SWEeneyThreads

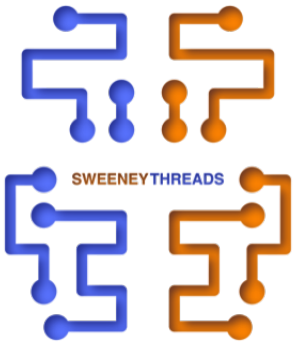
Actorbase

a NoSQL DB based on the Actor model

Definizione di prodotto

Redattori:
Maino Elia

Approvazione:
...
Verifica:
...



Versione 0.0.11

10 giugno 2016

Indice

1 Introduzione **6**
- 1.1 Scopo del documento . . . 6
- 1.2 Scopo del prodotto . . . 6
- 1.3 Glossario . . . 6
- 1.4 Riferimenti . . . 6

2 Standard di progetto **7**
- 2.1 Standard di progettazione . . . 7
- 2.2 Standard di codifica . . . 7
- 2.3 Standard di documentazione del codice . . . 7
- 2.4 Strumenti di lavoro . . . 7

3 Specifica componenti **8**
- 3.1 Actorbase . . . 8
- 3.2 Actorbase.server . . . 9
- 3.3 Actorbase.server.Server (Object) . . . 10
- 3.4 Actorbase.server.StaticSettings (Object) . . . 11
- 3.5 Actorbase.server.ClusterListener . . . 13
- 3.6 Actorbase.server.utils . . . 15
- 3.7 Actorbase.server.utils.Parser . . . 15
- 3.8 Actorbase.server.utils.Helper . . . 17
- 3.9 Actorbase.server.utils.ConfigurationManager . . . 18
- 3.10 Actorbase.server.utils.ReplyBuilder . . . 19
- 3.11 Actorbase.server.utils.Serializer . . . 22
- 3.12 Actorbase.server.utils.FileManager . . . 23
- 3.13 Actorbase.server.utils.fileManagerLibrary . . . 24
- 3.14 Actorbase.server.utils.fileManagerLibrary.SingleFileManager . . . 24
- 3.15 Actorbase.server.utils.fileManagerLibrary.Bounds . . . 26
- 3.16 Actorbase.server.utils.fileManagerLibrary.RemoveStrategy . . . 27
- 3.17 Actorbase.server.utils.fileManagerLibrary.SpoolerRemove . . . 27
- 3.18 Actorbase.server.actors . . . 29
- 3.19 Actorbase.server.actors.Doorkeeper . . . 29
- 3.20 Actorbase.server.actors.Usermanager . . . 31
- 3.21 Actorbase.server.actors.Main . . . 33
- 3.22 Actorbase.server.actors.MapManager . . . 37
- 3.23 Actorbase.server.actors.IndexManager . . . 39
- 3.24 Actorbase.server.actors.Storemanager . . . 40
- 3.25 Actorbase.server.actors.ReplyActor (trait) . . . 44
- 3.26 Actorbase.server.actors.ClusterAwareActor (trait) . . . 45
- 3.27 Actorbase.server.actors.Warehouseman . . . 46
- 3.28 Actorbase.server.enums . . . 46
- 3.29 Actorbase.server.enums.EnumPermission (object) . . . 47
- 3.30 Actorbase.server.enums.UserPermission (trait) . . . 47
- 3.31 Actorbase.server.enums.EnumPermission.Read . . . 47
- 3.32 Actorbase.server.enums.EnumPermission.Write . . . 48
- 3.33 Actorbase.server.enums.EnumReplyResult (object) . . . 48
- 3.34 Actorbase.server.enums.EnumReplyResult.ReplyResult (trait) . . . 49
- 3.35 Actorbase.server.enums.EnumReplyResult.Done . . . 49
- 3.36 Actorbase.server.enums.EnumReplyResult.Error . . . 50
- 3.37 Actorbase.server.enums.EnumStoremanagerType (object) . . . 50
- 3.38 Actorbase.server.enums.EnumStoremanagerType.StoremanagerType (trait) . . . 51
- 3.39 Actorbase.server.enums.EnumStoremanagerType.StorefinderType . . . 51
- 3.40 Actorbase.server.enums.EnumStoremanagerType.StorekeeperType . . . 51
- 3.41 Actorbase.server.enums.EnumStoremanagerType.StorekeeperNinjaType . . . 52
- 3.42 Actorbase.server.enums.EnumStoremanagerType.StorefinderNinjaType . . . 52
- 3.43 Actorbase.server.messages . . . 53
- 3.44 Actorbase.server.messages.internal . . . 54

3.45 Actorbase.server.messages.internal.AskMessages (object) . . . 54
3.46 Actorbase.server.messages.internal.AskMapMessage . . . 55
3.47 Actorbase.server.messages.internal.AskDatabaseMessage . . . 55
3.48 Actorbase.server.messages.internal.LinkMessages . . . 56
3.49 Actorbase.server.messages.internal.LinkMessages.LinkMessage (trait) . . . 56
3.50 Actorbase.server.messages.internal.LinkMessages.AddNinjaMessage . . . 57
3.51 Actorbase.server.messages.internal.LinkMessages.AddWarehousemanMessage . . . 57
3.52 Actorbase.server.messages.internal.LinkMessages.RemoveNinjaMessage . . . 58
3.53 Actorbase.server.messages.internal.LinkMessages.RemoveWarehousemanMessage . . . 58
3.54 Actorbase.server.messages.internal.LinkMessages.BecomeStorefinderNinjaMessage . . . 59
3.55 Actorbase.server.messages.internal.ScalabilityMessages . . . 59
3.56 Actorbase.server.messages.internal.ScalabilityMessages.ScalabilityMessage (trait) . . . 59
3.57 Actorbase.server.messages.internal.ScalabilityMessages.SendMapMessage . . . 60
3.58 Actorbase.server.messages.internal.StorageMessages . . . 60
3.59 Actorbase.server.messages.internal.StorageMessages.StorageMessages (trait) . . . 60
3.60 Actorbase.server.messages.internal.StorageMessages.WriteMapMessage . . . 61
3.61 Actorbase.server.messages.internal.StorageMessages.ReadMapMessage . . . 61
3.62 Actorbase.server.messages.query . . . 62
3.63 Actorbase.server.messages.query.QueryMessage (trait) . . . 62
3.64 Actorbase.server.messages.query.ServiceErrorInfo . . . 63
3.65 Actorbase.server.messages.query.LoginMessage . . . 63
3.66 Actorbase.server.messages.query.ReplyInfo (trait) . . . 64
3.67 Actorbase.server.messages.query.ReplyErrorInfo . . . 65
3.68 Actorbase.server.messages.query.ReplyMessage . . . 66
3.69 Actorbase.server.messages.query.ErrorMessages . . . 67
3.70 Actorbase.server.messages.query.ErrorMessages.ErrorMessage (trait) . . . 67
3.71 Actorbase.server.messages.query.ErrorMessages.InvalidQueryMessage . . . 68
3.72 Actorbase.server.messages.query.ErrorMessages.QueryErrorInfo . . . 68
3.73 Actorbase.server.messages.query.PermissionMessages . . . 69
3.74 Actorbase.server.messages.query.PermissionMessages.AdminPermissionMessage (trait) . . . 69
3.75 Actorbase.server.messages.query.PermissionMessages.NoPermissionMessage (trait) . . . 70
3.76 Actorbase.server.messages.query.PermissionMessages.ReadMessage (trait) . . . 70
3.77 Actorbase.server.messages.query.PermissionMessages.ReadWriteMessage (trait) . . . 71
3.78 Actorbase.server.messages.query.PermissionMessages.NoReadPermissionInfo . . . 72
3.79 Actorbase.server.messages.query.PermissionMessages.NoWritePermissionInfo . . . 72
3.80 Actorbase.server.messages.query.PermissionMessages.NoAdminPermissionInfo . . . 72
3.81 Actorbase.server.messages.query.admin . . . 73
3.82 Actorbase.server.messages.query.admin.AdminMessage (trait) . . . 73
3.83 Actorbase.server.messages.query.admin.PermissionsManagementMessages . . . 74
3.84 Actorbase.server.messages.query.admin.PermissionsManagementMessages.
PermissionManagementMessage (trait) . . . 74
3.85 Actorbase.server.messages.query.admin.PermissionsManagementMessages.
AddPermissionMessage . . . 75
3.86 Actorbase.server.messages.query.admin.PermissionsManagementMessages.
RemovePermissionMessage . . . 75
3.87 Actorbase.server.messages.query.admin.PermissionsManagementMessages.
ListPermissionMessage . . . 75
3.88 Actorbase.server.messages.query.admin.PermissionsManagementMessages.
ListPermissionsInfo . . . 76
3.89 Actorbase.server.messages.query.admin.SettingsMessages.RefreshSettingsMessage . . . 76
3.90 Actorbase.server.messages.query.admin.UserManagementMessages . . . 77
3.91 Actorbase.server.messages.query.admin.UserManagementMessages.
UserManagementMessage (trait) . . . 77
3.92 Actorbase.server.messages.query.admin.UserManagementMessages.
AddUserMessage . . . 77
3.93 Actorbase.server.messages.query.admin.UserManagementMessages.
RemoveUserMessage . . . 78

3.94 Actorbase.server.messages.query.admin.UserManagementMessages.ListUserMessage . . . 78
3.95 Actorbase.server.messages.query.admin.UserManagementMessages.ListUserInfo . . . 79
3.96 Actorbase.server.messages.query.admin.UserManagementMessages.NoUserInfo . . . 79
3.97 Actorbase.server.messages.query.admin.UserManagementMessages.AddUserInfo . . . 80
3.98 Actorbase.server.messages.query.admin.UserManagementMessages.RemoveUserInfo . . . 80
3.99 Actorbase.server.messages.query.user . . . 81
3.100 Actorbase.server.messages.query.user.UserMessage (trait) . . . 81
3.101 Actorbase.server.messages.query.user.RowMessages . . . 82
3.102 Actorbase.server.messages.query.user.RowMessages.RowMessage (trait) . . . 82
3.103 Actorbase.server.messages.query.user.RowMessages.InsertRowMessage . . . 83
3.104 Actorbase.server.messages.query.user.RowMessages.UpdateRowMessage . . . 84
3.105 Actorbase.server.messages.query.user.RowMessages.RemoveRowMessage . . . 85
3.106 Actorbase.server.messages.query.user.RowMessages.FindRowMessage . . . 85
3.107 Actorbase.server.messages.query.user.RowMessages.ListKeysMessage . . . 86
3.108 Actorbase.server.messages.query.user.RowMessages.KeyAlreadyExistInfo . . . 86
3.109 Actorbase.server.messages.query.user.RowMessages.KeyDoesNotExistInfo . . . 87
3.110 Actorbase.server.messages.query.user.RowMessages.ListKeyInfo . . . 87
3.111 Actorbase.server.messages.query.user.RowMessages.NoKeyInfo . . . 88
3.112 Actorbase.server.messages.query.user.RowMessages.FindInfo . . . 88
3.113 Actorbase.server.messages.query.user.MapMessages . . . 89
3.114 Actorbase.server.messages.query.user.MapMessages.MapMessage (trait) . . . 90
3.115 Actorbase.server.messages.query.user.MapMessages.CreateMapMessage . . . 90
3.116 Actorbase.server.messages.query.user.MapMessages.DeleteMapMessage . . . 91
3.117 Actorbase.server.messages.query.user.MapMessages.SelectMapMessage . . . 91
3.118 Actorbase.server.messages.query.user.MapMessages.ListMapMessage . . . 92
3.119 Actorbase.server.messages.query.user.MapMessages.MapAlreadyExistInfo . . . 92
3.120 Actorbase.server.messages.query.user.MapMessages.MapDoesNotExistInfo . . . 93
3.121 Actorbase.server.messages.query.user.MapMessages.ListMapInfo . . . 93
3.122 Actorbase.server.messages.query.user.MapMessages.NoMapInfo . . . 94
3.123 Actorbase.server.messages.query.user.MapMessages.NoMapSelectedInfo . . . 94
3.124 Actorbase.server.messages.query.user.DatabaseMessages . . . 95
3.125 Actorbase.server.messages.query.user.DatabaseMessages.DatabaseMessage (trait) . . . 95
3.126 Actorbase.server.messages.query.user.DatabaseMessages.CreateDatabaseMessage . . . 96
3.127 Actorbase.server.messages.query.user.DatabaseMessages.DeleteDatabaseMessage . . . 97
3.128 Actorbase.server.messages.query.user.DatabaseMessages.SelectDatabaseMessage . . . 97
3.129 Actorbase.server.messages.query.user.DatabaseMessages.ListDatabaseMessage . . . 98
3.130 Actorbase.server.messages.query.user.DatabaseMessages.DBAlreadyExistInfo . . . 98
3.131 Actorbase.server.messages.query.user.DatabaseMessages.DBDoesNotExistInfo . . . 99
3.132 Actorbase.server.messages.query.user.DatabaseMessages.ListDBInfo . . . 99
3.133 Actorbase.server.messages.query.user.DatabaseMessages.NoDBInfo . . . 100
3.134 Actorbase.server.messages.query.user.DatabaseMessages.NoDBSelectedInfo . . . 100
3.135 Actorbase.server.messages.query.user.HelpMessages . . . 101
3.136 Actorbase.server.messages.query.user.HelpMessages.HelpMessage (trait) . . . 101
3.137 Actorbase.server.messages.query.user.HelpMessages.CompleteHelp . . . 102
3.138 Actorbase.server.messages.query.user.HelpMessages.SpecificHelp . . . 102
3.139 Actorbase.server.messages.query.user.HelpMessages.CompleteHelpReplyInfo . . . 103
3.140 Actorbase.server.messages.query.user.HelpMessages.SpecificHelpReplyInfo . . . 103
3.141 Actorbase.client . . . 104
3.142 Actorbase.client.Client . . . 104
3.143 Actorbase.client.Welcome . . . 105
3.144 Actorbase.driver . . . 106
3.145 Actorbase.driver.Connection (trait) . . . 106
3.146 Actorbase.driver.ConcreteConnection . . . 107
3.147 Actorbase.driver.Driver . . . 108

4 Diagrammi di sequenza **110**
4.1 Ricezione messaggio 110
4.2 Ricezione comando a livello di riga 111
4.3 Scalabilità Storemanager 111

5 Tracciamento **113**
5.1 Tracciamento requisiti-classi 113
5.2 Tracciamento classi-requisiti 114
5.3 Tracciamento classi-test 116

Elenco delle figure **117**

Elenco delle tabelle **118**

Diario delle modifiche

Versione	Data	Autore	Descrizione
0.0.11	2016-06-02	*Progettista* Maino Elia	Inserimento immagini e correzione errori.
0.0.10	2016-05-31	*Progettista* Biggeri Mattia	Stesura tracciamento.
0.0.9	2016-05-31	*Progettista* Bonato Paolo	Stesura componenti `Driver` e `Client`.
0.0.8	2016-05-30	*Progettista* Bonato Paolo	Completamento stesura sezione `messages`.
0.0.7	2016-05-30	*Progettista* Bortolazzo Matteo	Stesura sezione diagrammi di sequenza.
0.0.6	2016-05-29	*Progettista* Maino Elia	Incremento stesura componente `messages`.
0.0.5	2016-05-28	*Progettista* Padovan Tommaso	Stesura definizione della componente del server `Filemanager`, inizio stesura componente `messages`.
0.0.4	2016-05-26	*Progettista* Maino Elia	Completamento stesura definizione della componente del server `actors`.
0.0.3	2016-05-25	*Progettista* Maino Elia	Stesura definizione della componente del server `actors`.
0.0.2	2016-05-25	*Progettista* Maino Elia	Stesura definizione della componente del server `utils`.
0.0.1	2016-05-24	*Progettista* Maino Elia	Creazione scheletro documento, stesura introduzione, definizione di metodo e formalismo di specifica.

Tabella 1: Diario delle modifiche

1 Introduzione

1.1 Scopo del documento

Il documento illustra la progettazione di dettaglio del software *Actorbase*. Le decisioni architetturali definite nel documento di *Specifica Tecnica* saranno sviluppate ad un livello di dettaglio superiore, tale da fornire uno strumento adeguato a guidare e supportare l'attività di programmazione del gruppo.

1.2 Scopo del prodotto

Il progetto consiste nella realizzazione di un Database NoSQL key-value basato sul modello ad Attori con l'obiettivo di fornire una tecnologia adatta allo sviluppo di moderne applicazioni che richiedono brevissimi tempi di risposta e che elaborano enormi quantità di dati. Lo sviluppo porterà al rilascio del software sotto licenza MIT.

1.3 Glossario

Al fine di evitare ambiguità di linguaggio e di massimizzare la comprensione dei documenti, il gruppo ha steso un documento interno che è il *Glossario v2.0.0*. In esso saranno definiti, in modo chiaro e conciso i termini che possono causare ambiguità o incomprensione del testo.

1.4 Riferimenti

- **Slide dell'insegnamento Ingegneria del software mod.A:**
  `http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E02.pdf`
- **Scala:**
  `http://www.scala-lang.org/`
- **Java:**
  `http://www.java.com/`
- **Akka:**
  `http://akka.io/`
- **IntelliJ:**
  `http://www.jetbrains.com/idea/`

Normativi

- **Norme di progetto:** *Norme di progetto v2.0.0*
- **Capitolato d'appalto Actorbase (C1):**
  `http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C1p.pdf`

2 Standard di progetto

Di seguito si riportano gli standard di progettazione e documentazione a cui i membri del gruppo dovranno attenersi durante l'attività di progettazione di dettaglio e programmazione.

2.1 Standard di progettazione

Gli standard di progettazione architetturale sono definiti nei documenti di *Specifica Tecnica 3.0.0* e *Norme di Progetto 3.0.0, sez 2.2.6*.

2.2 Standard di codifica

Gli standard di codifica sono definiti nel documento *Norme di Progetto 3.0.0, sez 2.2.11*.

2.3 Standard di documentazione del codice

Gli standard relativi alla documentazione del codice prodotto sono definiti nel documento *Norme di progetto 3.0.0, sez 2.2.11*.

2.4 Strumenti di lavoro

Gli strumenti di lavoro da utilizzare sono definiti nel documento *Norme di Progetto 3.0.0*.

3 Specifica componenti

In tale sezione verranno descritti il più dettagliatamente possibile i componenti architetturali definiti nel documento *Specifica Tecnica*.

3.1 Actorbase

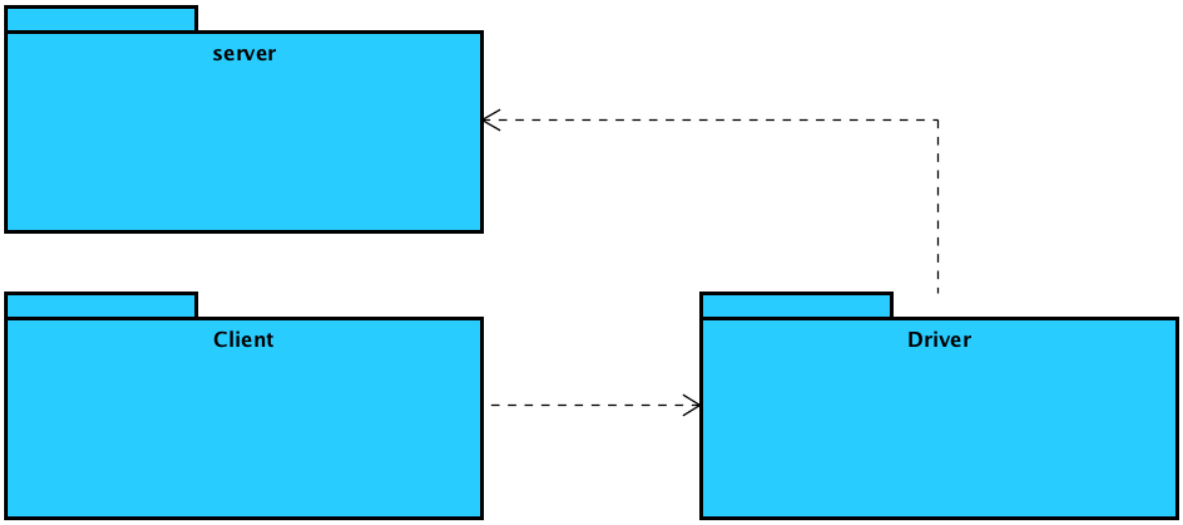


Figura 1: Actorbase architettura generale

L'architettura generale di *Actorbase* è formata da tre componenti: Server, Client e Driver. Il Client utilizza metodi e oggetti forniti dal Driver per comunicare con il Server.

