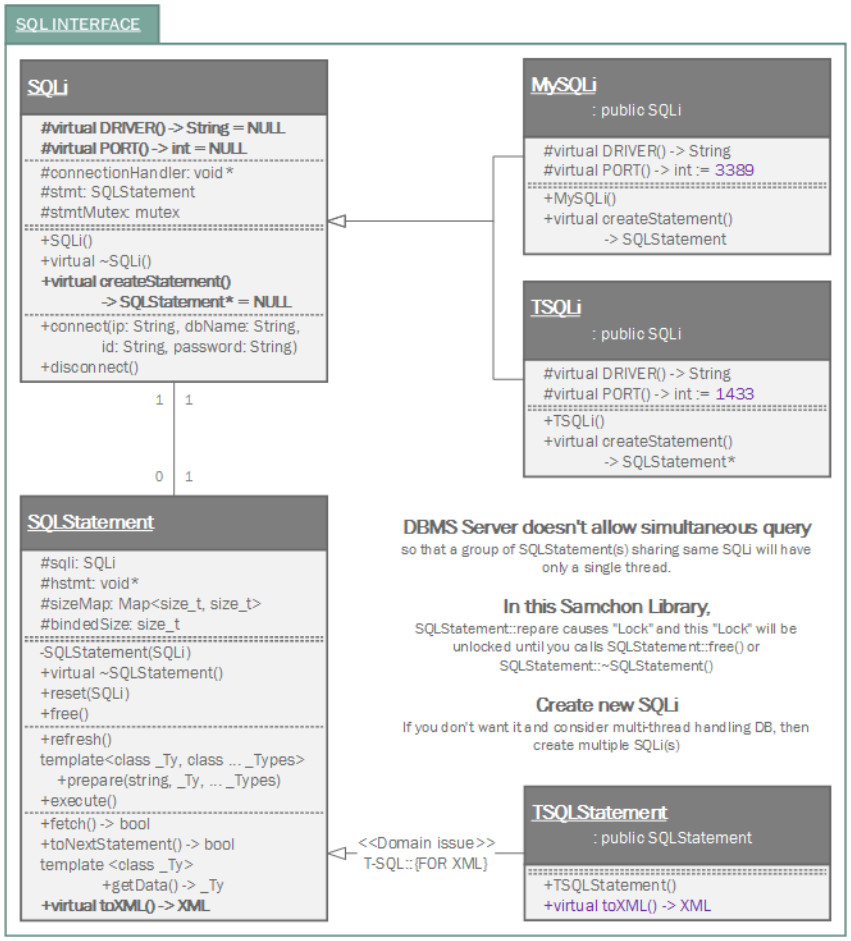
* void * hdbc
  Handler of DB-connector.
* SQLStatement * stmt
  SQLStatement's pointer linked with the SQLi.
* std::mutex stmtMutex
  Mutex ensuring concurrency with SQLStatement.

10.131.1 Detailed Description

A SQL interface; DBMS connector.

SQLi is an abstract class managing connection (session) with a specific database and manages SQL statements are executed and results are returned within the context of a connection.

The reason why SQLStatement(s) are managed by SQLi is because of a principle of DBMS, DBMS system doesn't allow simultaneous query from a session (connection). Only a query (process0) is allowed at a time. If you try to simultaneous query from a SQLi, ODBC throws error. To avoid the error, an execute of query from SQLStatement will lock a mutex of SQLi to ensure exclusiveness.



Note

To ensure the exclusiveness, you've to make SQLStatement from SQLi. Do not make SQLStatement by yourself. call SQLi::createSQLStatement() instead. Even you make a derived class from SQLStatement, don't make its constructor to have public accessor.

Becuase execution of a sql statement causes lock on mutex in SQLi, you've to destruct the SQLStatement or call SQLStatement::free() method(). If you don't, the mutex will not be unlocked, thus you can't do anything by the SQLi.

See also

samchon::library

Author

Jeongho Nam

Definition at line 42 of file SQLi.hpp.

10.131.2 Constructor & Destructor Documentation

10.131.2.1 SQLi::SQLi (const std::string & *driver,* int *port*)

Construct from driver name and port.

Parameters

driver	Driver name of DBMS
port	Port number of DBMS

Definition at line 21 of file SQLi.cpp.

10.131.3 Member Function Documentation

10.131.3.1 void SQLi::connect (const std::string & *ip*, const std::string & *db*, const std::string & *id*, const std::string & *pwd*) [virtual]

Connect to the DBMS

.

Parameters

ip	An IP address of the DBMS
db	A name of database to access
id	Your own ID authorified from DBMS
pwd	A password for the ID

Exceptions

exception	Error message from DBMS

Definition at line 38 of file SQLi.cpp.

10.131.3.2 void SQLi::disconnect ( ) [virtual]

Disconnect from DBMS.
Definition at line 67 of file SQLi.cpp.

10.131.3.3 auto SQLi::createStatement ( ) -> std::shared_ptr<SQLStatement> [virtual]

Factory method for creating SQLStatement.
Recommend to create SQLStatement by this method. Direct creation is not recommended as the reason of domain problem of each DBMS

Returns

A SQLStatement matched for the domain SQLi

Reimplemented in samchon::library::TSQLi.
Definition at line 33 of file SQLi.cpp.

10.131.4 Member Data Documentation

10.131.4.1 std::string samchon::library::SQLi::driver [protected]

Driver; name of DBMS used for connection.

* MySQL: "{MySQL Server}"
* MS-SQL: "{SQL Server}"
* PL-SQL: "{Oracle}"

Definition at line 55 of file SQLi.hpp.

10.131.4.2 int samchon::library::SQLi::port [protected]

Port number of DBMS.

* MySQL: 3306
* MS-SQL: 1433
* PL-SQL: 1521

Definition at line 65 of file SQLi.hpp.

10.131.4.3 void* samchon::library::SQLi::hdbc [private]

Handler of DB-connector.

Definition at line 76 of file SQLi.hpp.

Referenced by samchon::library::SQLStatement::SQLStatement().

10.131.4.4 SQLStatement* samchon::library::SQLi::stmt [private]

SQLStatement's pointer linked with the SQLi.

Definition at line 81 of file SQLi.hpp.

10.131.4.5 std::mutex samchon::library::SQLi::stmtMutex [private]

Mutex ensuring concurrency with SQLStatement.

Definition at line 86 of file SQLi.hpp.

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

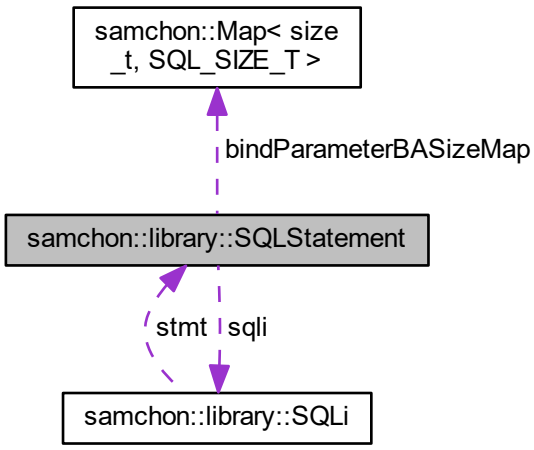
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/SQLi.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/SQLi.cpp

10.132 samchon::library::SQLStatement Class Reference

A sql statement.

`#include <SQLStatement.hpp>`

Collaboration diagram for samchon::library::SQLStatement:



Public Member Functions

* void free ()
  Free the sql statement.
* void refresh ()
  Refresh the sql statement.
* template<typename _Ty , typename... _Args>
  void prepare (const std::string &sql, const _Ty &val, const _Args &...args)
  Prepare a sql statement.
* void execute ()

Execute the prepared sql statement.

* void executeDirectly (const std::string &)
  Execute sql-statement direclty.
* auto fetch () const -> bool
  Fetch a record.
* auto next () const -> bool
  Move cursor to the next sql-statement.
* auto size () const -> size_t
  Get size of columns.
* template<typename _Ty >
  auto at (size_t index) const -> _Ty
  Get column's data by its index.
* template<typename _Ty >
  auto get (const std::string &) const -> _Ty
  Get a column data by its name.
* virtual auto toXML () const -> std::shared_ptr< XML >
  Result sets to XML.

Protected Member Functions

* SQLStatement (SQLi *sqli)
  Protected Constructor.

Protected Attributes

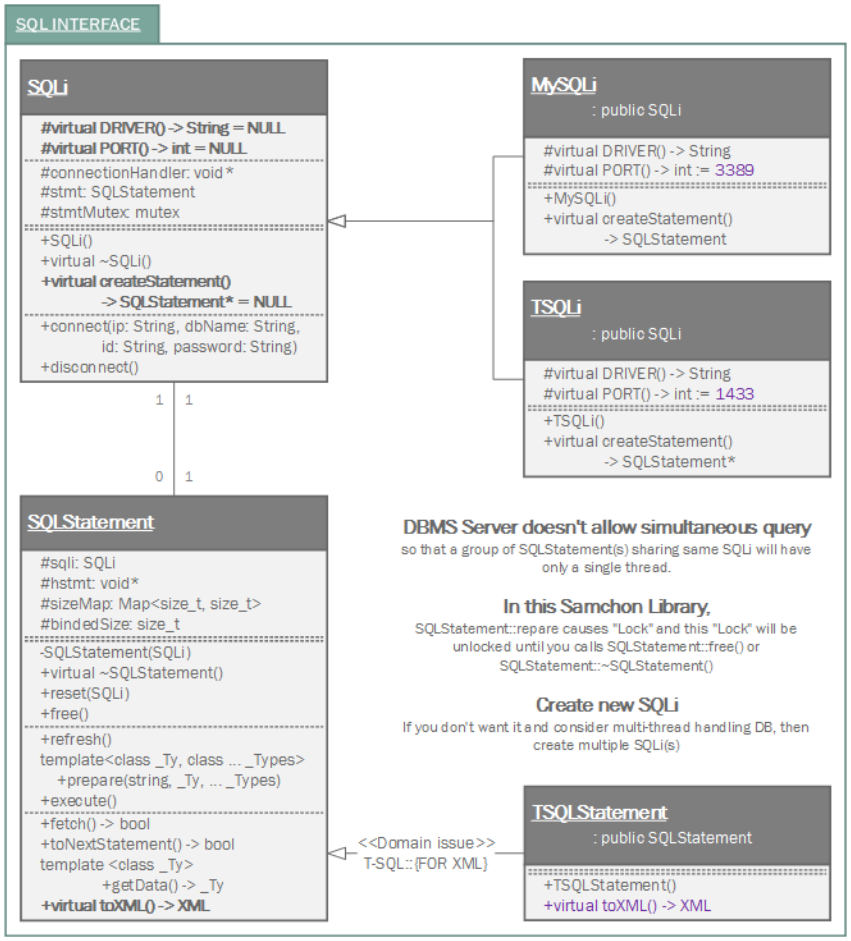
* SQLi * sqli
  SQLi who created the SQLStatement.
* void * hstmt
  Handler of sql statement (OBDC)
* size_t bindParameterCount
  Count of binded parameters
  .
* Map< size_t, SQL_SIZE_T > bindParameterBASizeMap
  A map for binary size.

10.132.1 Detailed Description

A sql statement.

A SQLStatement instance is used to executing a SQL statement and returning the results it produces against a SQL database that is opened through a SQLi instance.

Through the SQLi, it's the reason why a principle of DBMS, DBMS system doesn't allow simultaneous query from a session (connection). Only a query (process) is allowed at a time. If you try to simultaneous query from a SQLi, ODBC throws error. To avoid the error, an execute of query from SQLStatement will lock a mutex of SQLi to ensure exclusiveness.



Note

To ensure the exclusiveness, you've to make SQLStatement from SQLi. Do not make SQLStatement by yourself. call SQLi::createSQLStatement() instead. Even you make a derived class from SQLStatement, don't make its constructor to have public accessor.

Becuase execution of a sql statement causes lock on mutex in SQLi, you've to destruct the SQL↩ Statement or call SQLStatement::free() method(). If you don't, the mutex will not be unlocked, thus you can't do anything by the SQLi.

See also

samchon::library

Author

Jeongho Nam

Definition at line 44 of file SQLStatement.hpp.

10.132.2 Constructor & Destructor Documentation

10.132.2.1 SQLStatement::SQLStatement (SQLi * *sqli*) [protected]

Protected Constructor.

SQLStatement's constructor have to created by SQLi::createStatement().

Note

Don't create SQLStatement by yourself.

SQLStatement has to be created by SQLi::createStatement().

Parameters

sqli	Parent SQLi who created the SQLStatement

Definition at line 22 of file SQLStatement.cpp.
References samchon::library::SQLi::hdbc.

10.132.3 Member Function Documentation

10.132.3.1 void SQLStatement::free ( )

Free the sql statement.
Definition at line 41 of file SQLStatement.cpp.

10.132.3.2 void SQLStatement::refresh ( )

Refresh the sql statement.
Definition at line 53 of file SQLStatement.cpp.

10.132.3.3 template<typename _Ty , typename... _Args> void samchon::library::SQLStatement::prepare (const std::string & *sql,* const _Ty & *val,* const _Args &... *args*) [inline]

Prepare a sql statement.
Prepare a sql statement with parameters to bind for execution

Warning

Be careful for destructions of binded parameters

Parameters

sql	A sql-statement to prepare
...	args The parameters to bind

Definition at line 126 of file SQLStatement.hpp.

10.132.3.4 void SQLStatement::execute ( )

Execute the prepared sql statement.

Exceptions

exception	Error message from DBMS

Definition at line 73 of file SQLStatement.cpp.

10.132.3.5 void SQLStatement::executeDirectly (const std::string & *sql*)

Execute sql-statement direclty.
Executes the given sql-statement without preparing or binding any parameter

Warning

* Cannot use if prepare statement has already defined. Use execute instead.
* Not recommended when the case of dynamic sql and the sql-statement is not for procedure but for direct access to table. Use prepare and execute instead.

Parameters

sql	sql-statement you want to execute

Exceptions

exception	Error message from DBMS
exception	Method prepare is already called

Definition at line 79 of file SQLStatement.cpp.

10.132.3.6 auto SQLStatement::fetch ( ) const -> bool

Fetch a record.

Returns

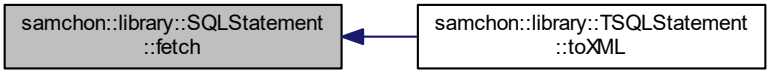
Whether succeded to fetch a record.

False means there's not any record or previous record was the last.

Definition at line 98 of file SQLStatement.cpp.

Referenced by samchon::library::TSQLStatement::toXML().

Here is the caller graph for this function:



10.132.3.7 auto SQLStatement::next ( ) const -> bool

Move cursor to the next sql-statement.

Returns

Whether succeded to move cursor to the next statement.

Definition at line 94 of file SQLStatement.cpp.

10.132.3.8 auto SQLStatement::size ( ) const -> size_t

Get size of columns.

Returns the number of columns in a result set.

Warning

Not size of rows.

Definition at line 118 of file SQLStatement.cpp.

10.132.3.9 template<typename _Ty > auto samchon::library::SQLStatement::at (size_t *index*) const -> _Ty [inline]

Get column's data by its index.

Returns column's data from fetched-record by specified column index

Parameters

index	Index number of a column wants to get

Returns

Data stored in the record at the position of specifield column

Definition at line 203 of file SQLStatement.hpp.

10.132.3.10 template<typename _Ty > auto samchon::library::SQLStatement::get (const std::string &) const -> _Ty [inline]

Get a column data by its name.

Returns column's data from fetchched-recrod by specified column name

Get data from fetched-record by specified column name

Parameters

name	Name of a column wants to get

Returns

Data stored in the record at the position of specifield column

Definition at line 224 of file SQLStatement.hpp.

10.132.3.11 auto SQLStatement::toXML ( ) const -> std::shared_ptr<XML> [virtual]

Result sets to XML.

Converts the records of current sql-statement to XML.

Recommends to override for each DBMS's domain XML rule.

Returns

XML representing records of the statement

Reimplemented in samchon::library::TSQLStatement.

Definition at line 172 of file SQLStatement.cpp.

10.132.4 Member Data Documentation

10.132.4.1 SQLi* samchon::library::SQLStatement::sqli [protected]

SQLi who created the SQLStatement.

Definition at line 61 of file SQLStatement.hpp.

10.132.4.2 void* samchon::library::SQLStatement::hstmt [protected]

Handler of sql statement (OBDC)

Definition at line 66 of file SQLStatement.hpp.

10.132.4.3 size_t samchon::library::SQLStatement::bindParameterCount [protected]

Count of binded parameters

.

Definition at line 74 of file SQLStatement.hpp.

10.132.4.4 Map<size_t, SQL_SIZE_T> samchon::library::SQLStatement::bindParameterBASizeMap [protected]

A map for binary size.

When calls ByteArray::size(), the returned size_t value can't be kept until

Definition at line 82 of file SQLStatement.hpp.

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

* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/SQLStatement.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/SQLStatement.cpp

10.133 samchon::library::StringUtil Class Reference

Utility class for string.

`#include <StringUtil.hpp>`

Static Public Member Functions

* template<typename _Ty , typename... _Args>
  static auto substitute (const std::string &format, const _Ty &val, const _Args &...args) -> std::string
  Substitutes "{n}" tokens within the specified string with the respective arguments passed in.
* template<typename _Ty , typename... _Args>
  static auto substituteSQL (const std::string &format, const _Ty &value, const _Args &...args) -> std::string
  Substitutes "{n}" tokens within the specified sql-string with the respective arguments passed in.
  .
* static auto isNumeric (const std::string &str) -> bool
  Returns wheter the std::string represents Number or not
  .
* static auto toNumber (const std::string &str) -> double
  Number std::string to Number having ',' symbols.
* static auto numberFormat (double val, int precision=2) -> std::string
* static auto percentFormat (double val, int precision=2) -> std::string
  Returns a percentage string converted from the number rounded off from specified precision with "," symbols
  ex) percentFormat(11.3391, 1) => 1,133.9%.
* static auto colorNumberFormat (double value, int precision=2, double delimiter=0.0) -> std::string
  Returns a string converted from the number rounded off from specified precision with "," symbols and color tag
  ex) numberFormat(17151.339, 2) => <font color="red">17,151.34</font>
* static auto colorPercentFormat (double value, int precision=2, double delimiter=0.0) -> std::string
  Returns a percentage string converted from the number rounded off from specified precision with "," symbols
  ex) percentFormat(11.3391, 1) => 1,133.9%.
* static auto trim (const std::string &val, const std::vector< std::string > &delims) -> std::string
  Removes all designated characters from the beginning and end of the specified string.
* static auto ltrim (const std::string &val, const std::vector< std::string > &delims) -> std::string
  Removes all designated characters from the beginning of the specified string.
* static auto rtrim (const std::string &val, const std::vector< std::string > &delims) -> std::string
  Removes all designated characters from the end of the specified string.
* static auto finds (const std::string &str, const std::vector< std::string > &delims, size_t start↩Index=0) -> IndexPair< std::string >
  Finds first occurence in string.
* static auto rfinds (const std::string &str, const std::vector< std::string > &delims, size_t end↩Index=SIZE_MAX) -> IndexPair< std::string >
  Finds last occurence in string.
* static auto substring (const std::string &str, size_t startIndex, size_t endIndex=SIZE_MAX) -> std↩::string
  Generates a substring.
* static auto between (const std::string &str, const std::string &start="", const std::string &end="") -> std::string
  Generates a substring.
* static auto addTab (const std::string &str, size_t n=1) -> std::string
  Adds tab() character to first position of each line.
* static auto split (const std::string &str, const std::string &delim) -> std::vector< std::string >
  Generates substrings.
* static auto betweens (const std::string &str, const std::string &start="", const std::string &end="") -> std::vector< std::string >
  Generates substrings.
* static auto toLowerCase (const std::string &str) -> std::string
  Returns a string that all uppercase characters are converted to lowercase
  .
* static auto toUpperCase (const std::string &str) -> std::string
* static auto replaceAll (const std::string &str, const std::string &before, const std::string &after) -> std::string
  Returns a string specified word is replaced.

* static auto replaceAll (const std::string &str, const std::vector< std::pair< std::string, std::string >> &pairs) -> std::string
  Returns a string specified words are replaced.

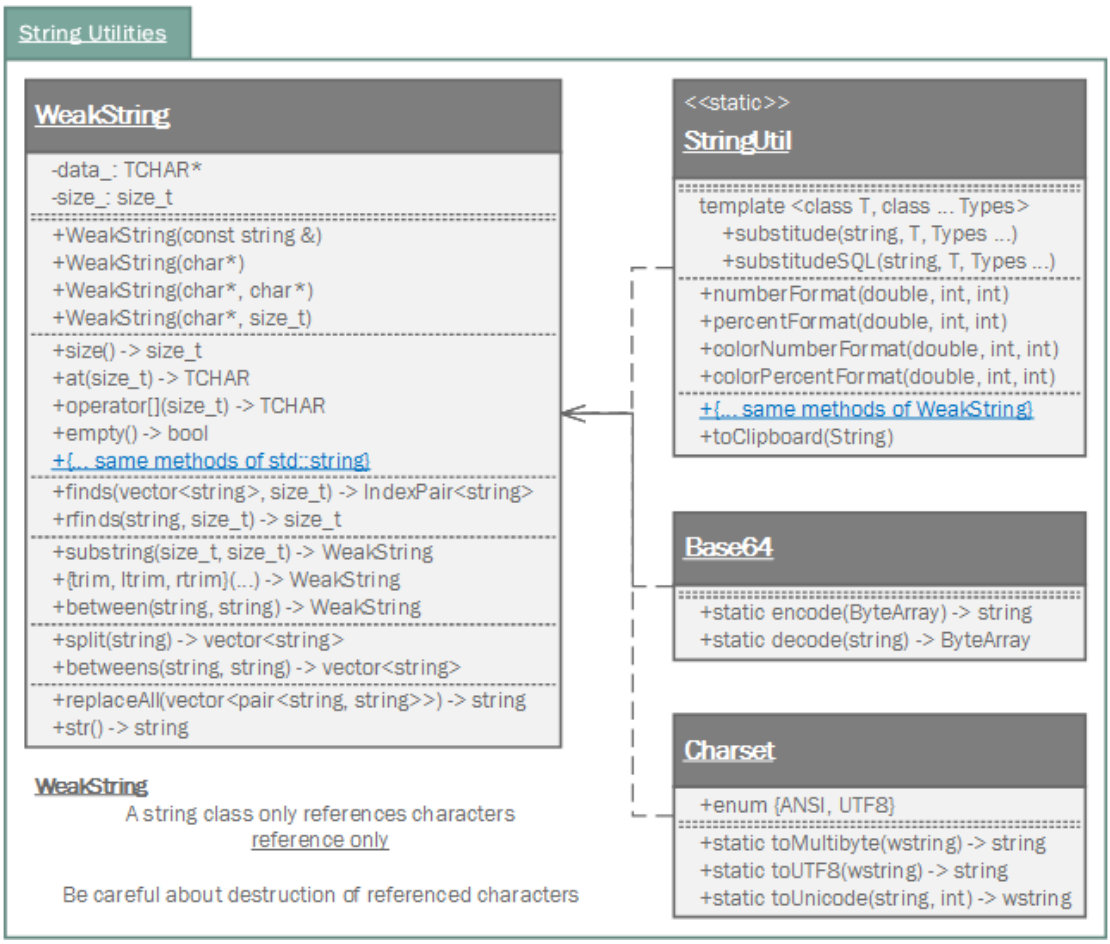
10.133.1 Detailed Description

Utility class for string.

StringUtil is an utility class providing lots of static methods for std::string.

There are two methods to strength std::string to have addictional uility methods like trim and split. The way of first is to make std::string class inheriting from std::string. The second is to make StringUtil class having static methods.

But those methods have problems. std::string class violates standard and StringUtil class violates principle of Object-Oriented Design. For the present, I've made the StringUtil class, but if you have a good opinion about the issue, please write your opinion on my github.



See also

library::WeakString
samchon::library

Author

Jeongho Nam

Definition at line 36 of file StringUtil.hpp.

10.133.2 Member Function Documentation

10.133.2.1 template<typename _Ty , typename... _Args> static auto samchon::library::StringUtil::substitute (const std::string & *format,* const _Ty & *val,* const _Args &... *args*) -> std::string [inline], [static]

Substitutes "{n}" tokens within the specified string with the respective arguments passed in.

Parameters

format	The string to make substitutions in. This string can contain special tokens of the form {n}, where n is a zero based index, that will be replaced with the additional parameters found at that index if specified
val	Target value to substitute the minimum {n} tokens
args	Additional parameters that can be substituted in the str parameter at each {n} location, where n is an integer (zero based) index value into the varadics of values specified.

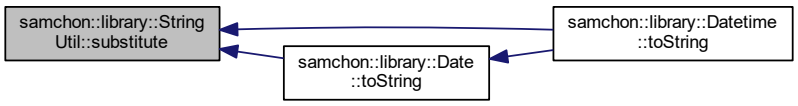
Returns

New string with all of the {n} tokens replaced with the respective arguments specified.

Definition at line 54 of file StringUtil.hpp.

Referenced by samchon::library::Datetime::toString(), and samchon::library::Date::toString().

Here is the caller graph for this function:



10.133.2.2 template<typename _Ty , typename... _Args> static auto samchon::library::StringUtil::substituteSQL (const std::string & *format,* const _Ty & *value,* const _Args &... *args*) -> std::string [inline], [static]

Substitutes "{n}" tokens within the specified sql-string with the respective arguments passed in.

.

Warning

substituteSQL creates the dynamic sql-statement.
Not recommended when the dynamic sql-statement is not only for procedure.

Parameters

format	The string to make substitutions in. This string can contain special tokens of the form {n}, where n is a zero based index, that will be replaced with the additional parameters found at that index if specified
val	Target value to substitute the minimum {n} tokens
args	Additional parameters that can be substituted in the str parameter at each {n} location, where n is an integer (zero based) index value into the varadics of values specified.

Returns

New sql-string with all of the {n} tokens replaced with the respective arguments specified.

Definition at line 79 of file StringUtil.hpp.

10.133.2.3 auto StringUtil::isNumeric (const std::string & *str*) -> bool [static]

Returns wheter the std::string represents Number or not

.

Parameters

str	Target std::string to check

Returns

Whether the std::string can be converted to Number or not

Definition at line 49 of file StringUtil.cpp.

10.133.2.4 auto StringUtil::toNumber (const std::string & *str*) -> double [static]

Number std::string to Number having ',' symbols.

Parameters

str	Target std::string you want to convert to Number

Returns

Number from std::string

Definition at line 63 of file StringUtil.cpp.

10.133.2.5 auto StringUtil::numberFormat (double *val*, int *precision* = 2) -> std::string [static]

Returns a string converted from the number rounded off from specified precision with "," symbols ex) numberFormat(17151.339, 2) => 17,151.34

Parameters

val	A number wants to convert to string
precision	Target precision of roundoff

Returns

A string representing the number with roundoff and "," symbols

Definition at line 69 of file StringUtil.cpp.

10.133.2.6 auto StringUtil::percentFormat (double *val*, int *precision* = 2) -> std::string [static]

Returns a percentage string converted from the number rounded off from specified precision with "," symbols
ex) percentFormat(11.3391, 1) => 1,133.9%.

Warning

Do not multiply by 100 to the value representing percent

Parameters

val	A number wants to convert to percentage string
precision	Target precision of roundoff

Definition at line 109 of file StringUtil.cpp.

10.133.2.7 auto StringUtil::colorNumberFormat (double *value*, int *precision* = 2, double *delimiter* = 0.0) -> std::string [static]

Returns a string converted from the number rounded off from specified precision with "," symbols and color tag
ex) numberFormat(17151.339, 2) => <font color="red">17,151.34</font>
Which color would be chosen

* Number is positive, color is RED
* Number is zero (0), color is BLACK
* Number is negative, color is BLUE

Parameters

val	A number wants to convert to colored string
precision	Target precision of roundoff

Returns

A colored string representing the number with roundoff and "," symbols

Definition at line 116 of file StringUtil.cpp.

10.133.2.8 auto StringUtil::colorPercentFormat (double *value*, int *precision* = 2, double *delimiter* = 0.0) -> std::string [static]

Returns a percentage string converted from the number rounded off from specified precision with "," symbols
ex) percentFormat(11.3391, 1) => 1,133.9%.

Warning

Do not multiply by 100 to the value representing percent

Parameters

val	A number wants to convert to percentage string
precision	Target precision of roundoff

Definition at line 131 of file StringUtil.cpp.

10.133.2.9 static auto samchon::library::StringUtil::trim (const std::string & *val*, const std::vector< std::string > & *delims*) -> std::string [static]

Removes all designated characters from the beginning and end of the specified string.

Parameters

str	The string should be trimmed
delims	Designated character(s)

Returns

Updated string where designated characters was removed from the beginning and end

10.133.2.10 static auto samchon::library::StringUtil::ltrim (const std::string & *val*, const std::vector< std::string > & *delims*) -> std::string [static]

Removes all designated characters from the beginning of the specified string.

Parameters

str	The string should be trimmed
delims	Designated character(s)

Returns

Updated string where designated characters was removed from the beginning

10.133.2.11 static auto samchon::library::StringUtil::rtrim (const std::string & *val*, const std::vector< std::string > & *delims*) -> std::string [static]

Removes all designated characters from the end of the specified string.

Parameters

str	The string should be trimmed
delims	Designated character(s)

Returns

Updated string where designated characters was removed from the end

10.133.2.12 auto StringUtil::finds (const std::string & *str*, const std::vector< std::string > & *delims*, size_t *startIndex* = 0) -> IndexPair<std::string> [static]

Finds first occurence in string.
Finds first occurence position of each delim in the string after startIndex and returns the minimum position of them

If startIndex is not specified, then starts from 0.
If failed to find any substring, returns -1 (std::string::npos)

Parameters

str	Target string to find
delims	The substrings of target(str) which to find
startIndex	Specified starting index of find. Default is 0

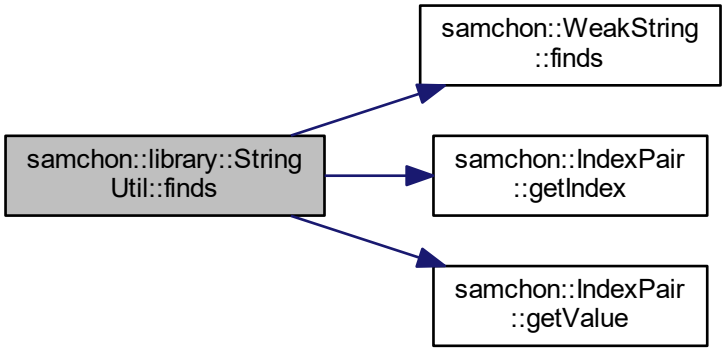
Returns

pair<size_t := position, string := matched substring>

Definition at line 194 of file StringUtil.cpp.

References samchon::WeakString::finds(), samchon::IndexPair< _Ty >::getIndex(), and samchon::IndexPair< _Ty >::getValue().