3.2 Actorbase.server

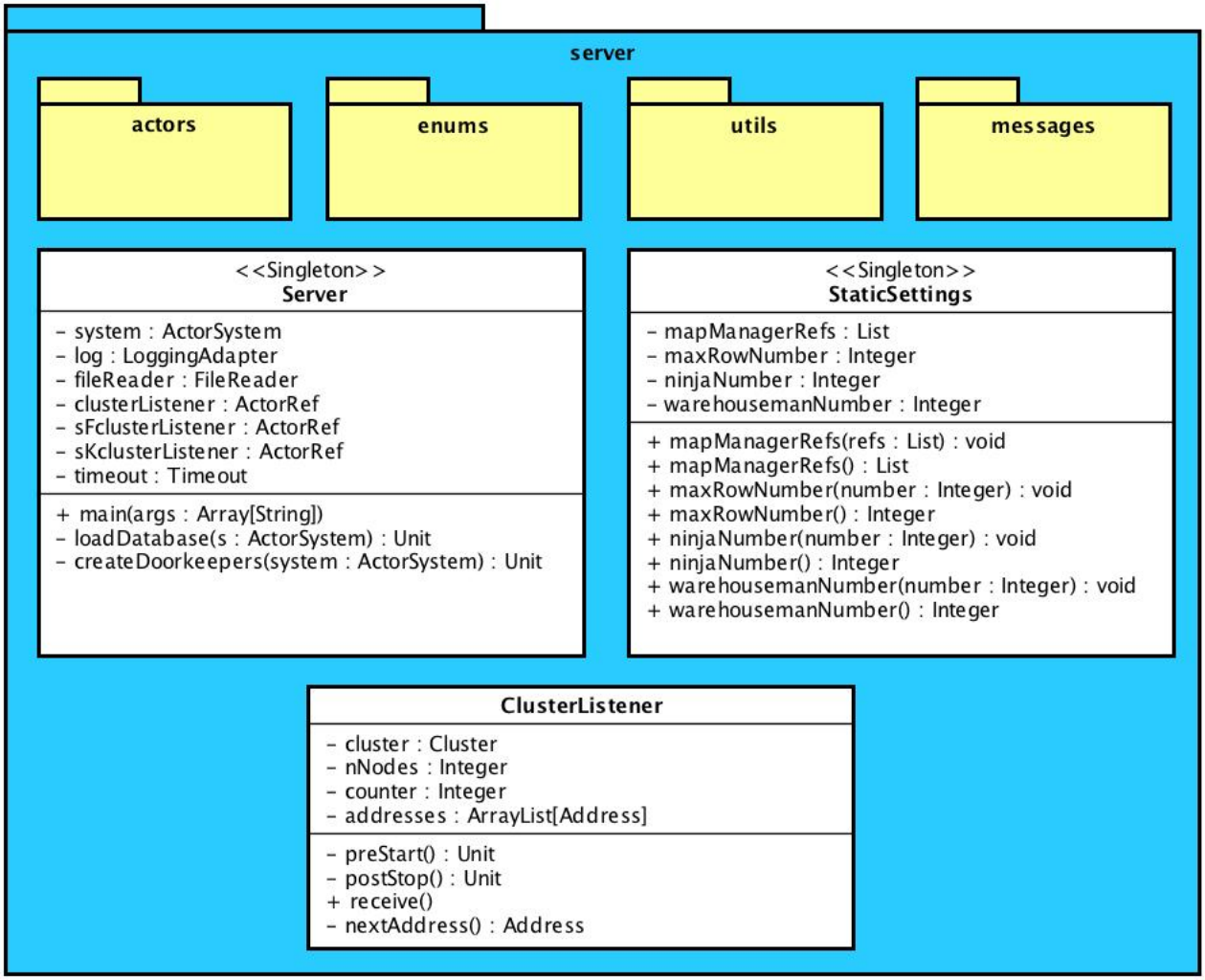


Figura 2: Componente Actorbase.server

La componente server di *Actorbase* è il nucleo dell'applicativo, è composta dai packages: utils, messages, actors ed enums e dalla classe Server.

3.3 Actorbase.server.Server (Object)

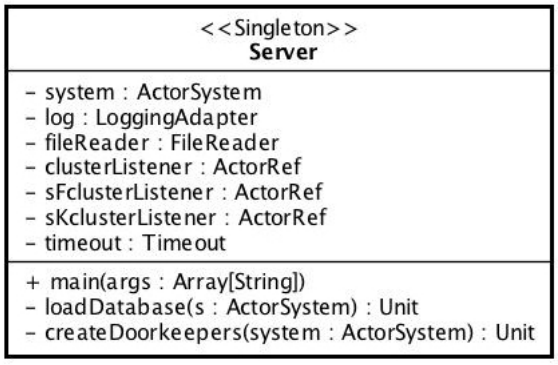


Figura 3: Classe Actorbase.server.Server

Descrizione

Classe principale della parte Server del programma. É di fatto l'entry point dello stesso, gestisce la configurazione iniziale e avvia il sistema. Utilizza il design pattern Singleton (Object).

Utilizzo

Classe che fornisce un punto di accesso al programma, la sua esecuzione avvia il server sulla macchina in cui viene lanciata (contiene il metodo `main` per la componente server di *Actorbase*).

Classi ereditate

Nessuna.

Ereditata da

Nessuna.

Attributi

- `val system: ActorSystem` - Istanza di ActorSystem di Akka.
- `var log: LoggingAdapter` - Permette di ottenere un log per l'ActorSystem.
- `implicit val timeout: Timeout` - Timeout di connessione.
- `var clusterListener: ActorRef` - Cluster
- `var sFclusterListener: ActorRef` - Cluster
- `var sKclusterListener: ActorRef` - Cluster

Metodo: `main(args: Array[String]`

Metodo main che permette di avviare l'applicativo lato server. Si occupa di impostare i valori dei campi dati e di invocare gli altri metodi di configurazione presenti nella classe.

Lista parametri del metodo:

- `args: Array[String]` - Parametro standard del metodo main di *Scala*.

Metodo: `private def loadDatabases(system: ActorSystem): Unit`

Il metodo carica i database da disco.

Lista parametri del metodo:

- `system: ActorSystem` - ActorSystem da utilizzare per accedere agli attori necessari.

Metodo: `private def createDoorkeepers(system: ActorSystem): Unit`

Legge le impostazioni di configurazione degli attori `Doorkeeper` e si occupa della conseguente creazione degli attori stessi.

Lista parametri del metodo:

- `system: ActorSystem` - ActorSystem da utilizzare per accedere agli attori necessari.

3.4 Actorbase.server.StaticSettings (Object)

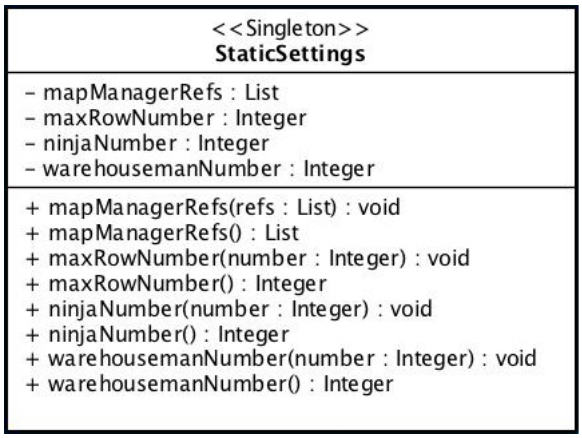


Figura 4: Classe Actorbase.server.StaticSettings

Descrizione

Classe statica che permette di accedere a dei dati (impostazioni) globali.

Utilizzo

La classe definisce i valori di alcune proprietà che devono essere utilizzati da diversi componenti del sistema, evitando il passaggio di tali dati tra le componenti. Alcuni dei dati che la classe contiene devono essere:

- Riferimento agli attori `MapManager` presenti
- Numero massimo di righe per `Storemanager` (di tipo `Storekeeper`)
- Numero di attori `Ninja`
- Numero di attori `Warehouseman`

Classi ereditate

Nessuna.

Ereditata da

Nessuna.

Attributi

- `private var mapManagerRefs: List` - Riferimento ai `MapManger`.
- `private var maxRowNumber: Integer` - Numero massimo di righe.
- `private var ninjaNumber: Integer` - Numero di `Ninja`.
- `private var warehousemanNumber: Integer` - Numero di `Warehousean`.

Metodo: `mapManagerRefs(refs: List) : Unit`

Metodo per impostare la lista di riferimenti agli attori `MapManager`.

Lista parametri del metodo:

- `refs: List` - La lista di riferimenti agli attori.

Metodo: `mapManagerRefs() : List`

Metodo per ottenere la lista di riferimenti agli attori `MapManager`.

Lista parametri del metodo:

Nessuno.

Metodo: `maxRowNumber(number: Integer) : Unit`

Metodo per impostare il numero massimo di righe di uno `Storemanager`.

Lista parametri del metodo:

- `number: Integer` - L'intero rappresentante il numero massimo di righe.

Metodo: `maxRowNumber() : Integer`

Metodo per ottenere il numero massimo di righe degli `Storemanager`.

Lista parametri del metodo:

Nessuno.

Metodo: `ninjaNumber(number: Integer) : Unit`

Metodo per impostare il numero di attori di tipo `Ninja` che uno `Storemanager` deve avere.

Lista parametri del metodo:

- `number: Integer` - L'intero rappresentante il numero di `Ninja`.

Metodo: `ninjaNumber() : Integer`

Metodo per ottenere il numero di `Ninja`.

Lista parametri del metodo:

Nessuno.

Metodo: `warehousemanNumber(number: Integer) : Unit`

Metodo per impostare il numero di attori di tipo `Warehouseman` per mappa.

Lista parametri del metodo:

- `number: Integer` - L'intero rappresentante il numero di `Warehouseman`.

Metodo: `warehousemanNumber() : Integer`

Metodo per ottenere il numero di `Warehouseman`.

Lista parametri del metodo:

Nessuno.

3.5 Actorbase.server.ClusterListener

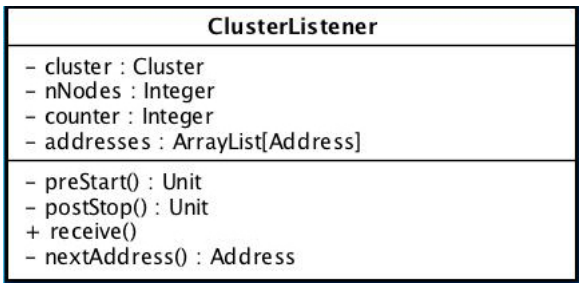


Figura 5: Classe Actorbase.server.ClusterListener

Descrizione

La classe rappresenta l'attore responsabile di mantenere gli indirizzi dei nodi segnati come *UP* nel cluster. Deve esserci un attore `ClusterListener` in ogni nodo del cluster. L'attore inoltre implementa una strategia Round Robin per selezionare un indirizzo dalla sua lista di nodi.

Utilizzo

Questo attore viene utilizzato per gestire le funzionalità del Cluster.

Classi ereditate

- `akka.actor.Actor`
- `akka.actor.ActorLogging`

Ereditata da

Nessuna.

Attributi

- `private val cluster: Cluster` - L'istanza del cluster.
- `private var nNodes: Integer` - Numero di nodi *UP* nel cluster (inizialmente 0).
- `var counter: Integer` - Contatore delle richieste (inizialmente a 0). Deve essere incrementato prima di ogni operazione.
- `var addresses: ArrayList[Address]` - Lista degli indirizzi dei nodi del cluster.

Metodo: `override def preStart(): Unit`

Override del metodo `preStart()` definito in `akka.actor.Actor`. Alla creazione dell'attore esso si sottoscrive al cluster e aggiunge l'indirizzo del suo nodo alla lista.

Lista parametri del metodo:

Nessuno.

Metodo: `override def postStop(): Unit`

Override del metodo `postStop()` definito in `akka.actor.Actor`. Allo stop l'attore deve rimuoversi dal cluster.

Lista parametri del metodo:

Nessuno.

Metodo: `def receive`

Metodo di ricezione dei messaggi dell'attore, il metodo riceve messaggi dal cluster e il messaggio (stringa) `"next"` (richiesta di rotazione Round Robin). I messaggi ricevuti dal cluster vengono gestiti in modo da mantenere la lista dei nodi aggiornata. Il metodo gestisce i seguenti messaggi:

- `MemberUp`
- `UnreachableMember`
- `MemberRemoved`

Lista parametri del metodo:

Nessuno.

Metodo: `def nextAddress(): Address`

Metodo che implementa la strategia Round Robin per selezionare un indirizzo.

Lista parametri del metodo:

Nessuno.

3.6 Actorbase.server.utils

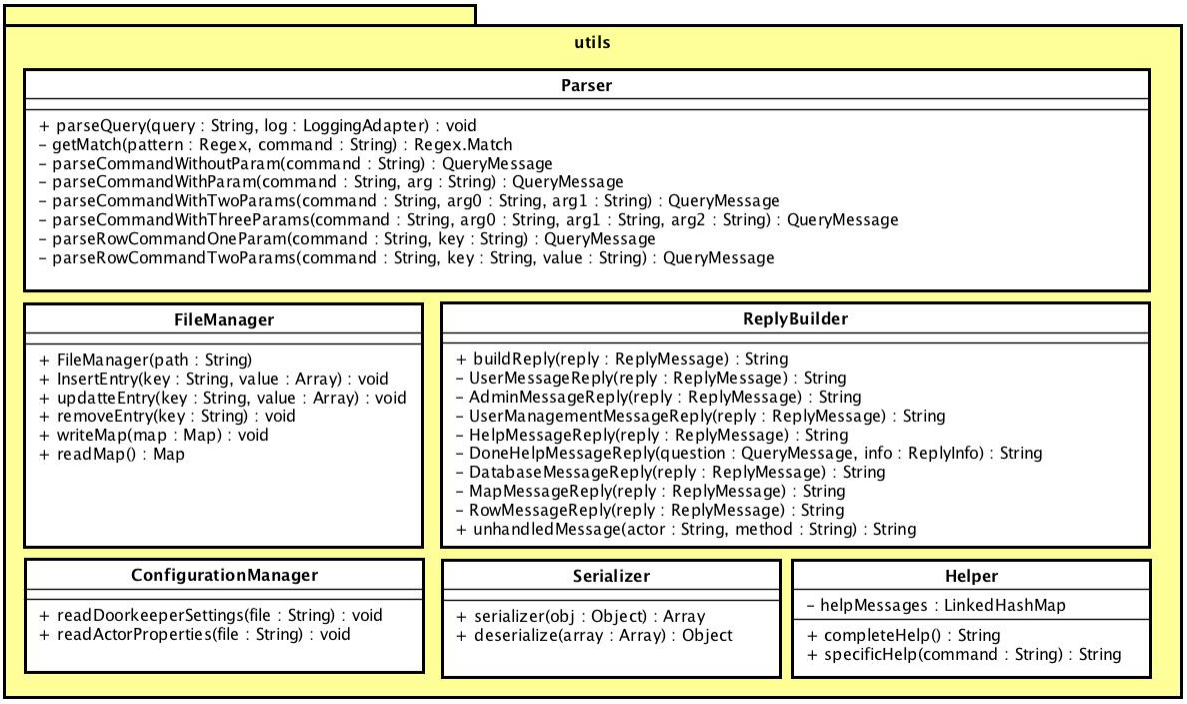


Figura 6: Componente Actorbase.server.utils

Package contenente le classi che effettuano operazioni varie a supporto delle varie componenti del server, e degli attori nello specifico.

3.7 Actorbase.server.utils.Parser

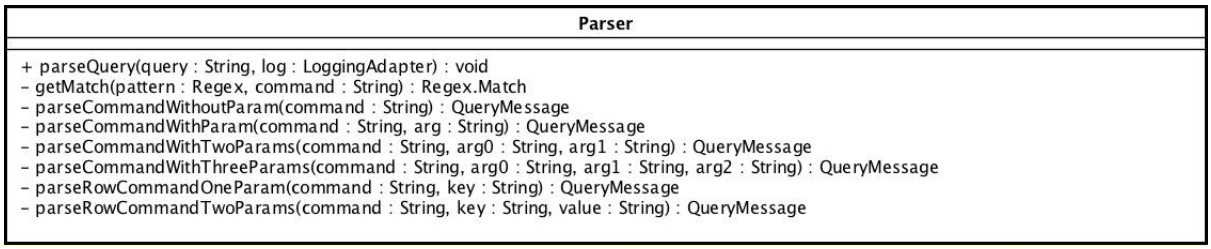


Figura 7: Componente Actorbase.server.utils.Parser

Descrizione

La classe `Parser` definisce i metodi per trasformare stringhe in messaggi `QueryMessage` utilizzabili dagli attori del sistema.

Utilizzo

Viene utilizzata da attori di tipo `Usermanager` per trasformare le richieste client in messaggi inviabili agli attori.

Classi ereditate

Nessuna.

Ereditata da

Nessuno.

Attributi

Nessuno.

Costruttore: `Parser()`

Costruttore senza parametri.

Lista parametri del metodo:

Nessuno.

Metodo: `parseQuery(query: String) : QueryMessage`

Effettua il parsing della stringa in base al numero di parametri che la compongono (utilizzando i metodi per il parsing a seconda dei parametri) e genera un `QueryMessage` che viene ritornato.

Lista parametri del metodo:

- `query: String` - Stringa da convertire in messaggio.

Metodo: `private getMatch(pattern:Regex, command:String): Regex.Match`

Effettua il match dell'espressione regolare sulla stringa passata e ritorna il risultato.

Lista parametri del metodo:

- `pattern: Regex` - Pattern da utilizzare per il match.
- `command: String` - La stringa su cui effettuare il match.

Metodo: `private parseCommandWithoutParam(command: String): QueryMessage`

Effettua il parsing di un comando senza parametri e ritorna il corrispondente `QueryMessage`.

Lista parametri del metodo:

- `command: String` - La stringa rappresentante il comando.

Metodo: `private parseCommandWithParam(command: String, arg: String): QueryMessage`

Effettua il parsing di un comando con un parametro e ritorna il corrispondente `QueryMessage`.

Lista parametri del metodo:

- `command: String` - La stringa rappresentante il comando.
- `arg: String` - La stringa rappresentante il parametro.

Metodo: `private parseCommandWithTwoParams(command:String, arg1: String, arg2: String): QueryMessage`

Effettua il parsing di un comando con due parametri e ritorna il corrispondente `QueryMessage`.

Lista parametri del metodo:

- `command: String` - La stringa rappresentante il comando.

- `arg1: String` - La stringa rappresentante il primo parametro.
- `arg2: String` - La stringa rappresentante il secondo parametro.

Metodo: `private parseCommandWithThreeParams(command:String, arg1: String, arg2: String, arg3: String): QueryMessage`

Effettua il parsing di un comando con tre parametri e ritorna il corrispondente `QueryMessage`.

Lista parametri del metodo:

- `command: String` - La stringa rappresentante il comando.
- `arg1: String` - La stringa rappresentante il primo parametro.
- `arg2: String` - La stringa rappresentante il secondo parametro.
- `arg3: String` - La stringa rappresentante il terzo parametro.

Metodo: `private parseRowCommandOneParam(command: String, key: String): QueryMessage`

Effettua il parsing di un comando al livello di item con un parametro (la chiave) e ritorna il corrispondente `QueryMessage`.

Lista parametri del metodo:

- `command: String` - La stringa rappresentante il comando a livello di item.
- `key: String` - La stringa rappresentante la chiave.

Metodo: `private parseRowCommandTwoParams(command: String, key: String, value: String): QueryMessage`

Effettua il parsing di un comando al livello di item con due parametri (la chiave e il valore) e ritorna il corrispondente `QueryMessage`.

Lista parametri del metodo:

- `command: String` - La stringa rappresentante il comando a livello di item.
- `key: String` - La stringa rappresentante la chiave.
- `value: String` - La stringa rappresentante il valore.

3.8 Actorbase.server.utils.Helper

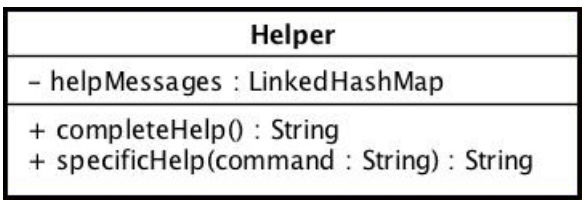


Figura 8: Componente Actorbase.server.utils.Helper

Descrizione

Classe che fornisce i metodi per ottenere una descrizione dei comandi di *Actorbase*.

Utilizzo

Viene utilizzata per soddisfare una richiesta di `help` da parte di un utente.

Classi ereditate

Nessuna.

Ereditata da

Nessuno.

Attributi

- `private helpMessages: LinkedHashMap[String, String]` - Mappa contenente i comandi come chiavi e le descrizioni degli stessi come valori.

Metodo: `completeHelp(): String`

Il metodo costruisce una stringa contenente l'aiuto completo, basandosi sugli elementi della mappa `helpMessages`.

Lista parametri del metodo:

Nessuno.

Metodo: `specificHelp(command: String): String`

Il metodo costruisce una stringa contenente l'aiuto per un comando specifico, basandosi sugli elementi della mappa `helpMessages`.

Lista parametri del metodo:

- `command: String` - Stringa rappresentante il comando per cui si vuole generare il messaggio di aiuto.

3.9 Actorbase.server.utils.ConfigurationManager

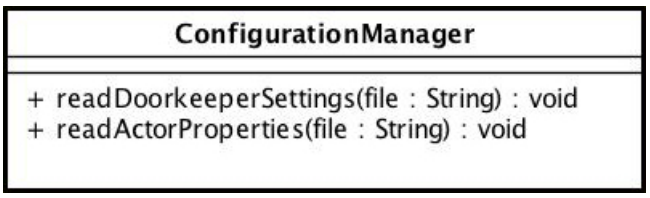


Figura 9: Componente Actorbase.server.utils.ConfigurationManager

Descrizione

Classe che fornisce i metodi di lettura e scrittura dei file di configurazione del server.

Utilizzo

Viene utilizzata per leggere le impostazioni del server dai file di configurazione all'avvio di esso. Inoltre viene utilizzata per scrivere modifiche alle configurazioni.

Classi ereditate

Nessuna.

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodo: `readDoorkeepersSettings(fileName: String): util.HashMap[String, Integer]`

Il metodo legge dal file di configurazione gli indirizzi e le porte che attori di tipo `Doorkeeper` dovranno utilizzare per gestire le connessioni. Tali informazioni vengono ritornate con una mappa in cui le chiavi sono gli indirizzi e i valori sono le porte.

Lista parametri del metodo:

- `fileName: String` - Nome del file che contiene la configurazione dei `Doorkeeper`.