Here is the call graph for this function:



10.133.2.13 auto StringUtil::rfinds (const std::string & *str*, const std::vector< std::string > & *delims*, size_t *endIndex* = SIZE_MAX) -> IndexPair<std::string> [static]

Finds last occurence in string.
Finds last occurence position of each delim in the string before endIndex and returns the maximum position of them

If index is not specified, then starts str.size() - 1
If failed to find any substring, returns -1 (std::string::npos)

Parameters

str	Target string to find
delims	The substrings of target(str) which to find
endIndex	Specified starting index of find. Default is str.size() - 1

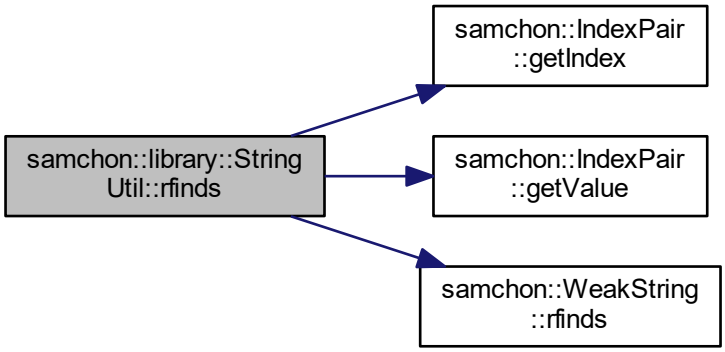
Returns

pair<size_t := position, string := matched substring>

Definition at line 200 of file StringUtil.cpp.

References samchon::IndexPair< _Ty >::getIndex(), samchon::IndexPair< _Ty >::getValue(), and samchon::WeakString::rfinds().

Here is the call graph for this function:



10.133.2.14 auto StringUtil::substring (const std::string & *str*, size_t *startIndex*, size_t *endIndex* = `SIZE_MAX`) -> std::string `[static]`

Generates a substring.

Extracts a string consisting of the character specified by startIndex and all characters up to endIndex - 1 If endIndex is not specified, string::size() will be used instead.
If endIndex is greater than startIndex, then those will be swapped

Parameters

str	Target string to be applied substring
startIndex	Index of the first character. If startIndex is greater than endIndex, those will be swapped
endIndex	Index of the last character - 1. If not specified, then string::size() will be used instead

Returns

Extracted string by specified index(es)

Definition at line 208 of file StringUtil.cpp.

10.133.2.15 auto StringUtil::between (const std::string & *str*, const std::string & *start* = " ", const std::string & *end* = " ") -> std::string `[static]`

Generates a substring.

Extracts a substring consisting of the characters from specified start to end It's same with str.substring(? = (str.find(start) + start.size()), str.find(end, ?))
ex) between("ABCD[EFGH]IJK", "[", "]") => "EFGH"

If start is not specified, extracts from begin of the string to end
If end is not specified, extracts from start to end of the string
If start and end are all omitted, returns str, itself.

Parameters

str	Target string to be applied between
start	A string for separating substring at the front
end	A string for separating substring at the end

Returns

substring by specified terms

Definition at line 213 of file StringUtil.cpp.

10.133.2.16 auto StringUtil::addTab (const std::string & *str*, size_t *n* = 1) -> std::string [static]

Adds tab() character to first position of each line.

Parameters

str	Target str to add tabs
n	The size of tab to be added for each line

Returns

A string added multiple tabs

Definition at line 218 of file StringUtil.cpp.

10.133.2.17 auto StringUtil::split (const std::string & *str*, const std::string & *delim*) -> std::vector<std::string> [static]

Generates substrings.

Splits a string into an array of substrings by dividing the specified delimiter

Parameters

str	Target string to split
delim	The pattern that specifies where to split this string

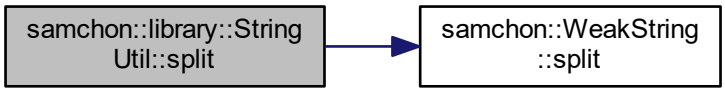
Returns

An array of substrings

Definition at line 239 of file StringUtil.cpp.

References samchon::WeakString::split().

Here is the call graph for this function:



10.133.2.18 auto StringUtil::betweens (const std::string & *str*, const std::string & *start* = " ", const std::string & *end* = " ") -> std::vector<std::string> [static]

Generates substrings.

Splits a string into an array of substrings dividing by specified delimeters of start and end
It's the array of substrings adjusted the between

If startStr is omitted, it's same with the split by endStr not having last item
If endStr is omitted, it's same with the split by startStr not having first item
If startStr and endStar are all omitted, returns {str}

Parameters

str	Target string to split by between
start	A string for separating substring at the front If omitted, it's same with split(end) not having last item
end	A string for separating substring at the end If omitted, it's same with split(start) not having first item

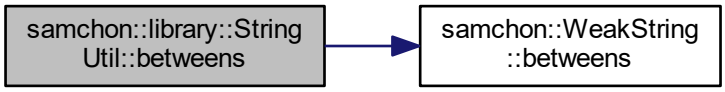
Returns

An array of substrings

Definition at line 249 of file StringUtil.cpp.

References samchon::WeakString::betweens().

Here is the call graph for this function:



10.133.2.19 auto StringUtil::toLowerCase (const std::string & *str*) -> std::string [static]

Returns a string that all uppercase characters are converted to lowercase
.

Parameters

str	Target string to convert uppercase to lowercase

Returns

A string converted to lowercase

Definition at line 264 of file StringUtil.cpp.

10.133.2.20 auto StringUtil::toUpperCase (const std::string & *str*) -> std::string [static]

Returns a string all lowercase characters are converted to uppercase

Parameters

str	Target string to convert lowercase to uppercase

Returns

A string converted to uppercase

Definition at line 268 of file StringUtil.cpp.

10.133.2.21 static auto samchon::library::StringUtil::replaceAll (const std::string & *str*, const std::string & *before*, const std::string & *after*) -> std::string [static]

Returns a string specified word is replaced.

Parameters

str	Target string to replace
before	Specific word you want to be replaced
after	Specific word you want to replace

Returns

A string specified word is replaced

10.133.2.22 static auto samchon::library::StringUtil::replaceAll (const std::string & *str*, const std::vector< std::pair< std::string, std::string >> & *pairs*) -> std::string [static]

Returns a string specified words are replaced.

Parameters

str	Target string to replace
pairs	A specific word's pairs you want to replace and to be replaced

Returns

A string specified words are replaced

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

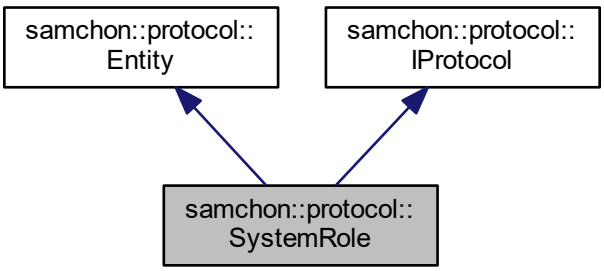
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/StringUtil.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/StringUtil.cpp

10.134 samchon::protocol::SystemRole Class Reference

A role belongs to a system.

`#include <SystemRole.hpp>`

Collaboration diagram for samchon::protocol::SystemRole:



Public Member Functions

* SystemRole ()
  Default Constructor.
* virtual void construct (std::shared_ptr< library::XML >) override
  Construct data of the Entity from an XML object.
* virtual auto key () const -> std::string override
  Get a key that can identify the Entity uniquely.
* auto hasListener (const std::string &) const -> bool
  Test whether has a listener in the role.
* virtual auto TAG () const -> std::string override
  A tag name when represented by XML.
* virtual auto toXML () const -> std::shared_ptr< library::XML > override
  Get an XML object represents the Entity.

Protected Attributes

* std::string name
  A name representing the role.
* std::set< std::string > listeners
  Listeners belongs to the role.

10.134.1 Detailed Description

A role belongs to a system.

SystemRole is an abstract and a 'control' class groupping methods, handling Invoke messages interacting with an system that the SystemRole is belonged to, by a subject or unit of a module.

Author

Jeongho Nam

Definition at line 23 of file SystemRole.hpp.

10.134.2 Constructor & Destructor Documentation

10.134.2.1 SystemRole::SystemRole ( )

Default Constructor.

Definition at line 12 of file SystemRole.cpp.

10.134.3 Member Function Documentation

10.134.3.1 void SystemRole::construct (std::shared_ptr< library::XML >) [override], [virtual]

Construct data of the Entity from an XML object.

Overrides the construct() method and fetch data of member variables from the XML.

By recommended guidance, data representing member variables are contained in properties of the put XML object.

Parameters

xml	An xml used to construct data of entity

Implements samchon::protocol::Entity.

Definition at line 17 of file SystemRole.cpp.

References listeners, and name.

10.134.3.2 auto SystemRole::key ( ) const -> std::string [override], [virtual]

Get a key that can identify the Entity uniquely.

If identifier of the Entity is not atomic value, returns a string represents the composite identifier. If identifier of the Entity is not string, converts the identifier to string and returns the string.

Returns

An identifier

Reimplemented from samchon::protocol::Entity.

Definition at line 33 of file SystemRole.cpp.

References name.

10.134.3.3 auto SystemRole::hasListener (const std::string &) const -> bool

Test whether has a listener in the role.

Definition at line 38 of file SystemRole.cpp.

10.134.3.4 auto SystemRole::TAG ( ) const -> std::string [override], [virtual]

A tag name when represented by XML.

Returns

A tag name

Implements samchon::protocol::Entity.

Definition at line 46 of file SystemRole.cpp.

10.134.3.5 auto SystemRole::toXML ( ) const -> std::shared_ptr<library::XML> [override], [virtual]

Get an XML object represents the Entity.

Returns an XML object that can represents the Entity containing member variables into properties.

A member variable (not object, but atomic value like number, string or date) is categorized as a property within the framework of entity side. Thus, when overriding a toXML() method and archiving member variables to an XML object to return, puts each variable to be a property belongs to only an XML object.

Don't archive the member variable of atomic value to XML::value causing enormouse creation of XML objects to number of member variables. An Entity must be represented by only an XML instance (tag).

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Returns

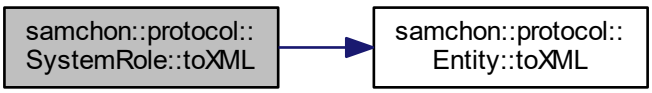
An XML object representing the Entity.

Reimplemented from samchon::protocol::Entity.

Definition at line 51 of file SystemRole.cpp.

References listeners, name, and samchon::protocol::Entity::toXML().

Here is the call graph for this function:



10.134.4 Member Data Documentation

10.134.4.1 std::string samchon::protocol::SystemRole::name [protected]

A name representing the role.

Definition at line 33 of file SystemRole.hpp.

Referenced by construct(), key(), and toXML().

10.134.4.2 std::set<std::string> samchon::protocol::SystemRole::listeners [protected]

Listeners belongs to the role.

Definition at line 38 of file SystemRole.hpp.

Referenced by construct(), and toXML().

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

* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/SystemRole.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/SystemRole.cpp

10.135 samchon::example::tsp::Travel Class Reference

Represent a travel containning Point(s)

```
#include <Travel.hpp>
```

Public Member Functions

* Travel ()
  Default Constructor.
* Travel (const Travel &)
  Copy Constructor.
* Travel (Travel &&)
  Move Constructor.
* auto operator< (const Travel &) const -> bool
  Compare which object is less.
* auto toString () const -> std::string
  Convert the Travel to String.

Private Member Functions

* auto calcDistance () const -> double
  Calculate distance to move.

Private Attributes

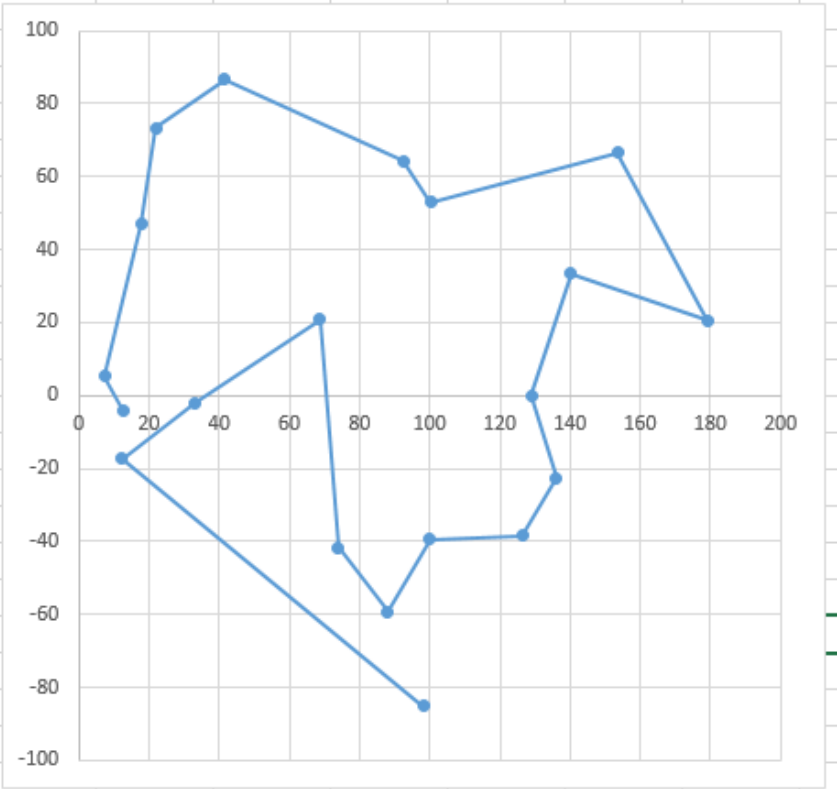
* double distance
  Estimated hours to move.

10.135.1 Detailed Description

Represent a travel containning Point(s)

Travel class is not only an array containing Point(s) but also a sequence listing of the Point(s).

Distance: 64,896.70 km		
uid	longitude	latitude
18	12.5233	-3.95529
15	7.23275	5.18342
7	17.7528	46.8665
2	22.0434	73.4133
20	41.6615	86.5372
13	92.427	64.3369
1	100.345	53.1067
9	153.521	66.5119
3	179.118	20.5073
10	140.113	33.3537
17	128.885	0.054607
4	136.012	-22.4915
6	126.352	-38.5031
16	99.9775	-39.3696
11	87.8859	-59.3663
14	73.8477	-41.5867
5	68.6563	20.8528
12	33.2162	-1.92823
8	12.1839	-17.3552
19	98.0967	-85.195



Author

Jeongho Nam

Definition at line 24 of file Travel.hpp.

10.135.2 Constructor & Destructor Documentation

10.135.2.1 Travel::Travel ( )

Default Constructor.

Definition at line 13 of file Travel.cpp.

References distance.

10.135.2.2 Travel::Travel (const Travel & *travel*)

Copy Constructor.

Travel is an array of Point(s) as gene(s). The copy constructor of Travel doesn't copy member variables which are related to G.A. because the sequence of gene(Point)s can be shuffled by process of evolution of genetic algorithm.

Definition at line 18 of file Travel.cpp.

References distance.

10.135.2.3 Travel::Travel (Travel && *travel*)

Move Constructor.

Definition at line 23 of file Travel.cpp.

References distance.

10.135.3 Member Function Documentation

10.135.3.1 auto Travel::calcDistance ( ) const -> double [private]

Calculate distance to move.

Definition at line 32 of file Travel.cpp.

References distance.

Referenced by toString().

Here is the caller graph for this function:

10.135.3.2 auto Travel::operator< (const Travel & *travel*) const -> bool

Compare which object is less.

Compares distances to move. The result is used to optimize and evolve genes by G.A.

Parameters

Target	object to compare

Returns

Whether this object is less than target

Definition at line 44 of file Travel.cpp.

10.135.3.3 auto Travel::toString ( ) const -> std::string

Convert the Travel to String.

Have of form of tab and enter delimeters for Excel.

```
{$uid1}    {$x}    {$y}



{$uid2}    {$x}    {$y}



{$uid3}    {$x}    {$y}
```

...

Returns

A string can represent the Travel

Definition at line 52 of file Travel.cpp.

References calcDistance().

Here is the call graph for this function:

10.135.4 Member Data Documentation

10.135.4.1 double samchon::example::tsp::Travel::distance [private]

Estimated hours to move.

A variable for avoiding duplicated calculation

See also

calcDistance()

Definition at line 36 of file Travel.hpp.

Referenced by calcDistance(), and Travel().

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

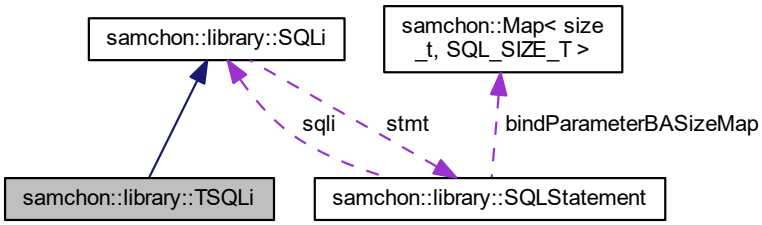
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/tsp/Travel.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/tsp/Travel.cpp

10.136 samchon::library::TSQLi Class Reference

A SQL interface for T-SQL.

`#include <TSQLi.hpp>`

Collaboration diagram for samchon::library::TSQLi:



Public Member Functions

* TSQLi (int port=1433)
  Construct from port.
* virtual auto createStatement () -> std::shared_ptr< SQLStatement >
  Factory method for creating T-SQL statement.

Additional Inherited Members

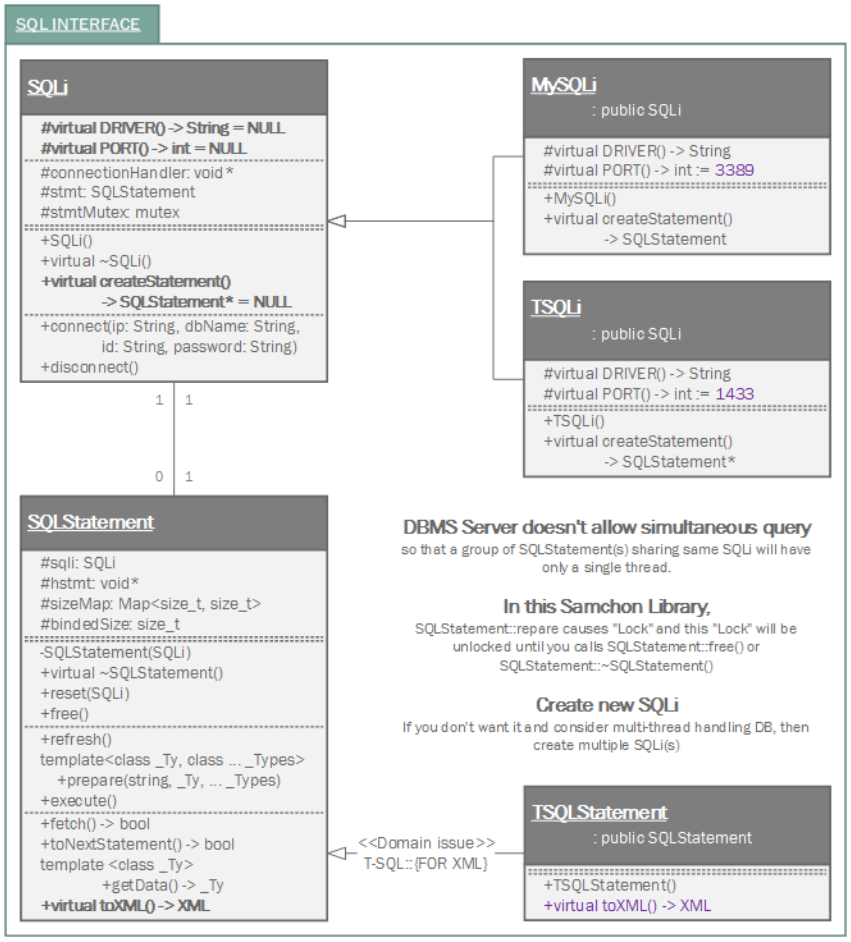
10.136.1 Detailed Description

A SQL interface for T-SQL.

[Inherited]

SQLi is an abstract class managing connection (session) with a specific database and manages SQL statements are executed and results are returned within the context of a connection.

The reason why SQLStatement(s) are managed by SQLi is because of a principle of DBMS, DBMS system doesn't allow simultaneous query from a session (connection). Only a query (process0) is allowed at a time. If you try to simultaneous query from a SQLi, ODBC throws error. To avoid the error, an execute of query from SQLStatement will lock a mutex of SQLi to ensure exclusiveness.



Note

To ensure the exclusiveness, you've to make SQLStatement from SQLi. Do not make SQLStatement by yourself. call SQLi::createSQLStatement() instead. Even you make a derived class from SQLStatement, don't make its constructor to have public accessor.

Becuase execution of a sql statement causes lock on mutex in SQLi, you've to destruct the SQLStatement or call SQLStatement::free() method(). If you don't, the mutex will not be unlocked, thus you can't do anything by the SQLi.

See also

samchon::library

Author

Jeongho Nam

Definition at line 19 of file TSQLi.hpp.

10.136.2 Constructor & Destructor Documentation

10.136.2.1 TSQLi::TSQLi (int *port* = 1433)

Construct from port.

Parameters

port	Port number of T-SQL. default is 1433.

Definition at line 8 of file MSSQLi.cpp.

10.136.3 Member Function Documentation

10.136.3.1 auto TSQLi::createStatement ( ) -> std::shared_ptr<SQLStatement> [virtual]

Factory method for creating T-SQL statement.

To realize domain function in T-SQL, 'FOR XML' statement.

[Inherited]

SQLi::createStatement()

Returns

A T-SQL statement

Reimplemented from samchon::library::SQLi.

Definition at line 13 of file MSSQLi.cpp.

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

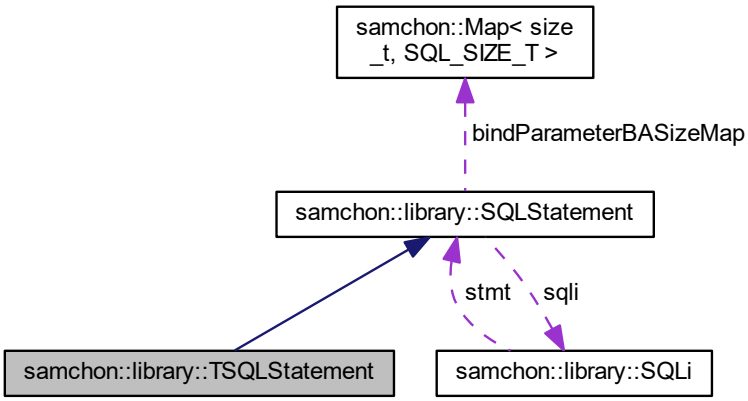
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/TSQLi.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/MSSQLi.cpp

10.137 samchon::library::TSQLStatement Class Reference

A t-sql statement.

`#include <TSQLStatement.hpp>`

Collaboration diagram for samchon::library::TSQLStatement:



Public Member Functions

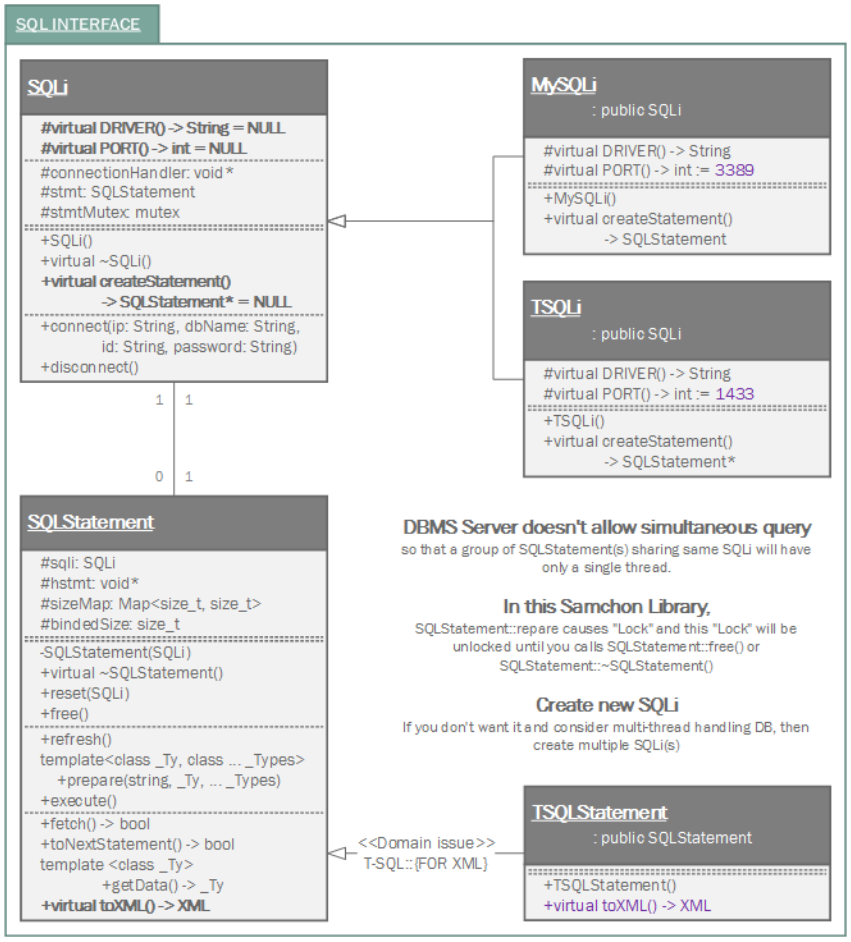
* virtual auto toXML () const -> std::shared_ptr< XML > override
  Return an XML representing records.

Additional Inherited Members

10.137.1 Detailed Description

A t-sql statement.
TSQLStatement is a SQLStatement for adjusting domain function f T-SQL, 'FOR XML' statement.
[Inherited]

A SQLStatement instance is used to executing a SQL statement and returning the results it produces against a SQL database that is opened through a SQLi instance.
Through the SQLi, it's the reason why a principle of DBMS, DBMS system doesn't allow simultaneous query from a session (connection). Only a query (process) is allowed at a time. If you try to simultaneous query from a SQLi, ODBC throws error. To avoid the error, an execute of query from SQLStatement will lock a mutex of SQLi to ensure exclusiveness.



Note

To ensure the exclusiveness, you've to make SQLStatement from SQLi. Do not make SQLStatement by yourself. call SQLi::createSQLStatement() instead. Even you make a derived class from SQLStatement, don't make its constructor to have public accessor.
Becuase execution of a sql statement causes lock on mutex in SQLi, you've to destruct the SQL↩ Statement or call SQLStatement::free() method(). If you don't, the mutex will not be unlocked, thus you can't do anything by the SQLi.

See also

samchon::library

Author

Jeongho Nam

Definition at line 21 of file TSQLStatement.hpp.

10.137.2 Member Function Documentation

10.137.2.1 auto TSQLStatement::toXML ( ) const -> std::shared_ptr<XML> [override], [virtual]

Return an XML representing records.

Get an XML object generated from a sql statement containing 'FOR XML'

Returns

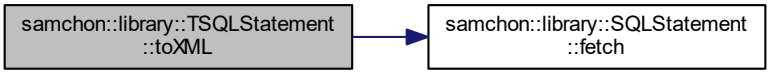
XML represents records

Reimplemented from samchon::library::SQLStatement.

Definition at line 14 of file MSSQLStatement.cpp.

References samchon::library::SQLStatement::fetch().

Here is the call graph for this function:



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

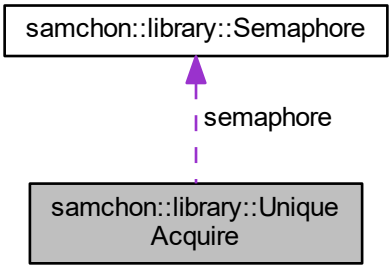
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/TSQLStatement.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/MSSQLStatement.cpp

10.138 samchon::library::UniqueAcquire Class Reference

Unique acquire from a Semaphore.

`#include <UniqueAcquire.hpp>`

Collaboration diagram for samchon::library::UniqueAcquire:



Public Member Functions

* UniqueAcquire (Semaphore &, bool=true)
  Construct from semaphore.
* UniqueAcquire (const UniqueAcquire &)=delete
  Prohibited Copy Constructor.
* UniqueAcquire (UniqueAcquire &&)
  Move Constructor.
* ~UniqueAcquire ()
  Default Destructor.
* void acquire ()
  Acquire admission.
* void release ()
  Release an admission.

Private Attributes

* Semaphore * semaphore
  Managed semaphore.
* bool isLocked
  Whether the semaphore was acquired by the UniqueLock.

10.138.1 Detailed Description

Unique acquire from a Semaphore.

A UniqueAcquire is an object manages a Semaphore with unique ownership in both states.

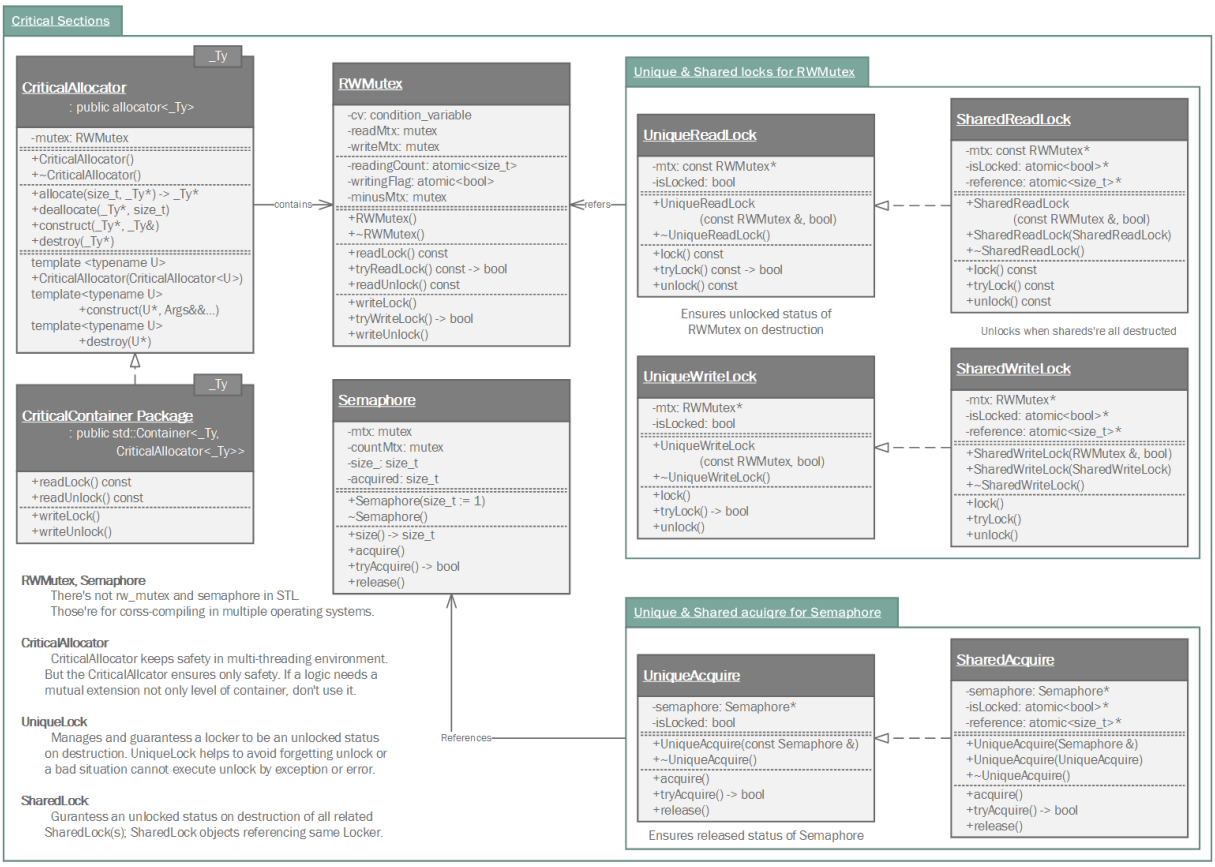
* acquired
* released

On construction (or by move-assigning to it), the object acquires a semaphore object, for whose acquiring and releasing operations becomes responsible.

This class guarantees a released status on destruction (even if not called explicitly). Therefore it is especially useful as an object with automatic duration, as it guarantees the semaphore object is properly released in case an exception is thrown.

Referenced comments of std::unique_lock

* http://www.cplusplus.com/reference/mutex/unique_lock/



Note

Though, that the UniqueAcquire object does not manage the lifetime of the semaphore object in any way: the duration of the semaphore object shall extend at least until the destruction of the UniqueAcquire that manages it.

The UniqueAcquire is movable, but not copyable.

See also

samchon::library

Author

Jeongho Nam

Definition at line 41 of file UniqueAcquire.hpp.

10.138.2 Constructor & Destructor Documentation

10.138.2.1 UniqueAcquire::UniqueAcquire (Semaphore & *semaphore*, bool *doLock* = `true`)

Construct from semaphore.

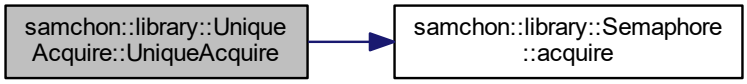
Parameters

semaphore	Semaphore to manage
doLock	Whether to lock directly or not

Definition at line 9 of file UniqueAcquire.cpp.

References samchon::library::Semaphore::acquire(), isLocked, and semaphore.

Here is the call graph for this function:



10.138.2.2 samchon::library::UniqueAcquire::UniqueAcquire (const UniqueAcquire &) [delete]

Prohibited Copy Constructor.

UniqueAcquire can't be copied. Use pointer, reference instead.

If what you want is UniqueAcquire(s) references each other and unlock when all related UniqueAcquire objects are destructed, SharedAcquire is the best way.

Parameters

obj	Tried object to copy.

10.138.2.3 UniqueAcquire::UniqueAcquire (UniqueAcquire && *obj*)

Move Constructor.

Parameters

An	object to move

Definition at line 17 of file UniqueAcquire.cpp.

References isLocked, and semaphore.

10.138.2.4 UniqueAcquire::~UniqueAcquire ( )

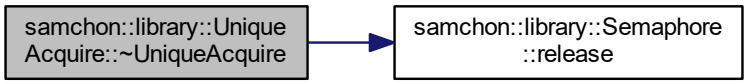
Default Destructor.

If read lock has done by the UniqueLock, unlock it

Definition at line 27 of file UniqueAcquire.cpp.

References isLocked, samchon::library::Semaphore::release(), and semaphore.

Here is the call graph for this function:



10.138.3 Member Function Documentation

10.138.3.1 void UniqueAcquire::acquire ( )

Acquire admission.

Acquires an admission and increases count of admission by 1.

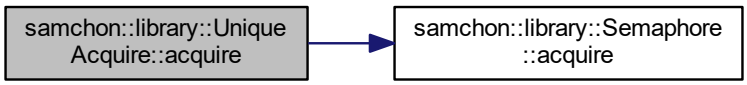
If the count is over permitted size, wait until other admissions to be released.

* Lock on mutex

Definition at line 36 of file UniqueAcquire.cpp.

References samchon::library::Semaphore::acquire(), isLocked, and semaphore.

Here is the call graph for this function:



10.138.3.2 void UniqueAcquire::release ( )

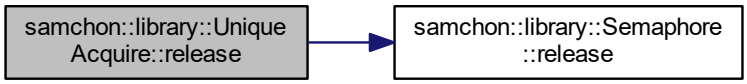
Release an admission.

Releases an admission what you've acquired. If the admission count was over the limited size, unlock the mutex.

Definition at line 44 of file UniqueAcquire.cpp.

References isLocked, samchon::library::Semaphore::release(), and semaphore.

Here is the call graph for this function:



10.138.4 Member Data Documentation

10.138.4.1 Semaphore* samchon::library::UniqueAcquire::semaphore [private]

Managed semaphore.

Definition at line 47 of file UniqueAcquire.hpp.

Referenced by acquire(), release(), UniqueAcquire(), and ~UniqueAcquire().

10.138.4.2 bool samchon::library::UniqueAcquire::isLocked [private]

Whether the semaphore was acquired by the UniqueLock.

Definition at line 52 of file UniqueAcquire.hpp.

Referenced by acquire(), release(), UniqueAcquire(), and ~UniqueAcquire().

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

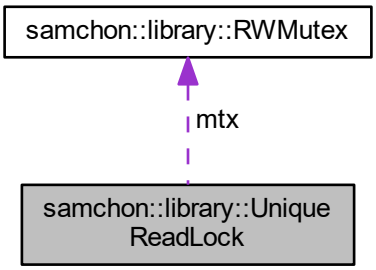
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/UniqueAcquire.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/UniqueAcquire.cpp

10.139 samchon::library::UniqueReadLock Class Reference

Unique lock for reading.

`#include <UniqueReadLock.hpp>`

Collaboration diagram for samchon::library::UniqueReadLock:



Public Member Functions

- UniqueReadLock (const RWMutex &, bool=true)
  Construct from mutex.
- UniqueReadLock (const UniqueReadLock &)=delete
  Prohibited Copy Constructor.
- UniqueReadLock (UniqueReadLock &&)
  Move Constructor.
- ~UniqueReadLock ()
  Default Destructor.
- void lock () const
  Lock on read.
- void unlock () const
  Unlock of read.

Private Attributes

- const RWMutex * mtx
  Managed mutex.
- bool isLocked
  Whether the mutex was locked by UniqueLock.

10.139.1 Detailed Description

Unique lock for reading.

A UniqueAcquire is an object manages a RWMutex with unique ownership in both states.

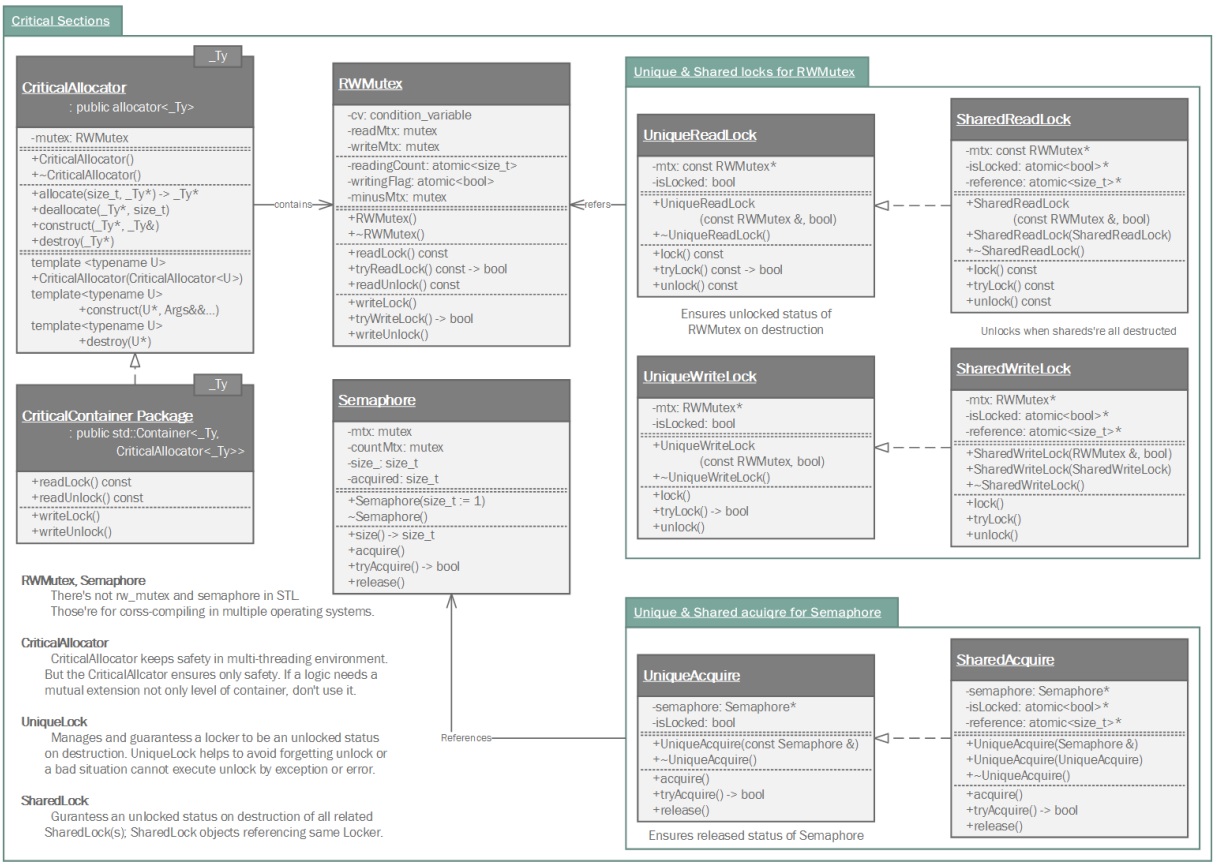
- locked
- unlocked

On construction (or by move-assigning to it), the object locks a RWMutex object on reading side, for whose locking and unlocking operations becomes responsible.

This class guarantees a unlocked status on destruction (even if not called explicitly). Therefore it is especially useful as an object with automatic duration, as it guarantees the RWMutex object is properly unlocked in case an exception is thrown.

Referenced comments of std::unique_lock

- http://www.cplusplus.com/reference/mutex/unique_lock/



Note

Though, that the UniqueReadLock object does not manage the lifetime of the RWMutex object in any way: the duration of the RWMutex object shall extend at least until the destruction of the UniqueReadLock that manages it.

See also

samchon::library

Author

Jeongho Nam

Definition at line 39 of file UniqueReadLock.hpp.

10.139.2 Constructor & Destructor Documentation

10.139.2.1 UniqueReadLock::UniqueReadLock (const RWMutex & *mtx*, bool *doLock* = `true`)

Construct from mutex.

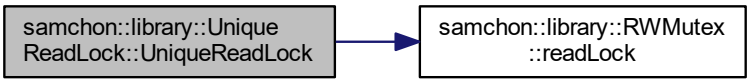
Parameters

mtx	Mutex to manage
doLock	Whether to lock directly or not

Definition at line 7 of file UniqueReadLock.cpp.

References isLocked, mtx, and samchon::library::RWMutex::readLock().

Here is the call graph for this function:



10.139.2.2 samchon::library::UniqueReadLock::UniqueReadLock (const UniqueReadLock &) [delete]

Prohibited Copy Constructor.

UniqueAcquire can't be copied. Use pointer, reference instead.

If what you want is UniqueAcquire(s) references each other and unlock when all related UniqueAcquire objects are destructed, SharedAcquire is the best way.

Parameters

obj	Tried object to copy.

10.139.2.3 UniqueReadLock::UniqueReadLock (UniqueReadLock && *obj*)

Move Constructor.

Parameters

An	object to move

Definition at line 15 of file UniqueReadLock.cpp.

References isLocked, and mtx.

10.139.2.4 UniqueReadLock::~UniqueReadLock ( )

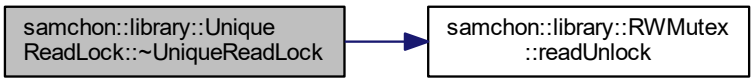
Default Destructor.

If read lock has done by the UniqueLock, unlock it

Definition at line 25 of file UniqueReadLock.cpp.

References isLocked, mtx, and samchon::library::RWMutex::readUnlock().

Here is the call graph for this function:



10.139.3 Member Function Documentation

10.139.3.1 void UniqueReadLock::lock ( ) const

Lock on read.

Increases a reading count.

When write_lock is on a progress, wait until write_unlock to be called.

* Reading can be done by multiple sections.
* Reading can't be done when writing.

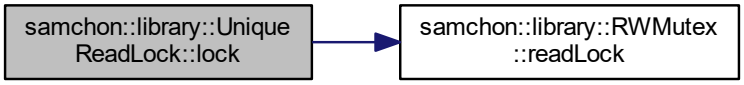
Warning

You've to call read_unlock when the reading work is terminated.

Definition at line 34 of file UniqueReadLock.cpp.

References isLocked, mtx, and samchon::library::RWMutex::readLock().

Here is the call graph for this function:



10.139.3.2 void UniqueReadLock::unlock ( ) const

Unlock of read.

Decreases a reading count.

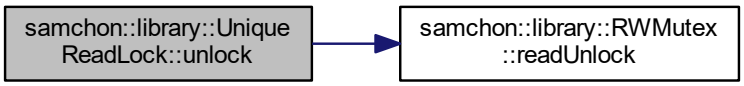
When write_lock had done after read_lock, it continues by read_unlock if the reading count was 1 (read_unlock makes the count to be zero).

Definition at line 44 of file UniqueReadLock.cpp.

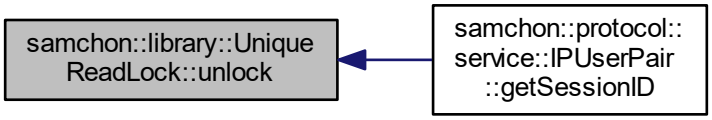
References isLocked, mtx, and samchon::library::RWMutex::readUnlock().

Referenced by samchon::protocol::service::IPUserPair::getSessionID().

Here is the call graph for this function:



Here is the caller graph for this function:



10.139.4 Member Data Documentation

10.139.4.1 const RWMutex* samchon::library::UniqueReadLock::mtx [private]

Managed mutex.

Definition at line 45 of file UniqueReadLock.hpp.

Referenced by lock(), UniqueReadLock(), unlock(), and ~UniqueReadLock().

10.139.4.2 bool samchon::library::UniqueReadLock::isLocked [private]

Whether the mutex was locked by UniqueLock.

Definition at line 50 of file UniqueReadLock.hpp.

Referenced by lock(), UniqueReadLock(), unlock(), and ~UniqueReadLock().

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

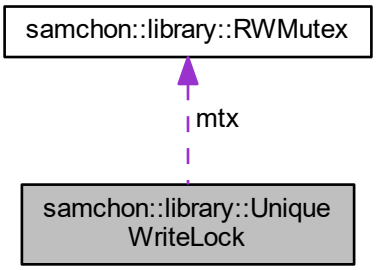
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/UniqueReadLock.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/UniqueReadLock.cpp

10.140 samchon::library::UniqueWriteLock Class Reference

Unique lock for writing.

`#include <UniqueWriteLock.hpp>`

Collaboration diagram for samchon::library::UniqueWriteLock:



Public Member Functions

* UniqueWriteLock (RWMutex &, bool=true)
  Construct from mutex.
* UniqueWriteLock (const UniqueWriteLock &)=delete
  Prohibited Copy Constructor.
* UniqueWriteLock (UniqueWriteLock &&)
  Move Constructor.
* ~UniqueWriteLock ()
  Destructor.
* void lock ()
  Lock on writing.
* void unlock ()
  Unlock on writing.

Private Attributes

* RWMutex * mtx

Managed mutex.

* bool isLocked

  Whether the mutex was locked by UniqueLock.

10.140.1 Detailed Description

Unique lock for writing.

A UniqueWriteLock is an object manages a RWMutex with unique ownership in both states.

* acquired
* released

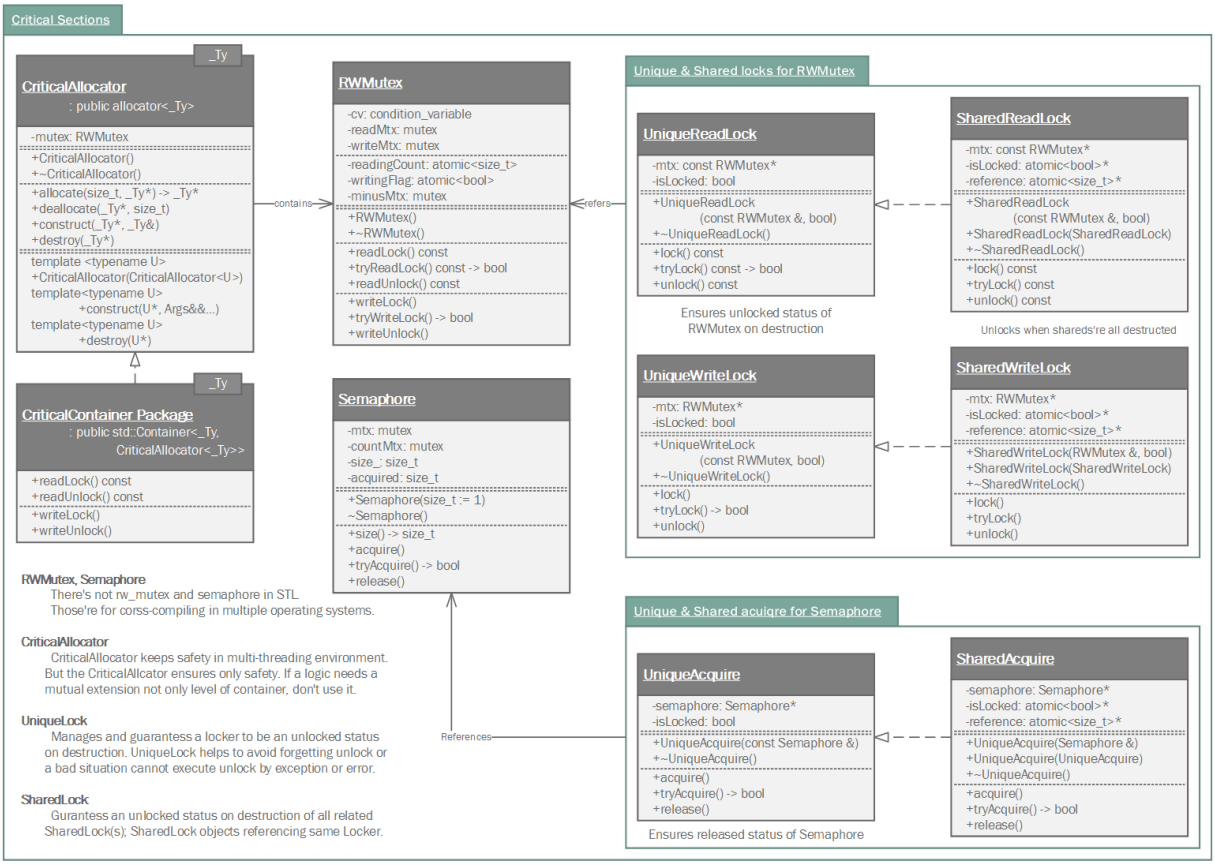
On construction (or by move-assigning to it), the object locks a RWMutex object on writing side, for whose locking and unlocking operations becomes responsible.

The object supports both states: locked and unlocked.

This class guarantees a unlocked status on destruction (even if not called explicitly). Therefore it is especially useful as an object with automatic duration, as it guarantees the RWMutex object is properly unlocked in case an exception is thrown.

Referenced comments of std::unique_lock

* http://www.cplusplus.com/reference/mutex/unique_lock/



Note

Though, that the UniqueWriteLock object does not manage the lifetime of the RWMutex object in any way: the duration of the RWMutex object shall extend at least until the destruction of the UniqueWriteLock that manages it.

See also

samchon::library

Author

Jeongho Nam

Definition at line 41 of file UniqueWriteLock.hpp.

10.140.2 Constructor & Destructor Documentation

10.140.2.1 UniqueWriteLock::UniqueWriteLock (RWMutex & *mtx*, bool *doLock* = `true`)

Construct from mutex.

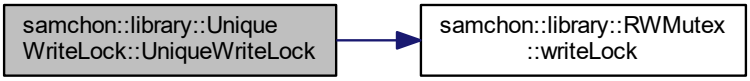
Parameters

mtx	Mutex to manage
doLock	Whether to lock directly or not

Definition at line 7 of file UniqueWriteLock.cpp.

References isLocked, mtx, and samchon::library::RWMutex::writeLock().

Here is the call graph for this function:



10.140.2.2 samchon::library::UniqueWriteLock::UniqueWriteLock (const UniqueWriteLock &) `[delete]`

Prohibited Copy Constructor.

UniqueAcquire can't be copied. Use pointer, reference instead.

If what you want is UniqueAcquire(s) references each other and unlock when all related UniqueAcquire objects are destructed, SharedAcquire is the best way.

Parameters

obj	Tried object to copy.

10.140.2.3 UniqueWriteLock::UniqueWriteLock (UniqueWriteLock && *obj*)

Move Constructor.

Parameters

An	object to move

Definition at line 15 of file UniqueWriteLock.cpp.

References isLocked, and mtx.

10.140.2.4 UniqueWriteLock::~UniqueWriteLock ( )

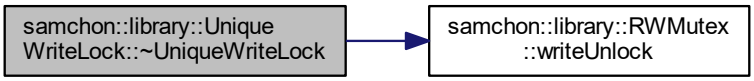
Destructor.

If write lock has done by the UniqueLock, unlock it

Definition at line 25 of file UniqueWriteLock.cpp.

References isLocked, mtx, and samchon::library::RWMutex::writeUnlock().

Here is the call graph for this function:



10.140.3 Member Function Documentation

10.140.3.1 void UniqueWriteLock::lock ( )

Lock on writing.

Changes writing flag to true.

If another write_lock or read_lock is on a progress, wait until them to be unlocked.

* Writing can be done by only a section at once.
* Writing can't be done when reading.

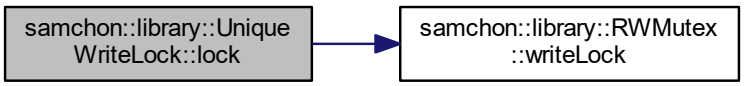
Note

You've to call write_unlock when writing work was terminated.

Definition at line 34 of file UniqueWriteLock.cpp.

References isLocked, mtx, and samchon::library::RWMutex::writeLock().

Here is the call graph for this function:



10.140.3.2 void UniqueWriteLock::unlock ( )

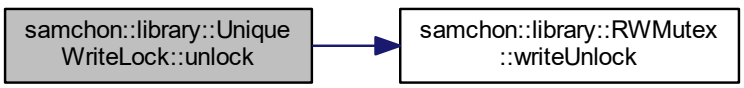
Unlock on writing.

Definition at line 42 of file UniqueWriteLock.cpp.

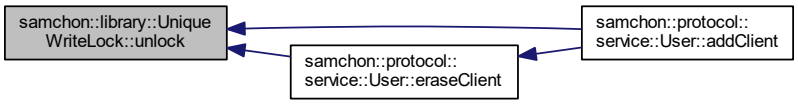
References isLocked, mtx, and samchon::library::RWMutex::writeUnlock().

Referenced by samchon::protocol::service::User::addClient(), and samchon::protocol::service::User↩ ::eraseClient().

Here is the call graph for this function:



Here is the caller graph for this function:



10.140.4 Member Data Documentation

10.140.4.1 RWMutex* samchon::library::UniqueWriteLock::mtx [private]

Managed mutex.

Definition at line 47 of file UniqueWriteLock.hpp.

Referenced by lock(), UniqueWriteLock(), unlock(), and ~UniqueWriteLock().

10.140.4.2 bool samchon::library::UniqueWriteLock::isLocked [private]

Whether the mutex was locked by UniqueLock.

Definition at line 52 of file UniqueWriteLock.hpp.

Referenced by lock(), UniqueWriteLock(), unlock(), and ~UniqueWriteLock().

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

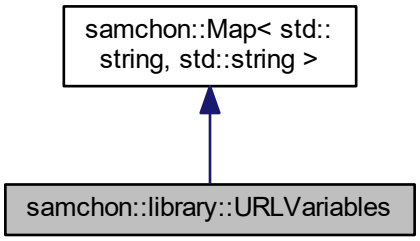
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/UniqueWriteLock.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/UniqueWriteLock.cpp

10.141 samchon::library::URLVariables Class Reference

URLVariables class is for representing variables of HTTP.

`#include <URLVariables.hpp>`

Collaboration diagram for samchon::library::URLVariables:



Public Member Functions

* URLVariables ()
  Default Constructor.
* URLVariables (const WeakString &flashVars)
  Constructor by a string representing encoded properties.
* auto toString () const -> std::string
  Get the string representing URLVariables.

Static Public Member Functions

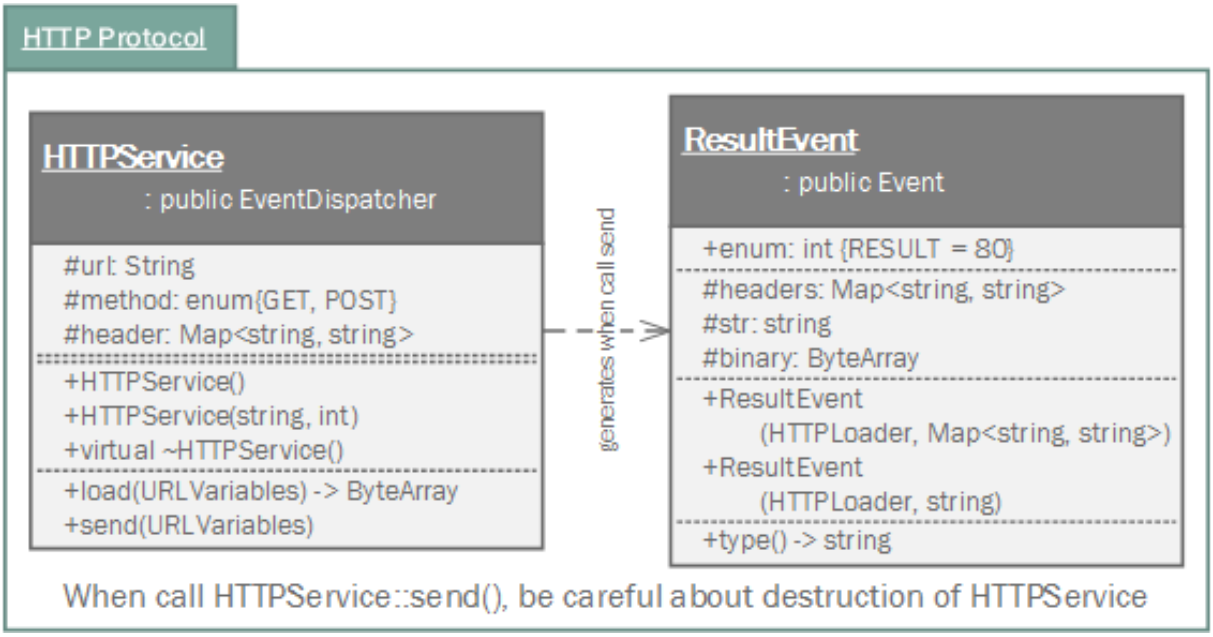
* static auto encode (const WeakString &) -> std::string
  Encode a string into a valid URI.
* static auto decode (const WeakString &) -> std::string
  Decode a URI string.

10.141.1 Detailed Description

URLVariables class is for representing variables of HTTP.
URLVariables class allows you to transfer variables between an application and server. When transfering, URLVariables will be converted to a URI string.

* URI: Uniform Resource Identifier

Use URLVariabels objects with methods of HTTPLoader class.



Example code

See also

samchon::library

Author

Jeongho Nam

Definition at line 30 of file URLVariables.hpp.

10.141.2 Constructor & Destructor Documentation

10.141.2.1 URLVariables::URLVariables ( )

Default Constructor.
Definition at line 11 of file URLVariables.cpp.

10.141.2.2 URLVariables::URLVariables (const WeakString & *flashVars*)

Constructor by a string representing encoded properties.
Converts the variable string to properties of the specified URLVariables object.

* URLVariables("id=jhnam88&name=Jeongho+Nam") => {{\"id", "jhnam88"}, {"name", "Jeongho Nam"}}

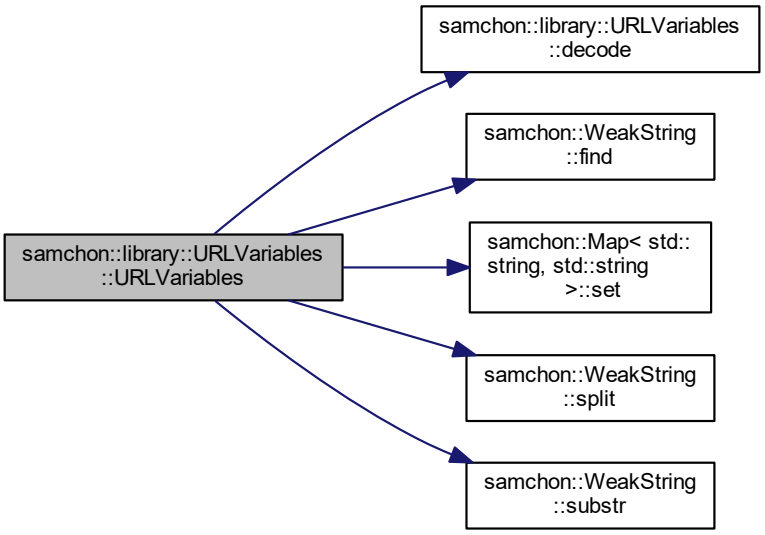
Parameters

A	uri-encoded string containing pair of properties

Definition at line 15 of file URLVariables.cpp.

References decode(), samchon::WeakString::find(), samchon::Map< std::string, std::string >::set(), samchon::WeakString::split(), and samchon::WeakString::substr().

Here is the call graph for this function:



10.141.3 Member Function Documentation

10.141.3.1 auto URLVariables::encode (const WeakString & *wstr*) -> std::string [static]

Encode a string into a valid URI.

Encodes a string to follow URI standard format.

Parameters

A	string to encode to URI

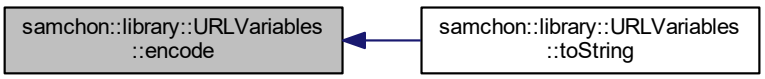
Returns

A string converted to URI

Definition at line 43 of file URLVariables.cpp.

Referenced by toString().

Here is the caller graph for this function:



10.141.3.2 auto URLVariables::decode (const WeakString & *wstr*) -> std::string [static]

Decode a URI string.

Decodes a URI string to its original

Parameters

A	string encoded

Returns

A string decoded from URI

Definition at line 66 of file URLVariables.cpp.

Referenced by URLVariables().

Here is the caller graph for this function:

10.141.3.3 auto URLVariables::toString ( ) const -> std::string

Get the string representing URLVariables.

Returns a string object representing URLVariables following the URI

* URLVariables({{"id", "jhnam88"}, {"name", "Jeongho Nam"}}).toString() => "id=jhnam88&name=↩ Jeongho+Nam"

Returns

A string representing URLVariables following the URI

Definition at line 114 of file URLVariables.cpp.

References encode().