Metodo: `readActorsProperties(fileName: String): util.HashMap[ActorProperties, Integer]`

Il metodo legge dal file di configurazione le proprietà relative agli attori (come ad esempio il numero massimo di attori di tipo `Ninja`). Tali informazioni vengono ritornate con una mappa in cui le chiavi sono i nomi delle proprietà e i valori sono i valori di tali proprietà.

Lista parametri del metodo:

- `fileName: String` - Nome del file che contiene la configurazione degli attori.

3.10 Actorbase.server.utils.ReplyBuilder

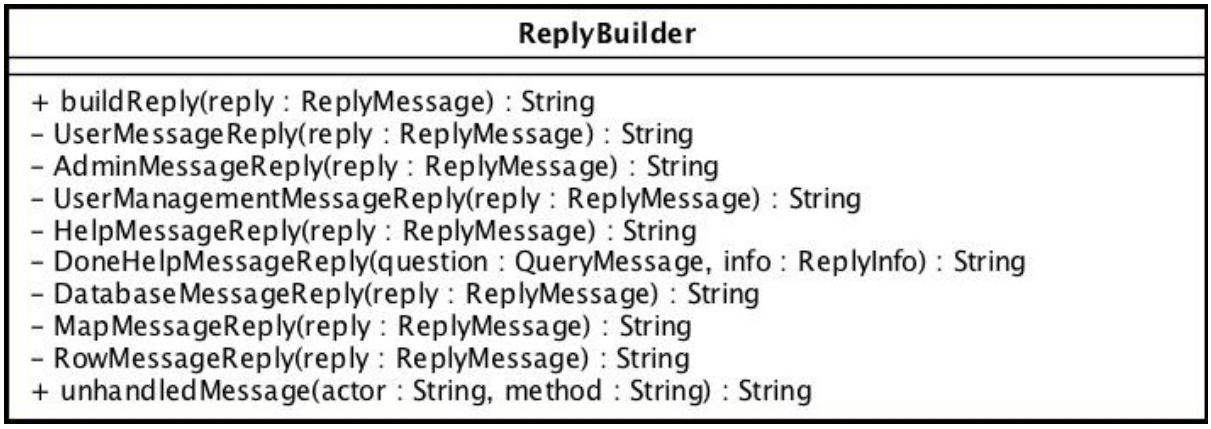


Figura 10: Componente Actorbase.server.utils.ReplyBuilder

Descrizione

Classe che fornisce i metodi di creazione delle stringhe da mandare in risposta a richieste client.

Utilizzo

Viene utilizzata per costruire delle risposte in formato stringa a partire da messaggi. Tali risposte possono così essere inviate ad un client.

Classi ereditate

Nessuna.

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodo: `buildReply(reply: ReplyMessage): String`

Il metodo permette di costruire una stringa a partire da un `ReplyMessage`. In particolare questo metodo si occupa di stabilire se il messaggio è di tipo amministratore o utente e delegare di conseguenza l'elaborazione al metodo più appropriato. Gestisce i seguenti messaggi:

- `UserMessage`
- `AdminMessage`

Lista parametri del metodo:

- `reply: ReplyMessage` - Il messaggio da cui ricavare la stringa.

Metodo: `private UserMessageReply(reply: ReplyMessage): String`

Il metodo permette di costruire una stringa a partire da un `ReplyMessage`. In particolare questo metodo si occupa di stabilire che tipo di `UserMessage` si sia ricevuto. Gestisce i seguenti messaggi:

- `HelpMessage`
- `DatabaseMessage`
- `MapMessage`
- `RowMessage`

Lista parametri del metodo:

- `reply: ReplyMessage` - Il messaggio da cui ricavare la stringa.

Metodo: `private AdminMessageReply(reply: ReplyMessage): String`

Il metodo permette di costruire una stringa a partire da un `ReplyMessage`. In particolare questo metodo si occupa di stabilire che tipo di `AdminMessage` si sia ricevuto. Gestisce i seguenti messaggi:

- `UsersManagementMessage`
- `PermissionsManagementMessage`

Lista parametri del metodo:

- `reply: ReplyMessage` - Il messaggio da cui ricavare la stringa.

Metodo: `private UserManagementMessageReply(reply: ReplyMessage): String`

Il metodo permette di costruire una stringa a partire da un `ReplyMessage`. In particolare questo metodo si occupa di gestire messaggi di tipo `UsersManagementMessage`. Gestisce i seguenti messaggi:

- `ListUserMessage`
- `AddUserMessage`
- `RemoveUserMessage`

Lista parametri del metodo:

- `reply: ReplyMessage` - Il messaggio da cui ricavare la stringa.

Metodo: `private PermissionsManagementMessageReply(reply: ReplyMessage): String`

Il metodo permette di costruire una stringa a partire da un `ReplyMessage`. In particolare questo metodo si occupa di gestire messaggi di tipo `PermissionManagementMessage`. Gestisce i seguenti messaggi:

- `ListPermissionMessage`
- `AddPermissionMessage`
- `RemovePermissionMessage`

Lista parametri del metodo:

- `reply: ReplyMessage` - Il messaggio da cui ricavare la stringa.

Metodo: `private HelpMessageReply(reply: ReplyMessage): String`

Il metodo permette di costruire una stringa a partire da un `ReplyMessage`. In particolare questo metodo si occupa di gestire messaggi di tipo `HelpMessage` invocando gli opportuni metodi. Gestisce i seguenti messaggi:

- `HelpMessage`

Lista parametri del metodo:

- `reply: ReplyMessage` - Il messaggio da cui ricavare la stringa.

Metodo: `private DoneHelpMessageReply(question: QueryMessage, info: ReplyInfo): String`

Il metodo permette di costruire una stringa a partire da un `ReplyMessage`. In particolare questo metodo si occupa di gestire messaggi di tipo `HelpMessage` maggiormente nel dettaglio. Gestisce i seguenti messaggi:

- `CompleteHelpMessage`
- `SpecificHelpMessage`

Lista parametri del metodo:

- `reply: ReplyMessage` - Il messaggio da cui ricavare la stringa.

Metodo: `private DatabaseMessageReply(reply: ReplyMessage): String`

Il metodo permette di costruire una stringa a partire da un `ReplyMessage`. In particolare questo metodo si occupa di gestire messaggi di tipo `DatabaseMessage`. Gestisce i seguenti messaggi:

- `ListDatabaseMessage`
- `SelectDatabaseMessage`
- `CreateDatabaseMessage`
- `DeleteDatabaseMessage`

Lista parametri del metodo:

- `reply: ReplyMessage` - Il messaggio da cui ricavare la stringa.

Metodo: `private MapMessageReply(reply: ReplyMessage): String`

Il metodo permette di costruire una stringa a partire da un `ReplyMessage`. In particolare questo metodo si occupa di gestire messaggi di tipo `MapMessage`. Gestisce i seguenti messaggi:

- `ListMapMessage`
- `SelectMapMessage`
- `CreateMapMessage`

- `DeleteMapMessage`

Lista parametri del metodo:

- `reply: ReplyMessage` - Il messaggio da cui ricavare la stringa.

Metodo: `private RowMessageReply(reply: ReplyMessage): String`

Il metodo permette di costruire una stringa a partire da un `ReplyMessage`. In particolare questo metodo si occupa di gestire messaggi di tipo `RowMessage`. Gestisce i seguenti messaggi:

- `ListKeysMessage`
- `FindRowMessage`
- `InsertRowMessage`
- `UpdateRowMessage`
- `RemoveRowMessage`

Lista parametri del metodo:

- `reply: ReplyMessage` - Il messaggio da cui ricavare la stringa.

Metodo: `unhandledMessage(actor: String, method: String): String`

Il metodo permette di costruire una stringa per i messaggi che non sono stati gestiti. Lista parametri del metodo:

- `actor: String` - Il percorso dell'attore che non ha gestito il messaggio
- `method: String` - Il nome del metodo in cui non è stato gestito il messaggio

3.11 Actorbase.server.utils.Serializer

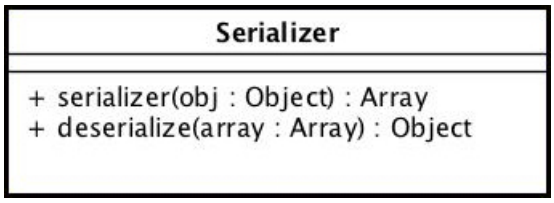


Figura 11: Componente Actorbase.server.utils.Serializer

Descrizione

Classe che gestisce la serializzazione e la deserializzazione di oggetti.

Utilizzo

Viene utilizzata per serializzare e deserializzare oggetti in Array di Byte in modo da poterli trattare come dati di *Actorbase*.

Classi ereditate

Nessuna.

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodo: `serialize(obj: Object): Array[Byte]`

Il metodo serializza un oggetto in un array di Byte.

Lista parametri del metodo:

- `obj: Object` - L'oggetto da serializzare.

Metodo: `deserialize(array: Array[Byte]): Object`

Il metodo genera un Oggetto a partire da un array di Byte.

Lista parametri del metodo:

- `array: Array[Byte]` - L'array da utilizzare per generare l'oggetto.

3.12 Actorbase.server.utils.FileManager

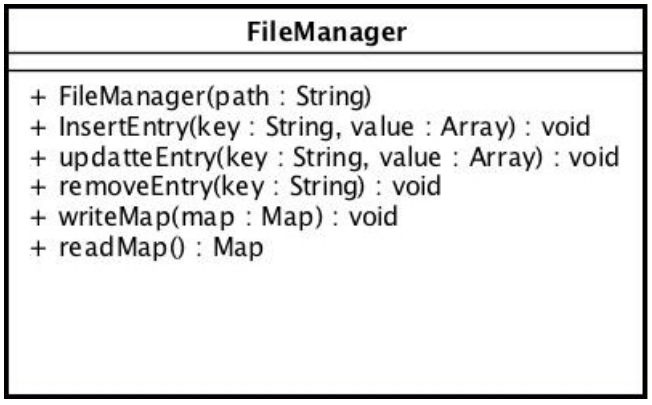


Figura 12: Componente Actorbase.server.utils.FileManager

Descrizione

Interfaccia che dichiara i metodi per leggere e scrivere dati su disco.

Utilizzo

Viene utilizzata da attori di tipo `Warehouseman` per gestire la persistenza dei dati.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.utils.fileManagerLibrary.SingleFileManager`

Attributi

- `removeStrategy: RemoveStrategy` - La strategia con cui verranno rimosse le value dal/dai file in cui sono salvate.

Metodo: `InsertEntry(key: String, value: Array[Byte])`

Metodo astratto per salvare la chiave ed il valore specificato su disco.

Lista parametri del metodo:

- `key: String` - La chiave da inserire.
- `value: Array[Byte]` - Il valore da inserire.

Metodo: `def UpdateEntry(key:String, value: Array[Byte])`

Metodo astratto per aggiornare il valore della chiave specificata su disco.

Lista parametri del metodo:

- `key: String` - La chiave di cui modificare il valore.
- `value: Array[Byte]` - Il nuovo valore da inserire.

Metodo: `def RemoveEntry(key: String)`

Metodo astratto per rimuovere la chiave indicata ed il relativo valore da disco.

Lista parametri del metodo:

- `key: String` - La chiave di da eliminare.

Metodo: `def WriteMap(map: util.HashMap[String, Array[Byte]])`

Metodo astratto per salvare una intera mappa chiave-valore su disco.

Lista parametri del metodo:

- `map: util.HashMap[String, Array[Byte]]` - La mappa da salvare su disco.

Metodo: `def ReadMap(): ConcurrentHashMap[String, Array[Byte]]`

Metodo astratto per leggere una intera mappa da disco.

Lista parametri del metodo: Nessuno.

3.13 Actorbase.server.utils.fileManagerLibrary

Package contenente le classi che effettuano operazioni per gestire i file su disco. Le classi di questo package vengono usate principalmente dall'attore di tipo `Warehouseman` che si occupa della persistenza dei dati su disco.

3.14 Actorbase.server.utils.fileManagerLibrary.SingleFileManager

Descrizione

Classe che implementa una semplice strategia per `Actorbase.server.utils.FileManager`.

Utilizzo

Viene utilizzata da attori di tipo `Warehouseman` per gestire la persistenza dei dati. In questa semplice strategia si usano due soli file per salvare una mappa. Il primo (`keyMap`) è una *HashMap* serializzata che indicizza puntatore all'inizio della value e la lunghezza della stessa rispetto alle chiavi. Il secondo (`valueFile`) è un file che contiene tutte le value in Byte concatenate.

Classi ereditate

- `Actorbase.server.utils.FileManager`

Ereditata da

Nessuna.

Attributi

- `path : String` - Il path assoluto del file `keyMap` che contiene chiavi e bounds dei valori.
- `valuesPath : String` - Il path assoluto del file `valueFile` che contiene i valori in Byte concatenati.
- `removeStrategy : RemoveStrategy` - La strategia di rimozione dal file delle *values*.

Metodo: `override def InsertEntry(key: String, value: Array[Byte])`

Prima inserisce la chiave e salva il puntatore all'inizio di dove sarà la value. Dopodiché inserisce la value nel `valueFile` appendendola alla fine.

Lista parametri del metodo:

- `key: String` - La chiave de inserire.
- `value: Array[Byte]` - Il valore da inserire.

Metodo: `override def UpdateEntry(key: String, value: Array[Byte]`

Prima rimuove la chiave da aggiornare e poi la reinserisce con il nuovo valore mediante i metodi `InsertEntry` e `RemovetEntry`.

Lista parametri del metodo:

- `key: String` - La chiave di cui modificare il valore.
- `value: Array[Byte]` - Il nuovo valore da inserire.

Metodo: `override def RemoveEntry(key: String)`

Prima rimuove la chiave selezionata e salva il puntatore all'inizio della value. Dopodiché rimuove il valore da `valueFile` secondo la strategia impostata.

Lista parametri del metodo:

- `key: String` - La chiave da rimuovere.

Metodo: `override def ReadMap(): ConcurrentHashMap[String, Array[Byte]]`

Legge una intera mappa da disco, la deserializza e la ritorna sotto forma di *ConcurrentHashMap*.

Lista parametri del metodo:

Nessuno.

Metodo: `WriteMap(map: util.HashMap[String, Array[Byte]])`

Salva su disco l'intera mappa sovrascrivendo i file precedentemente salvati.

Lista parametri del metodo:

- `map: util.HashMap[String, Array[Byte]]` - La mappa da salvare su disco.

Metodo: `private def insertKey(key: String, from: Long, off: Long)`

Inserisce una chiave nella mappa delle chiavi e la salva su disco.

Lista parametri del metodo:

- `key: String` - La chiave da inserire.
- `from: Long` - Il puntatore all'inizio della value.
- `off: Long` - La lunghezza in Byte della value.

Metodo: `private def removeKey(keyMap: ConcurrentHashMap[String,Bounds], key: String)`

Salva puntatore all'inizio della value e offset del valore puntato dalla chiave. Rimuove la chiave dalla mappa delle chiavi in RAM. Dopodiché scorre tutta la mappa per spostare indietro i puntatori agli inizi delle values che prima della rimozione avevano punto di inizio successivo a quello del valore della chiave rimossa.

Lista parametri del metodo:

- `keyMap: ConcurrentHashMap[String,Bounds]` - La mappa delle chiavi.
- `key: String` - La chiave da rimuovere.

Metodo: `private def insertValue(file: RandomAccessFile, value: Array[Byte])`

Inserisce il valore specificato alla fine del file `valueFile`.

Lista parametri del metodo:

- `file: RandomAccessFile` - Il file delle value.
- `value: Array[Byte]` - Il valore da inserire.

Metodo: `private def removeValue(file: RandomAccessFile, init: Long, off: Long)`

Rimuove la porzione indicata del file delle value secondo la strategia impostata.

Lista parametri del metodo:

- `file: RandomAccessFile` - Il file dei valori.
- `init: Long` - Il puntatore all'inizio della sezione da rimuovere.
- `off: Long` - La lunghezza della sezione da rimuovere.

Metodo: `private def readMap(): ConcurrentHashMap[String,Bounds]`

Legge su disco il file contenente la mappa delle chiavi, lo serializza e ritorna il risultato sotto forma di *ConcurrentHashMap*

Lista parametri del metodo:

Nessuno.

Metodo: `private def writeMap(map: ConcurrentHashMap[String,Bounds])`

Serializza la mappa delle chiavi e la salva su disco.

Lista parametri del metodo:

- `map: ConcurrentHashMap[String,Bounds]` - La mappa delle chiavi.

3.15 Actorbase.server.utils.fileManagerLibrary.Bounds

Descrizione

Classe che definisce una coppia di puntatori all'inizio di una value e lunghezza Byte.

Utilizzo

Viene utilizzata per definire una porzione del file delle value.

Classi ereditate

Nessuna.

Ereditata da

Nessuna.

Attributi

- `var init: Long` - Il puntatore all'inizio di una value.
- `var offset: Long` - La lunghezza in Byte.

3.16 Actorbase.server.utils.fileManagerLibrary.RemoveStrategy

Descrizione

Interfaccia che espone un metodo per rimuovere una sezione indicata di Byte da un file.

Utilizzo

Viene utilizzata per definire l'interfaccia del Design Pattern Strategy riguardo alla strategia di rimozione da array di Byte.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.utils.fileManagerLibrary.SpoolerRemove`

Attributi

Nessuno.

Metodo: `def remove(file: RandomAccessFile, init: Long, off: Long)`

Metodo astratto per rimuovere una sezione di Byte da un file.

Lista parametri del metodo:

- `file: RandomAccessFile` - Il file da cui rimuovere.
- `init: Long` - L'inizio della sezione da rimuovere.
- `off: Long` - La lunghezza in Byte della sezione da rimuovere.

3.17 Actorbase.server.utils.fileManagerLibrary.SpoolerRemove

Descrizione

Classe che implementa una strategia di rimozione da array di Byte. In questa strategia se la sezione che deve essere rimossa non si trova alla fine allora semplicemente la sezione successiva del file viene spostata indietro (a blocchi di 8kB l'uno) della lunghezza della sezione da rimuovere.

Utilizzo

Viene utilizzata per rimuovere sezioni interne del file dei valori all'interno del `FileManager`.

Classi ereditate

- `Actorbase.server.utils.fileManagerLibrary.RemoveStrategy`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodo: `override def remove(file: RandomAccessFile, init: Long, off: Long)`

Rimuove la sezione indicata di Byte da file.

Lista parametri del metodo:

- `file: RandomAccessFile` - Il file da cui rimuovere.
- `init: Long` - L'inizio della sezione da rimuovere.
- `off: Long` - La lunghezza in Byte della sezione da rimuovere.

3.18 Actorbase.server.actors

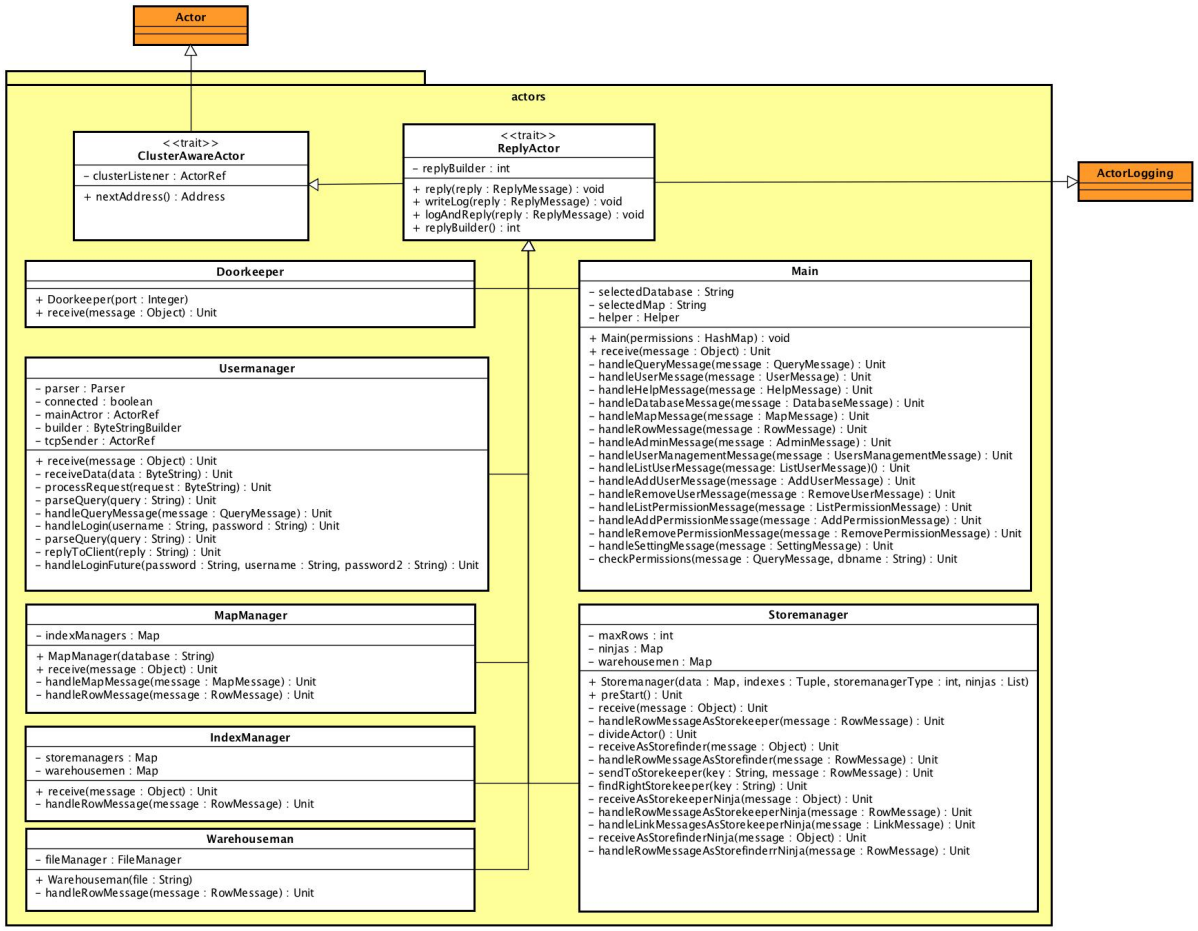


Figura 13: Componente Actorbase.server.actors

Package contenente le classi che definiscono gli attori di *Actorbase*. Gli attori sono le entità che compongono la logica vera e propria del sistema, gli attori definiti in questo package si scambiano i messaggi definiti in `Actorbase.server.messages`.