Here is the call graph for this function:

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

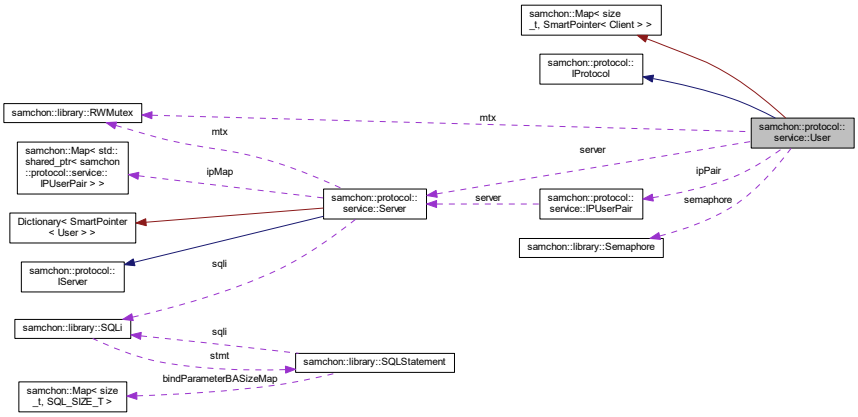
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/URLVariables.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/URLVariables.cpp

10.142 samchon::protocol::service::User Class Reference

User containing Client(s) with session-id.

```
#include <User.hpp>
```

Collaboration diagram for samchon::protocol::service::User:



Public Member Functions

- User (Server *)
  Construct from server.
- auto getServer () const -> Server *
  Get server.
- auto getSemaphore () const -> library::Semaphore *
  Get semaphore.
- auto getID () const -> std::string
  Get user's account id.
- auto getAuthority () const -> int
  Get authority of user.
- virtual void sendData (std::shared_ptr< Invoke >) override
  Send a message.
- virtual void replyData (std::shared_ptr< Invoke >) override
  Reply a message.

Protected Member Functions

- virtual auto createClient () -> Client *=0
  Factory method of a Client.
- virtual auto doLogin (std::shared_ptr< Invoke >) -> bool=0
  Login and return whether success.
- virtual auto doJoin (std::shared_ptr< Invoke >) -> bool=0
  Join and returns whether success.

Private Member Functions

- void addClient (Socket *)
  Add a client in user.
- void eraseClient (size_t)
  Erase a Client.
- void goLogin (std::shared_ptr< Invoke >)
  Log-in.
- void goJoin (std::shared_ptr< Invoke >)
  Join as a member.
- virtual void goLogout ()
  Log-out.

Private Attributes

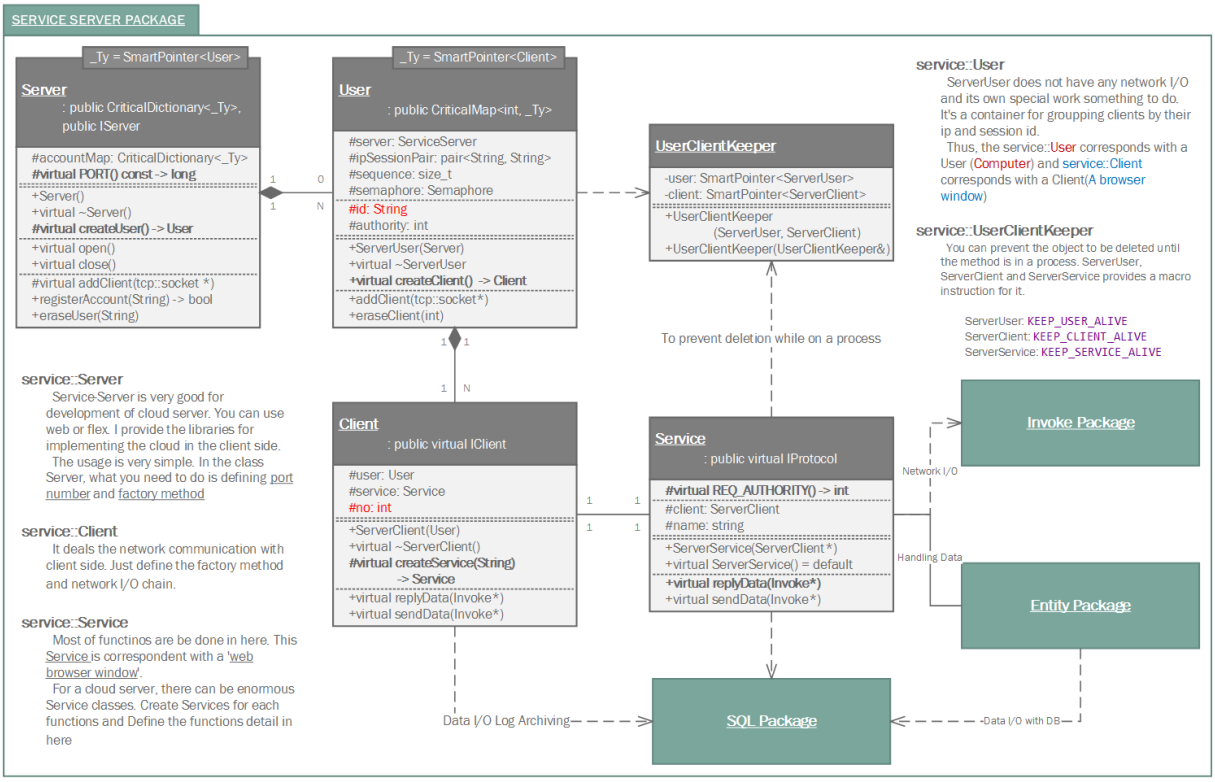
- Server * server
  Server containing the user.

* std::string sessionID
  Session ID of the user.
* library::RWMutex mtx
  Mutex for container.
* library::Semaphore * semaphore
  Semaphore to limit number of thread.
* std::string id
  Account id.
* int authority
  Authority allocated to user.

10.142.1 Detailed Description

User containing Client(s) with session-id.

The User does not have any network I/O and its own special work something to do. It's a container for groupping clients by their session id. Thus, the User corresponds with an user (Computer) and Client corresponds with a client (A browser window)



Override those methods

* createClient()
* doLogin()
* doJoin()

Author

Jeongho Nam

Definition at line 56 of file User.hpp.

10.142.2 Constructor & Destructor Documentation

10.142.2.1 User::User (Server * *server*)

Construct from server.

Parameters

server	A server who containing users and clients
sessionID	A session ID for discriminating User; A logical identifier

Definition at line 24 of file User.cpp.

References authority, semaphore, and server.

10.142.3 Member Function Documentation

10.142.3.1 auto User::getServer ( ) const -> Server*

Get server.

Definition at line 42 of file User.cpp.

References server.

10.142.3.2 auto User::getSemaphore ( ) const -> library::Semaphore*

Get semaphore.

See also

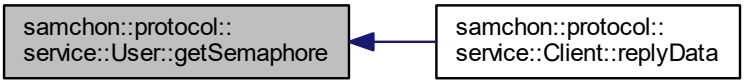
User::semaphore

Definition at line 55 of file User.cpp.

References semaphore.

Referenced by samchon::protocol::service::Client::replyData().

Here is the caller graph for this function:



10.142.3.3 auto User::getID ( ) const -> std::string

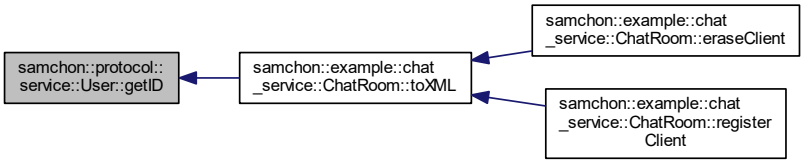
Get user's account id.

Definition at line 46 of file User.cpp.

References id.

Referenced by samchon::example::chat_service::ChatRoom::toXML().

Here is the caller graph for this function:



10.142.3.4 auto User::getAuthority ( ) const -> int

Get authority of user.

See also

User::authority

Definition at line 50 of file User.cpp.

References authority.

Referenced by samchon::protocol::service::Client::constructService().

Here is the caller graph for this function:

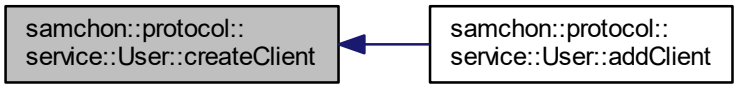
10.142.3.5 virtual auto samchon::protocol::service::User::createClient ( ) -> Client * [protected], [pure virtual]

Factory method of a Client.

Implemented in samchon::example::chat_service::ChatUser.

Referenced by addClient().

Here is the caller graph for this function:



10.142.3.6 void User::addClient (Socket * *socket*) [private]

Add a client in user.

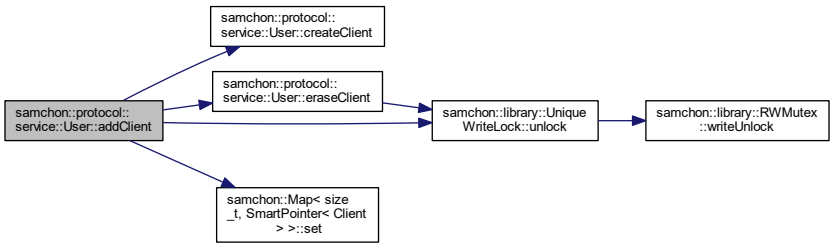
Parameters

socket	Socket for interacting with (physical) client

Definition at line 76 of file User.cpp.

References createClient(), eraseClient(), mtx, samchon::Map< size_t, SmartPointer< Client > >::set(), and samchon::library::UniqueWriteLock::unlock().

Here is the call graph for this function:



10.142.3.7 void User::eraseClient (size_t *no*) [private]

Erase a Client.

Erase a Client from User by its key(no:int).

When a network connection between server and user(person) was disconnected, matched Client is erased from User and the Client object will be destructed.

Some reasons of disconnection

* An user(not object but person using UI) closed the Window
* A "closeWindow" command has sent to the Window
* A network failure has occured

If the Client was the last object remained in the User, the User will be destructed too. However, the User is not destructed directly but waits for 15 seconds. In the 15 seconds, another Client object was created in User, the User doesn't be destructed.

Some reasons of postponing destruction for 15 seconds

* Page direction(href) or redirection in a browser.
* Temporary network failure.

Note

There're some tasks that have to completed irrespectively of network connection.

In that case, although user closed the window or logged-out, matched User, Client and Service objects haven't be destructed. They must stay alive until the tasks have completed.

To stay those objects User, Client and Service, use follwing macros. Those macros assure the objects to stay alive until the termination matched each stack.

* KEEP_USER_ALIVE(Client∗)
* KEEP_CLIENT_ALIVE
* KEEP_SERVICE_ALIVE

Parameters

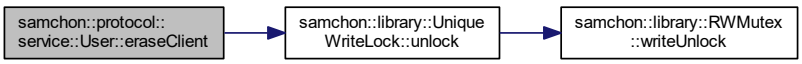
no	Sequence number of the client in user

Definition at line 94 of file User.cpp.

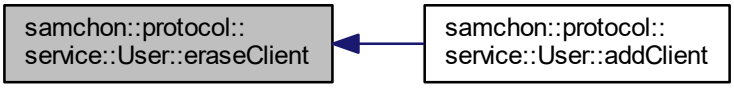
References mtx, server, sessionID, and samchon::library::UniqueWriteLock::unlock().

Referenced by addClient().

Here is the call graph for this function:



Here is the caller graph for this function:



10.142.3.8 void User::goLogin (std::shared_ptr< Invoke >) [private]

Log-in.

Parameters

client	
invoke	

See also

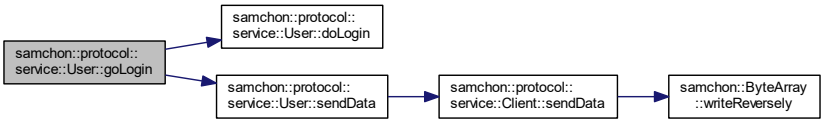
User::doLogin()

Definition at line 111 of file User.cpp.

References authority, doLogin(), and sendData().

Referenced by replyData().

Here is the call graph for this function:



Here is the caller graph for this function:

10.142.3.9 void User::goJoin (std::shared_ptr< Invoke >) [private]

Join as a member.

Parameters

client	An application; who requested the join command
invoke	AN invoke message containing parameters of joinning

See also

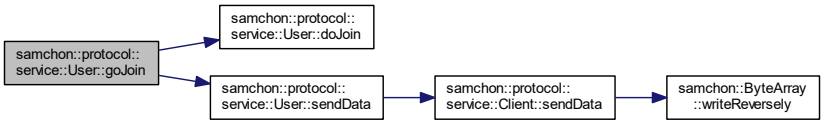
User::doJoin()

Definition at line 121 of file User.cpp.

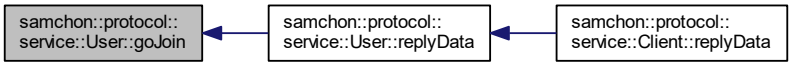
References doJoin(), and sendData().

Referenced by replyData().

Here is the call graph for this function:



Here is the caller graph for this function:



10.142.3.10 void User::goLogout ( ) [private], [virtual]

Log-out.

Log-out account; change id and authority to "guest" and 0.

Default is to send "closeWindow" message to all clients except the applicant.

* Perspective on UI:
  All windows of the user will be closed by getting an invoke message "closeWindow" except who requested the command log-out.
  The applicant window will moved to default log-in page.

Parameters

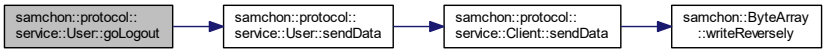
client	A window of browser; client who requested the log-out command

Definition at line 130 of file User.cpp.

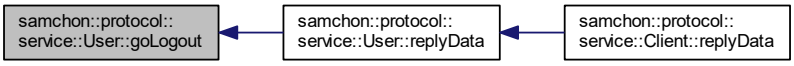
References authority, and sendData().

Referenced by replyData().

Here is the call graph for this function:



Here is the caller graph for this function:



10.142.3.11 virtual auto samchon::protocol::service::User::doLogin (std::shared_ptr< Invoke >) -> bool [protected],[pure virtual]

Login and return whether success.

Parameters

invoke	Invoke message containing parameters for logging-in

Returns

Whether the log-in is succeded or not

Referenced by goLogin().

Here is the caller graph for this function:

10.142.3.12 virtual auto samchon::protocol::service::User::doJoin (std::shared_ptr< Invoke >) -> bool [protected],[pure virtual]

Join and returns whether success.

Parameters

invoke	Invoke message containing paramters for joinning

Returns

Whether the join is succeded or not

Referenced by goJoin().

Here is the caller graph for this function:

10.142.3.13 void User::sendData (std::shared_ptr< Invoke >) [override],[virtual]

Send a message.

Sends Invoke message to a network system or shifts the responsibility to related chain.

Parameters

invoke	An Invoke message to send to a network system

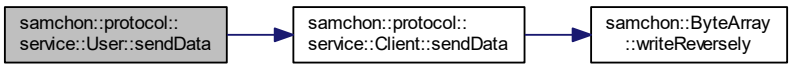
Reimplemented from samchon::protocol::IProtocol.

Definition at line 142 of file User.cpp.

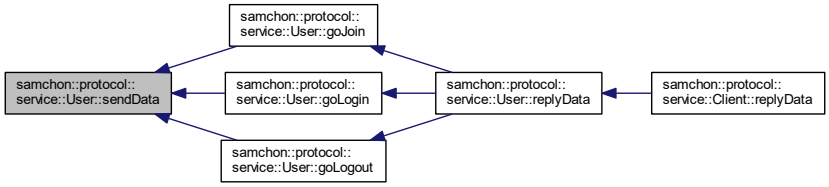
References mtx, and samchon::protocol::service::Client::sendData().

Referenced by goJoin(), goLogin(), and goLogout().

Here is the call graph for this function:



Here is the caller graph for this function:



10.142.3.14 void User::replyData (std::shared_ptr< Invoke >) [override], [virtual]

Reply a message.

Handles a replied Invoke message or shifts the responsibility to related chain.

Parameters

invoke	An Invoke message gotten from a network system

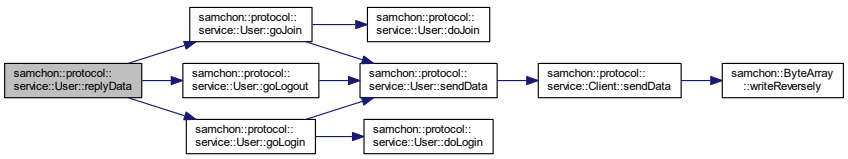
Reimplemented from samchon::protocol::IProtocol.

Definition at line 150 of file User.cpp.

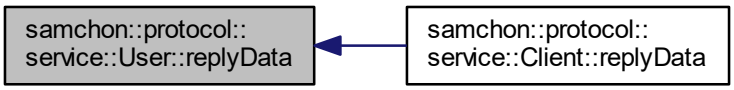
References goJoin(), goLogin(), and goLogout().

Referenced by samchon::protocol::service::Client::replyData().

Here is the call graph for this function:



Here is the caller graph for this function:



10.142.4 Member Data Documentation

10.142.4.1 Server* samchon::protocol::service::User::server [private]

Server containing the user.

Definition at line 73 of file User.hpp.

Referenced by eraseClient(), getServer(), and User().

10.142.4.2 std::string samchon::protocol::service::User::sessionID [private]

Session ID of the user.

Organization of the Session ID

* {$name}::{$port}::{$sequence}::{$linux_time}

1. $name: Name of the Server; Server::NAME()
2. $port: Port number of the Server; Server::PORT()
3. $sequence: Sequence number of creation of User in Server
4. $linux_time: Linux time of current time.

   If the object Server is re-constructed, $sequence can be duplicated. $linux_time is to avoid the duplication so that ensures exclusiveness on session id

Definition at line 94 of file User.hpp.

Referenced by eraseClient().

10.142.4.3 library::RWMutex samchon::protocol::service::User::mtx [private]

Mutex for container.

Definition at line 104 of file User.hpp.

Referenced by addClient(), eraseClient(), and sendData().

10.142.4.4 library::Semaphore* samchon::protocol::service::User::semaphore [private]

Semaphore to limit number of thread.

Semaphore in User is designed to constraint number of thread.

It's recommended to acquire the semaphore when a task from Client or Service occurs a new thread and you want to avoid exhausted allocation of new threads.

Note

If a task influences to all clients in a user globally, use lock instead of acquire

Definition at line 118 of file User.hpp.

Referenced by getSemaphore(), and User().

10.142.4.5 std::string samchon::protocol::service::User::id [private]

Account id.
Default is "guest"
Definition at line 133 of file User.hpp.
Referenced by getID().

10.142.4.6 int samchon::protocol::service::User::authority [private]

Authority allocated to user.
The authority means authority(right, power) level allocated to the user.
The default value is zero, which means not logged-in user.

Note

Each number of level is not defined clearly. Follwing table is just an example.

Level	Definition
-1	Banned member The -1 level member is banished by manager or policy
0	Guest, Not logged in
1	Unconfired member The member have applied to the join, but the join was not confirmed by manager or policy yet.
2	Confirmed member The member have joined and have already confirmed by manager or policy
3	Manager A manager of the application(cloud service). The manager can * Access to the monitoring service. (MonitorService) * Access historical log of members whose level are below Manager. * Banish or modify level of members whose level are below Manager. </u>

4	Administrator or super Manager Administrator is a Manager of Managers. An administrator can do same thing with Managers, but adminstrator even possible to target Managers as below.

Definition at line 200 of file User.hpp.

Referenced by getAuthority(), goLogin(), goLogout(), and User().

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

* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/service/User.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/service/User.cpp

10.143 samchon::WeakString Class Reference

A string class only references characeters, reference only.

`#include <WeakString.hpp>`

Public Member Functions

* WeakString ()
  Default Constructor does not reference any character.
* WeakString (const char *data, size_t size)
  Constructor by characters with specified size.
* WeakString (const char *begin, const char *end)
  Constructor by characters of begin and end.
* WeakString (const char *data)
  Constructor by characters.
* WeakString (const char &ch)
  Constructor by a single character.
* WeakString (std::initializer_list< char > &il)
  Constructor by a initializer list.
* WeakString (const std::string &str)
  Constructor by string.
* auto data () const -> const char *
  Get string data; referenced characeters.
* auto size () const -> size_t
  Returns size of the characters which are being referenced.
* auto empty () const -> bool
  Tests wheter string is emtpy.
* auto at (size_t index) const -> const char &
  Get character of string Returns a const reference to the character at the specified position.
* auto operator[] (size_t index) const -> const char &
  Get character of string Returns a const reference to the character at the specified position.
* auto find (const WeakString &delim, size_t startIndex=NULL) const -> size_t
  Finds first occurence in string.
* auto rfind (const WeakString &delim, size_t endIndex=SIZE_MAX) const -> size_t
  Finds last occurence in string.
* auto finds (const std::vector< std::string > &delims, size_t startIndex=0) const -> IndexPair< WeakString >
  Finds first occurence in string.
* auto rfinds (const std::vector< std::string > &delims, size_t endIndex=SIZE_MAX) const -> IndexPair< WeakString >
  Finds last occurence in string.
* auto substr (size_t startIndex, size_t endIndex=SIZE_MAX) const -> WeakString
  Generates a substring.
* auto substring (size_t startIndex, size_t size=SIZE_MAX) const -> WeakString
  Generates a substring.
* auto between (const WeakString &start={}, const WeakString &end={}) const -> WeakString
  Generates a substring.
* auto split (const WeakString &delim) const -> std::vector< WeakString >
  Generates substrings.
* auto betweens (const WeakString &start={}, const WeakString &end={}) const -> std::vector< WeakString >

Generates substrings.

* auto trim (const std::vector< std::string > &delims) const -> WeakString
  Removes all designated characters from the beginning and end of the specified string.
* auto ltrim (const std::vector< std::string > &delims) const -> WeakString
  Removes all designated characters from the beginning of the specified string.
* auto rtrim (const std::vector< std::string > &delims) const -> WeakString
  Removes all designated characters from the end of the specified string.
* auto replace (const WeakString &before, const WeakString &after) const -> std::string
  Replace portion of string once.
* auto replaceAll (const WeakString &before, const WeakString &after) const -> std::string
  Returns a string specified word is replaced.
* auto replaceAll (const std::vector< std::pair< std::string, std::string >> &pairs) const -> std::string
  Returns a string specified words are replaced.
* auto toLowerCase () const -> std::string
  Convert uppercase letters to lowercase.
* auto toUpperCase () const -> std::string
  Convert uppercase letters to lowercase.
* auto str () const -> std::string
  Get the string content.

Static Public Attributes

* static const size_t npos = -1
  Maximum value for size_t.

Private Attributes

* const char * data_
  Referenced characters's pointer of begining position.
* size_t size_
  (Specified) size of referenced characters

Static Private Attributes

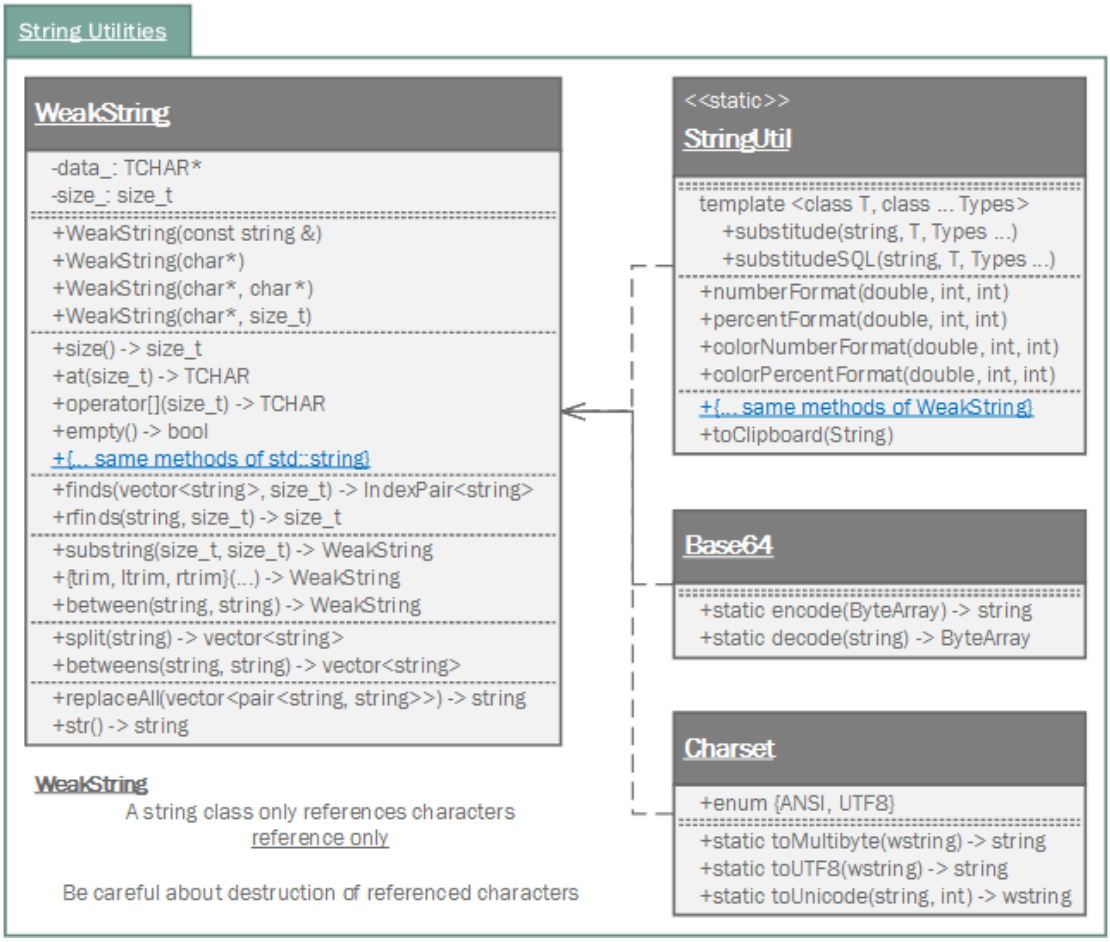
* static const std::vector< std::string > SPACE_ARRAY = { " ", "\t", "\r", "\n" }
  An array containing whitespaces.

10.143.1 Detailed Description

A string class only references characeters, reference only.

WeakSring does not consider any construction, modification and destruction of characters.

Thus, you can have greater advantages than std::string on the side of performance and memory, but of course, you can't modify the characeters at all.



Warning

- WeakString not copy(strcpy) characeters but only references characeters. Be careful about destruction of the characters being referenced by the WeakString
- WeakString will be used for basic data type in most case. Avoid to use WeakString by pre-definition in header (∗.hpp)

See also

samchon::library

Author

Jeongho Nam

Definition at line 32 of file WeakString.hpp.

10.143.2 Constructor & Destructor Documentation

10.143.2.1 WeakString::WeakString ( )

Default Constructor does not reference any character.
Constructs an empty string, with zero size
Definition at line 18 of file WeakString.cpp.

10.143.2.2 WeakString::WeakString (const char ∗ *data*, size_t *size*)

Constructor by characters with specified size.
Constructs by characters to be referenced with limited size.
Although the original size of data is over the specified size, you can limit referencing size of the characters

- Referencing a part of characters

Warning

WeakString only references. Be careful about destruction of the characters (data)

Parameters

data	Target characters to be referenced by string
size	

Specified limit-size of characters to be referenced.

If the specified size is greater than original size, it will be ignored

Definition at line 23 of file WeakString.cpp.

10.143.2.3 WeakString::WeakString (const char * *begin*, const char * *end*)

Constructor by characters of begin and end.

Constructs by characters to be referenced with its end position.

Although the original end point of data is over the specified end, you can limit end point of the characters.

* Referencing a part of characters

Warning

WeakString only references. Be careful about destruction of the characters (data).

Parameters

begin	Target characters to be referenced by string
end	

Specified end point of characters to be referenced.

If the specified end point is greater than original end point, it will be ignored.

Definition at line 28 of file WeakString.cpp.

10.143.2.4 WeakString::WeakString (const char * *data*)

Constructor by characters.

References the null-terminated character sequence pointed by ptr

Warning

WeakString only references. Be careful about destruction of the characeters (data)

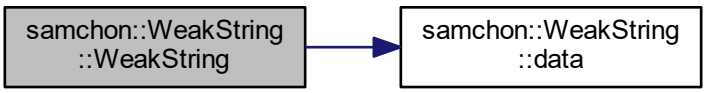
Parameters

data	Target characters to be referenced by string

Definition at line 32 of file WeakString.cpp.

References data(), data_, and size_.

Here is the call graph for this function:



10.143.2.5 WeakString::WeakString (const char & *ch*)

Constructor by a single character.

References a single character

Warning

WeakString only references. Be careful about destruction of the characeter (data)

Parameters

ch	Target character to be referenced by string

Definition at line 41 of file WeakString.cpp.

References data_, and size_.

10.143.2.6 WeakString::WeakString (std::initializer_list< char > & *il*)

Constructor by a initializer list.

References initializer list of character

Warning

WeakString only references. Be careful about destruction of the characeters (data)

Parameters

il	Target initializer list of characters to be referenced by string

Definition at line 46 of file WeakString.cpp.

References data_, and size_.

10.143.2.7 WeakString::WeakString (const std::string & *str*)

Constructor by string.

References whole chracters of the string

Warning

WeakString only references. Be careful about destruction of the string

Parameters

str	Target string to be referenced by string

Definition at line 55 of file WeakString.cpp.

References data_, and size_.

10.143.3 Member Function Documentation

10.143.3.1 auto WeakString::data ( ) const -> const char*

Get string data; referenced characeters.

Returns a pointer to an array that contains a null-terminated sequence of characters representing the current value of the string object

Warning

Returned pointer's size can be longer than string's specified size if the string references only a part of the characters

Returns

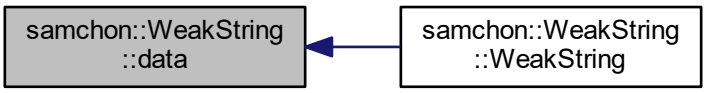
A pointer of characters being referenced by the string

Definition at line 64 of file WeakString.cpp.

References data_.

Referenced by WeakString().

Here is the caller graph for this function:



10.143.3.2 auto WeakString::size ( ) const -> size_t

Returns size of the characters which are being referenced.

Returns the length of the string, in terms of number of referenced characters

Returns

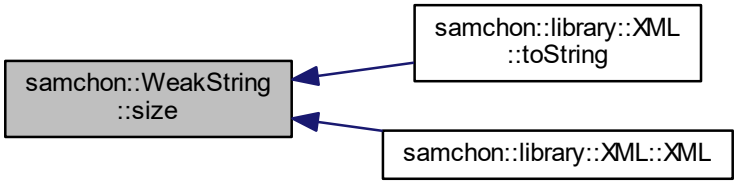
size of characters being referenced by string

Definition at line 68 of file WeakString.cpp.

References size_.

Referenced by samchon::library::XML::toString(), and samchon::library::XML::XML().

Here is the caller graph for this function:



10.143.3.3 auto WeakString::empty ( ) const -> bool

Tests wheter string is emtpy.

Returns wheter characters' size is zero or not.

Of course, string references nothing, then returns false, too.

Returns

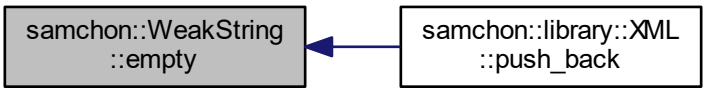
Wheter size is zero or not

Definition at line 73 of file WeakString.cpp.

References data_, and size_.

Referenced by samchon::library::XML::push_back().

Here is the caller graph for this function:



10.143.3.4 auto WeakString::at (size_t *index*) const -> const char&

Get character of string Returns a const reference to the character at the specified position.

Returns

const reference of character at the specified index

Definition at line 78 of file WeakString.cpp.

10.143.3.5 auto WeakString::operator[ ] (size_t *index*) const -> const char&

Get character of string Returns a const reference to the character at the specified position.

Returns

const reference of character at the specified index

Definition at line 82 of file WeakString.cpp.

10.143.3.6 auto WeakString::find (const WeakString & *delim*, size_t *startIndex* = NULL) const -> size_t

Finds first occurence in string.