3.19 Actorbase.server.actors.Doorkeeper

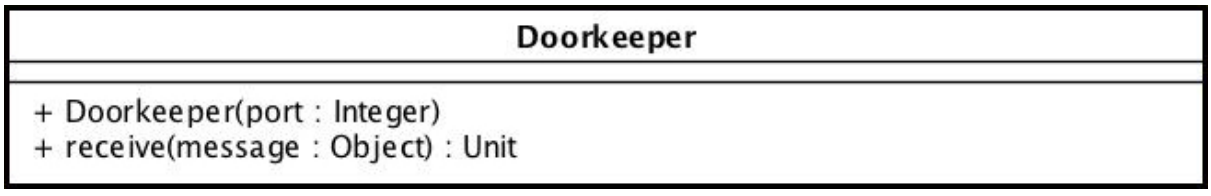


Figura 14: Componente Actorbase.server.actors.Doorkeeper

Descrizione

Classe che definisce l'attore di tipo `Doorkeeper`. Tale attore rappresenta il punto di ingresso al server, apre una porta nell'host e si mette in ascolto di eventuali richieste di connessione. Quando un nuovo client si connette, il `Doorkeeper` crea un nuovo attore di tipo `Usermanager` a cui delega la gestione delle richieste per quella determinata connessione.

Utilizzo

Viene utilizzato per creare e gestire un punto di accesso generale al server.

Classi ereditate

- `akka.actor.Actor`
- `akka.actor.ActorLogging`

Ereditata da

Nessuno.

Attributi

Nessuno.

Costruttore: `Doorkeeper(port: Integer)`

Costruisce un attore di tipo `Doorkeeper` a partire da un Integer rappresentante la porta da aprire.

Lista parametri del metodo:

- `array: Array[Byte]` - L'array da utilizzare per generare l'oggetto.

Metodo: `receive`

Il metodo è un implementazione del metodo di ricezione messaggi definito in *Akka*. Gestisce i messaggi provenienti dall'attore TCP della libreria. In particolare gestisce i seguenti messaggi:

- `Bound messages` - effettua il log sullo stato della porta
- `CommandFailed` - l'attore "uccide" se stesso nel caso ricevesse questo messaggio
- `Connected messages` - crea un `Usermanager` per ogni connessione

Lista parametri del metodo:

Nessuno.

3.20 Actorbase.server.actors.Usermanager

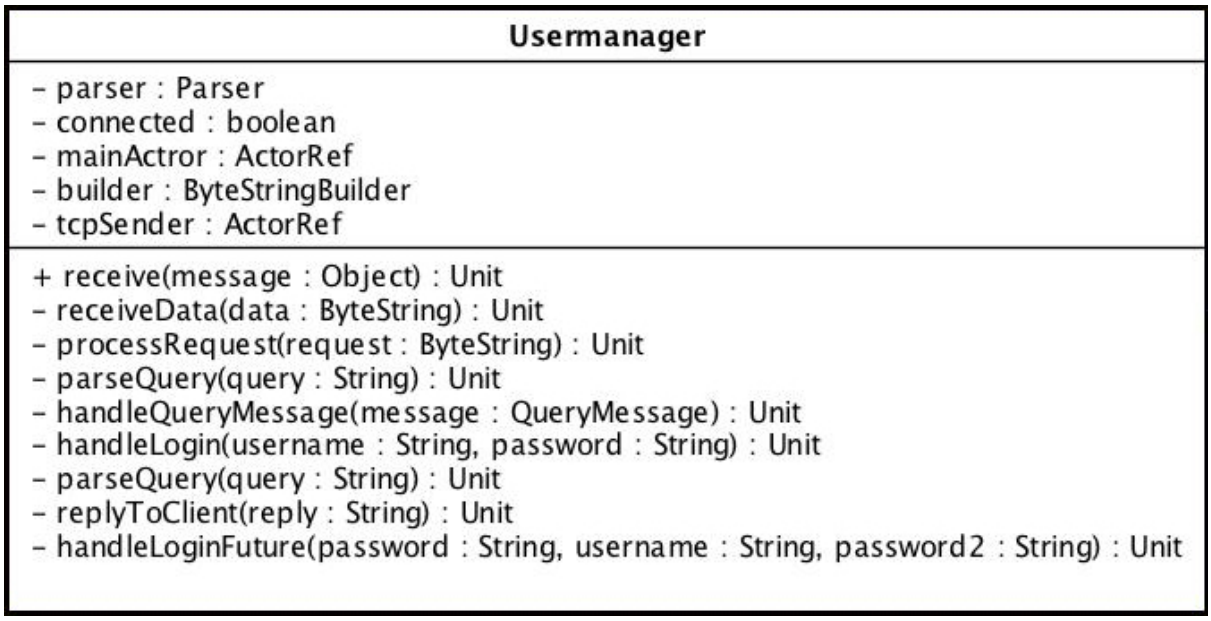


Figura 15: Componente Actorbase.server.actors.Usermanager

Descrizione

Classe che definisce l'attore di tipo `Usermanager`. Tale attore gestisce le richieste TCP provenienti da uno specifico client: si occupa di comprendere il contenuto delle query, di inoltrare le richieste e di fornire le risposte al client.

Utilizzo

Viene utilizzato gestire una singola connessione al server.

Classi ereditate

- `Actorbase.server.actors.ReplyActor`

Ereditata da

Nessuno.

Attributi

- `private parser: Parser` - Parser per effettuare l'elaborazione delle richieste utente.
- `private connected: Boolean` - Booleano per controllare lo stato della connessione.
- `private mainActor: ActorRef` - Riferimento all'attore di tipo `Main` per la connessione gestita.
- `private builder: ByteStringBuilder` - Costruttore di stringhe a partire da Byte.
- `private tcpSender: ActorRef` - Riferimento all'attore di tipo `TCP`.

Costruttore: `Usermanager()`

Costruisce un attore di tipo `Usermanager` senza parametri.

Lista parametri del metodo:

Nessuno.

Metodo: `receive`

Il metodo è un implementazione del metodo di ricezione messaggi definito in *Akka*. Gestisce i pacchetti inviati dall'attore `TCP`, li salva in un buffer, effettua il parsing di essi e inoltra il risultato all'attore di tipo `Main`.

- `Received` - gestisce la ricezione di un pacchetto invocando il metodo `receiveData`.
- `PeerClosed` - gestisce la disconnessione del client.

Lista parametri del metodo:

Nessuno.

Metodo: `private receiveData(data: ByteString): Unit`

Effettua il buffer dei Byte provenienti dal client e controlla che il messaggio sia nella forma corretta.

Lista parametri del metodo:

- `data: ByteString` - I Byte provenienti dal client.

Metodo: `private processRequest(request: ByteString): Unit`

Processa i Byte ricevuti nel metodo `receiveData` comprendendo il tipo di richiesta del client. Genera il corrispondente messaggio utilizzando il `Parser` e lo inoltra di conseguenza.

Lista parametri del metodo:

- `request: ByteString` - Richiesta del client.

Metodo: `private handleQueryMessage(message: QueryMessage): Unit`

Gestisce un messaggio di tipo `QueryMessage` prodotto dal metodo `processRequest`. Nel caso si tratti di un `LoginMessage` gestisce personalmente la richiesta, altrimenti inoltra il messaggio all'attore `Main`.

Lista parametri del metodo:

- `message: QueryMessage` - Il messaggio da gestire.

Metodo: `private handleLogin(username: String, password: String): Unit`

Effettua l'operazione di login per il client, nel caso quest'ultimo non fosse già autenticato. Si occupa di controllare la correttezza dei dati di login (username e password) rispetto alla lista di utenti che hanno accesso al server. Infine comunica al client l'esito dell'operazione.

Lista parametri del metodo:

- `username: String` - L'username dell'utente.
- `password: String` - La password dell'utente.

Metodo: `private replyToClient(reply: String): Unit`

Invia il `ReplyMessage` al mittente originario (l'attore TCP).

Lista parametri del metodo:

- `reply: String` - La stringa da inviare come risposta.

Metodo: `private handleLoginFuture(psw: String, username : String, password : String): Unit`

Implementa nel dettaglio la gestione del login differenziando la gestione di utenti normali da quella di un utente amministratore. Inoltre si occupa di generare la risposta per il client nel caso di login fallito.

Lista parametri del metodo:

- `psw: String` - La password da gestire.
- `password: String` - La password dell'utente.
- `username: String` - L'username dell'utente.

3.21 Actorbase.server.actors.Main

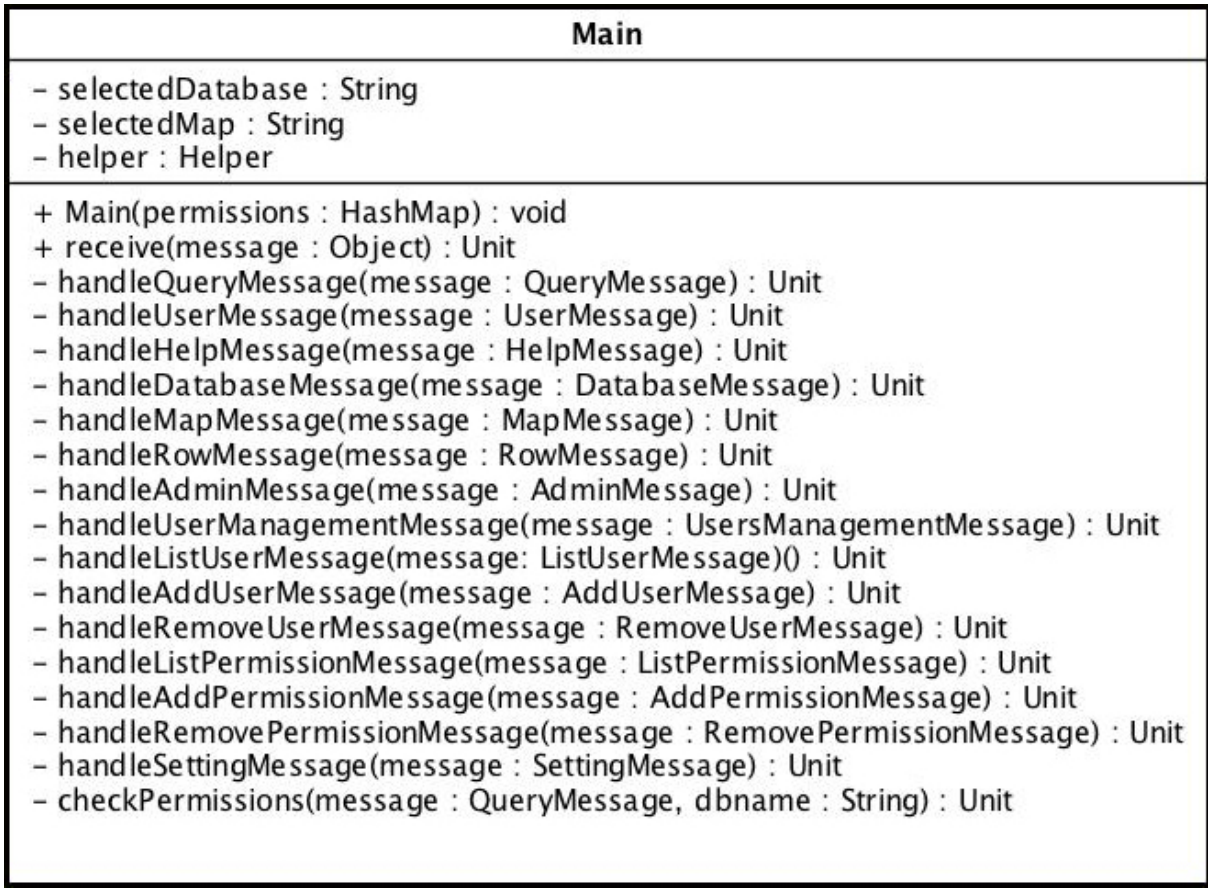


Figura 16: Componente Actorbase.server.actors.Main

Descrizione

Classe che definisce l'attore di tipo `Main`. Tale attore si occupa di eseguire le richieste effettuate da un client. Processa autonomamente le query a livello database e le query amministratore, per tutte le altre query si occupa di inoltrarle all'attore appropriato. É l'unico attore che interagisce con l'attore di tipo `Usermanager`, tutte le risposte generate vengono inviate ad esso.

Utilizzo

Viene utilizzato eseguire le richieste utente ed ottenere le risposte.

Classi ereditate

- `Actorbase.server.actors.ReplyActor`

Ereditata da

Nessuno.

Attributi

- `private helper: Helper` - istanza della classe `Helper` per gestire le richieste di aiuto.
- `private selectedDatabase: String` - stringa che rappresenta il database selezionato dal client.
- `private selectedMap: String` - stringa che rappresenta la mappa selezionata dal client.

Costruttore: `Main(perms: util.HashMap[String, UserPermission] = null)`

Costruisce un attore di tipo `Main` a partire da una mappa di permessi.

Lista parametri del metodo:

- `perms: util.HashMap[String, UserPermission]` - la mappa di permessi.

Metodo: `receive`

Il metodo è un implementazione del metodo di ricezione messaggi definito in *Akka*. Gestisce solo messaggi di tipo `QueryMessage`.

Lista parametri del metodo:

Nessuno.

Metodo: `private handleQueryMessage(message: QueryMessage): Unit`

Processa messaggi di tipo `QueryMessage`. Si occupa di differenziare tra messaggi `UserMessage` e `AdminMessage` chiamando per essi il metodo corretto. Gestisce i seguenti tipi di messaggi:

- `UserMessage`
- `AdminMessage`

Lista parametri del metodo:

- `message: QueryMessage` - Il messaggio da processare.

Metodo: `private handleUserMessage(message: UserMessage): Unit`

Processa messaggi di tipo `UserMessage`. Si occupa di differenziare tra messaggi chiamando per essi il metodo corretto. Gestisce i seguenti tipi di messaggi:

- `HelpMessage`
- `DatabaseMessage`
- `MapMessage`
- `RowMessage`

Lista parametri del metodo:

- `message: UserMessage` - Il messaggio da processare.

Metodo: `private handleAdminMessage(message: AdminMessage): Unit`

Processa messaggi di tipo `AdminMessage`. Si occupa di differenziare tra messaggi chiamando per essi il metodo corretto. Gestisce i seguenti tipi di messaggi:

- `UsersManagementMessage`
- `PermissionsManagementMessage`
- `SettingMessage`

Lista parametri del metodo:

- `message: AdminMessage` - Il messaggio da processare.

Metodo: `private handleUserManagementMessage(message: UsersManagementMessage): Unit`

Processa messaggi di tipo `UsersManagementMessage`. Si occupa di differenziare tra messaggi chiamando per essi il metodo corretto. Gestisce i seguenti tipi di messaggi:

- `ListUserMessage`
- `AddUserMessage`
- `RemoveUserMessage`

Lista parametri del metodo:

- `message: UsersManagementMessage` - Il messaggio da processare.

Metodo: `private handlePermissionsManagementMessage(message: PermissionsManagementMessage): Unit`

Processa messaggi di tipo `PermissionsManagementMessage`. Si occupa di differenziare tra messaggi chiamando per essi il metodo corretto. Gestisce i seguenti tipi di messaggi:

- `ListPermissionMessage`
- `AddPermissionMessage`
- `RemovePermissionMessage`

Lista parametri del metodo:

- `message: PermissionsManagementMessage` - Il messaggio da processare.

Metodo: `private handleSettingMessage(message: SettingMessage): Unit`

Processa messaggi di tipo `SettingMessage`. Si occupa di differenziare tra messaggi chiamando per essi il metodo corretto. Gestisce i seguenti tipi di messaggi:

- `RefreshSettingsMessage`

Lista parametri del metodo:

- `message: SettingMessage` - Il messaggio da processare.

Metodo: `private handleHelpMessage(message: HelpMessage): Unit`

Processa messaggi di tipo `HelpMessage`. Si occupa di elaborare una richiesta definita da un messaggio di help. Gestisce i seguenti tipi di messaggi:

- `CompleteHelpMessage` - risponde al messaggio generando una risposta di aiuto completo con l'utilizzo dell'istanza di `Helper`.
- `SpecificHelpMessage` - risponde al messaggio generando una risposta di aiuto per il comando specifico con l'utilizzo dell'istanza di `Helper`.

Lista parametri del metodo:

- `message: HelpMessage` - Il messaggio da processare.

Metodo: `private handleDatabaseMessage(message: DatabaseMessage): Unit`

Processa messaggi di tipo `DatabaseMessage`. Si occupa di elaborare una richiesta a livello database. Gestisce i seguenti tipi di messaggi:

- `ListDatabaseMessage` - risponde al messaggio generando la lista dei database a cui il client ha accesso (almeno permessi di lettura).
- `SelectDatabaseMessage` - seleziona il database richiesto, salvandolo in `selectedDatabase`.
- `CreateDatabaseMessage` - crea un nuovo attore di tipo `MapManager` che rappresenti il database da creare. Gestisce anche il caso in cui il database da creare sia già presente.
- `DeleteDatabaseMessage` - rimuove il database richiesto rimuovendo l'attore `MapManager` che lo rappresenta.

Lista parametri del metodo:

- `message: DatabaseMessage` - Il messaggio da processare.

Metodo: `private handleMapMessage(message: MapMessage): Unit`

Processa messaggi di tipo `MapMessage`. Si occupa di elaborare una richiesta a livello mappa. Gestisce i seguenti tipi di messaggi:

- `SelectMapMessage` - seleziona la mappa richiesta salvando il nome in `selectedMap`. Si occupa di richiederne l'esistenza al `MapManager`.
- `MapMessage` - tutti gli altri `MapMessage` sono inoltrati al corretto MapManager.

Lista parametri del metodo:

- `message: MapMessage` - Il messaggio da processare.

Metodo: `private handleRowMessage(message: RowMessage): Unit`

Processa messaggi di tipo `RowMessage`. Si occupa di elaborare una richiesta a livello item. Controlla che vi siano un database e una mappa selezionati, in tal caso inoltra la richiesta al corretto `MapManager`.

Lista parametri del metodo:

- `message: RowMessage` - Il messaggio da processare.

Metodo: `private checkPermissions(message: QueryMessage, dbName: String): Boolean`

Questo metodo controlla che l'utente abbia i permessi necessari ad eseguire la query. Nel caso l'utente fosse amministratore egli dispone di tutti i permessi automaticamente, altrimenti vengono controllati i permessi utenti. Nel caso i permessi risultino sufficienti ad effettuare la query il metodo ritorna `true`, altrimenti ritorna `false`.

Lista parametri del metodo:

- `message: QueryMessage` - Il messaggio contenente la query utente.
- `dbName: String` - Il database selezionato dall'utente.

Metodo: `private handlePermissionsList(message: ListPermissionMessage): Unit`

Il metodo gestisce i messaggi di richiesta della lista dei permessi degli utenti.

Lista parametri del metodo:

- `message: ListPermissionMessage` - Il messaggio da processare.

Metodo: `private handleAddPermission(message: AddPermissionMessage): Unit`

Il metodo gestisce i messaggi di aggiunta alla lista dei permessi degli utenti.

Lista parametri del metodo:

- `message: AddPermissionMessage` - Il messaggio da processare.

Metodo: `private handleRemovePermissions(message: RemovePermissionMessage): Unit`

Il metodo gestisce i messaggi di rimozione dalla lista dei permessi degli utenti.

Lista parametri del metodo:

- `message: RemovePermissionMessage` - Il messaggio da processare.

Metodo: `private handleListUserMessage(message: ListUserMessage): Unit`

Il metodo gestisce i messaggi di richiesta della lista degli utenti.

Lista parametri del metodo:

- `message: ListUserMessage` - Il messaggio da processare.

Metodo: `private handleAddUser(message: AddUserMessage, username: String, password : String): Unit`

Il metodo gestisce i messaggi di aggiunta alla lista degli utenti.

Lista parametri del metodo:

- `message: AddUserMessage` - Il messaggio da processare.
- `username: String` - L'username dell'utente da aggiungere.
- `password : String` -La password dell'utente da aggiungere.

Metodo: `private handleRemoveUser(message: RemoveUserMessage, username: String): Unit`

Il metodo gestisce i messaggi di rimozione dalla lista degli utenti.

Lista parametri del metodo:

- `message: RemoveUserMessage` - Il messaggio da processare.
- `username: String` - L'username dell'utente da rimuovere.