Finds the string after startIndex and returns the position of first occurence of delim.

If delim is not found, returns -1 (npos)

Parameters

delim	The substring of the string which to find
startIndex	Specified starting index of find. Default is 0

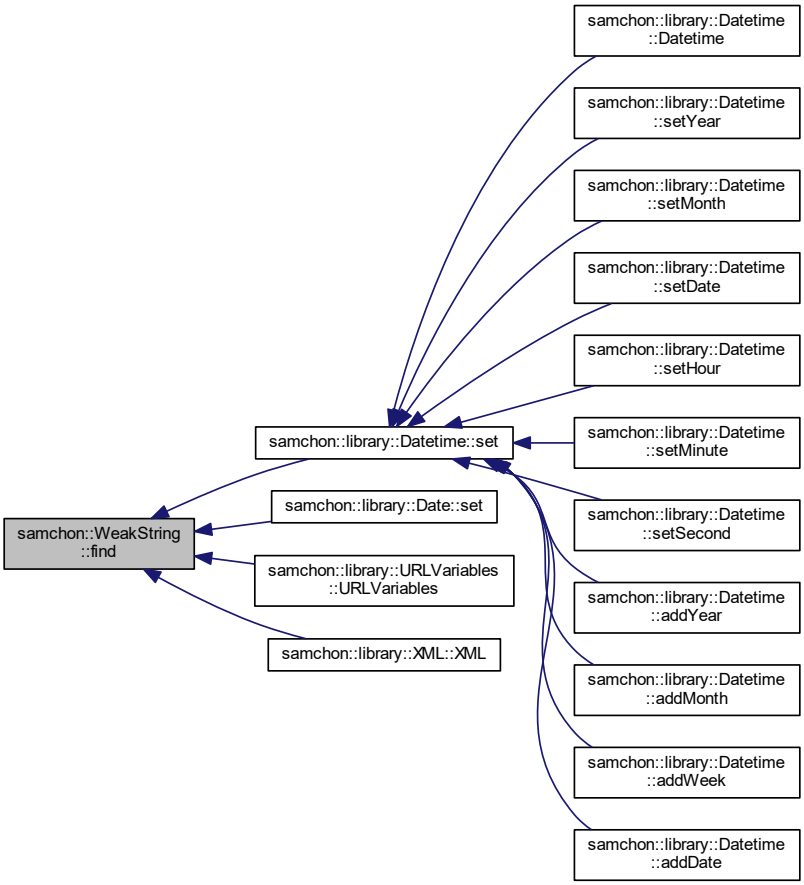
Returns

Index of first occurence of the specified substring or -1

Definition at line 90 of file WeakString.cpp.

Referenced by samchon::library::Datetime::set(), samchon::library::Date::set(), samchon::library::URLVariables::URLVariables(), and samchon::library::XML::XML().

Here is the caller graph for this function:



10.143.3.7 auto WeakString::rfind (const WeakString & *delim*, size_t *endIndex* = SIZE_MAX) const -> size_t

Finds last occurence in string.

Finds the string before endIndex and returns the position of last occurence of delim. If delim is not found, returns -1 (npos)

Parameters

delim	The substring of the string which to find
endIndex	Specified last index of find. Default is size() - 1

Returns

Index of first occurence of the specified substring or -1

Definition at line 102 of file WeakString.cpp.

10.143.3.8 auto WeakString::finds (const std::vector< std::string > & *delims*, size_t *startIndex* = 0) const -> IndexPair<WeakString>

Finds first occurence in string.

Finds first occurence position of each delim in the string after startIndex and returns the minimum position of them.

Note

* If startIndex is not specified, then starts from 0.
* If failed to find any substring, returns -1 (npos)

Parameters

delims	The substrings of target(str) which to find
startIndex	Specified starting index of find. Default is 0

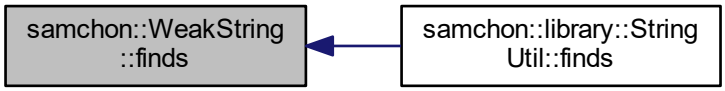
Returns

pair<size_t := position, string := matched substring>

Definition at line 120 of file WeakString.cpp.

Referenced by samchon::library::StringUtil::finds().

Here is the caller graph for this function:



10.143.3.9 auto WeakString::rfinds (const std::vector< std::string > & *delims*, size_t *endIndex* = SIZE_MAX) const -> IndexPair<WeakString>

Finds last occurence in string.

Finds last occurence position of each delim in the string before endIndex and returns the maximum position of them.

Note

* If index is not specified, then starts str.size() - 1
* If failed to find any substring, returns -1 (npos)

Parameters

delims	The substrings of target(str) which to find
endIndex	Specified starting index of find. Default is size() - 1

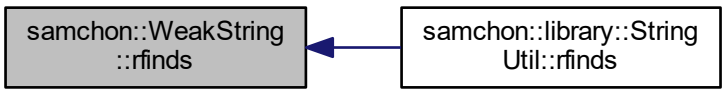
Returns

pair<size_t := position, string := matched substring>

Definition at line 140 of file WeakString.cpp.

Referenced by samchon::library::StringUtil::rfinds().

Here is the caller graph for this function:



10.143.3.10 auto WeakString::substr (size_t *startIndex*, size_t *endIndex* = SIZE_MAX) const -> WeakString

Generates a substring.

Extracts a substring consisting of the characters starts from startIndex and with a size specified size.

Parameters

startIndex	

Index of the first character.

If startIndex is greater than endIndex, those will be swapped.

Parameters

endIndex	

Number of characters to include in substring.

If the specified size is greater than last index of characeters, it will be shrinked.

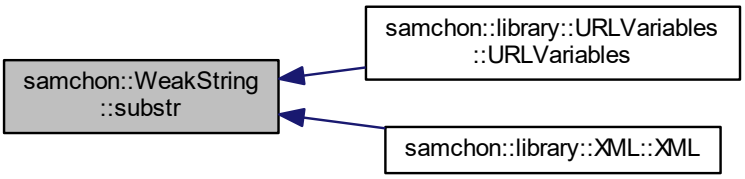
Returns

Sub string by specified index and size

Definition at line 173 of file WeakString.cpp.

Referenced by samchon::library::URLVariables::URLVariables(), and samchon::library::XML::XML().

Here is the caller graph for this function:



10.143.3.11 auto WeakString::substring (size_t *startIndex*, size_t *size* = SIZE_MAX) const -> WeakString

Generates a substring.

Extracts a substring consisting of the character specified by startIndex and all characters up to endIndex - 1.

* If endIndex is not specified, string::size() will be used instead.
* If endIndex is greater than startIndex, then those will be swapped

Parameters

startIndex	

Index of the first character.

If startIndex is greater than endIndex, those will be swapped.

Parameters

size	

Index of the last character - 1.

If not specified, then string::size() will be used instead.

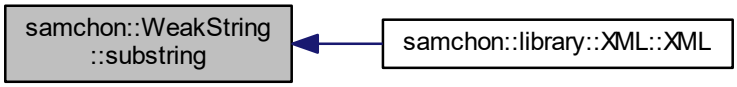
Returns

Sub string by specified index(es)

Definition at line 180 of file WeakString.cpp.

Referenced by samchon::library::XML::XML().

Here is the caller graph for this function:



10.143.3.12 auto WeakString::between (const WeakString & *start* = { }, const WeakString & *end* = { }) const -> WeakString

Generates a substring.

Extracts a substring consisting of the characters from specified start to end.

It's same with substring(? = (str.find(start) + start.size()), find(end, ?))

* between("ABCD[EFGH]IJK", "[", "]") => "EFGH"

Note

* If start is not specified, extracts from begin of the string to end
* If end is not specified, extracts from start to end of the string
* If start and end are all omitted, returns str, itself.

Parameters

start	A string for separating substring at the front
end	A string for separating substring at the end

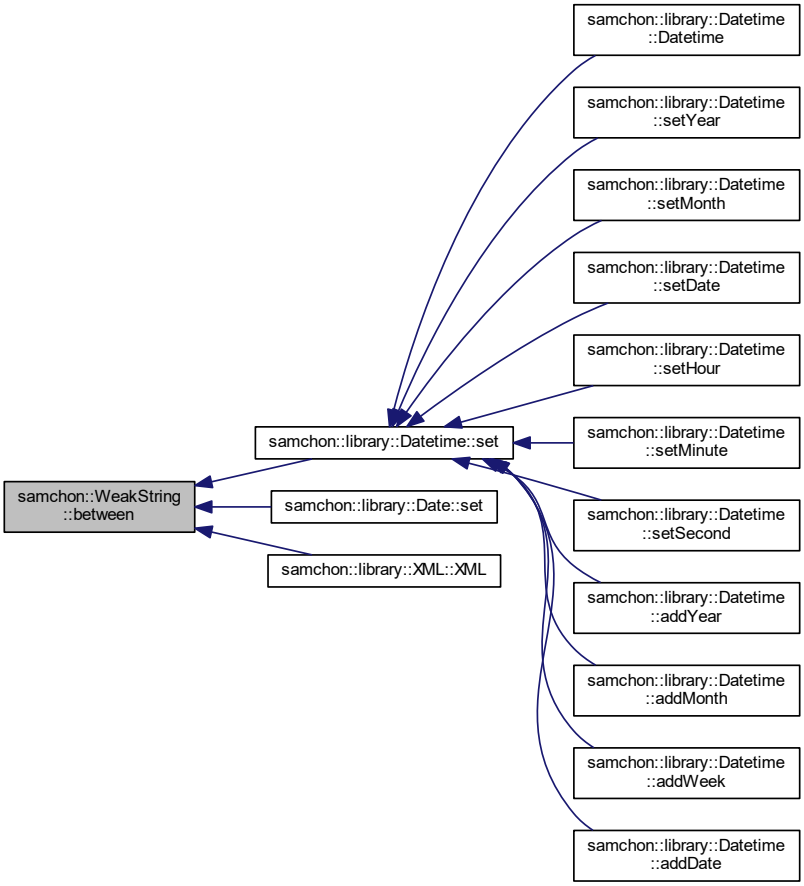
Returns

substring by specified terms

Definition at line 193 of file WeakString.cpp.

Referenced by samchon::library::Datetime::set(), samchon::library::Date::set(), and samchon::library↩ ::XML::XML().

Here is the caller graph for this function:



10.143.3.13 auto WeakString::split (const WeakString & *delim*) const -> std::vector<WeakString>

Generates substrings.

Splits a string in to an array of substrings dividing by the specified delimiter

Parameters

delim	The pattern which specifies where to split the string

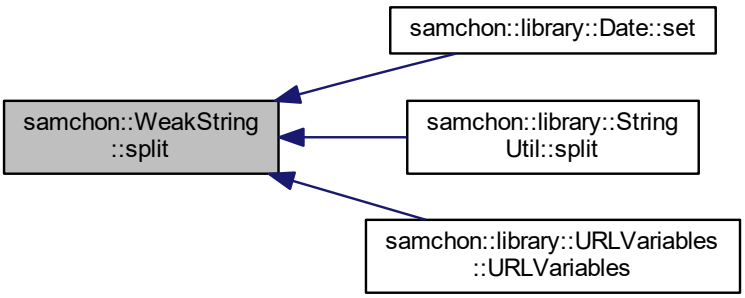
Returns

An array of substrings

Definition at line 209 of file WeakString.cpp.

Referenced by samchon::library::Date::set(), samchon::library::StringUtil::split(), and samchon↩ ::library::URLVariables::URLVariables().

Here is the caller graph for this function:



10.143.3.14 auto WeakString::betweens (const WeakString & *start* = { }, const WeakString & *end* = { }) const -> std::vector<WeakString>

Generates substrings.

Splits a string into an array of substrings dividing by delimeters of start and end.

It's the array of substrings adjusted the between.

Note

* If start is omitted, it's same with the split by endStr not having last item
* If end is omitted, it's same with the split by startStr not having first item
* If start and end are all omitted, returns string, itself

Parameters

start	

A string for separating substring at the front.

If omitted, it's same with split(end) not having last item.

Parameters

end	

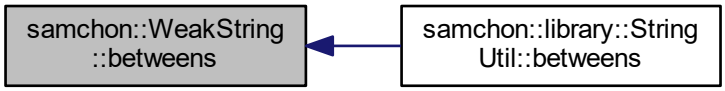
A string for separating substring at the end.

If omitted, it's same with split(start) not having first item.

Returns

An array of substrings

Definition at line 234 of file WeakString.cpp.
Referenced by samchon::library::StringUtil::betweens().
Here is the caller graph for this function:



10.143.3.15 auto WeakString::trim (const std::vector< std::string > & *delims*) const -> WeakString

Removes all designated characters from the beginning and end of the specified string.

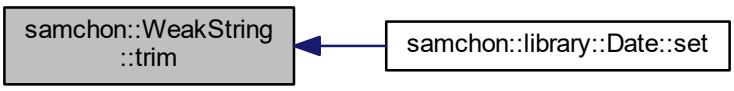
Parameters

delims	Designated character(s)

Returns

Updated string where designated characters was removed from the beginning and end

Definition at line 371 of file WeakString.cpp.
Referenced by samchon::library::Date::set().
Here is the caller graph for this function:



10.143.3.16 auto WeakString::ltrim (const std::vector< std::string > & *delims*) const -> WeakString

Removes all designated characters from the beginning of the specified string.

Parameters

delims	Designated character(s)

Returns

Updated string where designated characters was removed from the beginning

Definition at line 379 of file WeakString.cpp.

10.143.3.17 auto WeakString::rtrim (const std::vector< std::string > & *delims*) const -> WeakString

Removes all designated characters from the end of the specified string.

Parameters

delims	Designated character(s)

Returns

Updated string where designated characters was removed from the end

Definition at line 387 of file WeakString.cpp.

10.143.3.18 auto WeakString::replace (const WeakString & *before,* const WeakString & *after*) const -> std::string

Replace portion of string once.

Parameters

before	A specific word you want to be replaced
after	A specific word you want to replace

Returns

A string specific word is replaced once

Definition at line 399 of file WeakString.cpp.

10.143.3.19 auto WeakString::replaceAll (const WeakString & *before,* const WeakString & *after*) const -> std::string

Returns a string specified word is replaced.

Parameters

before	A specific word you want to be replaced
after	A specific word you want to replace

Returns

A string specified word is replaced

Definition at line 415 of file WeakString.cpp.

10.143.3.20 auto WeakString::replaceAll (const std::vector< std::pair< std::string, std::string >> & *pairs*) const -> std::string

Returns a string specified words are replaced.

Parameters

str	Target string to replace
pairs	A specific word's pairs you want to replace and to be replaced

Returns

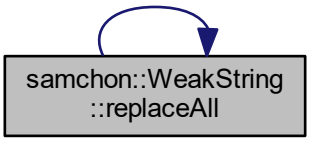
A string specified words are replaced

Definition at line 420 of file WeakString.cpp.

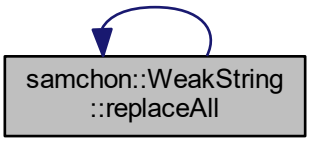
References replaceAll().

Referenced by replaceAll().

Here is the call graph for this function:



Here is the caller graph for this function:



10.143.3.21 auto WeakString::toLowerCase ( ) const -> std::string

Convert uppercase letters to lowercase.

Returns a string that all uppercase characters are converted to lowercase.

Parameters

wstr	Target string to convert uppercase to lowercase

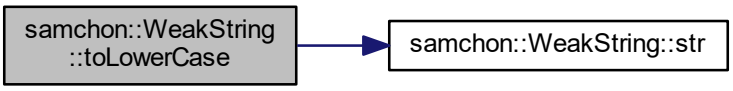
Returns

A string converted to lowercase

Definition at line 488 of file WeakString.cpp.

References str().

Here is the call graph for this function:



10.143.3.22 auto WeakString::toUpperCase ( ) const -> std::string

Convert uppercase letters to lowercase.

Returns a string all lowercase characters are converted to uppercase.

Parameters

str	Target string to convert lowercase to uppercase

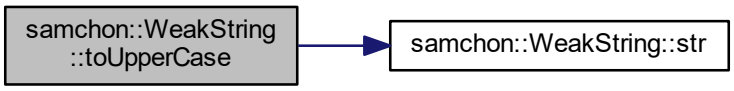
Returns

A string converted to uppercase

Definition at line 497 of file WeakString.cpp.

References str().

Here is the call graph for this function:



10.143.3.23 auto WeakString::str ( ) const -> std::string

Get the string content.

Returns a string object with a copy of the current contents in the WeakString.

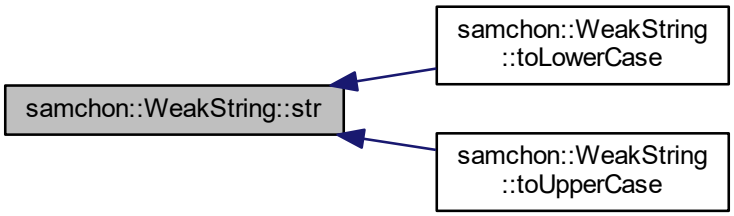
Returns

A new string copied from the WeakString

Definition at line 543 of file WeakString.cpp.

Referenced by toLowerCase(), and toUpperCase().

Here is the caller graph for this function:



10.143.4 Member Data Documentation

10.143.4.1 const vector< std::string > WeakString::SPACE_ARRAY = { " ", "\t", "\r", "\n" } [static], [private]

An array containing whitespaces.

Definition at line 38 of file WeakString.hpp.

10.143.4.2 const size_t samchon::WeakString::npos = -1 [static]

Maximum value for size_t.

npos is a static member constant value with the greatest possible value for an element of type size_t.

This value, when used as the value for a len (or sublen) parameter in string's member functions, means "until the end of the string". As a return value, it is usually used to indicate no matches.

This constant is defined with a value of -1, which because size_t is an unsigned integral type, it is the largest possible representable value for this type.

Definition at line 53 of file WeakString.hpp.

10.143.4.3 const char* samchon::WeakString::data_ [private]

Referenced characters's pointer of begining position.

Definition at line 59 of file WeakString.hpp.

Referenced by data(), empty(), and WeakString().

10.143.4.4 size_t samchon::WeakString::size_ [private]

(Specified) size of referenced characters

Definition at line 64 of file WeakString.hpp.

Referenced by empty(), size(), and WeakString().

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

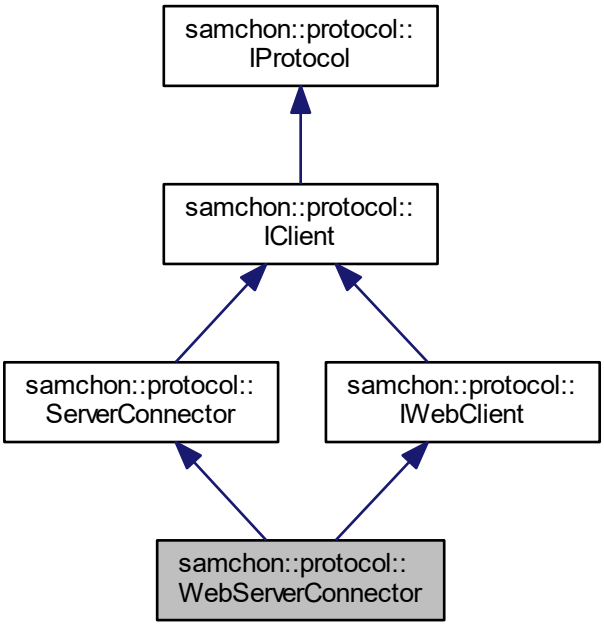
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/WeakString.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/WeakString.cpp

10.144 samchon::protocol::WebServerConnector Class Reference

A web-socket server connector.

```
#include <WebServerConnector.hpp>
```

Collaboration diagram for samchon::protocol::WebServerConnector:



Public Member Functions

* WebServerConnector ()
  Default Constructor.
* virtual void connect () override
  Connect to a server following web-socket protoocol.

Additional Inherited Members

10.144.1 Detailed Description

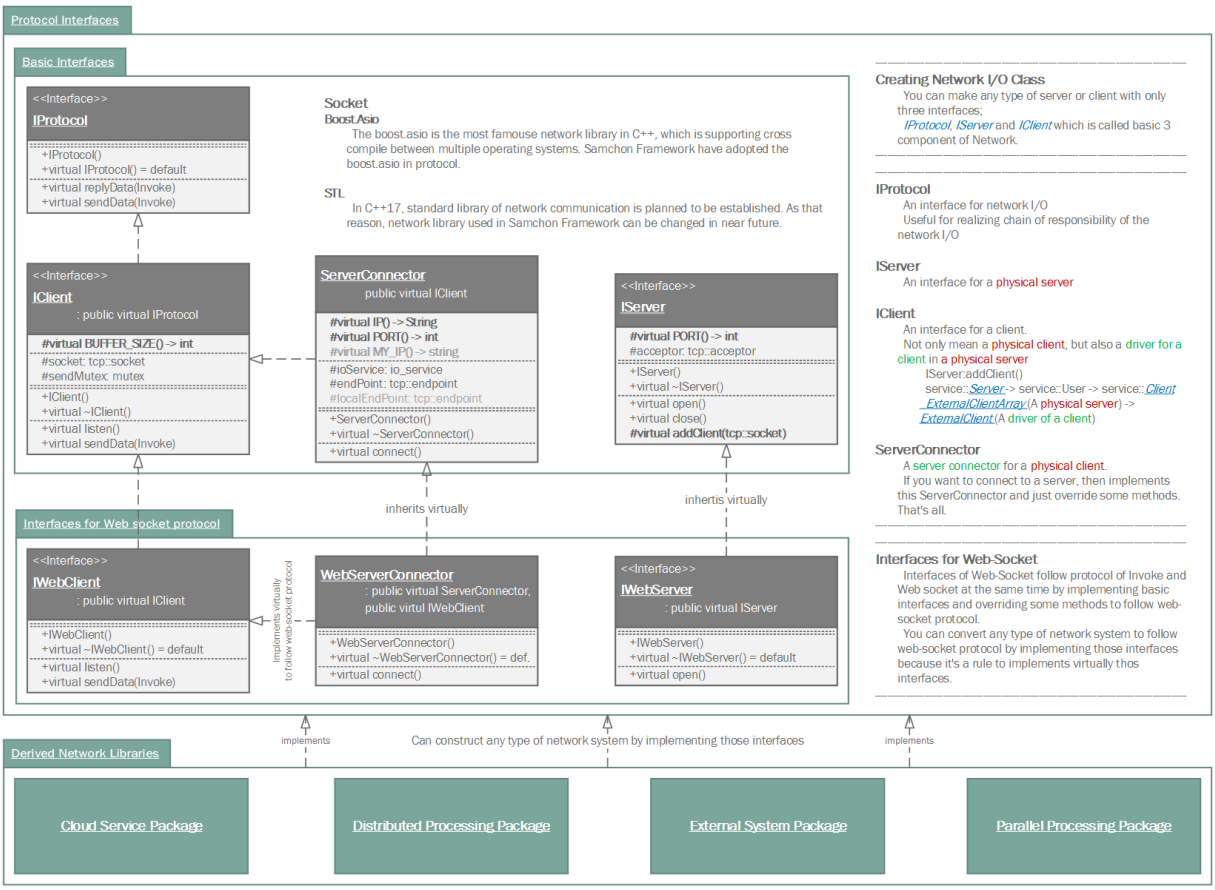
A web-socket server connector.

WebServerConnector is a ServerConnector following web-socket protocol.

[Inherited]

ServerConnector is a class for a physical client connecting a server. If you want to connect to a server, then implements this ServerConnector and just override some methods like getIP(), getPort() and replyData(). That's all.

In Samchon Framework, package protocol, There are basic 3 + 1 components that can make any type of network system in Samchon Framework. The basic 3 components are IProtocol, IServer and IClient. The last, surplus one is the ServerConnector. Looking around classes in Samchon Framework, especially module master and slave which are designed for realizing distributed processing systems and parallel processing systems, physical client classes are all derived from this ServerConnector.



Author

Jeongho Nam

Definition at line 20 of file WebServerConnector.hpp.

10.144.2 Constructor & Destructor Documentation

10.144.2.1 samchon::protocol::WebServerConnector::WebServerConnector ( )

Default Constructor.

10.144.3 Member Function Documentation

10.144.3.1 virtual void samchon::protocol::WebServerConnector::connect ( ) [override], [virtual]

Connect to a server following web-socket protoocol.

Reimplemented from samchon::protocol::ServerConnector.

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

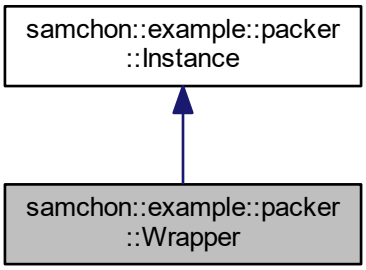
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/protocol/WebServerConnector.hpp

10.145 samchon::example::packer::Wrapper Class Reference

A wrapper can contain products.

```
#include <Wrapper.hpp>
```

Collaboration diagram for samchon::example::packer::Wrapper:



Public Member Functions

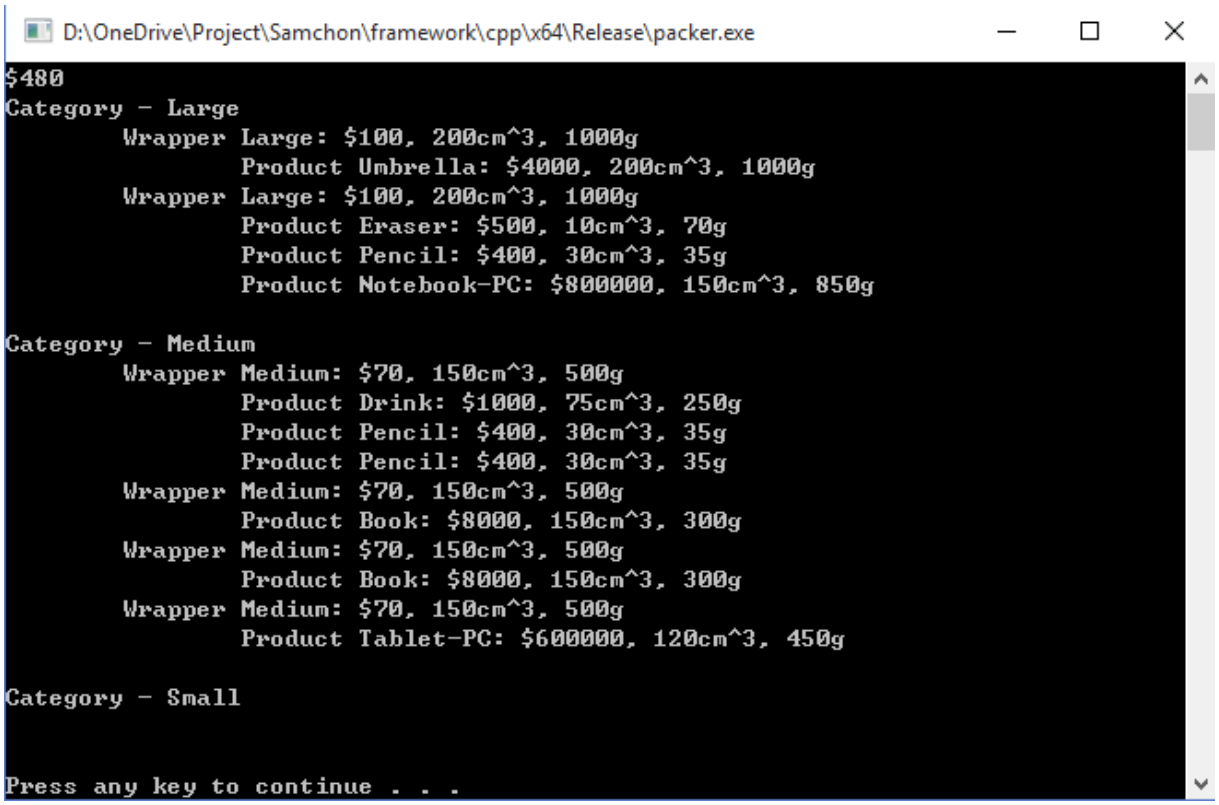
* Wrapper (const std::string &name, int price, int volume, int weight)
  Construct from argument of a wrapper.
* Wrapper (const Wrapper &wrapper)
  Copy Constructor.
* auto tryInsert (Product *) -> bool
  Try to insert a product into the wrapper.
* virtual auto toString () const -> std::string override
  Return a string represent the wrapper.

Additional Inherited Members

10.145.1 Detailed Description

A wrapper can contain products.

A wrapper containing products with its limitation of weight and volume.



Author

Jeongho Nam

Definition at line 24 of file Wrapper.hpp.

10.145.2 Constructor & Destructor Documentation

10.145.2.1 samchon::example::packer::Wrapper::Wrapper (const std::string & *name*, int *price*, int *volume*, int *weight*)

Construct from argument of a wrapper.

Parameters

name	Category name of a wrapper, can be identified
price	Price of a wrapper
volume	Limited volume of a wrapper can put in.
weight	Limited weight of a wrapper can put in.

10.145.2.2 Wrapper::Wrapper (const Wrapper & *wrapper*)

Copy Constructor.

Copy constructor of wrapper does not copy children items. Only copies arguments of Instance's.

Definition at line 15 of file Wrapper.cpp.

10.145.3 Member Function Documentation

10.145.3.1 auto Wrapper::tryInsert (Product * *product*) -> bool

Try to insert a product into the wrapper.

If the Wrapper has enough volume and limit of weight, inserts the Product into the Wrapper and returns *true*. If not enough, does not insert and just return *false*.

Parameters

product	A product try to insert in.

Returns

Whether to success put in

Definition at line 21 of file Wrapper.cpp.

Referenced by samchon::example::packer::WrapperArray::optimize().

Here is the caller graph for this function:

10.145.3.2 auto Wrapper::toString ( ) const -> std::string [override], [virtual]

Return a string represent the wrapper.

Returns a string of the Wrapper and Product(s) packaged in.

Returns

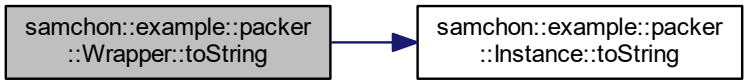
A string represents the Wrapper and contained Product(s).

Reimplemented from samchon::example::packer::Instance.

Definition at line 45 of file Wrapper.cpp.

References samchon::example::packer::Instance::toString().

Here is the call graph for this function:



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

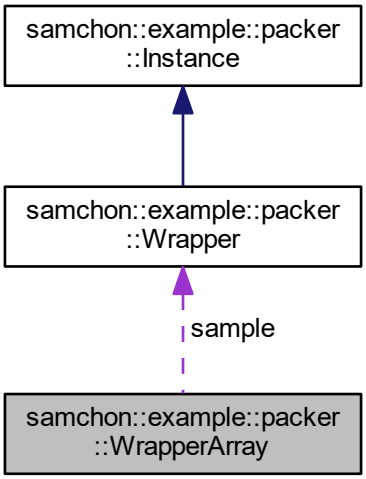
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/packer/Wrapper.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/packer/Wrapper.cpp

10.146 samchon::example::packer::WrapperArray Class Reference

An array of wrapper with same category (name).

```
#include <WrapperArray.hpp>
```

Collaboration diagram for samchon::example::packer::WrapperArray:



Public Member Functions

- WrapperArray (Wrapper ∗)
  Construct from a sample wrapper.
- auto tryInsert (Product ∗) -> bool
  Try to insert a product into reserved list.
- void optimize ()
  Optimize to retrieve the best solution.
- auto calcPrice () const -> int
  Calculate price of the Wrapper(s)
- auto toString () const -> std::string
  Return a string represents Wrapper(s) of same type.

Private Attributes

- std::vector< Product ∗ > reserved
  A list for reserved Product(s).
- Wrapper ∗ sample
  A sample wrapper used to copy.

10.146.1 Detailed Description

An array of wrapper with same category (name).

WrapperArray reserves Product(s) and generates enough number of Wrapper(s).

WrapperArray will retrieve the best arrange and packaging method for reserved Product(s).

```
D:\OneDrive\Project\Samchon\framework\cpp\x64\Release\packer.exe
$480
Category - Large
        Wrapper Large: $100, 200cm^3, 1000g
                Product Umbrella: $4000, 200cm^3, 1000g
        Wrapper Large: $100, 200cm^3, 1000g
                Product Eraser: $500, 10cm^3, 70g
                Product Pencil: $400, 30cm^3, 35g
                Product Notebook-PC: $800000, 150cm^3, 850g

Category - Medium
        Wrapper Medium: $70, 150cm^3, 500g
                Product Drink: $1000, 75cm^3, 250g
                Product Pencil: $400, 30cm^3, 35g
                Product Pencil: $400, 30cm^3, 35g
        Wrapper Medium: $70, 150cm^3, 500g
                Product Book: $8000, 150cm^3, 300g
        Wrapper Medium: $70, 150cm^3, 500g
                Product Book: $8000, 150cm^3, 300g
        Wrapper Medium: $70, 150cm^3, 500g
                Product Tablet-PC: $600000, 120cm^3, 450g

Category - Small


Press any key to continue . . .
```

Author

Jeongho Nam

Definition at line 26 of file WrapperArray.hpp.

10.146.2 Constructor & Destructor Documentation

10.146.2.1 WrapperArray::WrapperArray (Wrapper * *sample*)

Construct from a sample wrapper.

Parameters

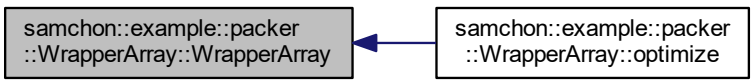
sample	A sample wrapper used to copy wrappers.

Definition at line 15 of file WrapperArray.cpp.

References sample.

Referenced by optimize().

Here is the caller graph for this function:



10.146.3 Member Function Documentation

10.146.3.1 auto WrapperArray::tryInsert (Product * *product*) -> bool

Try to insert a product into reserved list.

If the Product's volume and weight is equal or less than the Wrapper categorized so that enable to insert in a Wrapper, reserve the Product and returns *true*. If not, does not reserve and just return *false*.

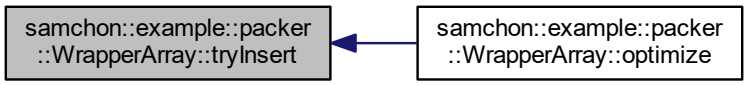
Returns

Whether the Product's volume and weight is equal or less than the Wrapper.

Definition at line 24 of file WrapperArray.cpp.

Referenced by optimize().

Here is the caller graph for this function:



10.146.3.2 void WrapperArray::optimize ( )

Optimize to retrieve the best solution.

Retrieve the best solution of packaging in level of WrapperArray.

Shuffles sequence of reserved Product(s) by samchon::library::FactorialGenerator and insert the reserved Products(s) following the sequence creating Wrapper(s) as needed. Between the sequences from FactorialGenerator, retrieve and determine the best solution.

Note

Sequence of inserting Product can affeact to numbers of Wrapper(s) to be used.

It's the reason why even WrapperArray has the optimize() method.

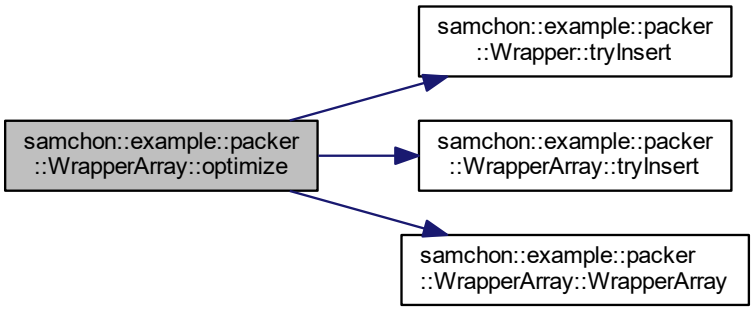
See also

samchon::library::FactorialGenerator

Definition at line 35 of file WrapperArray.cpp.

References reserved, sample, samchon::example::packer::Wrapper::tryInsert(), tryInsert(), and WrapperArray().

Here is the call graph for this function:



10.146.3.3 auto WrapperArray::calcPrice ( ) const -> int

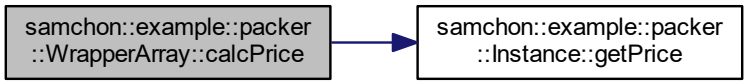
Calculate price of the Wrapper(s)

Calculate price of all wrapppers'. The price does not contain inserted products'

Definition at line 75 of file WrapperArray.cpp.

References samchon::example::packer::Instance::getPrice(), and sample.

Here is the call graph for this function:



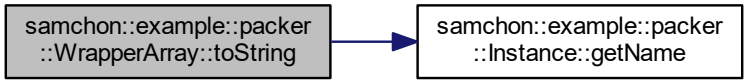
10.146.3.4 auto WrapperArray::toString ( ) const -> std::string

Return a string represents Wrapper(s) of same type.

Definition at line 83 of file WrapperArray.cpp.

References samchon::example::packer::Instance::getName(), and sample.

Here is the call graph for this function:



10.146.4 Member Data Documentation

10.146.4.1 std::vector<Product*> samchon::example::packer::WrapperArray::reserved [private]

A list for reserved Product(s).

Definition at line 35 of file WrapperArray.hpp.

Referenced by optimize().

10.146.4.2 Wrapper* samchon::example::packer::WrapperArray::sample [private]

A sample wrapper used to copy.

Definition at line 40 of file WrapperArray.hpp.

Referenced by calcPrice(), optimize(), toString(), and WrapperArray().

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

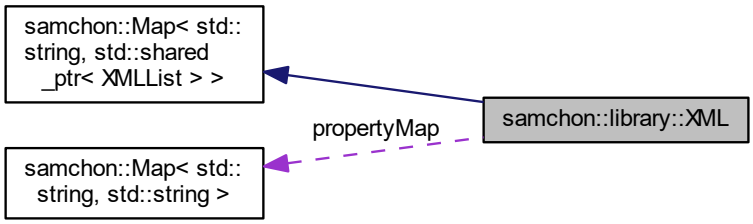
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/packer/WrapperArray.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/examples/packer/WrapperArray.cpp

10.147 samchon::library::XML Class Reference

XML is a class representing xml object.

`#include <XML.hpp>`

Collaboration diagram for samchon::library::XML:



Public Member Functions

* XML ()
  Default Constructor.
* XML (const XML &)
  Copy Constructor.
* XML (XML &&)
  Move Constructor.
* XML (WeakString)
  Constructor by string.
* void push_back (const WeakString &)
  Add children xml objects by string representing them.
* void push_back (const std::shared_ptr< XML >)
  Add children xml.
* void addAllProperty (const std::shared_ptr< XML >)
  Add all properties from another XML.
* void setTag (const std::string &)
  Set tag (identifier) of the XML.
* template<typename _Ty >
  void setValue (const _Ty &val)
  Set value of the XML.
* template<typename _Ty >
  void setProperty (const std::string &name, const _Ty &val)
  Set a property with its key.
* void eraseProperty (const std::string &)
  Erase a property by its key.
* void clearProperties ()
  Remove all properties in the XML.
* auto getTag () const -> std::string
  Get key; identifer of the XML.
* template<class _Ty = std::string>
  auto getValue () const -> _Ty
  Get value of the XML.
* template<class _Ty = std::string>
  auto getProperty (const std::string &key) const -> _Ty
  Get property.
* auto hasProperty (const std::string &) const -> bool
  Test wheter a property exists or not.
* auto getPropertyMap () const -> const Map< std::string, std::string > &
  Get propertyMap.
* auto toString (size_t level=0) const -> std::string
  Get the string content.

Private Member Functions

* XML (XML *, WeakString &)
  Protected Constructor by string for child.

Private Attributes

* std::string tag
  Tag name.
* std::string value
  Value of the XML.
* Map< std::string, std::string > propertyMap
  Properties belongs to the XML.

10.147.1 Detailed Description

XML is a class representing xml object.

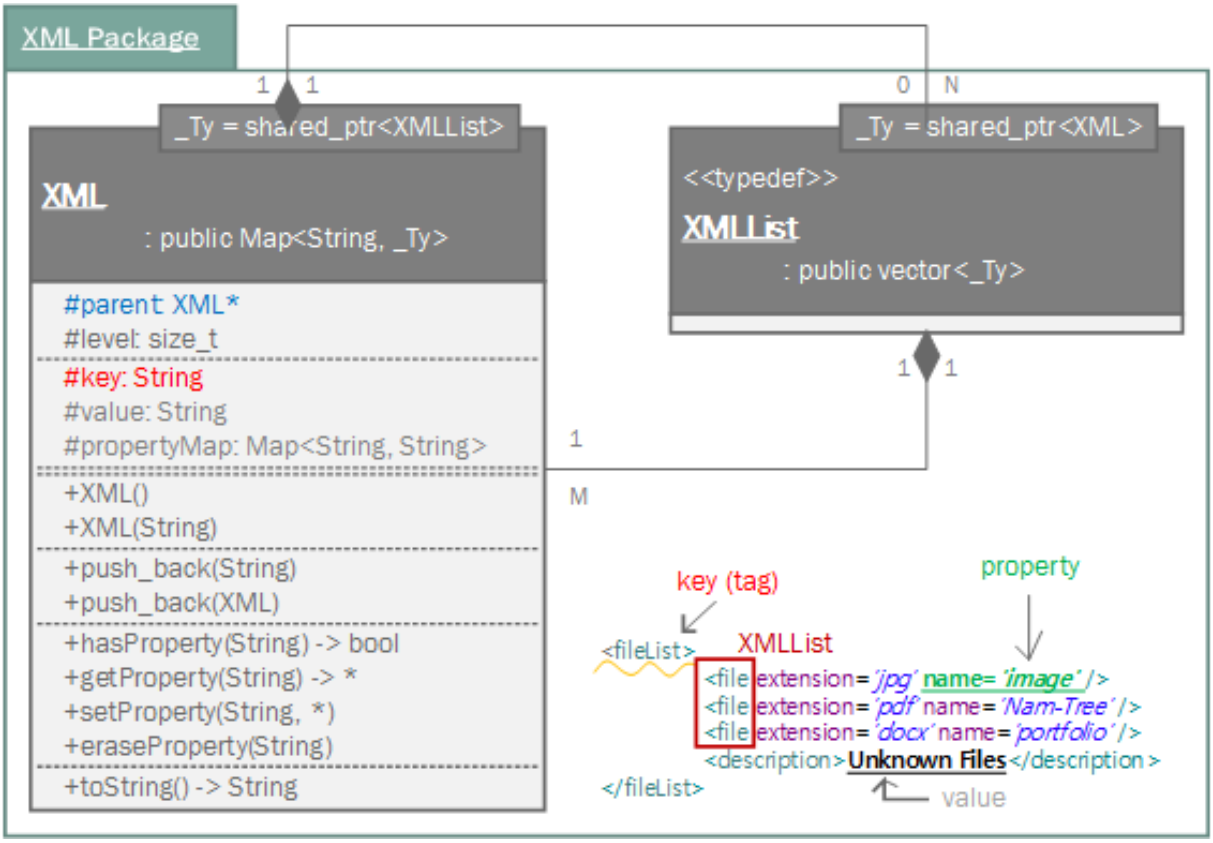
The XML class provides methods and properties for working with XML objects.

The XML class (along with the XMLList and Namespace) implements the powerful XML-handling standard defined in ECMAScript for XML (E4X) specification.

XML class has a recursive, hierarchical relationship.

All XML objects're managed by shared_ptr.

* XML contains XMLList from dictionary of shared pointer<XMLList>
* XMLList contains XML from vector of shared pointer<XML>
* Even if user creates an XML object directly, it's the basic principle to use shared pointer



Note

Parsing comment is not supported yet.

It's not recommeded to creating an XML object which is not being managed by shared pointer.

Warning

Do not abuse values for expressing member variables.

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

```
#include <iostream>
#include <vector>
#include <string>

#include <samchon/library/XML.hpp>

#ifdef _WIN64
#   ifdef _DEBUG
#       pragma comment(lib, "x64/Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "x64/Release/SamchonFramework.lib")
#   endif
#else
#   ifdef _DEBUG
#       pragma comment(lib, "Debug/SamchonFramework.lib")
#   else
#       pragma comment(lib, "Release/SamchonFramework.lib")
#   endif
#endif

using namespace std;
using namespace samchon::library;

struct Member
{
    string id;
    string name;
    int age;
    int grade;

    auto toXML() const -> shared_ptr<XML>
    {
        shared_ptr<XML> xml(new XML());
        xml->setTag("member");
        xml->setProperty("id", id);
        xml->setProperty("name", name);
        xml->setProperty("age", age);
        xml->setProperty("grade", grade);

        return xml;
    };
};

void main()
{
    //DECLARE An XML WITH MEMBER_ARRAY TAG
    shared_ptr<XML> xml(new XML());
    xml->setTag("memberArray");

    //ADD MEMBER(S) TO XML
    vector<struct Member> memberArray =
    {
        {"samchon", "Jeongho Nam", 27, 1},
        {"gkyu", "Kwangkyu Ko", 25, 1},
        {"guest", "John Doe", 99, 4}
    };
    for(size_t i = 0; i < memberArray.size(); i++)
        xml->push_back( memberArray[i].toXML() );

    //ADD FILE_LIST BY STRING DIRECTLY
    xml->push_back
    (
        string("") +
        "<fileList>\n" +
            "<file extension='pdf' name='API' />" +
            "<file extension='pdf' name='Guidance+For+Developer'>Damaged</file>" +
            "<file extension='docx' name='Resume' />" +
            "<file extension='jpg' name='My+House' />" +
            "<file extension='xlsx' name='Grades' />" +
        "</fileList>"
```

```
    );

    cout << "-----------------------------------------------------------------" << endl;
    cout << " Get Properties And Values" << endl;
    cout << "-----------------------------------------------------------------" << endl;
    cout << "Age of 2nd member: " << xml->get("member")->at(1)->getProperty<int>("age") << endl;
    cout << "Age of 1st member: " << xml->get("member")->at(0)->getProperty("id") << endl << endl;

    cout << "File name and extension of 5th: "
        << xml->get("fileList")->at(0)->get("file")->at(4)->getProperty("name") << "."
        << xml->get("fileList")->at(0)->get("file")->at(4)->getProperty("extension") << endl;
    cout << "Value of 2nd file: " << xml->get("fileList")->at(0)->get("file")->at(1)->getValue() << endl << endl;

    cout << "-----------------------------------------------------------------" << endl;
    cout << " XML to String: " << endl;
    cout << "-----------------------------------------------------------------" << endl;
    cout << xml->toString() << endl;

    system("pause");
}
```

See also

samchon::library

Author

Jeongho Nam

Definition at line 72 of file XML.hpp.

10.147.2 Constructor & Destructor Documentation

10.147.2.1 XML::XML ( )

Default Constructor.

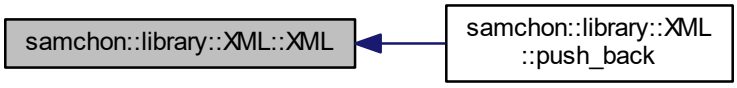
Warning

Declare XML to managed by shared pointer

Definition at line 36 of file XML.cpp.

Referenced by push_back().

Here is the caller graph for this function:



10.147.2.2 XML::XML (const XML & *xml*)

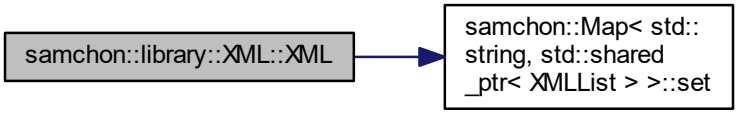
Copy Constructor.

Not copying (shared) pointer of children xml objects, but copying the real objects of children xml

Definition at line 42 of file XML.cpp.

References propertyMap, samchon::Map< std::string, std::shared_ptr< XMLList > >::set(), tag, and value.

Here is the call graph for this function:



10.147.2.3 XML::XML (XML && *xml*)

Move Constructor.

Definition at line 65 of file XML.cpp.

References propertyMap, tag, and value.

10.147.2.4 XML::XML (WeakString *wStr*)

Constructor by string.

Parses a string so that constructs an XML object

Parameters

str	A string representing xml object

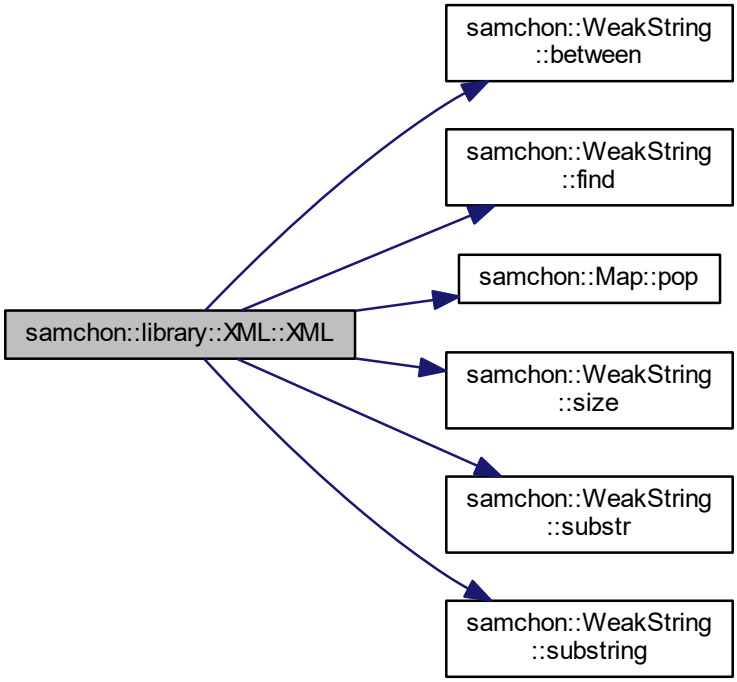
Warning

Declare XML to managed by shared pointer

Definition at line 74 of file XML.cpp.

References samchon::WeakString::between(), samchon::WeakString::find(), samchon::Map< _Kty, _Ty, _Pr, _Alloc >::pop(), samchon::WeakString::size(), samchon::WeakString::substr(), and samchon::WeakString::substring().

Here is the call graph for this function:



10.147.2.5 XML::XML (XML * *parent*, WeakString & *str*) [private]

Protected Constructor by string for child.

Parses a string so that creates an XML object It is called for creating children XML objects from parent XML object.

Parameters

parent	Parent object who will contains this XML object
str	A string to be parsed

Definition at line 117 of file XML.cpp.

10.147.3 Member Function Documentation

10.147.3.1 void XML::push_back (const WeakString & *str*)

Add children xml objects by string representing them.

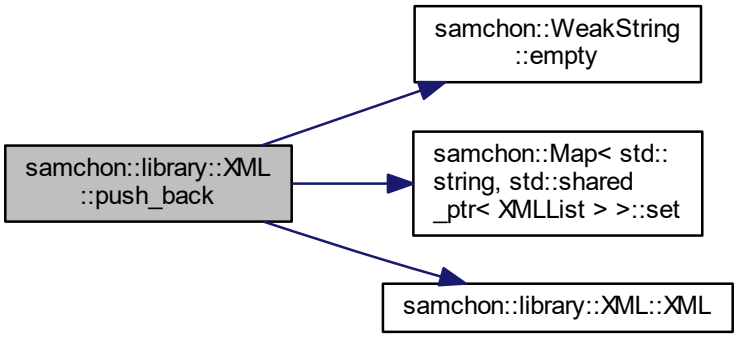
Parameters

str	A string representing xml objects whould be belonged to this XML

Definition at line 436 of file XML.cpp.

References samchon::WeakString::empty(), samchon::Map< std::string, std::shared_ptr< XMLList > >::set(), and XML().

Here is the call graph for this function:



10.147.3.2 void samchon::library::XML::push_back (const std::shared_ptr< XML >)

Add children xml.

Parameters

xml	An xml object you want to add

10.147.3.3 void XML::addAllProperty (const std::shared_ptr< XML >)

Add all properties from another XML.

Copies all properties from target to here.

Warning

Not a category of assign, but an insert.

Parameters

xml	Target xml object to deliver its properties

Definition at line 466 of file XML.cpp.

References propertyMap.

10.147.3.4 void XML::setTag (const std::string & *tag*)

Set tag (identifier) of the XML.

See also

XML::tag

Definition at line 381 of file XML.cpp.

References tag.

10.147.3.5 template<typename _Ty > void samchon::library::XML::setValue (const _Ty & *val*) [inline]

Set value of the XML.

Template Parameters

_Ty	Type of the value

Parameters

val	The value to set

Warning

Do not abuse values for expressing member variables

Standard Usage	Non-standard usage abusing value
<memberList> <member id='jhnam88' name='Jeongho+Nam' birthdate='1988-03-11' /> <member id='master' name='Administartor' birthdate='2011-07-28' /> </memberList>	<member> <id>jhnam88</id> <name>Jeongho+Nam</name> <birthdate>1988-03-11</birthdate> </member>

Definition at line 229 of file XML.hpp.

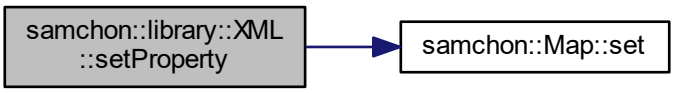
10.147.3.6 template<typename _Ty > void samchon::library::XML::setProperty (const std::string & *name,* const _Ty & *val*) [inline]

Set a property with its key.

Definition at line 241 of file XML.hpp.

References samchon::Map< _Kty, _Ty, _Pr, _Alloc >::set().

Here is the call graph for this function:



10.147.3.7 void XML::eraseProperty (const std::string & *tag*)

Erase a property by its key.

Parameters

key	The key of the property to erase

Exceptions

exception	Unable to find the element

Definition at line 473 of file XML.cpp.

References propertyMap.

10.147.3.8 void XML::clearProperties ( )

Remove all properties in the XML.

Definition at line 477 of file XML.cpp.

References propertyMap.

10.147.3.9 auto XML::getTag ( ) const -> std::string

Get key; identifer of the XML.

Returns

tag, identifer of the XML

See also

XML::tag

Definition at line 327 of file XML.cpp.

References tag.

10.147.3.10 template<class _Ty = std::string> auto samchon::library::XML::getValue ( ) const -> _Ty [inline]

Get value of the XML.

Definition at line 277 of file XML.hpp.

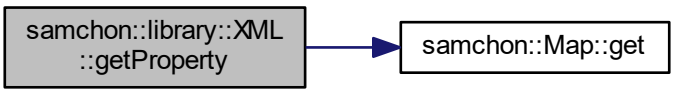
10.147.3.11 template<class _Ty = std::string> auto samchon::library::XML::getProperty (const std::string & *key*) const -> _Ty [inline]

Get property.

Definition at line 291 of file XML.hpp.

References samchon::Map< _Kty, _Ty, _Pr, _Alloc >::get().

Here is the call graph for this function:



10.147.3.12 auto XML::hasProperty (const std::string & *tag*) const -> bool

Test wheter a property exists or not.

Definition at line 488 of file XML.cpp.

10.147.3.13 auto XML::getPropertyMap ( ) const -> const Map<std::string, std::string>&

Get propertyMap.

Definition at line 482 of file XML.cpp.

References propertyMap.

10.147.3.14 auto XML::toString (size_t *level* = 0) const -> std::string

Get the string content.

Returns a string representation of the XML and its all children

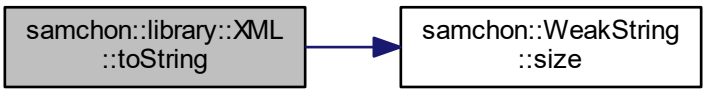
Returns

A string representing the xml

Definition at line 644 of file XML.cpp.

References samchon::WeakString::size().

Here is the call graph for this function:



10.147.4 Member Data Documentation

10.147.4.1 std::string samchon::library::XML::tag [private]

Tag name.

* <**tag** label='property' />: tag => "tag"
* <**price** high='1500' low='1300' open='1450' close='1320' />: tag => "price"

Definition at line 85 of file XML.hpp.

Referenced by getTag(), setTag(), and XML().

10.147.4.2 std::string samchon::library::XML::value [private]

Value of the XML.

* <parameter name='age' type='int'>**26**</parameter>: value => 26
* <price high='1500' low='1300' open='1450' close='1320' />: tag => null

Definition at line 94 of file XML.hpp.

Referenced by XML().

10.147.4.3 Map<std::string, std::string> samchon::library::XML::propertyMap [private]

Properties belongs to the XML.

A Dictionary of properties accessing each property by its key.

* <price **high='1500' low='1300' open='1450' close='1320'** />: propertyMap => {{"high": 1500}, {"low": 1300}, {"open": 1450}, {"close", 1320}}
* <member **id='jhnam88' name='Jeongho+Nam' comment='Hello.+My+name+is+Jeongho+Nam'** >: propertyMap => {{"id", "jhnam88"}, {"name", "Jeongho Nam"}, {"comment", "Hello. My name is Jeongho Nam"}}

Definition at line 106 of file XML.hpp.

Referenced by addAllProperty(), clearProperties(), eraseProperty(), getPropertyMap(), and XML().

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

* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/XML.hpp
* D:/OneDrive/Project/Samchon/framework/cpp/samchon/library/XML.cpp

Index

_replyData
samchon::protocol::IClient, 376
samchon::protocol::master::ParallelSystem, 540
samchon::protocol::slave::ParallelSystem, 537
samchon::protocol::slave::SlaveSystem, 609
~ChatClient
samchon::example::console_chat_server::ChatClient, 98
~SharedAcquire
samchon::library::SharedAcquire, 595
~SharedReadLock
samchon::library::SharedReadLock, 599
~SharedWriteLock
samchon::library::SharedWriteLock, 602
~SmartPointer
samchon::SmartPointer, 612
~UniqueAcquire
samchon::library::UniqueAcquire, 647
~UniqueReadLock
samchon::library::UniqueReadLock, 651
~UniqueWriteLock
samchon::library::UniqueWriteLock, 655

accept
samchon::protocol::FlashPolicyServer, 331
acceptor
samchon::protocol::IServer, 437
acquire
samchon::library::Semaphore, 577
samchon::library::SharedAcquire, 595
samchon::library::UniqueAcquire, 647
acquired
samchon::library::Semaphore, 579
acquiredSize
samchon::library::Semaphore, 577
addAllProperty
samchon::library::XML, 705
addClient
samchon::example::console_chat_server::ChatServer, 115
samchon::protocol::ExternalClientArray, 299
samchon::protocol::IServer, 436
samchon::protocol::service::Server, 583
samchon::protocol::service::User, 664
addDate
samchon::library::Date, 151
samchon::library::Datetime, 174
addEventListener
samchon::library::EventDispatcher, 291
addHour
samchon::library::Datetime, 175
addMinute
samchon::library::Datetime, 175
addMonth
samchon::library::Date, 149
samchon::library::Datetime, 172
addSecond
samchon::library::Datetime, 176
addTab
samchon::library::StringUtil, 632
addWeek
samchon::library::Date, 150
samchon::library::Datetime, 173
addYear
samchon::library::Date, 148
samchon::library::Datetime, 172
allocate
samchon::library::CriticalAllocator, 135
allocateRoles
samchon::protocol::master::DistributedSystemArray, 225
allocatedSystems
samchon::protocol::master::DistributedSystemRole, 236
allocationHistoryList
samchon::protocol::master::DistributedSystemRole, 237
application
samchon::library::FTFactory, 336
archive
samchon::protocol::ISQLEntity, 439
at
samchon::WeakString, 678
samchon::library::CaseGenerator, 90
samchon::library::SQLStatement, 622
authority
samchon::protocol::service::User, 671
avgElapsedTime
samchon::protocol::master::DistributedSystem, 219

BUFFER_SIZE
samchon::protocol::IClient, 375
begin
samchon::namtree::NTEntityGroup, 468
samchon::protocol::service::Server, 581
between
samchon::WeakString, 682
samchon::library::StringUtil, 631
betweens
samchon::WeakString, 684
samchon::library::StringUtil, 632
bindParameterBASizeMap
samchon::library::SQLStatement, 623
bindParameterCount
samchon::library::SQLStatement, 623

ByteArray
samchon::ByteArray, 77
byteArray
samchon::protocol::InvokeParameter, 424