3.22 Actorbase.server.actors.MapManager

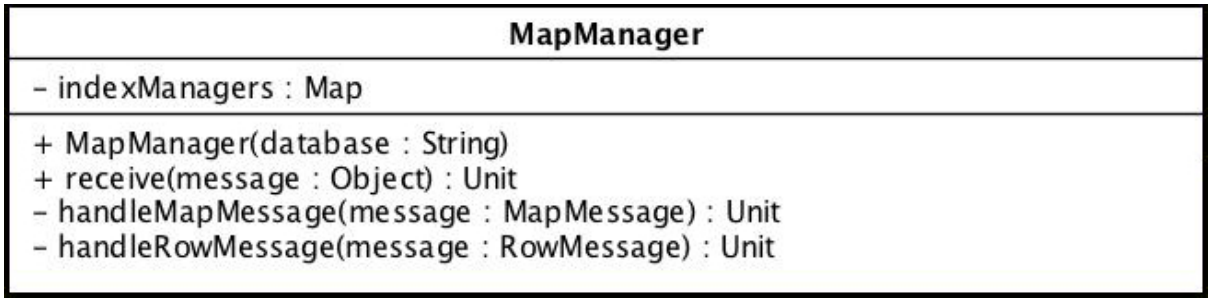


Figura 17: Componente Actorbase.server.actors.MapManager

Descrizione

Classe che definisce l'attore di tipo `MapManager`. Questo tipo di attore rappresenta un singolo database di *Actorbase*, gestisce le diverse mappe che lo compongono (attori `IndexManager`).

Utilizzo

Gestisce ad alto livello tutti i dati che compongono un database, attori di tipo `Main` inoltrano a lui le richieste per il database che rappresenta.

Classi ereditate

- `Actorbase.server.actors.ReplyActor`

Ereditata da

Nessuno.

Attributi

- `var database: String` - Il nome del database che l'attore rappresenta.
- `val indexManagers: ConcurrentHashMap[String, ActorRef]` - La mappa contenente i nomi e i riferimenti alle mappe del database.

Costruttore: `MapManager(database: String)`

Costruisce un attore di tipo `MapManager`. Alla creazione un attore di questo tipo deve registrarsi alla lista di database presente in `StaticSettings`.

Lista parametri del metodo:

- `database: String` - Il nome del database che l'attore rappresenta.

Metodo: `receive`

Il metodo è un implementazione del metodo di ricezione messaggi definito in *Akka*. Gestisce i seguenti messaggi:

- `AskMapMessage` - Ricerca la mappa in `indexManagers` e risponde.
- `MapMessage` - Chiama il metodo `handleMapMessage`.
- `RowMessage` - Chiama il metodo `handleRowMessage`.

Lista parametri del metodo:

Nessuno.

Metodo: `private handleMapMessage(message: MapMessage): Unit`

Processa messaggi di tipo `MapMessage`. Gestisce i seguenti tipi di messaggi:

- `ListMapMessage` - Crea e risponde con la lista di mappe che compongono il database.
- `CreateMapMessage` - Crea un nuovo `IndexManager` rappresentante la mappa richiesta e lo aggiunge alla propria lista se la mappa non è già presente.
- `DeleteMapMessage` - Elimina la mappa richiesta (se presente) eliminando l'attore `IndexManager` che la rappresenta.

Lista parametri del metodo:

- `message: MapMessage` - Il messaggio da processare.

Metodo: `private handleRowMessage(message: RowMessage): Unit`

Processa messaggi di tipo `RowMessage`. Trova il corretto `IndexManager` a cui inoltrare il messaggio.

Lista parametri del metodo:

- `message: RowMessage` - Il messaggio da inoltrare.

3.23 Actorbase.server.actors.IndexManager

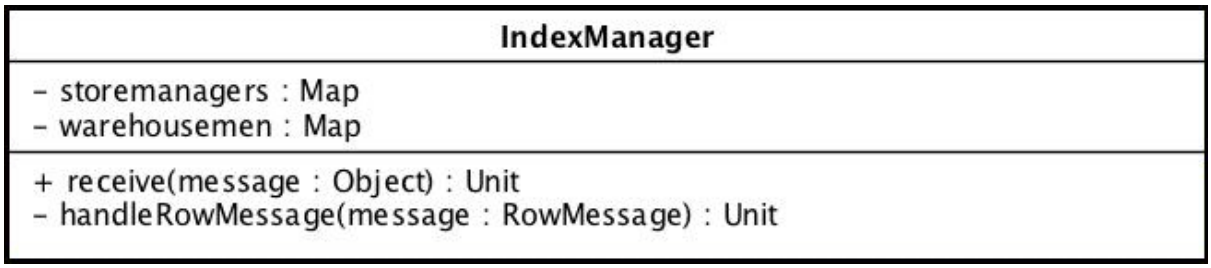


Figura 18: Componente Actorbase.server.actors.IndexManager

Descrizione

Classe che definisce l'attore di tipo `IndexManager`. Questo tipo di attore rappresenta una singola mappa di *Actorbase*. Gestisce i dati che compongono la mappa sia in memoria principale (RAM) che su disco, utilizzando attori di tipo `Storemanager` e `Warehouseman`.

Utilizzo

Gestisce i dati che compongono la mappa sia in memoria principale (RAM) che su disco, utilizzando attori di tipo `Storemanager` e `Warehouseman`. Riceve le richieste da attori di tipo `MapManager`.

Classi ereditate

- `Actorbase.server.actors.ReplyActor`

Ereditata da

Nessuno.

Attributi

- `val storemanager: ActorRef` - Il riferimento al primo `Storemanager` dell'albero.
- `val warehousemen: Array[ActorRef]` - Il riferimento ai `Warhouseman` che gestiscono la mappa su disco.

Costruttore: `IndexManager()`

Costruisce un attore di tipo `IndexManager`. Vengono inizializzati i riferimenti a `Storemanager` e `Warehouseman`.

Lista parametri del metodo:

Nessuno.

Metodo: `receive`

Il metodo è un implementazione del metodo di ricezione messaggi definito in *Akka*. Gestisce i seguenti messaggi:

- `RowMessage` - Chiama il metodo `handleRowMessage`.

Lista parametri del metodo:

Nessuno.

Metodo: `private handleRowMessage(message: RowMessage): Unit`

Processa messaggi di tipo `RowMessage`. Inoltra il messaggio all'albero di `Storemanager` e ai `Warehouseman`.

Lista parametri del metodo:

- `message: RowMessage` - Il messaggio da inoltrare.

3.24 Actorbase.server.actors.Storemanager

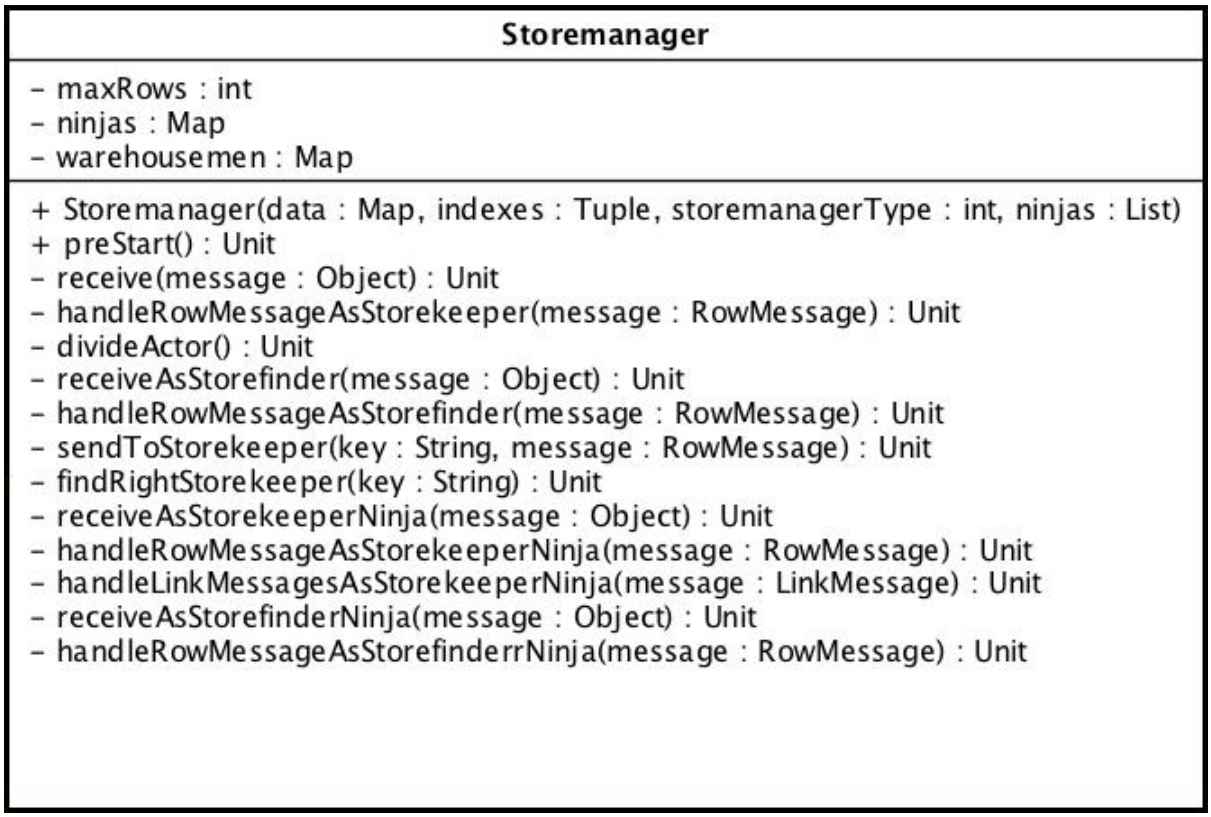


Figura 19: Componente Actorbase.server.actors.Storemanager

Descrizione

Classe che definisce l'attore di tipo `Storemanager`. Questo tipo di attore si occupa di mantenere i dati in memoria principale secondo una struttura gerarchica. Uno `Storemanager` può avere quattro tipologie di comportamento differenti:

- `Storekeeper`
- `StorekeeperNinja`
- `Storefinder`
- `StorefinderNinja`

Alla creazione dell'attore è possibile impostare il comportamento attraverso un parametro.

Utilizzo

Viene utilizzato da un attore `Indexmanager` per gestire i dati in memoria principale.

Classi ereditate

- `Actorbase.server.actors.ReplyActor`

Ereditata da

Nessuno.

Attributi

- `private var map: ConcurrentHashMap[String, Array[Byte]]` - Mappa contenente gli item.
- `private var index: (String, String)` - Indice dei dati contenuti nell'attore (utilizzato per trovare l'attore corretto).
- `private var storemanagerType: StoremanagerType` - Tipo di comportamento.
- `private var ninjas: Array[ActorRef]` - Riferimento ai propri attori `Ninja`.

Costruttore: `Storemanager(var map: ConcurrentHashMap[String, Array[Byte]], index: (String, String), storemanagerType: StoremanagerType, ninjas: Array[ActorRef]=null)`

Costruisce un attore di tipo `Storemanager`.

Lista parametri del metodo:

- `map: ConcurrentHashMap[String, Array[Byte]]` - la mappa di item che l'attore dovrà gestire.
- `index: (String, String)` - gli indici che identificano il range di valori gestiti dall'attore.
- `storemanagerType: StoremanagerType` - il tipo di comportamento che l'attore deve avere.
- `ninjas: Array[ActorRef]=null` - i riferimenti ai `Ninja` dell'attore (può essere `null` nel caso in cui si stia creando un `Ninja`).

Metodo: `override def preStart(): Unit`

Override del metodo `preStart()` di `akka.actor.Actor`. Il metodo effettua un controllo sul tipo di comportamento passato nel costruttore e invoca correttamente il metodo `become` di *Akka* per cambiare il comportamento del metodo di ricezione messaggi (`receive`).

Lista parametri del metodo:

Nessuno.

Metodo: `receive`

Il metodo è un implementazione del metodo di ricezione messaggi definito in *Akka*. Di default rappresenta la gestione dei messaggi come `Storekeeper`, riconosce messaggi di tipo `RowMessage`, invocando il metodo `handleRowMessageAsStorekeeper` per la gestione vera e propria.

Lista parametri del metodo:

Nessuno.

Metodo: `private handleRowMessageAsStorekeeper(message: RowMessage): Unit`

Processa messaggi di tipo `RowMessage` quando l'attore si comporta come `Storekeeper`. Riconosce il messaggio e gestisce la richiesta completamente, producendo una risposta. Gestisce i seguenti tipi di messaggi:

- `InsertRowMessage` - Inserisce una riga nella mappa se c'è spazio, altrimenti richiede la propria divisione.
- `UpdateRowMessage` - Aggiorna il valore della riga richiesta nella propria mappa.
- `RemoveRowMessage` - Rimuove la riga richiesta dalla propria mappa.
- `FindRowMessage` - Restituisce il valore della riga contenente la chiave richiesta.
- `ListKeysMessage` - Restituisce la lista di tutte le chiavi che compongono la sua mappa.

Lista parametri del metodo:

- `message: RowMessage` - Il messaggio da processare.

Metodo: `private divideActor() : Unit`

Il metodo effettua la divisione dell'attore in due quando si raggiunge il numero massimo di item contenuti in esso. La divisione si effettua creando due `Storemanager` figli con comportamento da `Storekeeper` a cui si passa metà della mappa. Una volta creati i figli l'attore svuota la propria mappa e inizia a comportarsi da `Storefinder`.

Lista parametri del metodo:

Nessuno.

Metodo: `private receiveAsStoreFinder: Receive`

Metodo di ricezione dei messaggi utilizzato quando il comportamento dell'attore è `Storefinder`. Riconosce messaggi di tipo `RowMessage`, e passa la gestione di essi al metodo `handleRowMessageAsStorefinder`.

Lista parametri del metodo:

Nessuno.

Metodo: `private handleRowMessageAsStorefinder(message: RowMessage) : Unit`

Processa messaggi di tipo `RowMessage` quando l'attore si comporta come `Storefinder`. Riconosce il messaggio e inoltra la richiesta ai figli. Gestisce i seguenti tipi di messaggi:

- `InsertRowMessage` - Chiama il metodo `sendToStorekeeper`.
- `UpdateRowMessage` - Chiama il metodo `sendToStorekeeper`.
- `RemoveRowMessage` - Chiama il metodo `sendToStorekeeper`.
- `FindRowMessage` - Chiama il metodo `sendToStorekeeper`.
- `ListKeysMessage` - Inoltra la richiesta ai figli e costruisce la lista completa delle chiavi unificando le informazioni ricevute dai figli.

Lista parametri del metodo:

- `message: RowMessage` - Il messaggio da processare.

Metodo: `private sendToStorekeeper(key: String, message: RowMessage): Unit`

Il metodo si occupa di trovare inoltrare al figlio corretto il messaggio.

Lista parametri del metodo:

- `key: String` - La chiave della richiesta da inoltrare.
- `message: RowMessage` - Il messaggio da inoltrare.

Metodo: `private findRightStorekeeper(key:String): Child`

Il metodo si occupa di trovare il figlio corretto confrontando la chiave con gli indici dei figli.

Lista parametri del metodo:

- `key: String` - La chiave della richiesta da inoltrare.

Metodo: `private receiveAsStorekeeperNinja: Receive`

Metodo di ricezione dei messaggi utilizzato quando il comportamento dell'attore è `StorekeeperNinja`. Gestisce i seguenti messaggi:

- `RowMessage` - Chiama il metodo `handleRowMessagesAsStorekeeperNinja`.
- `LinkMessage` - Chiama il metodo `handleLinkMessagesAsStorekeeperNinja`.

Lista parametri del metodo:

Nessuno.

Metodo: `private handleRowMessagesAsStorekeeperNinja(message: RowMessage): Unit`

Il comportamento del metodo è simile a quello di `handleRowMessageAsStorekeeper`, con la differenza che un Ninja si occupa solo di tenere i dati aggiornati dunque non vengono generate risposte alle richieste.

Lista parametri del metodo:

- `message: RowMessage` - Il messaggio da processare.

Metodo: `private handleLinkMessagesAsStorekeeperNinja(message: LinkMessage): Unit`

Gestisce la ricezione di messaggi di tipo `LinkMessage`. Nello specifico il metodo deve gestire un messaggio di tipo `BecomeStorefinderNinjaMessage` che modifica il comportamento dell'attore in `StorefinderNinja`.

Lista parametri del metodo:

- `message: LinkMessage` - Il messaggio da processare.

Metodo: `private receiveAsStorefinderNinja: Receive`

Metodo di ricezione dei messaggi utilizzato quando il comportamento dell'attore è `StorefinderNinja`. Gestisce i seguenti messaggi:

- `RowMessage` - Chiama il metodo `handleRowMessagesAsStorefinderNinja`.

Lista parametri del metodo:

Nessuno.

Metodo: `private handleRowMessagesAsStorefinderNinja(message: RowMessage): Unit`

Poiché uno `StrefinderNinja` è una semplice copia dello `Storefinder` originale, con cui condivide i figli, non è necessaria alcuna operazione alla ricezione di un messaggio di tipo `RowMessage`.

Lista parametri del metodo:

- `message: RowMessage` - Il messaggio da processare.

3.25 Actorbase.server.actors.ReplyActor (trait)

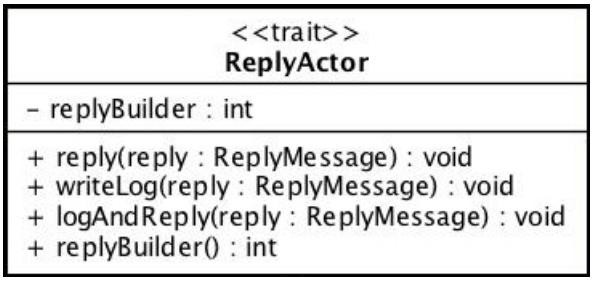


Figura 20: Componente Actorbase.server.actors.ReplyActor

Descrizione

Trait che definisce le funzionalità di risposta e log di un attore.

Utilizzo

Viene esteso dagli attori che devono effettuare risposte strutturate e che vogliono eseguire il log delle proprie operazioni.

Classi ereditate

- `Actorbase.server.actors.ClusterAwareActor`
- `akka.actor.ActorLogging`

Ereditata da

- `Actorbase.server.actors.Usermanager`
- `Actorbase.server.actors.Main`
- `Actorbase.server.actors.MapManager`
- `Actorbase.server.actors.IndexManager`
- `Actorbase.server.actors.Storemanager`
- `Actorbase.server.actors.Warehouseman`

Attributi

- `val replyBuilder: ReplyBuilder` - Il costruttore di risposte.

Metodo: `def logAndReply(reply: ReplyMessage, sender: ActorRef = sender): Unit`

Effettua il log dell'operazione rappresentata dal `ReplyMessage` utilizzando il metodo `writeLog` e invia il messaggio al `sender` utilizzando il metodo `reply`.

Lista parametri del metodo:

- `reply: ReplyMessage` - Il messaggio di cui effettuare il log.
- `sender: ActorRef = sender` - Il sender a cui inoltrare il messaggio.

Metodo: `def reply(reply: ReplyMessage, sender: ActorRef = sender): Unit`

Invia il messaggio al `sender`.

Lista parametri del metodo:

- `reply: ReplyMessage` - Il messaggio di cui effettuare il log.
- `sender: ActorRef = sender` - Il sender a cui inoltrare il messaggio.

Metodo: `def writeLog(reply: ReplyMessage): Unit`

Effettua il log dell'operazione definita dal messaggio.

Lista parametri del metodo:

- `reply: ReplyMessage` - Il messaggio di cui effettuare il log.

Metodo: `def currentMethodName() : String`

Ritorna il nome del metodo attualmente in esecuzione.

Lista parametri del metodo:

Nessuno.

3.26 Actorbase.server.actors.ClusterAwareActor (trait)

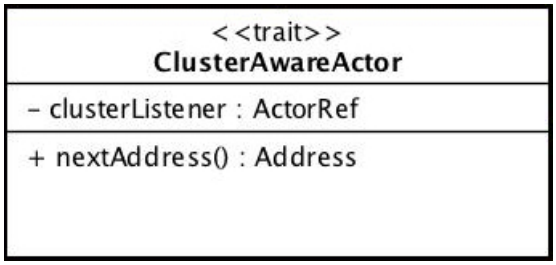


Figura 21: Componente Actorbase.server.actors.ClusterAwareActor

Descrizione

Trait che definisce un attore che si interfaccia con il cluster.

Utilizzo

Fornisce ad un attore il metodo `nextAddress`, dovrebbe essere esteso da tutti gli attori che necessitano di creare attori in altri nodi del cluster. La politica di selezione degli indirizzi è responsabilità del `ClusterListener` del nodo.

Classi ereditate

- `akka.actor.Actor`

Ereditata da

- `Actorbase.server.actors.ReplyActor`

Attributi

- `implicit val timeout: Timeout` - Timeout per le futures.
- `var clusterListener: ActorRef` - Istanza del cluster listener.

Metodo: `def nextAddress: Address`

Ritorna un indirizzo di un nodo del cluster. Questo metodo invia un messaggio al `ClusterListener` dello stesso nodo di questo attore.

Lista parametri del metodo:

Nessuno.

3.27 Actorbase.server.actors.Warehouseman

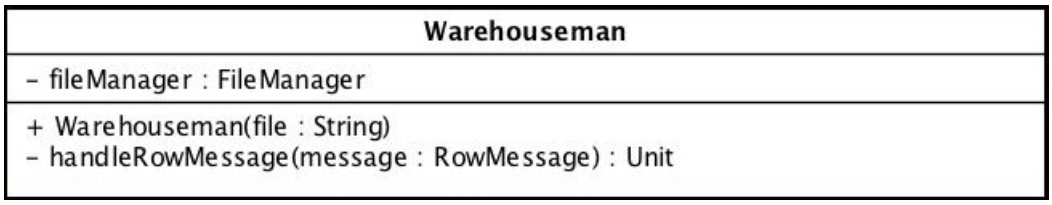


Figura 22: Componente Actorbase.server.actors.Warehouseman

3.28 Actorbase.server.enums

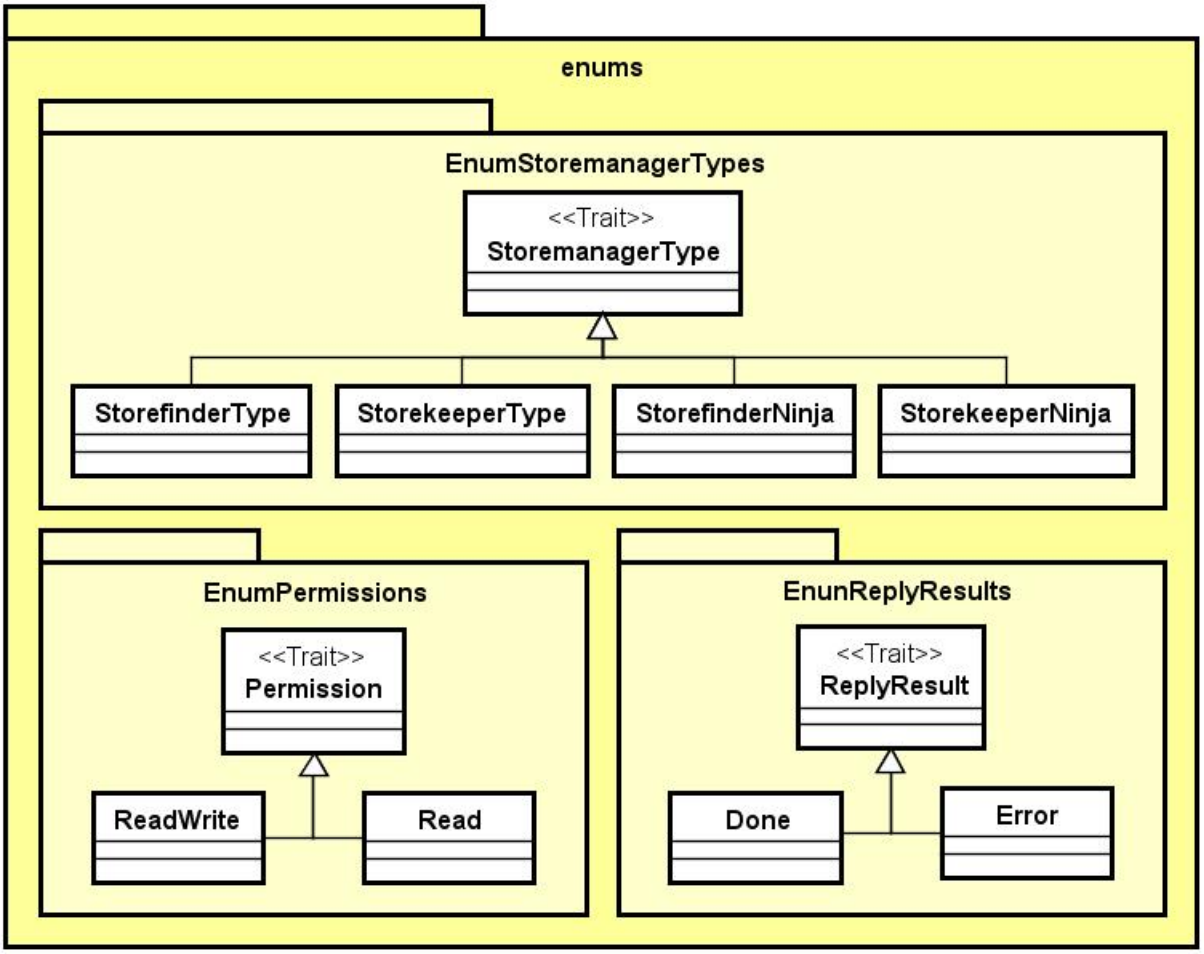


Figura 23: Actorbase.server.enums

Package contenente tutte le classi che rappresentano un'enumerazione. Le classi definite in questo package servono come enumerazioni di alcune proprietà di *Actorbase*.

3.29 Actorbase.server.enums.EnumPermission (object)

Descrizione

Rappresenta un'enumerazione dei permessi utente.

Utilizzo

Viene utilizzata per rappresentare i permessi di un utente.

Classi ereditate

Nessuna.

Ereditata da

Nessuna.

Attributi

- `val permissionsType: Seq` - Lista dei permessi

Metodi

Nessuno.

3.30 Actorbase.server.enums.UserPermission (trait)

Descrizione

Trait che le classi che rappresentano permessi utente devono estendere.

Utilizzo

Fornire un'interfaccia base per le classi che definiscono permessi utente.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.enums.EnumPermission.Read`
- `Actorbase.server.enums.EnumPermission.Write`

Attributi

Nessuno.

Metodi

Nessuno.

3.31 Actorbase.server.enums.EnumPermission.Read

Descrizione

Tipo che rappresenta i permessi di lettura.

Utilizzo

Consente di definire un oggetto per i permessi di lettura. Viene utilizzato per definire operazioni che richiedono tali permessi.

Classi ereditate

- `Actorbase.server.enums.EnumPermission.UserPermission`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.32 Actorbase.server.enums.EnumPermission.Write

Descrizione

Tipo che rappresenta i permessi di scrittura.

Utilizzo

Consente di definire un oggetto per i permessi di scrittura. Viene utilizzato per definire operazioni che richiedono tali permessi.

Classi ereditate

- `Actorbase.server.enums.EnumPermission.UserPermission`

Ereditata da

Nessuno.
Attributi

Nessuno.

Metodi

Nessuno.

3.33 Actorbase.server.enums.EnumReplyResult (object)

Descrizione

Rappresenta un'enumerazione dei possibili risultati di un'operazione.

Utilizzo

Viene utilizzata per rispondere il risultato di un'operazione a chi l'ha richiesta.

Classi ereditate

Nessuna.

Ereditata da

Nessuna.

Attributi

- `val replyResultType: Seq` - lista dei risultati possibili.

Metodi

Nessuno.

3.34 Actorbase.server.enums.EnumReplyResult.ReplyResult (trait)

Descrizione

Trait che le classi che rappresentano il risultato di un'operazione devono estendere..

Utilizzo

Rispondere all'attore che ha richiesto un'operazione il risultato definendo un'insieme di possibili risposte.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.enums.EnumReplyResult.Done`
- `Actorbase.server.enums.EnumReplyResult.Error`

Attributi

Nessuno.

Metodi

Nessuno.

3.35 Actorbase.server.enums.EnumReplyResult.Done

Descrizione

Tipo che rappresenta un operazione avvenuta con successo.

Utilizzo

Segnalare che l'operazione è avvenuta con successo.

Classi ereditate

- `Actorbase.server.enums.EnumReplyResult.ReplyResult`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.36 Actorbase.server.enums.EnumReplyResult.Error

Descrizione

Tipo che rappresenta il risultato di un operazione completata in modo anomalo.

Utilizzo

Segnalare che un operazione è stata completata in modo anomalo o non è stata completata.

Classi ereditate

- `Actorbase.server.enums.EnumReplyResult.ReplyResult`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.37 Actorbase.server.enums.EnumStoremanagerType (object)

Descrizione

Rappresenta un enumerazione dei possibili comportamenti di uno `Storemanager`.

Utilizzo

Permette di definire l'insieme di comportamenti di uno `Storemanager`, in modo da poter impostare un comportamento alla creazione di un attore di tale tipo.

Classi ereditate

Nessuna.

Ereditata da

Nessuna.

Attributi

- `val storemanagerTypes: Seq` - lista dei comportamenti di uno `Storemanager`.

Metodi

Nessuno.

3.38 Actorbase.server.enums.EnumStoremanagerType.StoremanagerType (trait)

Descrizione

Trait che deve essere esteso dalle classi che definiscono il comportamento di uno `Storemanager`.

Utilizzo

Fornire una interfaccia base per i comportamenti dello `Storemanager`.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.enums.EnumStoremanagerType.StorefinderType`
- `Actorbase.server.enums.EnumStoremanagerType.StorekeeperType`
- `Actorbase.server.enums.EnumStoremanagerType.StorekeeperNinjaType`
- `Actorbase.server.enums.EnumStoremanagerType.StorefinderNinjaType`

Attributi

Nessuno.

Metodi

Nessuno.

3.39 Actorbase.server.enums.EnumStoremanagerType.StorefinderType

Descrizione

Classe che rappresenta il comportamento `Storefinder` da parte di un attore `Storemanager`.

Utilizzo

Definire il comportamento `Storefinder` per un attore `Storemanager`.

Classi ereditate

- `Actorbase.server.enums.EnumStoremanagerType.StoremanagerType`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.40 Actorbase.server.enums.EnumStoremanagerType.StorekeeperType

Descrizione

Classe che rappresenta il comportamento `Storekeeper` da parte di un attore `Storemanager`.

Utilizzo

Definire il comportamento `Storekeeper` per un attore `Storemanager`.

Classi ereditate

- `Actorbase.server.enums.EnumStoremanagerType.StoremanagerType`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.41 Actorbase.server.enums.EnumStoremanagerType.StorekeeperNinjaType

Descrizione

Classe che rappresenta il comportamento `StorekeeperNinja` da parte di un attore `Storemanager`.

Utilizzo

Definire il comportamento `StorekeeperNinja` per un attore `Storemanager`.

Classi ereditate

- `Actorbase.server.enums.EnumStoremanagerType.StoremanagerType`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.42 Actorbase.server.enums.EnumStoremanagerType.StorefinderNinjaType

Descrizione

Classe che rappresenta il comportamento `StorefinderNinja` da parte di un attore `Storemanager`.

Utilizzo

Definire il comportamento `StorefinderNinja` per un attore `Storemanager`.

Classi ereditate

- `Actorbase.server.enums.EnumStoremanagerType.StoremanagerType`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.43 Actorbase.server.messages

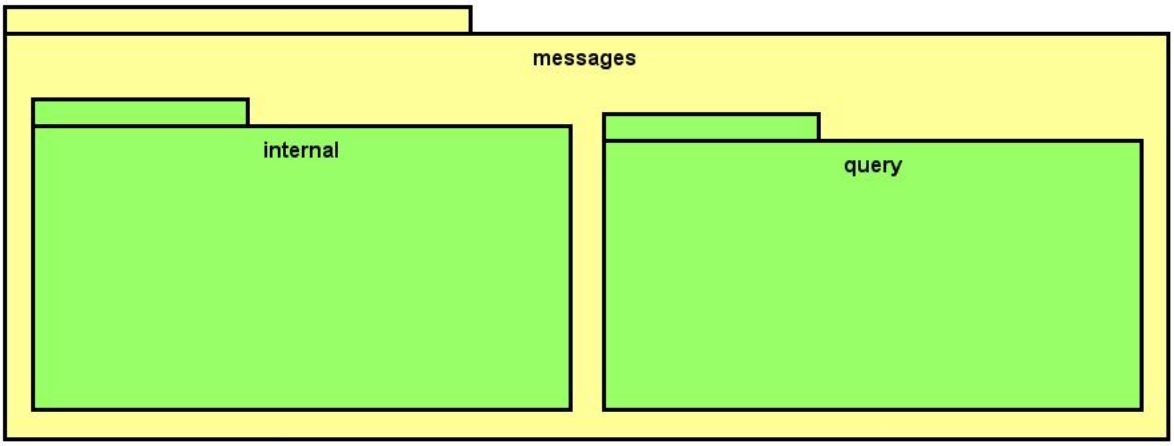


Figura 24: Componente Actorbase.server.messages

La componente messages di *Actorbase* è la raccolta di tutti i messaggi che vengono scambiati tra attori. Comprende sia i messaggi interni al sistema che i messaggi che rappresentano le richieste di un utente.

3.44 Actorbase.server.messages.internal

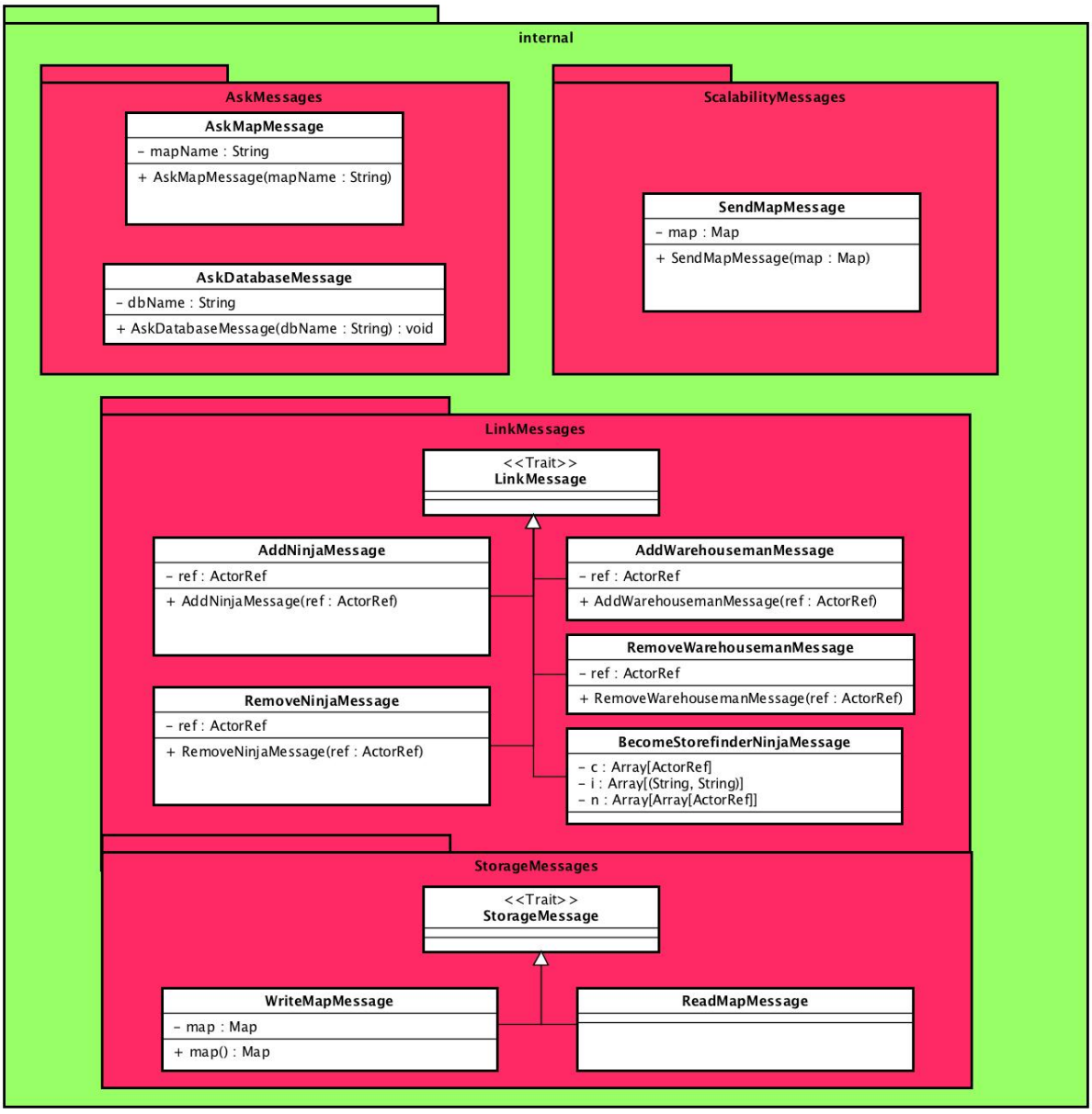


Figura 25: Componente Actorbase.server.messages.internal

La componente internal di *Actorbase* è la raccolta dei messaggi che vengono scambiati tra attori internamente al sistema, ovvero non sono collegati direttamente ad azione dell'utente. Svolgono attività di configurazione e influenzano il comportamento degli attori.

3.45 Actorbase.server.messages.internal.AskMessages (object)

Descrizione

Un `AskMessage` definisce una richiesta di controllo dell'esistenza di un elemento.

Utilizzo

Messaggi di questo tipo sono utilizzati per controllare se un elemento (un database, una mappa, ...)

è presente.

Classi ereditate

Nessuna.

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.46 Actorbase.server.messages.internal.AskMapMessage

Descrizione

Un `AskMapMessage` definisce una richiesta di controllo dell'esistenza di una mappa.

Utilizzo

Messaggi di questo tipo sono utilizzati per controllare se una mappa è presente.

Classi ereditate

Nessuna.

Ereditata da

Nessuno.

Attributi

- `val mapName: String` - Il nome della mappa di cui si vuole controllare l'esistenza.

Costruttore: `AskMapMessage(mapName:String)`

Costruisce un `AskMapMessage` a partire dalla stringa contenente il nome della mappa.

Lista parametri del metodo:

- `mapName: String` - Il nome della mappa.

3.47 Actorbase.server.messages.internal.AskDatabaseMessage

Descrizione

Un `AskDatabaseMessage` definisce una richiesta di controllo dell'esistenza di un database.

Utilizzo

Messaggi di questo tipo sono utilizzati per controllare se un database è presente.

Classi ereditate

Nessuna.

Ereditata da

Nessuno.

Attributi

- `val dbName: String` - Il nome del database di cui si vuole controllare l'esistenza.

Costruttore: `AskDatabaseMessage(dbName: String)`

Costruisce un `AskDatabaseMessage` a partire dalla stringa contenente il nome del database.

Lista parametri del metodo:

- `dbName: String` - Il nome del database.

3.48 Actorbase.server.messages.internal.LinkMessages

I LinkMessages sono i messaggi usati per gestire i collegamenti tra attori. Questo tipo di messaggi deve contenere il riferimento all'attore che deve essere aggiunto o eliminato dalla mappa degli attori conosciuti.

3.49 Actorbase.server.messages.internal.LinkMessages.LinkMessage (trait)

Descrizione

Trait che ogni messaggio che definisce operazioni di collegamento tra attori deve estendere.

Utilizzo

Viene utilizzato per fornire un'interfaccia comune per quanto riguarda la gestione di messaggi che riguardano il collegamento tra attori.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.messages.internal.LinkMessages.AddNinjaMessage`
- `Actorbase.server.messages.internal.LinkMessages.AddWarehousemanMessage`
- `Actorbase.server.messages.internal.LinkMessages.RemoveNinjaMessage`
- `Actorbase.server.messages.internal.LinkMessages.RemoveWarehousemanMessage`
- `Actorbase.server.messages.internal.LinkMessages.BecomeStorefinderNinjaMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.50 Actorbase.server.messages.internal.LinkMessages.AddNinjaMessage

Descrizione

Un `AddNinjaMessage` definisce una richiesta di aggiunta di un attore di tipo `Ninja`.

Utilizzo

Viene utilizzato per richiedere l'aggiunta di un attore di tipo `Ninja`.

Classi ereditate

- `Actorbase.server.messages.internal.LinkMessages.LinkMessage`

Ereditata da

Nessuno.

Attributi

- `val ref : ActorRef` - Il riferimento all'attore.

Costruttore: `AddNinjaMessage(ref : ActorRef)`

Costruisce un `AddNinjaMessage` a partire dal riferimento all'attore.

Lista parametri del metodo:

- `ref : ActorRef` - Il riferimento all'attore.

3.51 Actorbase.server.messages.internal.LinkMessages.AddWarehousemanMessage

Descrizione

Un `AddWarehousemanMessage` definisce una richiesta di aggiunta di un attore di tipo `Warehouseman`.

Utilizzo

Viene utilizzato per richiedere l'aggiunta di un attore di tipo `Warehouseman`.

Classi ereditate

- `Actorbase.server.messages.internal.LinkMessages.LinkMessage`

Ereditata da

Nessuno.

Attributi

- `val ref : ActorRef` - Il riferimento all'attore.

Costruttore: `AddWarehousemanMessage(ref : ActorRef)`

Costruisce un `AddWarehousemanMessage` a partire dal riferimento all'attore.

Lista parametri del metodo:

- `ref : ActorRef` - Il riferimento all'attore.

3.52 Actorbase.server.messages.internal.LinkMessages.RemoveNinjaMessage

Descrizione

Un `RemoveNinjaMessage` definisce una richiesta di rimozione di un attore di tipo `Ninja`.

Utilizzo

Viene utilizzato per richiedere la rimozione di un attore di tipo `Ninja`.

Classi ereditate

- `Actorbase.server.messages.internal.LinkMessages.LinkMessage`

Ereditata da

Nessuno.

Attributi

- `val ref : ActorRef` - Il riferimento all'attore.

Costruttore: `RemoveNinjaMessage(ref : ActorRef)`

Costruisce un `RemoveNinjaMessage` a partire dal riferimento all'attore.

Lista parametri del metodo:

- `ref : ActorRef` - Il riferimento all'attore.

3.53 Actorbase.server.messages.internal.LinkMessages.RemoveWarehousemanMessage

Descrizione

Un `RemoveWarehousemanMessage` definisce una richiesta di rimozione di un attore di tipo `Warehouseman`.

Utilizzo

Viene utilizzato per richiedere la rimozione di un attore di tipo `Warehouseman`.

Classi ereditate

- `Actorbase.server.messages.internal.LinkMessages.LinkMessage`

Ereditata da

Nessuno.

Attributi

- `val ref : ActorRef` - Il riferimento all'attore.

Costruttore: `RemoveWarehousemanMessage(ref : ActorRef)`

Costruisce un `RemoveWarehousemanMessage` a partire dal riferimento all'attore.

Lista parametri del metodo:

- `ref : ActorRef` - Il riferimento all'attore.

3.54 Actorbase.server.messages.internal.LinkMessages.BecomeStorefinderNinjaMessage

Descrizione

Un `BecomeStorefinderNinjaMessage` definisce una richiesta di cambiamento del comportamento di uno `Storemanager` in `StorefinderNinja`.