CHILD_TAG
samchon::example::chat_service::ChatRoomArray, 110
samchon::library::FTFolder, 341
samchon::namtree::NTCriteria, 466
samchon::namtree::NTParameter, 483
samchon::namtree::NTParameterArray, 487
samchon::protocol::ExternalSystem, 314
samchon::protocol::ExternalSystemArray, 320
samchon::protocol::IEntityGroup, 381
samchon::protocol::Invoke, 405
samchon::protocol::InvokeHistoryArray, 416
calcAverageElapsedTime
samchon::protocol::master::PRInvokeHistory, 559
calcDistance
samchon::example::tsp::GeometryPoint, 371
samchon::example::tsp::Travel, 639
calcElapsedTime
samchon::protocol::InvokeHistory, 411
calcLastDates
samchon::library::Date, 142
calcPrice
samchon::example::packer::Packer, 498
samchon::example::packer::WrapperArray, 697
calcSeconds
samchon::library::Date, 141
CaseGenerator
samchon::library::CaseGenerator, 89
category
samchon::library::FTFactory, 336
ChatClient
samchon::example::chat_service::ChatClient, 96
samchon::example::console_chat_server::ChatClient, 98
ChatMessage
samchon::example::chat_service::ChatMessage, 101
ChatRoom
samchon::example::chat_service::ChatRoom, 105
ChatRoomArray
samchon::example::chat_service::ChatRoomArray, 110
ChatServer
samchon::example::chat_service::ChatServer, 118
samchon::example::console_chat_server::ChatServer, 114
ChatService
samchon::example::chat_service::ChatService, 120
ChatUser
samchon::example::chat_service::ChatUser, 122
children
samchon::library::GAPopulation, 355
clearProperties
samchon::library::XML, 706
Client
samchon::protocol::service::Client, 124
client
samchon::protocol::service::Service, 590
samchon::protocol::service::ServiceKeeper, 593
clientSet
samchon::example::console_chat_server::ChatServer, 11
close
samchon::protocol::IServer, 436
colorNumberFormat
samchon::library::StringUtil, 628
colorPercentFormat
samchon::library::StringUtil, 629
CombinedPermutationGenerator
samchon::library::CombinedPermutationGenerator, 131
comment
samchon::library::FTInstance, 347
completed
samchon::protocol::master::PRMasterHistory, 565
compress
samchon::ByteArray, 79
connect
samchon::library::SQLi, 617
samchon::protocol::ServerConnector, 586
samchon::protocol::WebServerConnector, 691
construct
samchon::example::chat_service::ChatMessage, 101
samchon::library::CriticalAllocator, 134
samchon::namtree::INTExplore, 397
samchon::namtree::NTCriteria, 465
samchon::namtree::NTFile, 475
samchon::namtree::NTParameter, 481
samchon::namtree::NTParameterDetermined, 490
samchon::namtree::NTSide, 494
samchon::protocol::Entity, 248
samchon::protocol::EntityArray, 257
samchon::protocol::EntityGroup, 267
samchon::protocol::EntityList, 279
samchon::protocol::ExternalClientArray, 298
samchon::protocol::ExternalServer, 304
samchon::protocol::ExternalSystem, 312
samchon::protocol::ExternalSystemRole, 323
samchon::protocol::Invoke, 403
samchon::protocol::InvokeHistory, 409
samchon::protocol::InvokeParameter, 421
samchon::protocol::SystemRole, 635
samchon::protocol::master::DistributedClientArray, 186
samchon::protocol::master::DistributedClientArrayMediato
samchon::protocol::master::DistributedServer, 199
samchon::protocol::master::DistributedSystem, 218
samchon::protocol::master::DistributedSystemArray, 223
samchon::protocol::master::DistributedSystemArrayMedia
samchon::protocol::master::PRInvokeHistory, 558
samchon::protocol::master::ParallelClientArrayMediator, 5
samchon::protocol::master::ParallelServer, 512
samchon::protocol::master::ParallelSystem, 540
samchon::protocol::master::ParallelSystemArrayMediator
constructService
samchon::protocol::service::Client, 128
createChild
samchon::namtree::NTCriteria, 465
samchon::namtree::NTParameter, 482
samchon::namtree::NTParameterArray, 487

samchon::protocol::EntityGroup, 268
samchon::protocol::Invoke, 404
samchon::protocol::InvokeHistoryArray, 415
samchon::protocol::master::DSInvokeHistoryList, 242
samchon::protocol::master::PRInvokeHistoryArray, 562
samchon::protocol::master::ParallelSystem, 540
samchon::protocol::slave::ParallelSystem, 537
createClient
samchon::example::chat_service::ChatUser, 122
samchon::protocol::service::User, 664
createCriteria
samchon::namtree::NTFactory, 471
createFile
samchon::library::FTFactory, 335
createRole
samchon::protocol::master::DistributedSystemArray, 225
createRoom
samchon::example::chat_service::ListService, 447
createService
samchon::example::chat_service::ChatClient, 96
samchon::protocol::service::Client, 125
createSide
samchon::namtree::NTFactory, 471
createSlave
samchon::protocol::master::DistributedSystemArrayMediator, 262
createStatement
samchon::library::SQLi, 617
samchon::library::TSQLi, 642
createUser
samchon::example::chat_service::ChatServer, 118
samchon::protocol::service::Server, 582
CriticalDictionary
samchon::library, 37
CriticalList
samchon::library, 37
CriticalMap
samchon::library, 38
CriticalSet
samchon::library, 38
CriticalVector
samchon::library, 38
crossover
samchon::library::GeneticAlgorithm, 366

DSInvokeHistory
samchon::protocol::master::DSInvokeHistory, 240
DSInvokeHistoryList
samchon::protocol::master::DSInvokeHistoryList, 242
data
samchon::WeakString, 676
samchon::library::FTByteFile, 334
samchon::library::FTTextFile, 350
samchon::namtree::NTFactory, 472
samchon::namtree::NTIterator, 479
data_
samchon::WeakString, 689
Date
samchon::library::Date, 139–141
Datetime
samchon::library::Datetime, 162, 163
deallocate
samchon::library::CriticalAllocator, 136
decode
samchon::library::Base64, 74
samchon::library::URLVariables, 659
decompress
samchon::ByteArray, 79
degree_to_radian
samchon::library::Math, 454
denominator
samchon::library::ProgressEvent, 569
destroy
samchon::library::CriticalAllocator, 135
Dictionary
samchon, 24
disconnect
samchon::library::SQLi, 617
dispatchEvent
samchon::library::EventDispatcher, 291
dispatchProgressEvent
samchon::library::EventDispatcher, 292
distance
samchon::example::tsp::Travel, 640
DistributedClient
samchon::protocol::master::DistributedClient, 184
DistributedClientArray
samchon::protocol::master::DistributedClientArray, 186
DistributedClientArrayMediator
samchon::protocol::master::DistributedClientArrayMediat
DistributedClientServerMediator
samchon::protocol::master::DistributedClientServerMedia
DistributedServer
samchon::protocol::master::DistributedServer, 198
DistributedServerArray
samchon::protocol::master::DistributedServerArray, 202
DistributedServerArrayMediator
samchon::protocol::master::DistributedServerArrayMedia
DistributedSlaveServerMediator
samchon::protocol::master::DistributedSlaveServerMedia
DistributedSlaveSystemMediator
samchon::protocol::master::DistributedSlaveSystemMedi
DistributedSystem
samchon::protocol::master::DistributedSystem, 217
DistributedSystemArray
samchon::protocol::master::DistributedSystemArray, 223
DistributedSystemArrayMediator
samchon::protocol::master::DistributedSystemArrayMedia
DistributedSystemRole
samchon::protocol::master::DistributedSystemRole, 235
doJoin
samchon::protocol::service::User, 668
doLogin
samchon::protocol::service::User, 668
driver
samchon::library::SQLi, 617

E
samchon::library::Math, 458

empty
samchon::WeakString, 677
encode
samchon::library::Base64, 74
samchon::library::URLVariables, 659
end
samchon::namtree::NTEntityGroup, 468
samchon::protocol::service::Server, 582
endPoint
samchon::protocol::ServerConnector, 587
endTime
samchon::protocol::InvokeHistory, 414
Entity
samchon::protocol::Entity, 247
entity
samchon::protocol::IEntityChain, 379
EntityArray
samchon::protocol::EntityArray, 257
EntityGroup
samchon::protocol::EntityGroup, 266
EntityList
samchon::protocol::EntityList, 279
eraseClient
samchon::example::chat_service::ChatRoom, 106
samchon::example::console_chat_server::ChatServer, 114
samchon::protocol::service::User, 665
eraseProperty
samchon::library::XML, 706
ErrorEvent
samchon::library::ErrorEvent, 283
Event
samchon::library::Event, 286
EventDispatcher
samchon::library::EventDispatcher, 290, 291
eventSetMap
samchon::library::EventDispatcher, 292
evolveGeneArray
samchon::library::GeneticAlgorithm, 363
evolvePopulation
samchon::library::GeneticAlgorithm, 363
execute
samchon::library::SQLStatement, 621
executeDirectly
samchon::library::SQLStatement, 621
extension
samchon::library::FTFile, 339
ExternalClient
samchon::protocol::ExternalClient, 294
ExternalClientArray
samchon::protocol::ExternalClientArray, 297
ExternalServer
samchon::protocol::ExternalServer, 303
ExternalSystem
samchon::protocol::ExternalSystem, 311
ExternalSystemArray
samchon::protocol::ExternalSystemArray, 318
ExternalSystemRole
samchon::protocol::ExternalSystemRole, 322
FTByteFile
samchon::library::FTByteFile, 333
FTFactory
samchon::library::FTFactory, 335
FTFile
samchon::library::FTFile, 338
FTFolder
samchon::library::FTFolder, 341
FTInstance
samchon::library::FTInstance, 344
FTTextFile
samchon::library::FTTextFile, 349
FactorialGenerator
samchon::library::FactorialGenerator, 329
factory
samchon::library::FTFolder, 342
samchon::namtree::NTFile, 476
fetch
samchon::library::SQLStatement, 622
find
samchon::WeakString, 678
finds
samchon::WeakString, 679
samchon::library::StringUtil, 630
fitTest
samchon::library::GAPopulation, 354
FlashPolicyServer
samchon::protocol::FlashPolicyServer, 330
free
samchon::library::SQLStatement, 621
function
samchon::namtree::NTFile, 477

GAPopulation
samchon::library::GAPopulation, 354
gaParameters
samchon::example::tsp::Scheduler, 575
samchon::protocol::master::DistributedSystemArray, 226
generation
samchon::library::GAParameters, 351
GeneticAlgorithm
samchon::library::GeneticAlgorithm, 362
GeometryPoint
samchon::example::tsp::GeometryPoint, 370
get
samchon::Map, 449
samchon::SmartPointer, 612
samchon::library::SQLStatement, 623
samchon::protocol::EntityGroup, 269, 271
getAllocationHistoryList
samchon::protocol::master::DistributedSystemRole, 236
getAuthority
samchon::protocol::service::User, 663
getChatRoomArray
samchon::example::chat_service::ChatServer, 119
getClient
samchon::protocol::service::Service, 590
getComment
samchon::library::FTInstance, 346

getData
samchon::library::FTByteFile, 333
samchon::library::FTTextFile, 349
getDate
samchon::library::Date, 154
getDay
samchon::library::Date, 155
getDenominator
samchon::library::ProgressEvent, 569
getEndTime
samchon::protocol::InvokeHistory, 410
getExtension
samchon::library::FTFile, 338
getFunction
samchon::namtree::NTFile, 475
getHour
samchon::library::Datetime, 176
getID
samchon::protocol::service::User, 663
getIP
samchon::protocol::ExternalServer, 305
samchon::protocol::ServerConnector, 585
getIndex
samchon::IndexPair, 390
samchon::protocol::master::PRInvokeHistory, 559
getInitialValue
samchon::namtree::NTParameter, 482
getInvokeHistoryList
samchon::protocol::master::DistributedSystemRole, 236
getLabel
samchon::namtree::NTParameterDetermined, 490
getListener
samchon::example::chat_service::ChatMessage, 102
samchon::protocol::Invoke, 404
samchon::protocol::InvokeHistory, 410
getMaximum
samchon::namtree::INTExplore, 397
getMessage
samchon::library::ErrorEvent, 283
getMinimum
samchon::namtree::INTExplore, 397
getMinute
samchon::library::Datetime, 178
getMonth
samchon::library::Date, 153
getMutex
samchon::library::CriticalAllocator, 134
getMyIP
samchon::protocol::ExternalServer, 305
samchon::protocol::ServerConnector, 586
getName
samchon::example::packer::Instance, 393
samchon::library::FTInstance, 346
samchon::namtree::NTParameter, 482
samchon::protocol::InvokeParameter, 421
getNo
samchon::protocol::service::Client, 125
getNumerator
samchon::library::ProgressEvent, 569
getOtherside
samchon::namtree::NTFile, 475
getParameterArray
samchon::namtree::NTFile, 475
getParent
samchon::library::FTInstance, 345
getPercent
samchon::library::ProgressEvent, 569
getPerformance
samchon::protocol::master::DistributedSystemRole, 235
getPort
samchon::protocol::ExternalServer, 305
samchon::protocol::ServerConnector, 586
getPosition
samchon::ByteArray, 77
getPrecision
samchon::namtree::INTExplore, 398
getPrice
samchon::example::packer::Instance, 393
getProperty
samchon::library::XML, 707
getPropertyMap
samchon::library::XML, 707
getRole
samchon::protocol::ExternalSystemArray, 319
samchon::protocol::master::DSInvokeHistory, 240
samchon::protocol::master::DistributedSystemArray, 225
getSecond
samchon::library::Datetime, 179
getSection
samchon::namtree::INTExplore, 398
getSemaphore
samchon::protocol::service::User, 663
getServer
samchon::protocol::service::User, 663
getService
samchon::protocol::service::Client, 125
getSessionID
samchon::protocol::service::IPUserPair, 430
getSize
samchon::protocol::master::PRInvokeHistory, 559
getSource
samchon::library::Event, 287
getStartTime
samchon::protocol::InvokeHistory, 410
getSystem
samchon::protocol::ExternalSystemRole, 323
samchon::protocol::master::DSInvokeHistory, 240
samchon::protocol::master::DistributedSystemRole, 235
getTag
samchon::library::XML, 706
getType
samchon::library::Event, 287
samchon::protocol::InvokeParameter, 421
getUID
samchon::library::FTInstance, 345
samchon::protocol::InvokeHistory, 410

getUser
samchon::protocol::service::Client, 125
getValue
samchon::IndexPair, 390, 391
samchon::library::XML, 707
samchon::namtree::NTParameterDetermined, 490
samchon::protocol::InvokeParameter, 422
getVolume
samchon::example::packer::Instance, 393
getWeight
samchon::example::packer::Instance, 393
getYear
samchon::library::Date, 151
getvalueAsXML
samchon::protocol::InvokeParameter, 422
goJoin
samchon::protocol::service::User, 666
goLogin
samchon::protocol::service::User, 666
goLogout
samchon::protocol::service::User, 667

has
samchon::Map, 449
samchon::protocol::EntityGroup, 269
hasListener
samchon::protocol::SystemRole, 636
hasProperty
samchon::library::XML, 707
hasReplyListener
samchon::protocol::ExternalSystemRole, 323
hasRole
samchon::protocol::ExternalSystemArray, 319
hasSendListener
samchon::protocol::ExternalSystemRole, 323
hdbc
samchon::library::SQLi, 617
historyArray
samchon::protocol::master::PRMasterHistory, 565
samchon::protocol::master::ParallelSystem, 542
samchon::protocol::master::ParallelSystemArray, 547
host
samchon::example::chat_service::ChatRoom, 108
hstmt
samchon::library::SQLStatement, 623

IClient
samchon::protocol::IClient, 375
IEntityChain
samchon::protocol::IEntityChain, 379
IEntityGroup
samchon::protocol::IEntityGroup, 380
IHTMLEntity
samchon::protocol::IHTMLEntity, 386
INTExplore
samchon::namtree::INTExplore, 397
IOperator
samchon::library::IOperator, 425
IPUserPair
samchon::protocol::service::IPUserPair, 430
IProtocol
samchon::protocol::IProtocol, 427
ISQLEntity
samchon::protocol::ISQLEntity, 438
IServer
samchon::protocol::IServer, 435
IWebClient
samchon::protocol::IWebClient, 441
IWebServer
samchon::protocol::IWebServer, 445
id
samchon::protocol::service::User, 670
inProgress
samchon::protocol::master::DistributedSystem, 219
index
samchon::protocol::master::PRInvokeHistory, 560
initialValue
samchon::namtree::NTParameter, 484
Instance
samchon::example::packer::Instance, 392
instanceMap
samchon::library::FTFactory, 336
Invoke
samchon::protocol::Invoke, 403
InvokeHistory
samchon::protocol::InvokeHistory, 409
InvokeHistoryArray
samchon::protocol::InvokeHistoryArray, 415
invokeHistoryList
samchon::protocol::master::DistributedSystem, 220
samchon::protocol::master::DistributedSystemRole, 237
InvokeParameter
samchon::protocol::InvokeParameter, 419, 421
ioService
samchon::protocol::ServerConnector, 587
ip
samchon::protocol::ExternalSystem, 316
samchon::protocol::service::IPUserPair, 431
ipMap
samchon::protocol::service::Server, 583
isLocked
samchon::library::SharedAcquire, 596
samchon::library::SharedReadLock, 600
samchon::library::SharedWriteLock, 603
samchon::library::UniqueAcquire, 648
samchon::library::UniqueReadLock, 653
samchon::library::UniqueWriteLock, 657
isNumeric
samchon::library::StringUtil, 627
issueSessionID
samchon::protocol::service::IPUserPair, 431

key
samchon::library::FTInstance, 345
samchon::namtree::NTParameter, 482
samchon::namtree::NTParameterDetermined, 490
samchon::protocol::Entity, 249
samchon::protocol::ExternalSystem, 313

samchon::protocol::ExternalSystemRole, 323
samchon::protocol::InvokeHistory, 410
samchon::protocol::InvokeParameter, 421
samchon::protocol::SystemRole, 635

LN10
samchon::library::Math, 458
LN2
samchon::library::Math, 458
LOG10E
samchon::library::Math, 459
LOG2E
samchon::library::Math, 459
label
samchon::namtree::NTParameterDetermined, 491
latitude
samchon::example::tsp::GeometryPoint, 372
ListService
samchon::example::chat_service::ListService, 447
listen
samchon::protocol::IClient, 375
samchon::protocol::IWebClient, 441
listener
samchon::example::chat_service::ChatMessage, 103
samchon::protocol::Invoke, 406
samchon::protocol::InvokeHistory, 413
listeners
samchon::protocol::SystemRole, 637
load
samchon::protocol::ISQLEntity, 438
localEndPoint
samchon::protocol::ServerConnector, 587
lock
samchon::library::SharedReadLock, 599
samchon::library::SharedWriteLock, 602
samchon::library::UniqueReadLock, 651
samchon::library::UniqueWriteLock, 656
longitude
samchon::example::tsp::GeometryPoint, 372
ltrim
samchon::WeakString, 685
samchon::library::StringUtil, 629

MY_IP
samchon::protocol::ExternalClientArray, 299
samchon::protocol::IServer, 436
master
samchon::protocol::master::DistributedSlaveSystemMediator, 214
samchon::protocol::master::PRMasterHistory, 565
samchon::protocol::master::PRMasterHistoryArray, 566
samchon::protocol::master::ParallelSlaveSystemMediator, 534
masterHistory
samchon::protocol::master::PRInvokeHistory, 560
maximum
samchon::library::Math, 454
samchon::namtree::INTExplore, 399
mean
samchon::library::Math, 455
median
samchon::library::Math, 455
member
samchon::library::FTFactory, 336
message
samchon::example::chat_service::ChatMessage, 103
samchon::library::ErrorEvent, 284
minimum
samchon::library::Math, 454
samchon::namtree::INTExplore, 399
mode
samchon::library::Math, 456
moveValue
samchon::protocol::InvokeParameter, 422
mtx
samchon::SmartPointer, 613
samchon::library::CriticalAllocator, 137
samchon::library::EventDispatcher, 292
samchon::library::Semaphore, 579
samchon::library::UniqueReadLock, 653
samchon::library::UniqueWriteLock, 657
samchon::protocol::service::User, 670
mutate
samchon::library::GeneticAlgorithm, 367
mutationRate
samchon::library::GAParameters, 351
samchon::library::GeneticAlgorithm, 368
myIP
samchon::protocol::ExternalClientArray, 301
samchon::protocol::ExternalServer, 306
MySQLi
samchon::library::MySQLi, 461

n
samchon::library::CaseGenerator, 90
n_
samchon::library::CaseGenerator, 91
NAME
samchon::protocol::service::Server, 581
NTEntityGroup
samchon::namtree::NTEntityGroup, 468
NTFactory
samchon::namtree::NTFactory, 471
NTFile
samchon::namtree::NTFile, 474
NTIterator
samchon::namtree::NTIterator, 478
NTParameter
samchon::namtree::NTParameter, 481
NTParameterArray
samchon::namtree::NTParameterArray, 486
NTParameterDetermined
samchon::namtree::NTParameterDetermined, 490
name
samchon::example::chat_service::ChatRoom, 108
samchon::example::packer::Instance, 394
samchon::library::FTInstance, 347
samchon::namtree::NTParameter, 484
samchon::protocol::ExternalSystem, 316
samchon::protocol::ExternalSystemRole, 327

samchon::protocol::InvokeParameter, 423
samchon::protocol::SystemRole, 637
samchon::protocol::service::Service, 590
next
samchon::library::SQLStatement, 622
no
samchon::protocol::service::Client, 129
notify
samchon::example::chat_service::ChatRoomArray, 11
notifyEnd
samchon::protocol::InvokeHistory, 409
samchon::protocol::master::PRInvokeHistory, 558
samchon::protocol::master::PRMasterHistory, 564
samchon::protocol::master::ParallelSystemArray, 546
samchon::protocol::master::ParallelSystemArrayMediator, 552
npos
samchon::WeakString, 688
numberFormat
samchon::library::StringUtil, 628
numerator
samchon::library::ProgressEvent, 569

open
samchon::protocol::IServer, 436
samchon::protocol::IWebServer, 445
openServer
samchon::protocol::FlashPolicyServer, 330
operator<
samchon::example::tsp::Travel, 639
samchon::library::IOperator, 425
samchon::namtree::NTIterator, 479
operator*
samchon::SmartPointer, 613
operator++
samchon::namtree::NTIterator, 479
operator->
samchon::SmartPointer, 613
operator--
samchon::namtree::NTIterator, 479
operator==
samchon::library::IOperator, 425
samchon::namtree::NTIterator, 479
operator[]
samchon::WeakString, 678
samchon::library::CaseGenerator, 90
samchon::library::CombinedPermutationGenerator, 132
samchon::library::PermutationGenerator, 556
optimize
samchon::example::packer::Packer, 497
samchon::example::packer::WrapperArray, 697
samchon::example::tsp::Scheduler, 574
orator
samchon::example::chat_service::ChatMessage, 103
otherside
samchon::namtree::NTFile, 476

PI
samchon::library::Math, 458
PORT
samchon::example::chat_service::ChatServer, 118
samchon::example::console_chat_server::ChatServer, 11
samchon::protocol::ExternalClientArray, 299
samchon::protocol::IServer, 436
PRInvokeHistory
samchon::protocol::master::PRInvokeHistory, 558
PRInvokeHistoryArray
samchon::protocol::master::PRInvokeHistoryArray, 562
PRMasterHistory
samchon::protocol::master::PRMasterHistory, 564
PRMasterHistoryArray
samchon::protocol::master::PRMasterHistoryArray, 566
Packer
samchon::example::packer::Packer, 497
ParallelClient
samchon::protocol::master::ParallelClient, 502
samchon::protocol::slave::ParallelClient, 501
ParallelClientArray
samchon::protocol::master::ParallelClientArray, 504
ParallelClientArrayMediator
samchon::protocol::master::ParallelClientArrayMediator, 5
ParallelServer
samchon::protocol::master::ParallelServer, 512
samchon::protocol::slave::ParallelServer, 515
ParallelServerArray
samchon::protocol::master::ParallelServerArray, 517
ParallelServerArrayMediator
samchon::protocol::master::ParallelServerArrayMediator,
ParallelSlaveClientMediator
samchon::protocol::master::ParallelSlaveClientMediator, 5
ParallelSlaveServerMediator
samchon::protocol::master::ParallelSlaveServerMediator,
ParallelSlaveSystemMediator
samchon::protocol::master::ParallelSlaveSystemMediator
ParallelSystem
samchon::protocol::master::ParallelSystem, 539
samchon::protocol::slave::ParallelSystem, 537
ParallelSystemArray
samchon::protocol::master::ParallelSystemArray, 545
ParallelSystemArrayMediator
samchon::protocol::master::ParallelSystemArrayMediator
parameterArray
samchon::namtree::NTFile, 476
parent
samchon::library::FTInstance, 347
participants
samchon::example::chat_service::ChatRoom, 108
percentFormat
samchon::library::StringUtil, 628
performance
samchon::protocol::master::DistributedSystem, 220
samchon::protocol::master::DistributedSystemRole, 236
samchon::protocol::master::ParallelSystem, 542
PermutationGenerator
samchon::library::PermutationGenerator, 555
policy
samchon::protocol::FlashPolicyServer, 331
pop

samchon::Map, 451
population
samchon::library::GAParameters, 351
port
samchon::library::SQLi, 617
samchon::protocol::ExternalClientArray, 301
samchon::protocol::ExternalSystem, 316
position
samchon::ByteArray, 80
precision
samchon::namtree::INTExplore, 399
prepare
samchon::library::SQLStatement, 621
price
samchon::example::packer::Instance, 394
processed
samchon::protocol::master::DistributedSystem, 219
Product
samchon::example::packer::Product, 567
productArray
samchon::example::packer::Packer, 498
progressArray
samchon::protocol::master::ParallelSystem, 542
samchon::protocol::master::ParallelSystemArray, 547
ProgressEvent
samchon::library::ProgressEvent, 569
propertyMap
samchon::library::XML, 708
ptr
samchon::SmartPointer, 613
push_back
samchon::library::XML, 704, 705

r
samchon::library::CaseGenerator, 90
r_
samchon::library::CaseGenerator, 91
REQUIRE_AUTHORITY
samchon::protocol::service::Service, 589
RWMutex
samchon::library::RWMutex, 571
radian_to_degree
samchon::library::Math, 454
random
samchon::library::Math, 453
read
samchon::ByteArray, 77
readLock
samchon::library::RWMutex, 571
readReversely
samchon::ByteArray, 78
readUnlock
samchon::library::RWMutex, 572
referValue
samchon::protocol::InvokeParameter, 422
reference
samchon::library::SharedAcquire, 596
samchon::library::SharedReadLock, 599
samchon::library::SharedWriteLock, 603
refresh
samchon::library::SQLStatement, 621
registerClient
samchon::example::chat_service::ChatRoom, 106
registerInstance
samchon::library::FTFactory, 335
release
samchon::library::Semaphore, 578
samchon::library::SharedAcquire, 595
samchon::library::UniqueAcquire, 648
removeEventListener
samchon::library::EventDispatcher, 291
replace
samchon::WeakString, 686
replaceAll
samchon::WeakString, 686
samchon::library::StringUtil, 634
replyData
samchon::example::console_chat_server::ChatClient, 99
samchon::example::console_chat_server::ChatServer, 11
samchon::protocol::ExternalSystem, 314
samchon::protocol::ExternalSystemArray, 320
samchon::protocol::IProtocol, 427
samchon::protocol::master::DistributedSlaveSystemMedia
samchon::protocol::master::DistributedSystem, 218
samchon::protocol::service::Client, 126
samchon::protocol::service::User, 669
replyListeners
samchon::protocol::ExternalSystemRole, 327
replyPieceData
samchon::protocol::master::ParallelSlaveSystemMediator
samchon::protocol::slave::ParallelSystem, 537
reserved
samchon::example::packer::WrapperArray, 698
reset
samchon::SmartPointer, 612
reverse
samchon::ByteArray, 77
rfind
samchon::WeakString, 679
rfinds
samchon::WeakString, 680
samchon::library::StringUtil, 630
role
samchon::protocol::master::DSInvokeHistory, 241
roleDictionary
samchon::protocol::master::DistributedSystemArray, 226
roleHistoryList
samchon::protocol::master::DistributedSystem, 220
room
samchon::example::chat_service::ChatService, 120
roomArray
samchon::example::chat_service::ChatRoom, 108
samchon::example::chat_service::ChatServer, 119
rtrim
samchon::WeakString, 685
samchon::library::StringUtil, 629
rwMutex

samchon::example::console_chat_server::ChatServer, 116
SPACE_ARRAY
samchon::WeakString, 688
SQLStatement
samchon::library::SQLStatement, 620
SQLi
samchon::library::SQLi, 616
SQRT1_2
samchon::library::Math, 459
SQRT2
samchon::library::Math, 459
samchon, 23
Dictionary, 24
samchon::ByteArray, 75
ByteArray, 77
compress, 79
decompress, 79
getPosition, 77
position, 80
read, 77
readReversely, 78
reverse, 77
setPosition, 77
write, 78
writeReversely, 78
samchon::IndexPair
getIndex, 390
getValue, 390, 391
samchon::IndexPair< _Ty >, 389
samchon::Map
get, 449
has, 449
pop, 451
set, 450
samchon::Map< _Kty, _Ty, _Pr, _Alloc >, 448
samchon::SmartPointer
~SmartPointer, 612
get, 612
mtx, 613
operator*, 613
operator->, 613
ptr, 613
reset, 612
SmartPointer, 611
useCountMap, 613
samchon::SmartPointer< _Ty >, 610
samchon::WeakString, 672
at, 678
between, 682
betweens, 684
data, 676
data_, 689
empty, 677
find, 678
finds, 679
ltrim, 685
npos, 688
operator[], 678
replace, 686
replaceAll, 686
rfind, 679
rfinds, 680
rtrim, 685
SPACE_ARRAY, 688
size, 677
size_, 689
split, 683
str, 688
substr, 681
substring, 681
toLowerCase, 687
toUpperCase, 688
trim, 685
WeakString, 674–676
samchon::example::chat_service, 25
samchon::example::chat_service::ChatClient, 95
ChatClient, 96
createService, 96
samchon::example::chat_service::ChatMessage, 99
ChatMessage, 101
construct, 101
getListener, 102
listener, 103
message, 103
orator, 103
TAG, 101
toXML, 102
samchon::example::chat_service::ChatRoom, 103
ChatRoom, 105
eraseClient, 106
host, 108
name, 108
participants, 108
registerClient, 106
roomArray, 108
TAG, 105
toXML, 106
samchon::example::chat_service::ChatRoomArray, 109
CHILD_TAG, 110
ChatRoomArray, 110
notify, 111
server, 112
toInvoke, 112
toXML, 111
samchon::example::chat_service::ChatServer, 117
ChatServer, 118
createUser, 118
getChatRoomArray, 119
PORT, 118
roomArray, 119
samchon::example::chat_service::ChatService, 119
ChatService, 120
room, 120
samchon::example::chat_service::ChatUser, 121
ChatUser, 122
createClient, 122

samchon::example::chat_service::ListService, 446
createRoom, 447
ListService, 447
samchon::example::console_chat_server, 26
samchon::example::console_chat_server::ChatClient, 97
~ChatClient, 98
ChatClient, 98
replyData, 99
server, 99
samchon::example::console_chat_server::ChatServer, 113
addClient, 115
ChatServer, 114
clientSet, 116
eraseClient, 114
PORT, 114
replyData, 115
rwMutex, 116
sendData, 116
samchon::example::packer::Instance, 391
getName, 393
getPrice, 393
getVolume, 393
getWeight, 393
Instance, 392
name, 394
price, 394
toString, 394
volume, 394
weight, 394
samchon::example::packer::Packer, 495
calcPrice, 498
optimize, 497
Packer, 497
productArray, 498
toString, 498
wrapperArray, 498
samchon::example::packer::Product, 566
Product, 567
toString, 567
samchon::example::packer::Wrapper, 691
toString, 694
tryInsert, 693
Wrapper, 693
samchon::example::packer::WrapperArray, 694
calcPrice, 697
optimize, 697
reserved, 698
sample, 698
toString, 698
tryInsert, 696
WrapperArray, 696
samchon::example::tsp, 27
samchon::example::tsp::GAParameters, 350
samchon::example::tsp::GeometryPoint, 369
calcDistance, 371
GeometryPoint, 370
latitude, 372
longitude, 372
toString, 372
uid, 372
samchon::example::tsp::Scheduler, 573
gaParameters, 575
optimize, 574
Scheduler, 574
travel, 575
samchon::example::tsp::Travel, 637
calcDistance, 639
distance, 640
operator<, 639
toString, 639
Travel, 638
samchon::library, 30
CriticalDictionary, 37
CriticalList, 37
CriticalMap, 38
CriticalSet, 38
CriticalVector, 38
XMLList, 38
samchon::library::Base64, 73
decode, 74
encode, 74
samchon::library::CaseGenerator, 80
at, 90
CaseGenerator, 89
n, 90
n_, 91
operator[], 90
r, 90
r_, 91
size, 89
size_, 91
toMatrix, 91
samchon::library::Charset, 92
toMultibyte, 93, 94
toUTF8, 94
toUnicode, 95
samchon::library::CombinedPermutationGenerator, 130
CombinedPermutationGenerator, 131
operator[], 132
samchon::library::CriticalAllocator
allocate, 135
construct, 134
deallocate, 136
destroy, 135
getMutex, 134
mtx, 137
samchon::library::CriticalAllocator< _Ty >, 132
samchon::library::Date, 137
addDate, 151
addMonth, 149
addWeek, 150
addYear, 148
calcLastDates, 142
calcSeconds, 141
Date, 139–141
getDate, 154