Utilizzo

Viene utilizzato da un attore `Storemanager` quando cambia comportamento e passa da `Storekeeper` a `Storefinder`, serve ad informare i suoi ninja del cambio.

Classi ereditate

- `Actorbase.server.messages.internal.LinkMessages.LinkMessage`

Ereditata da

Nessuno.

Attributi

- `val c : Array[ActorRef]` - I riferimenti agli attori figli dello `Storefinder` originale.
- `val i : Array[(String, String)]` - I riferimenti agli indici degli attori figli dello `Storefinder` originale.
- `val n : Array[Array[ActorRef]]` - I riferimenti ai ninja degli attori figli dello `Storefinder` originale.

Costruttore: `BecomeStorefinderNinjaMessage(c : Array[ActorRef], i : Array[(String, String)], n : Array[Array[ActorRef]])`

Costruisce un `BecomeStorefinderNinjaMessage` a partire dai dati rappresentanti i figli dello `Storefinder` originale.

Lista parametri del metodo:

- `c : Array[ActorRef]` - I riferimenti agli attori figli dello `Storefinder` originale.
- `i : Array[(String, String)]` - I riferimenti agli indici degli attori figli dello `Storefinder` originale.
- `n : Array[Array[ActorRef]]` - I riferimenti ai ninja degli attori figli dello `Storefinder` originale.

3.55 Actorbase.server.messages.internal.ScalabilityMessages

Gli ScalabilityMessages sono messaggi usati per gestire le proprietà di scalabilità del sistema.

3.56 Actorbase.server.messages.internal.ScalabilityMessages.ScalabilityMessage (trait)

Descrizione

Trait che ogni messaggio che definisce operazioni relative alla scalabilità del sistema deve estendere.

Utilizzo

Viene utilizzato per fornire un'interfaccia comune per quanto riguarda la gestione di messaggi che riguardano la scalabilità del sistema.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.messages.internal.ScalabilityMessages.SendMapMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.57 Actorbase.server.messages.internal.ScalabilityMessages.SendMapMessage

Descrizione

Messaggio che definisce una richiesta di aggiunta di una mappa ad un attore preesistente.

Utilizzo

Viene utilizzato per passare mappe tra attori.

Classi ereditate

- `Actorbase.server.messages.internal.ScalabilityMessages.ScalabilityMessage`

Ereditata da

Nessuno.

Attributi

- `val map: mutable.HashMap[String, Array[Byte]]` -La mappa che deve essere inviata.

Costruttore: `SendMapMessage (map: mutable.HashMap[String, Array[Byte]])`

Costruisce un `SendMapMessage` a partire dalla mappa che deve essere inviata.

Lista parametri del metodo:

- `map: mutable.HashMap[String, Array[Byte]]` -La mappa che deve essere inviata.

3.58 Actorbase.server.messages.internal.StorageMessages

Gli StorageMessages sono messaggi usati per effettuare operazioni relative alla gestione dei dati su file.

3.59 Actorbase.server.messages.internal.StorageMessages.StorageMessages (trait)

Descrizione

Trait che ogni messaggio che definisce operazioni relative alla gestione dei dati su disco deve estendere.

Utilizzo

Viene utilizzato per fornire un'interfaccia comune per quanto riguarda la gestione di messaggi che riguardano la gestione dei dati su disco

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.messages.internal.StorageMessages.WriteMapMessage`
- `Actorbase.server.messages.internal.StorageMessages.ReadMapMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.60 Actorbase.server.messages.internal.StorageMessages.WriteMapMessage

Descrizione

Messaggio che definisce una richiesta di scrittura di una mappa su disco.

Utilizzo

Viene utilizzato per scrivere una mappa su disco.

Classi ereditate

- `Actorbase.server.messages.internal.StorageMessages.StorageMessages`

Ereditata da

Nessuno.

Attributi

- `val map: mutable.HashMap[String, Array[Byte]]` -La mappa che deve essere scritta su disco.

Costruttore: `WriteMapMessage (map: HashMap[String, Array[Byte]])`

Costruisce un `WriteMapMessage` a partire dalla mappa che deve essere scritta.

Lista parametri del metodo:

- `map: mutable.HashMap[String, Array[Byte]]` -La mappa che deve essere scritta.

3.61 Actorbase.server.messages.internal.StorageMessages.ReadMapMessage

Descrizione

Messaggio che definisce una richiesta di lettura di una mappa da disco.

Utilizzo

Viene utilizzato per leggere una mappa da disco.

Classi ereditate

- `Actorbase.server.messages.internal.StorageMessages.StorageMessages`

Ereditata da

Nessuno.

Attributi

Nessuno.

Costruttore: `ReadMapMessage()`

Costruisce un `ReadMapMessage` senza parametri.

Nessuno.

3.62 Actorbase.server.messages.query

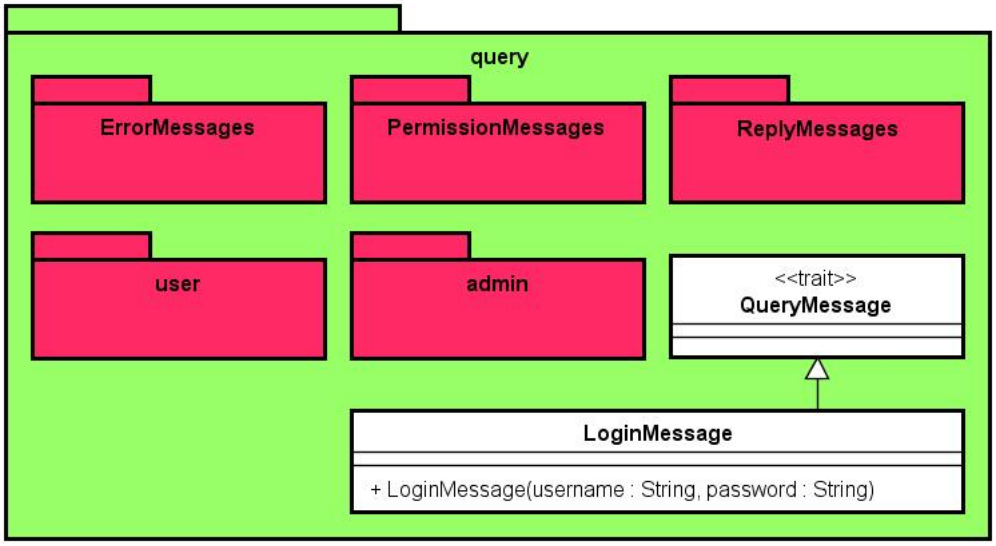


Figura 26: Componente Actorbase.server.messages.query

La componente query di *Actorbase* raccoglie tutti i messaggi che rappresentano richieste dirette di un utente e i messaggi di risposta a tali richieste. Comprende inoltre le richieste degli amministratori.

3.63 Actorbase.server.messages.query.QueryMessage (trait)

Descrizione

Interfaccia di base dei messaggi di tipo query.

Utilizzo

Questa interfaccia fornisce un tipo comune per tutti i messaggi che rappresentano una query ed è estesa dai messaggi concreti.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.messages.query.LoginMessage`

- `Actorbase.server.messages.query.user.UserMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.64 Actorbase.server.messages.query.ServiceErrorInfo

Descrizione

Classe che rappresenta una risposta di errore nel servizio.

Utilizzo

Questa classe viene utilizzata per comunicare un errore generale di servizio all'utente.

Classi ereditate

- `Actorbase.server.messages.ReplyErrorInfo`

Ereditata da

Nessuna.

Attributi

- `error: String` - Stringa che descrive l'errore da comunicare all'utente.

Costruttore: `ServiceErrorInfo(error : String)`

Metodo che costruisce un oggetto di tipo ServiceErrorInfo.

Lista parametri del metodo:

- `error: String` - Stringa che descrive l'errore da comunicare all'utente.

Metodo: `error(): String`

Metodo che ritorna l'attributo error della classe.

Lista parametri del metodo:

Nessuno.

3.65 Actorbase.server.messages.query.LoginMessage

Descrizione

Classe che rappresenta una richiesta di login da parte di un utente.

Utilizzo

Questa classe è un messaggio che viene creato quando viene riconosciuta una richiesta di login dal parser e serve a controllare le credenziali dell'utente che vuole accedere al sistema.

Classi ereditate

- `Actorbase.server.messages.query.QueryMessage`

Ereditata da

Nessuna.

Attributi

- `username: String` - Username dell'utente che richiede di accedere al sistema.
- `password: String` - Password dell'utente che richiede di accedere al sistema.

Costruttore: `LoginMessage(username: String, password: String)`

Metodo che costruisce un oggetto di tipo LoginMessage.

Lista parametri del metodo:

- `username: String` - Username dell'utente che richiede di accedere al sistema.
- `password: String` - Password dell'utente che richiede di accedere al sistema.

Metodo: `username(): String`

Metodo che ritorna l'attributo username della classe.

Lista parametri del metodo:

Nessuno.

Metodo: `password(): String`

Metodo che ritorna l'attributo password della classe.

Lista parametri del metodo:

Nessuno.

3.66 Actorbase.server.messages.query.ReplyInfo (trait)

Descrizione

Interfaccia di base dei messaggi di tipo ReplyInfo.

Utilizzo

Questa interfaccia fornisce un tipo comune per tutti i messaggi che rappresentano informazioni aggiuntive sul risultato di una query dell'utente.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.messages.query.ReplyErrorInfo`
- `Actorbase.server.messages.query.ServiceErrorInfo`
- `Actorbase.server.messages.query.ErrorMessages.QueryErrorInfo`
- `Actorbase.server.messages.query.PermissionMessages.NoReadPermissionInfo`
- `Actorbase.server.messages.query.PermissionMessages.NoWritePermissionInfo`

- Actorbase.server.messages.query.PermissionMessages.NoAdminPermissionInfo
- Actorbase.server.messages.query.admin.PermissionManagementMessages.ListPermissionsInfo
- Actorbase.server.messages.query.admin.UsersManagementMessages.ListUserInfo
- Actorbase.server.messages.query.admin.UsersManagementMessages.NoUserInfo
- Actorbase.server.messages.query.admin.UsersManagementMessages.AddUserInfo
- Actorbase.server.messages.query.admin.UsersManagementMessages.RemoveUserInfo
- Actorbase.server.messages.query.user.RowMessages.KeyAlreadyExistInfo
- Actorbase.server.messages.query.user.RowMessages.KeyDoesNotExistInfo
- Actorbase.server.messages.query.user.RowMessages.ListKeyInfo
- Actorbase.server.messages.query.user.RowMessages.NoKeyInfo
- Actorbase.server.messages.query.user.RowMessages.FindInfo
- Actorbase.server.messages.query.user.DatabaseMessages.DBAlreadyExistInfo
- Actorbase.server.messages.query.user.DatabaseMessages.DBDoesNotExistInfo
- Actorbase.server.messages.query.user.DatabaseMessages.ListDBInfo
- Actorbase.server.messages.query.user.DatabaseMessages.NoDBInfo
- Actorbase.server.messages.query.user.DatabaseMessages.NoDBSelectedInfo
- Actorbase.server.messages.query.user.HelpMessages.CompleteHelpReplyInfo
- Actorbase.server.messages.query.user.HelpMessages.SpecificHelpReplyInfo
- Actorbase.server.messages.query.user.MapMessages.MapAlreadyExistInfo
- Actorbase.server.messages.query.user.MapMessages.MapDoesNotExistInfo
- Actorbase.server.messages.query.user.MapMessages.ListMapInfo
- Actorbase.server.messages.query.user.MapMessages.NoMapInfo
- Actorbase.server.messages.query.user.MapMessages.NoMapSelectedInfo

Attributi

Nessuno.

Metodi

Nessuno.

3.67 Actorbase.server.messages.query.ReplyErrorInfo

Descrizione

Classe che rappresenta una risposta di errore generale.

Utilizzo

Questa classe viene utilizzata per rappresentare un errore generale e comunicarlo all'utente.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.messages.ServiceErrorInfo`

Attributi

Nessuno.

Metodi

Nessuno.

3.68 Actorbase.server.messages.query.ReplyMessage

Descrizione

Classe che rappresenta la risposta ad una richiesta dell'utente.

Utilizzo

Questa classe viene creata con le informazioni necessarie per rispondere all'utente in merito ad una richiesta.

Classi ereditate

Nessuna.

Ereditata da

Nessuna.

Attributi

- `result: ReplyResult` - Esito della query.
- `question: QueryMessage` - La query a cui rispondere.
- `info: ReplyInfo = null` - Risultato della query, contenente i dati richiesti.

Costruttore: `ReplyMessage(result: ReplyResult, question: QueryMessage, info: ReplyInfo = null)`

Metodo che costruisce un oggetto di tipo ReplyResult.

Lista parametri del metodo:

- `result: ReplyResult` - Esito della query.
- `question: QueryMessage` - La query a cui rispondere.
- `info: ReplyInfo = null` - Risultato della query, contenente i dati richiesti.

Metodo: `result(): ReplyResult`

Metodo che ritorna l'attributo result della classe.

Lista parametri del metodo:

Nessuno.

Metodo: `question(): QueryMessage`

Metodo che ritorna l'attributo question della classe.

Lista parametri del metodo:

Nessuno.

Metodo: `info(): ReplyInfo`

Metodo che ritorna l'attributo info della classe.

Lista parametri del metodo:

Nessuno.

3.69 Actorbase.server.messages.query.ErrorMessages

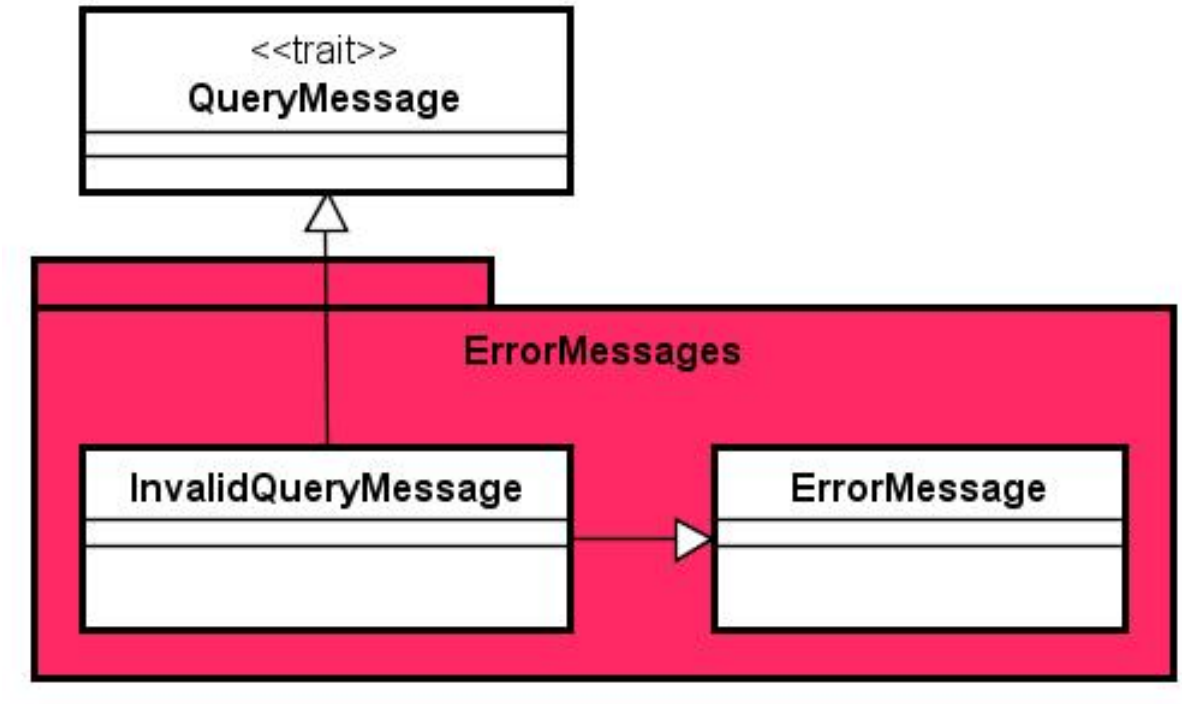


Figura 27: Componente Actorbase.server.messages.query.ErrorMessages

I messaggi contenuti in questo package sono utilizzati per gestire situazioni di errore e operazioni non valide.

3.70 Actorbase.server.messages.query.ErrorMessages.ErrorMessage (trait)

Descrizione

Trait che fornisce un'interfaccia di base a tutti i messaggi che modellano situazioni di errore.

Utilizzo

Viene esteso da tutti i messaggi che modellano situazioni di errore.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.messages.query.ErrorMessages.InvalidQueryMessage`
- `Actorbase.server.messages.query.ErrorMessages.QueryErrorInfo`

Attributi

Nessuno.

Metodi

Nessuno.

3.71 Actorbase.server.messages.query.ErrorMessages.InvalidQueryMessage

Descrizione

Questo tipo di messaggio definisce una query non valida.

Utilizzo

Viene utilizzato per informare che la query richiesta non è valida.

Classi ereditate

- `Actorbase.server.messages.query.ErrorMessages.ErrorMessage`
- `Actorbase.server.messages.query.QueryMessage`

Ereditata da

Nessuno.

Attributi

Nessuno.

Costruttore: `InvalidQueryMessage()`

Il costruttore del messaggio costruisce un `InvalidQueryMessage` senza alcun parametro.

Lista parametri del metodo: Nessuno.

3.72 Actorbase.server.messages.query.ErrorMessages.QueryErrorInfo

Descrizione

Questo tipo di messaggio definisce una query non eseguita correttamente.

Utilizzo

Viene utilizzato per informare che la query richiesta non è stata eseguita correttamente.

Classi ereditate

- `Actorbase.server.messages.query.ErrorMessages.ErrorMessage`
- `Actorbase.server.messages.query.ReplyInfo`

Ereditata da

Nessuno.

Attributi

Nessuno.

Costruttore: `QueryErrorInfo()`

Il costruttore del messaggio costruisce un `QueryErrorInfo` senza alcun parametro.

Lista parametri del metodo: Nessuno.

3.73 Actorbase.server.messages.query.PermissionMessages

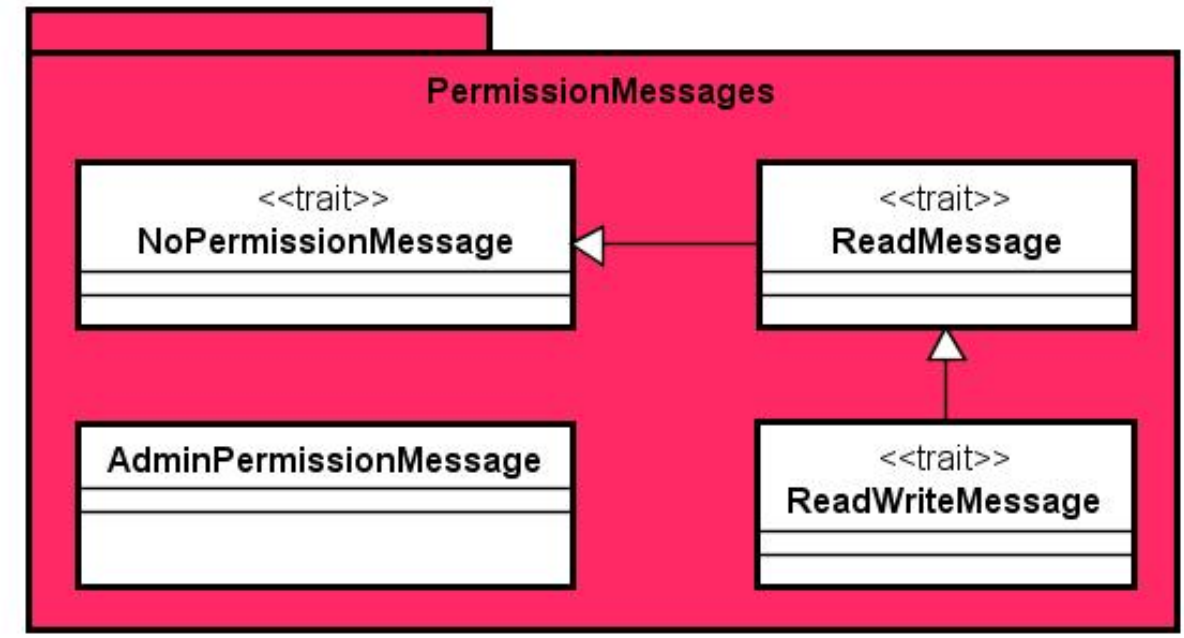


Figura 28: Componente Actorbase.server.messages.query.PermissionMessages

I `PermissionMessages` sono utilizzati per definire operazioni che trattano il livello di permessi degli utenti. Tutte le classi e interfacce contenute in questo package definiscono questo tipo di operazioni.

3.74 Actorbase.server.messages.query.PermissionMessages.AdminPermissionMessage (trait)

Descrizione

Trait che definisce l'interfaccia di base per i messaggi che riguardano operazioni per cui sono richiesti i permessi di amministratore.

Utilizzo

Viene esteso dai messaggi che trattano i permessi amministratore.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.messages.query.admin.AdminMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.75 Actorbase.server.messages.query.PermissionMessages.NoPermissionMessage (trait)

Descrizione

Trait che definisce l'interfaccia di base per i messaggi che riguardano operazioni per cui non sono richiesti permessi.