getDay, 155
getMonth, 153
getYear, 151
set, 142–144
setDate, 148
setMonth, 147
setYear, 146
TP_1970, 159
toLinuxTime, 157
toString, 158
toTM, 156
samchon::library::Datetime, 159
addDate, 174
addHour, 175
addMinute, 175
addMonth, 172
addSecond, 176
addWeek, 173
addYear, 172
Datetime, 162, 163
getHour, 176
getMinute, 178
getSecond, 179
set, 164, 166, 167
setDate, 169
setHour, 170
setMinute, 171
setMonth, 168
setSecond, 171
setYear, 167
toString, 180
samchon::library::ErrorEvent, 281
ErrorEvent, 283
getMessage, 283
message, 284
samchon::library::Event, 284
Event, 286
getSource, 287
getType, 287
source, 287
type, 287
samchon::library::EventDispatcher, 287
addEventListener, 291
dispatchEvent, 291
dispatchProgressEvent, 292
EventDispatcher, 290, 291
eventSetMap, 292
mtx, 292
removeEventListener, 291
semaphore, 292
samchon::library::FTByteFile, 331
data, 334
FTByteFile, 333
getData, 333
samchon::library::FTFactory, 334
application, 336
category, 336
createFile, 335
FTFactory, 335
instanceMap, 336
member, 336
registerInstance, 335
samchon::library::FTFile, 337
extension, 339
FTFile, 338
getExtension, 338
toXML, 338
samchon::library::FTFolder, 340
CHILD_TAG, 341
FTFolder, 341
factory, 342
toXML, 342
samchon::library::FTInstance, 343
comment, 347
FTInstance, 344
getComment, 346
getName, 346
getParent, 345
getUID, 345
key, 345
name, 347
parent, 347
TAG, 344
toXML, 346
uid, 347
samchon::library::FTTextFile, 348
data, 350
FTTextFile, 349
getData, 349
samchon::library::FactorialGenerator, 328
FactorialGenerator, 329
samchon::library::GAParameters, 351
generation, 351
mutationRate, 351
population, 351
tournament, 351
samchon::library::GAPopulation
children, 355
fitTest, 354
GAPopulation, 354
samchon::library::GAPopulation< GeneArray, Compare >, 35
samchon::library::GeneticAlgorithm
crossover, 366
evolveGeneArray, 363
evolvePopulation, 363
GeneticAlgorithm, 362
mutate, 367
mutationRate, 368
selection, 365
tournament, 368
unique, 368
samchon::library::GeneticAlgorithm< GeneArray, Compare >,
samchon::library::HTTPLoader, 372
samchon::library::IOperator
IOperator, 425
operator<, 425

operator==, 425
samchon::library::IOperator< _Ty >, 424
samchon::library::Math, 451
degree_to_radian, 454
E, 458
LN10, 458
LN2, 458
LOG10E, 459
LOG2E, 459
maximum, 454
mean, 455
median, 455
minimum, 454
mode, 456
PI, 458
radian_to_degree, 454
random, 453
SQRT1_2, 459
SQRT2, 459
stdev_p, 456
stdev_s, 456
variance_p, 457
variance_s, 457
samchon::library::MySQLi, 459
MySQLi, 461
samchon::library::PermutationGenerator, 553
operator[], 556
PermutationGenerator, 555
samchon::library::ProgressEvent, 568
denominator, 569
getDenominator, 569
getNumerator, 569
getPercent, 569
numerator, 569
ProgressEvent, 569
samchon::library::RWMutex, 570
RWMutex, 571
readLock, 571
readUnlock, 572
writeLock, 572
writeUnlock, 573
samchon::library::SQLStatement, 618
at, 622
bindParameterBASizeMap, 623
bindParameterCount, 623
execute, 621
executeDirectly, 621
fetch, 622
free, 621
get, 623
hstmt, 623
next, 622
prepare, 621
refresh, 621
SQLStatement, 620
size, 622
sqli, 623
toXML, 623
samchon::library::SQLi, 614
connect, 617
createStatement, 617
disconnect, 617
driver, 617
hdbc, 617
port, 617
SQLi, 616
stmt, 618
stmtMutex, 618
samchon::library::Semaphore, 575
acquire, 577
acquired, 579
acquiredSize, 577
mtx, 579
release, 578
Semaphore, 577
setSize, 577
size, 577
size_, 579
tryAcquire, 578
samchon::library::SharedAcquire, 593
~SharedAcquire, 595
acquire, 595
isLocked, 596
reference, 596
release, 595
semaphore, 596
SharedAcquire, 595
samchon::library::SharedReadLock, 596
~SharedReadLock, 599
isLocked, 600
lock, 599
reference, 599
semaphore, 599
SharedReadLock, 598, 599
unlock, 599
samchon::library::SharedWriteLock, 600
~SharedWriteLock, 602
isLocked, 603
lock, 602
reference, 603
semaphore, 603
SharedWriteLock, 602
unlock, 603
samchon::library::StringUtil, 624
addTab, 632
between, 631
betweens, 632
colorNumberFormat, 628
colorPercentFormat, 629
finds, 630
isNumeric, 627
ltrim, 629
numberFormat, 628
percentFormat, 628
replaceAll, 634
rfinds, 630

rtrim, 629
split, 632
substitute, 625
substituteSQL, 627
substring, 631
toLowerCase, 633
toNumber, 628
toUpperCase, 633
trim, 629
samchon::library::TSQLStatement, 642
toXML, 644
samchon::library::TSQLi, 640
createStatement, 642
TSQLi, 642
samchon::library::URLVariables, 657
decode, 659
encode, 659
toString, 660
URLVariables, 658
samchon::library::UniqueAcquire, 644
~UniqueAcquire, 647
acquire, 647
isLocked, 648
release, 648
semaphore, 648
UniqueAcquire, 646, 647
samchon::library::UniqueReadLock, 649
~UniqueReadLock, 651
isLocked, 653
lock, 651
mtx, 653
UniqueReadLock, 650, 651
unlock, 652
samchon::library::UniqueWriteLock, 653
~UniqueWriteLock, 655
isLocked, 657
lock, 656
mtx, 657
UniqueWriteLock, 655
unlock, 656
samchon::library::XML, 699
addAllProperty, 705
clearProperties, 706
eraseProperty, 706
getProperty, 707
getPropertyMap, 707
getTag, 706
getValue, 707
hasProperty, 707
propertyMap, 708
push_back, 704, 705
setProperty, 706
setTag, 705
setValue, 705
tag, 708
toString, 707
value, 708
XML, 702–704
samchon::namtree, 39
samchon::namtree::INTExplore, 395
construct, 397
getMaximum, 397
getMinimum, 397
getPrecision, 398
getSection, 398
INTExplore, 397
maximum, 399
minimum, 399
precision, 399
section, 399
toXML, 398
samchon::namtree::NTCriteria, 462
CHILD_TAG, 466
construct, 465
createChild, 465
TAG, 465
toXML, 466
samchon::namtree::NTEntityGroup, 467
begin, 468
end, 468
NTEntityGroup, 468
samchon::namtree::NTFactory, 468
createCriteria, 471
createSide, 471
data, 472
NTFactory, 471
sideFunctionMap, 472
samchon::namtree::NTFile, 472
construct, 475
factory, 476
function, 477
getFunction, 475
getOtherside, 475
getParameterArray, 475
NTFile, 474
otherside, 476
parameterArray, 476
toXML, 476
samchon::namtree::NTIterator, 477
data, 479
NTIterator, 478
operator<, 479
operator++, 479
operator--, 479
operator==, 479
samchon::namtree::NTParameter, 480
CHILD_TAG, 483
construct, 481
createChild, 482
getInitialValue, 482
getName, 482
initialValue, 484
key, 482
NTParameter, 481
name, 484
TAG, 483

toXML, 483
samchon::namtree::NTParameterArray, 484
CHILD_TAG, 487
createChild, 487
NTParameterArray, 486
TAG, 486
samchon::namtree::NTParameterDetermined, 487
construct, 490
getLabel, 490
getValue, 490
key, 490
label, 491
NTParameterDetermined, 490
TAG, 490
toXML, 491
value, 491
samchon::namtree::NTSide, 492
construct, 494
TAG, 494
toXML, 494
samchon::protocol, 43
SharedEntityArray, 54
SharedEntityList, 55
UniqueEntityArray, 55
UniqueEntityList, 56
samchon::protocol::Entity, 243
construct, 248
Entity, 247
key, 249
TAG, 248
toXML, 249
samchon::protocol::EntityArray
construct, 257
EntityArray, 257
toXML, 257
samchon::protocol::EntityArray< _Ty >, 251
samchon::protocol::EntityGroup
construct, 267
createChild, 268
EntityGroup, 266
get, 269, 271
has, 269
toXML, 271
samchon::protocol::EntityGroup< _Container, _ETy, _Ty >, 259
samchon::protocol::EntityList
construct, 279
EntityList, 279
toXML, 280
samchon::protocol::EntityList< _Ty >, 274
samchon::protocol::ExternalClient, 293
ExternalClient, 294
start, 295
samchon::protocol::ExternalClientArray, 295
addClient, 299
construct, 298
ExternalClientArray, 297
MY_IP, 299
myIP, 301
PORT, 299
port, 301
start, 298
toXML, 300
samchon::protocol::ExternalServer, 302
construct, 304
ExternalServer, 303
getIP, 305
getMyIP, 305
getPort, 305
myIP, 306
start, 304
toXML, 305
samchon::protocol::ExternalServerArray, 306
start, 308
samchon::protocol::ExternalSystem, 309
CHILD_TAG, 314
construct, 312
ExternalSystem, 311
ip, 316
key, 313
name, 316
port, 316
replyData, 314
start, 313
TAG, 314
toXML, 314
samchon::protocol::ExternalSystemArray, 317
CHILD_TAG, 320
ExternalSystemArray, 318
getRole, 319
hasRole, 319
replyData, 320
sendData, 319
start, 318
TAG, 320
samchon::protocol::ExternalSystemRole, 321
construct, 323
ExternalSystemRole, 322
getSystem, 323
hasReplyListener, 323
hasSendListener, 323
key, 323
name, 327
replyListeners, 327
sendData, 323
sendListeners, 327
system, 327
TAG, 325
toXML, 325
samchon::protocol::FlashPolicyServer, 329
accept, 331
FlashPolicyServer, 330
openServer, 330
policy, 331
samchon::protocol::IClient, 374
_replyData, 376
BUFFER_SIZE, 375

IClient, 375
listen, 375
sendData, 376
sendMtx, 377
socket, 377
samchon::protocol::IEntityChain, 377
entity, 379
IEntityChain, 379
samchon::protocol::IEntityGroup, 380
CHILD_TAG, 381
IEntityGroup, 380
samchon::protocol::IEntityPtrGroup< _Container >, 382
samchon::protocol::IHTMLEntity, 382
IHTMLEntity, 386
toHTML, 389
toTH, 387
toTR, 388
samchon::protocol::IProtocol, 426
IProtocol, 427
replyData, 427
sendData, 428
samchon::protocol::ISQLEntity, 437
archive, 439
ISQLEntity, 438
load, 438
toSQL, 439
samchon::protocol::IServer, 432
acceptor, 437
addClient, 436
close, 436
IServer, 435
MY_IP, 436
open, 436
PORT, 436
samchon::protocol::IWebClient, 439
IWebClient, 441
listen, 441
sendData, 441
samchon::protocol::IWebServer, 442
IWebServer, 445
open, 445
samchon::protocol::Invoke, 399
CHILD_TAG, 405
construct, 403
createChild, 404
getListener, 404
Invoke, 403
listener, 406
setListener, 404
TAG, 405
toSQL, 406
toXML, 405
samchon::protocol::InvokeHistory, 406
calcElapsedTime, 411
construct, 409
endTime, 414
getEndTime, 410
getListener, 410
getStartTime, 410
getUID, 410
InvokeHistory, 409
key, 410
listener, 413
notifyEnd, 409
startTime, 413
TAG, 411
toInvoke, 412
toXML, 411
uid, 413
samchon::protocol::InvokeHistoryArray, 414
CHILD_TAG, 416
createChild, 415
InvokeHistoryArray, 415
TAG, 415
samchon::protocol::InvokeParameter, 416
byteArray, 424
construct, 421
getName, 421
getType, 421
getValue, 422
getvalueAsXML, 422
InvokeParameter, 419, 421
key, 421
moveValue, 422
name, 423
referValue, 422
str, 424
TAG, 422
toSQL, 423
toXML, 422
type, 423
xml, 424
samchon::protocol::ServerConnector, 584
connect, 586
endPoint, 587
getIP, 585
getMyIP, 586
getPort, 586
ioService, 587
localEndPoint, 587
ServerConnector, 585
samchon::protocol::SystemRole, 634
construct, 635
hasListener, 636
key, 635
listeners, 637
name, 637
SystemRole, 635
TAG, 636
toXML, 636
samchon::protocol::WebServerConnector, 689
connect, 691
WebServerConnector, 691
samchon::protocol::master, 56
samchon::protocol::master::DSInvokeHistory, 237
DSInvokeHistory, 240

getRole, 240
getSystem, 240
role, 241
system, 241
toXML, 240
samchon::protocol::master::DSInvokeHistoryList, 241
createChild, 242
DSInvokeHistoryList, 242
samchon::protocol::master::DistributedClient, 181
DistributedClient, 184
samchon::protocol::master::DistributedClientArray, 184
construct, 186
DistributedClientArray, 186
start, 187
toXML, 187
samchon::protocol::master::DistributedClientArrayMediator, 188
construct, 191
DistributedClientArrayMediator, 191
start, 192
toXML, 192
samchon::protocol::master::DistributedClientServerMediator, 196
DistributedClientServerMediator, 196
samchon::protocol::master::DistributedServer, 196
construct, 199
DistributedServer, 198
toXML, 199
samchon::protocol::master::DistributedServerArray, 200
DistributedServerArray, 202
start, 202
samchon::protocol::master::DistributedServerArrayMediator, 203
DistributedServerArrayMediator, 206
start, 206
samchon::protocol::master::DistributedSlaveServerMediator, 207
DistributedSlaveServerMediator, 210
samchon::protocol::master::DistributedSlaveSystemMediator, 210
DistributedSlaveSystemMediator, 213
master, 214
replyData, 214
TAG, 214
samchon::protocol::master::DistributedSystem, 214
avgElapsedTime, 219
construct, 218
DistributedSystem, 217
inProgress, 219
invokeHistoryList, 220
performance, 220
processed, 219
replyData, 218
roleHistoryList, 220
sendData, 218
toXML, 218
samchon::protocol::master::DistributedSystemArray, 220
allocateRoles, 225
construct, 223
createRole, 225
DistributedSystemArray, 223
gaParameters, 226
getRole, 225
roleDictionary, 226
start, 225
toXML, 225
samchon::protocol::master::DistributedSystemArrayMediator, 2
construct, 230
createSlave, 232
DistributedSystemArrayMediator, 230
slave, 233
start, 232
toXML, 232
samchon::protocol::master::DistributedSystemRole, 233
allocatedSystems, 236
allocationHistoryList, 237
DistributedSystemRole, 235
getAllocationHistoryList, 236
getInvokeHistoryList, 236
getPerformance, 235
getSystem, 235
invokeHistoryList, 237
performance, 236
sendData, 236
toXML, 236
samchon::protocol::master::PRInvokeHistory, 556
calcAverageElapsedTime, 559
construct, 558
getIndex, 559
getSize, 559
index, 560
masterHistory, 560
notifyEnd, 558
PRInvokeHistory, 558
size, 561
system, 560
toXML, 560
samchon::protocol::master::PRInvokeHistoryArray, 561
createChild, 562
PRInvokeHistoryArray, 562
samchon::protocol::master::PRMasterHistory, 562
completed, 565
historyArray, 565
master, 565
notifyEnd, 564
PRMasterHistory, 564
samchon::protocol::master::PRMasterHistoryArray, 565
master, 566
PRMasterHistoryArray, 566
samchon::protocol::master::ParallelClient, 501
ParallelClient, 502
samchon::protocol::master::ParallelClientArray, 502
ParallelClientArray, 504
samchon::protocol::master::ParallelClientArrayMediator, 505
construct, 508
ParallelClientArrayMediator, 508
start, 509
toXML, 510
samchon::protocol::master::ParallelServer, 511
construct, 512
ParallelServer, 512

toXML, 513
samchon::protocol::master::ParallelServerArray, 515
ParallelServerArray, 517
samchon::protocol::master::ParallelServerArrayMediator, 517
ParallelServerArrayMediator, 521
start, 521
samchon::protocol::master::ParallelSlaveClientMediator, 522
ParallelSlaveClientMediator, 525
samchon::protocol::master::ParallelSlaveServerMediator, 526
ParallelSlaveServerMediator, 529
samchon::protocol::master::ParallelSlaveSystemMediator, 529
master, 534
ParallelSlaveSystemMediator, 532
replyPieceData, 533
TAG, 534
samchon::protocol::master::ParallelSystem, 538
_replyData, 540
construct, 540
createChild, 540
historyArray, 542
ParallelSystem, 539
performance, 542
progressArray, 542
sendPieceData, 541
systemArray, 542
toXML, 541
samchon::protocol::master::ParallelSystemArray, 543
historyArray, 547
notifyEnd, 546
ParallelSystemArray, 545
progressArray, 547
sendPieceData, 545
sendSegmentData, 545
uid, 547
samchon::protocol::master::ParallelSystemArrayMediator, 547
construct, 550
notifyEnd, 552
ParallelSystemArrayMediator, 550
slave, 553
start, 551
toXML, 552
samchon::protocol::service, 60
samchon::protocol::service::Client, 123
Client, 124
constructService, 128
createService, 125
getNo, 125
getService, 125
getUser, 125
no, 129
replyData, 126
sendData, 125
service, 129
user, 129
samchon::protocol::service::IPUserPair, 428
getSessionID, 430
IPUserPair, 430
ip, 431
issueSessionID, 431
server, 431
userSet, 432
samchon::protocol::service::Server, 579
addClient, 583
begin, 581
createUser, 582
end, 582
ipMap, 583
NAME, 581
sequence, 583
Server, 581
size, 581
sqli, 583
samchon::protocol::service::Service, 588
client, 590
getClient, 590
name, 590
REQUIRE_AUTHORITY, 589
sendData, 590
Service, 589
samchon::protocol::service::ServiceKeeper, 591
client, 593
ServiceKeeper, 592
user, 593
samchon::protocol::service::User, 660
addClient, 664
authority, 671
createClient, 664
doJoin, 668
doLogin, 668
eraseClient, 665
getAuthority, 663
getID, 663
getSemaphore, 663
getServer, 663
goJoin, 666
goLogin, 666
goLogout, 667
id, 670
mtx, 670
replyData, 669
semaphore, 670
sendData, 668
server, 670
sessionID, 670
User, 662
samchon::protocol::slave, 70
samchon::protocol::slave::ParallelClient, 499
ParallelClient, 501
samchon::protocol::slave::ParallelServer, 513
ParallelServer, 515
samchon::protocol::slave::ParallelSystem, 535
_replyData, 537
createChild, 537
ParallelSystem, 537
replyPieceData, 537
samchon::protocol::slave::SlaveClient, 603

SlaveClient, 605
samchon::protocol::slave::SlaveServer, 605
SlaveServer, 607
samchon::protocol::slave::SlaveSystem, 607
_replyData, 609
sendData, 608
SlaveSystem, 608
sample
samchon::example::packer::WrapperArray, 698
Scheduler
samchon::example::tsp::Scheduler, 574
section
samchon::namtree::INTExplore, 399
selection
samchon::library::GeneticAlgorithm, 365
Semaphore
samchon::library::Semaphore, 577
semaphore
samchon::library::EventDispatcher, 292
samchon::library::SharedAcquire, 596
samchon::library::SharedReadLock, 599
samchon::library::SharedWriteLock, 603
samchon::library::UniqueAcquire, 648
samchon::protocol::service::User, 670
sendData
samchon::example::console_chat_server::ChatServer, 116
samchon::protocol::ExternalSystemArray, 319
samchon::protocol::ExternalSystemRole, 323
samchon::protocol::IClient, 376
samchon::protocol::IProtocol, 428
samchon::protocol::IWebClient, 441
samchon::protocol::master::DistributedSystem, 218
samchon::protocol::master::DistributedSystemRole, 236
samchon::protocol::service::Client, 125
samchon::protocol::service::Service, 590
samchon::protocol::service::User, 668
samchon::protocol::slave::SlaveSystem, 608
sendListeners
samchon::protocol::ExternalSystemRole, 327
sendMtx
samchon::protocol::IClient, 377
sendPieceData
samchon::protocol::master::ParallelSystem, 541
samchon::protocol::master::ParallelSystemArray, 545
sendSegmentData
samchon::protocol::master::ParallelSystemArray, 545
sequence
samchon::protocol::service::Server, 583
Server
samchon::protocol::service::Server, 581
server
samchon::example::chat_service::ChatRoomArray, 112
samchon::example::console_chat_server::ChatClient, 99
samchon::protocol::service::IPUserPair, 431
samchon::protocol::service::User, 670
ServerConnector
samchon::protocol::ServerConnector, 585
Service
samchon::protocol::service::Service, 589
service
samchon::protocol::service::Client, 129
ServiceKeeper
samchon::protocol::service::ServiceKeeper, 592
sessionID
samchon::protocol::service::User, 670
set
samchon::Map, 450
samchon::library::Date, 142–144
samchon::library::Datetime, 164, 166, 167
setDate
samchon::library::Date, 148
samchon::library::Datetime, 169
setHour
samchon::library::Datetime, 170
setListener
samchon::protocol::Invoke, 404
setMinute
samchon::library::Datetime, 171
setMonth
samchon::library::Date, 147
samchon::library::Datetime, 168
setPosition
samchon::ByteArray, 77
setProperty
samchon::library::XML, 706
setSecond
samchon::library::Datetime, 171
setSize
samchon::library::Semaphore, 577
setTag
samchon::library::XML, 705
setValue
samchon::library::XML, 705
setYear
samchon::library::Date, 146
samchon::library::Datetime, 167
SharedAcquire
samchon::library::SharedAcquire, 595
SharedEntityArray
samchon::protocol, 54
SharedEntityList
samchon::protocol, 55
SharedReadLock
samchon::library::SharedReadLock, 598, 599
SharedWriteLock
samchon::library::SharedWriteLock, 602
sideFunctionMap
samchon::namtree::NTFactory, 472
size
samchon::WeakString, 677
samchon::library::CaseGenerator, 89
samchon::library::SQLStatement, 622
samchon::library::Semaphore, 577
samchon::protocol::master::PRInvokeHistory, 561
samchon::protocol::service::Server, 581
size_

samchon::WeakString, 689
samchon::library::CaseGenerator, 91
samchon::library::Semaphore, 579
slave
samchon::protocol::master::DistributedSystemArrayMediator, 233
samchon::protocol::master::ParallelSystemArrayMediator, 553
SlaveClient
samchon::protocol::slave::SlaveClient, 605
SlaveServer
samchon::protocol::slave::SlaveServer, 607
SlaveSystem
samchon::protocol::slave::SlaveSystem, 608
SmartPointer
samchon::SmartPointer, 611
socket
samchon::protocol::IClient, 377
source
samchon::library::Event, 287
split
samchon::WeakString, 683
samchon::library::StringUtil, 632
sqli
samchon::library::SQLStatement, 623
samchon::protocol::service::Server, 583
start
samchon::protocol::ExternalClient, 295
samchon::protocol::ExternalClientArray, 298
samchon::protocol::ExternalServer, 304
samchon::protocol::ExternalServerArray, 308
samchon::protocol::ExternalSystem, 313
samchon::protocol::ExternalSystemArray, 318
samchon::protocol::master::DistributedClientArray, 187
samchon::protocol::master::DistributedClientArrayMediator, 192
samchon::protocol::master::DistributedServerArray, 202
samchon::protocol::master::DistributedServerArrayMediator, 206
samchon::protocol::master::DistributedSystemArray, 225
samchon::protocol::master::DistributedSystemArrayMediator, 232
samchon::protocol::master::ParallelClientArrayMediator, 509
samchon::protocol::master::ParallelServerArrayMediator, 531
samchon::protocol::master::ParallelSystemArrayMediator, 551
startTime
samchon::protocol::InvokeHistory, 413
stdev_p
samchon::library::Math, 456
stdev_s
samchon::library::Math, 456
stmt
samchon::library::SQLi, 618
stmtMutex
samchon::library::SQLi, 618
str
samchon::WeakString, 688
samchon::protocol::InvokeParameter, 424
substitute
samchon::library::StringUtil, 625
substituteSQL
samchon::library::StringUtil, 627
substr
samchon::WeakString, 681
substring
samchon::WeakString, 681
samchon::library::StringUtil, 631
system
samchon::protocol::ExternalSystemRole, 327
samchon::protocol::master::DSInvokeHistory, 241
samchon::protocol::master::PRInvokeHistory, 560
systemArray
samchon::protocol::master::ParallelSystem, 542
SystemRole
samchon::protocol::SystemRole, 635

TAG
samchon::example::chat_service::ChatMessage, 101
samchon::example::chat_service::ChatRoom, 105
samchon::library::FTInstance, 344
samchon::namtree::NTCriteria, 465
samchon::namtree::NTParameter, 483
samchon::namtree::NTParameterArray, 486
samchon::namtree::NTParameterDetermined, 490
samchon::namtree::NTSide, 494
samchon::protocol::Entity, 248
samchon::protocol::ExternalSystem, 314
samchon::protocol::ExternalSystemArray, 320
samchon::protocol::ExternalSystemRole, 325
samchon::protocol::Invoke, 405
samchon::protocol::InvokeHistory, 411
samchon::protocol::InvokeHistoryArray, 415
samchon::protocol::InvokeParameter, 422
samchon::protocol::SystemRole, 636
samchon::protocol::master::DistributedSlaveSystemMedia
samchon::protocol::master::ParallelSlaveSystemMediator
TP_1970
samchon::library::Date, 159
TSQLi
samchon::library::TSQLi, 642
tag
samchon::library::XML, 708
toHTML
samchon::protocol::IHTMLEntity, 389
toInvoke
samchon::example::chat_service::ChatRoomArray, 112
samchon::protocol::InvokeHistory, 412
toLinuxTime
samchon::library::Date, 157
toLowerCase
samchon::WeakString, 687
samchon::library::StringUtil, 633
toMatrix
samchon::library::CaseGenerator, 91
toMultibyte
samchon::library::Charset, 93, 94
toNumber
samchon::library::StringUtil, 628
toSQL
samchon::protocol::ISQLEntity, 439
samchon::protocol::Invoke, 406
samchon::protocol::InvokeParameter, 423

toString
samchon::example::packer::Instance, 394
samchon::example::packer::Packer, 498
samchon::example::packer::Product, 567
samchon::example::packer::Wrapper, 694
samchon::example::packer::WrapperArray, 698
samchon::example::tsp::GeometryPoint, 372
samchon::example::tsp::Travel, 639
samchon::library::Date, 158
samchon::library::Datetime, 180
samchon::library::URLVariables, 660
samchon::library::XML, 707
toTH
samchon::protocol::IHTMLEntity, 387
toTM
samchon::library::Date, 156
toTR
samchon::protocol::IHTMLEntity, 388
toUTF8
samchon::library::Charset, 94
toUnicode
samchon::library::Charset, 95
toUpperCase
samchon::WeakString, 688
samchon::library::StringUtil, 633
toXML
samchon::example::chat_service::ChatMessage, 102
samchon::example::chat_service::ChatRoom, 106
samchon::example::chat_service::ChatRoomArray, 111
samchon::library::FTFile, 338
samchon::library::FTFolder, 342
samchon::library::FTInstance, 346
samchon::library::SQLStatement, 623
samchon::library::TSQLStatement, 644
samchon::namtree::INTExplore, 398
samchon::namtree::NTCriteria, 466
samchon::namtree::NTFile, 476
samchon::namtree::NTParameter, 483
samchon::namtree::NTParameterDetermined, 491
samchon::namtree::NTSide, 494
samchon::protocol::Entity, 249
samchon::protocol::EntityArray, 257
samchon::protocol::EntityGroup, 271
samchon::protocol::EntityList, 280
samchon::protocol::ExternalClientArray, 300
samchon::protocol::ExternalServer, 305
samchon::protocol::ExternalSystem, 314
samchon::protocol::ExternalSystemRole, 325
samchon::protocol::Invoke, 405
samchon::protocol::InvokeHistory, 411
samchon::protocol::InvokeParameter, 422
samchon::protocol::SystemRole, 636
samchon::protocol::master::DSInvokeHistory, 240
samchon::protocol::master::DistributedClientArray, 187
samchon::protocol::master::DistributedClientArrayMediators, 192
samchon::protocol::master::DistributedServer, 199
samchon::protocol::master::DistributedSystem, 218
samchon::protocol::master::DistributedSystemArray, 225
samchon::protocol::master::DistributedSystemArrayMedia
samchon::protocol::master::DistributedSystemRole, 236
samchon::protocol::master::PRInvokeHistory, 560
samchon::protocol::master::ParallelClientArrayMediator, 5
samchon::protocol::master::ParallelServer, 513
samchon::protocol::master::ParallelSystem, 541
samchon::protocol::master::ParallelSystemArrayMediator
tournament
samchon::library::GAParameters, 351
samchon::library::GeneticAlgorithm, 368
Travel
samchon::example::tsp::Travel, 638
travel
samchon::example::tsp::Scheduler, 575
trim
samchon::WeakString, 685
samchon::library::StringUtil, 629
tryAcquire
samchon::library::Semaphore, 578
tryInsert
samchon::example::packer::Wrapper, 693
samchon::example::packer::WrapperArray, 696
type
samchon::library::Event, 287
samchon::protocol::InvokeParameter, 423

URLVariables
samchon::library::URLVariables, 658
uid
samchon::example::tsp::GeometryPoint, 372
samchon::library::FTInstance, 347
samchon::protocol::InvokeHistory, 413
samchon::protocol::master::ParallelSystemArray, 547
unique
samchon::library::GeneticAlgorithm, 368
UniqueAcquire
samchon::library::UniqueAcquire, 646, 647
UniqueEntityArray
samchon::protocol, 55
UniqueEntityList
samchon::protocol, 56
UniqueReadLock
samchon::library::UniqueReadLock, 650, 651
UniqueWriteLock
samchon::library::UniqueWriteLock, 655
unlock
samchon::library::SharedReadLock, 599
samchon::library::SharedWriteLock, 603
samchon::library::UniqueReadLock, 652
samchon::library::UniqueWriteLock, 656
useCountMap
samchon::SmartPointer, 613
User
samchon::protocol::service::User, 662
user
samchon::protocol::service::Client, 129
samchon::protocol::service::ServiceKeeper, 593
userSet
samchon::protocol::service::IPUserPair, 432

value
samchon::library::XML, 708
samchon::namtree::NTParameterDetermined, 491
variance_p
samchon::library::Math, 457
variance_s
samchon::library::Math, 457
volume
samchon::example::packer::Instance, 394

WeakString
samchon::WeakString, 674–676
WebServerConnector
samchon::protocol::WebServerConnector, 691
weight
samchon::example::packer::Instance, 394
Wrapper
samchon::example::packer::Wrapper, 693
WrapperArray
samchon::example::packer::WrapperArray, 696
wrapperArray
samchon::example::packer::Packer, 498
write
samchon::ByteArray, 78
writeLock
samchon::library::RWMutex, 572
writeReversely
samchon::ByteArray, 78
writeUnlock
samchon::library::RWMutex, 573

XML
samchon::library::XML, 702–704
XMLList
samchon::library, 38
xml
samchon::protocol::InvokeParameter, 424