Utilizzo

Viene esteso da tutti i messaggi che definiscono operazioni che non richiedono permessi.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.messages.query.user.DatabaseMessages.CreateDatabaseMessage`
- `Actorbase.server.messages.query.user.HelpMessages.CompleteHelpMessage`
- `Actorbase.server.messages.query.user.HelpMessages.SpecificHelpMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.76 Actorbase.server.messages.query.PermissionMessages.ReadMessage (trait)

Descrizione

Trait che definisce l'interfaccia di base per i messaggi che riguardano operazioni per cui sono richiesti permessi di lettura.

Utilizzo

Viene esteso da tutti i messaggi che definiscono operazioni che richiedono permessi di lettura.

Classi ereditate

- `Actorbase.server.messages.query.PermissionMessages.NoPermissionMessage`

Ereditata da

- `Actorbase.server.messages.query.user.DatabaseMessages.SelectDatabaseMessage`
- `Actorbase.server.messages.query.user.DatabaseMessages.ListDatabaseMessage`
- `Actorbase.server.messages.query.user.MapMessages.SelectMapMessage`
- `Actorbase.server.messages.query.user.MapMessages.ListMapMessage`
- `Actorbase.server.messages.query.user.RowMessages.FindRowMessage`
- `Actorbase.server.messages.query.user.RowMessages.ListKeysMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.77 Actorbase.server.messages.query.PermissionMessages.ReadWriteMessage (trait)

Descrizione

Trait che definisce l'interfaccia di base per i messaggi che riguardano operazioni per cui sono richiesti permessi di scrittura.

Utilizzo

Viene esteso da tutti i messaggi che definiscono operazioni che richiedono permessi di dcrittura.

Classi ereditate

- `Actorbase.server.messages.query.PermissionMessages.ReadMessage`

Ereditata da

- `Actorbase.server.messages.query.user.DatabaseMessages.DeleteDatabaseMessage`
- `Actorbase.server.messages.query.user.MapMessages.CreateMapMessage`
- `Actorbase.server.messages.query.user.MapMessages.DeleteMapMessage`
- `Actorbase.server.messages.query.user.RowMessages.InsertRowMessage`
- `Actorbase.server.messages.query.user.RowMessages.UpdateRowMessage`
- `Actorbase.server.messages.query.user.RowMessages.RemoveRowMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.78 Actorbase.server.messages.query.PermissionMessages.NoReadPermissionInfo

Descrizione

Messaggio che informa della mancanza dei permessi di lettura necessari.

Utilizzo

Viene utilizzato in risposta a una richiesta di accesso ad un database, da parte di un client che non dispone dei permessi di lettura su tale database.

Classi ereditate

- `Actorbase.server.messages.query.ReplyInfo`

Ereditata da

Nessuno.

Attributi

Nessuno.

Costruttore: `NoReadPermissionInfo()`

Il costruttore del messaggio costruisce un `NoReadPermissionInfo` senza alcun parametro.

Lista parametri del metodo: Nessuno.

3.79 Actorbase.server.messages.query.PermissionMessages.NoWritePermissionInfo

Descrizione

Messaggio che informa della mancanza dei permessi di scrittura necessari.

Utilizzo

Viene utilizzato in risposta a una richiesta di modifica ad un database o a una mappa, da parte di un client che non dispone dei permessi di scrittura su tale database.

Classi ereditate

- `Actorbase.server.messages.query.ReplyInfo`

Ereditata da

Nessuno.

Attributi

Nessuno.

Costruttore: `NoWritePermissionInfo()`

Il costruttore del messaggio costruisce un `NoWritePermissionInfo` senza alcun parametro.

Lista parametri del metodo: Nessuno.

3.80 Actorbase.server.messages.query.PermissionMessages.NoAdminPermissionInfo

Descrizione

Messaggio che informa della mancanza dei permessi di amministrazione necessari.

Utilizzo

Viene utilizzato in risposta a una richiesta di operazione di amministrazione da parte di un client che non è amministratore.

Classi ereditate

- `Actorbase.server.messages.query.ReplyInfo`

Ereditata da

Nessuno.

Attributi

Nessuno.

Costruttore: `NoAdminPermissionInfo()`

Il costruttore del messaggio costruisce un `NoAdminPermissionInfo` senza alcun parametro.

Lista parametri del metodo: Nessuno.

3.81 Actorbase.server.messages.query.admin

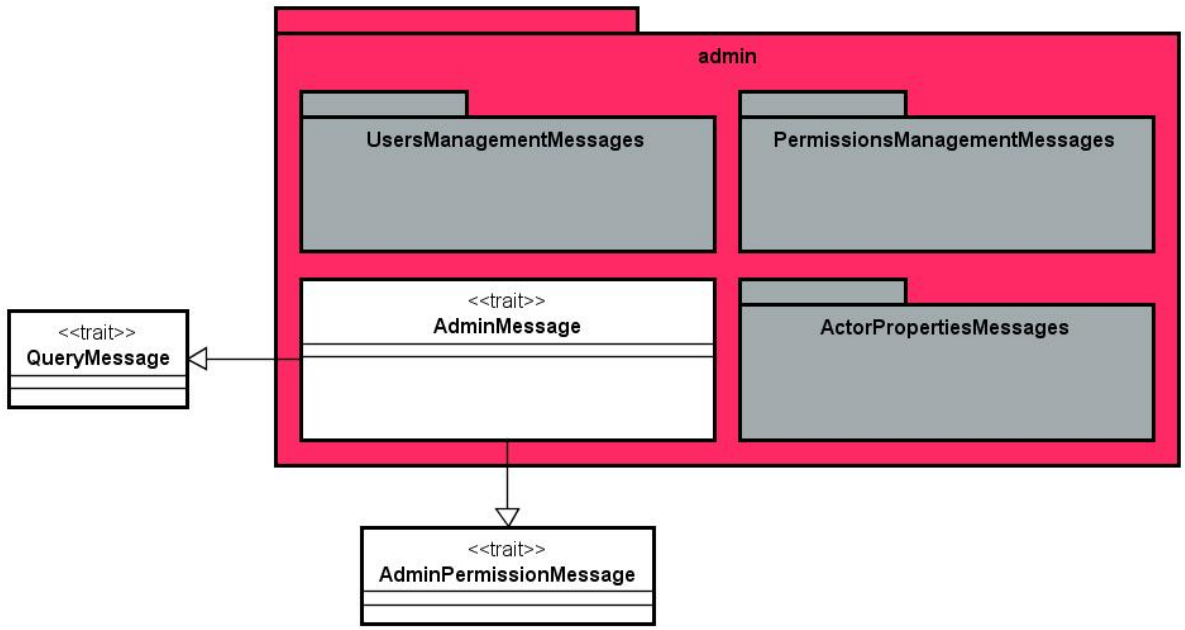


Figura 29: Componente Actorbase.server.messages.query.admin

Questo package racchiude tutte le classi ed interfacce che rappresentano messaggi di amministrazione, ovvero richieste fatte da un amministratore al server.

3.82 Actorbase.server.messages.query.admin.AdminMessage (trait)

Descrizione

Interfaccia di base per i messaggi di tipo `AdminMessage`.

Utilizzo

Questa interfaccia fornisce un tipo comune per tutti i messaggi che rappresentano dei comandi amministratore.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.messages.query.admin.PermissionsManagementMessage`
- `Actorbase.server.messages.query.admin.SettingMessage`
- `Actorbase.server.messages.query.admin.UsersManagementMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.83 Actorbase.server.messages.query.admin.PermissionsManagementMessages

Questo package contiene i messaggi di gestione dei permessi degli utenti.

3.84 Actorbase.server.messages.query.admin.PermissionsManagementMessages.PermissionManagementMessage (trait)

Descrizione

Interfaccia di base per i messaggi di tipo `PermissionManagementMessage`.

Utilizzo

Questa interfaccia fornisce un tipo comune per tutti i messaggi che rappresentano dei comandi amministratore riguardanti la gestione dei permessi.

Classi ereditate

- `Actorbase.server.messages.query.admin.AdminMessage`

Ereditata da

- `Actorbase.server.messages.query.admin.PermissionsManagementMessages.ListPermissionMessage`
- `Actorbase.server.messages.query.admin.PermissionsManagementMessages.AddPermissionMessage`
- `Actorbase.server.messages.query.admin.PermissionsManagementMessages.RemovePermissionMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.85 Actorbase.server.messages.query.admin.PermissionsManagementMessages.AddPermissionMessage

Descrizione

Messaggio per richiedere l'aggiunta di permessi ad un utente.

Utilizzo

Viene usato per modificare la tabella Master aggiungendo dei permessi all'utente specificato.

Classi ereditate

- `Actorbase.server.messages.query.admin.PermissionsManagementMessages.PermissionsManagementMessages`

Ereditata da

Nessuno.

Attributi

- `username : String` - L'username dell'utente a cui si vogliono aggiungere permessi.

Metodi

Nessuno.

3.86 Actorbase.server.messages.query.admin.PermissionsManagementMessages.RemovePermissionMessage

Descrizione

Messaggio per richiedere la rimozione di permessi ad un utente.

Utilizzo

Viene usato per modificare la tabella Master rimuovendo dei permessi all'utente specificato.

Classi ereditate

- `Actorbase.server.messages.query.admin.PermissionsManagementMessages.PermissionsManagementMessages`

Ereditata da

Nessuno.

Attributi

- `username : String` - L'username dell'utente a cui si vogliono rimuovere permessi.

Metodi

Nessuno.

3.87 Actorbase.server.messages.query.admin.PermissionsManagementMessages.ListPermissionMessage

Descrizione

Messaggio per richiedere la lista dei permessi ad un utente.

Utilizzo

Viene usato per ottenere dalla tabella Master la lista dei permessi all'utente specificato.

Classi ereditate

- `Actorbase.server.messages.query.admin.PermissionsManagementMessages.PermissionsManagementMessages`

Ereditata da

Nessuno.

Attributi

- `username : String` - L'username dell'utente di cui si vogliono visualizzare i permessi.

Metodi

Nessuno.

3.88 Actorbase.server.messages.query.admin.PermissionsManagementMessages.ListPermissionsInfo

Descrizione

Questa classe rappresenta la lista dei permessi assegnati ad un utente.

Utilizzo

Viene usata per aggiungere le informazioni riguardanti la lista dei permessi assegnati ad un utente nei messaggi di ritorno.

textbfClassi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuno.

Attributi

- `permissions: util.HashMap[String, UserPermission]` - La lista dei permessi assegnati ad un utente.

Metodi

Nessuno.

3.89 Actorbase.server.messages.query.admin.SettingsMessages.RefreshSettingsMessage

Descrizione

Messaggio per richiedere l'aggiornamento delle impostazioni dal file di configurazione.

Utilizzo

Viene usato per richiedere l'aggiornamento delle impostazioni dal file di configurazione.

Classi ereditate

- `Actorbase.server.messages.query.admin.SettingsMessages.SettingsMessages`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.90 Actorbase.server.messages.query.admin.UserManagementMessages

Questo package contiene i messaggi di amministrazione degli utenti.

3.91 Actorbase.server.messages.query.admin.UserManagementMessages.UserManagementMessage (trait)

Descrizione

Interfaccia di base per i messaggi di tipo UserManagementMessage.

Utilizzo

Questa interfaccia fornisce un tipo comune per tutti i messaggi che rappresentano dei comandi amministratore per la gestione degli utenti.

Classi ereditate

- `Actorbase.server.messages.query.admin.AdminMessage.AdminMessage`

Ereditata da

- `Actorbase.server.messages.query.admin.UserManagementMessages.AddUserMessage`
- `Actorbase.server.messages.query.admin.UserManagementMessages.RemoveUserMessage`
- `Actorbase.server.messages.query.admin.UserManagementMessages.ListUserMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.92 Actorbase.server.messages.query.admin.UserManagementMessages.AddUserMessage

Descrizione

Messaggio per richiedere l'aggiunta di un utente, con la relativa password alla lista degli utenti che hanno accesso al server.

Utilizzo

Viene usato per aggiungere un utente, con la relativa password alla lista degli utenti che hanno accesso al server.

Classi ereditate

- `Actorbase.server.messages.query.admin.UserManagementMessages.UserManagementMessages`

Ereditata da

Nessuno.

Attributi

- `username : String` - L'username che si vuole inserire.
- `password : String` - La password.

Metodi

Nessuno.

3.93 Actorbase.server.messages.query.admin.UserManagementMessages.RemoveUserMessage

Descrizione

Messaggio per richiedere la rimozione di un utente dalla lista degli utenti che hanno accesso al server.

Utilizzo

Viene usato per rimuovere un utente dalla lista degli utenti che hanno accesso al server.

Classi ereditate

- `Actorbase.server.messages.query.admin.UserManagementMessages.UserManagementMessages`

Ereditata da

Nessuno.

Attributi

- `username : String` - L'username che si vuole rimuovere.

Metodi

Nessuno.

3.94 Actorbase.server.messages.query.admin.UserManagementMessages.ListUserMessage

Descrizione

Messaggio per richiedere la lista degli utenti che hanno accesso al server.

Utilizzo

Viene usato per richiedere la lista degli utenti che hanno accesso al server.

Classi ereditate

- `Actorbase.server.messages.query.admin.UserManagementMessages.UserManagementMessages`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.95 Actorbase.server.messages.query.admin.UserManagementMessages.ListUserInfo

Descrizione

Questa classe rappresenta le informazioni riguardanti la lista degli utenti.

Utilizzo

Viene usata per aggiungere le informazioni riguardanti la lista degli utenti.

textbfClassi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuno.

Attributi

- `userList : List[String]` - La lista degli utenti.

Metodi

Nessuno.

3.96 Actorbase.server.messages.query.admin.UserManagementMessages.NoUserInfo

Descrizione

Questa classe rappresenta la non esistenza di un determinato utente nella mappa 'users' del database 'master'.

Utilizzo

Viene usata per segnalare la non esistenza di un determinato utente nella mappa 'users' del database 'master'.

textbfClassi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.97 Actorbase.server.messages.query.admin.UserManagementMessages.AddUserInfo

Descrizione

Questa classe rappresenta le informazioni riguardanti la risposta da mandare alla console quando un *AddUserMessage* è stato elaborato.

Utilizzo

Viene usata per aggiungere le informazioni riguardanti la risposta da mandare alla console quando un *AddUserMessage* è stato elaborato.

textbfClassi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.98 Actorbase.server.messages.query.admin.UserManagementMessages.RemoveUserInf

Descrizione

Questa classe rappresenta le informazioni riguardanti la risposta da mandare alla console quando un *RemoveUserMessage* è stato elaborato.

Utilizzo

Viene usata per aggiungere le informazioni riguardanti la risposta da mandare alla console quando un *RemoveUserMessage* è stato elaborato.

textbfClassi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.99 Actorbase.server.messages.query.user

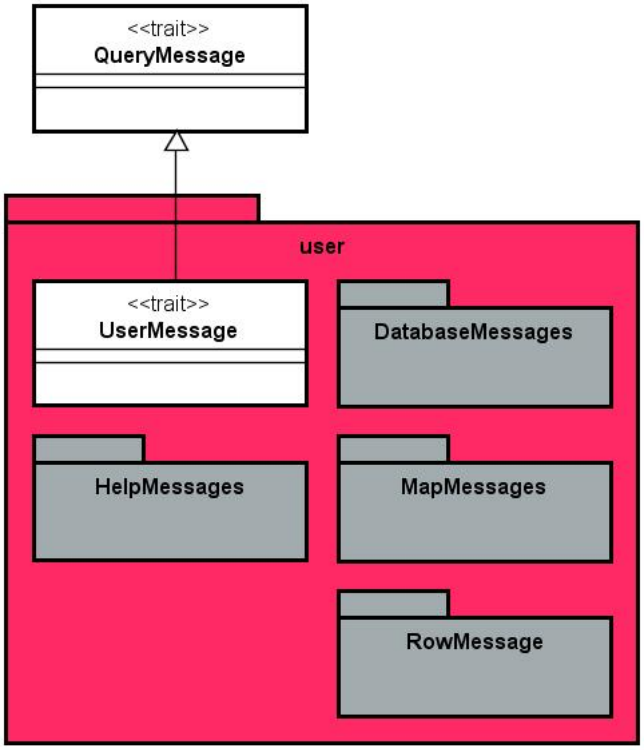


Figura 30: Componente Actorbase.server.messages.query.user

La componente user di *Actorbase* raccoglie tutti i messaggi che rappresentano richieste che l'utente sottopone al database. Si dividono in messaggi a livello di database, di mappa, di riga e in messaggi di aiuto. Raccoglie inoltre i rispettivi messaggi di risposta.

3.100 Actorbase.server.messages.query.user.UserMessage (trait)

Descrizione

Interfaccia comune a tutti i messaggi di tipo query dell'utente.

Utilizzo

Questa interfaccia viene utilizzata per dare un tipo comune a tutti i messaggi di query dell'utente.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.server.messages.query.user.RowMessage`
- `Actorbase.server.messages.query.user.MapMessage`
- `Actorbase.server.messages.query.user.DatabaseMessage`
- `Actorbase.server.messages.query.user.HelpMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.101 Actorbase.server.messages.query.user.RowMessages

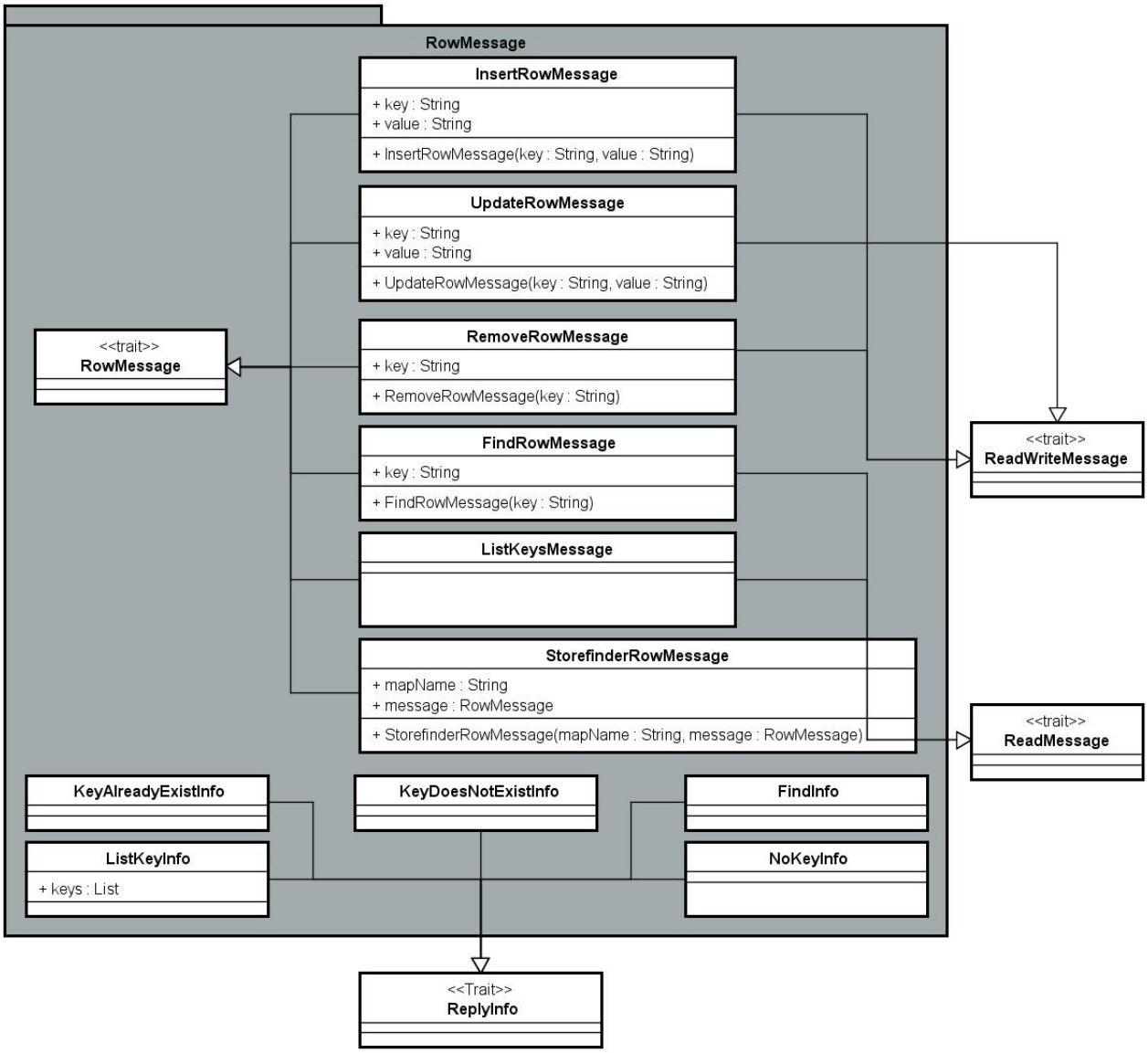


Figura 31: Componente Actorbase.server.messages.query.user.RowMessages

La componente RowMessages di *Actorbase* raccoglie tutti i messaggi che rappresentano richieste a livello di riga di un utente. Raccoglie inoltre i rispettivi messaggi di risposta.

3.102 Actorbase.server.messages.query.user.RowMessages.RowMessage (trait)

Interfaccia comune a tutti i messaggi di query a livello di riga.

Utilizzo

Questa interfaccia serve a dare un tipo comune a tutti i messaggi di query a livello di riga.

Classi ereditate

- `Actorbase.server.messages.query.user.UserMessage`

Ereditata da

- `Actorbase.server.messages.query.user.RowMessages.InsertRowMessage`
- `Actorbase.server.messages.query.user.RowMessages.UpdateRowMessage`
- `Actorbase.server.messages.query.user.RowMessages.RemoveRowMessage`
- `Actorbase.server.messages.query.user.RowMessages.FindRowMessage`
- `Actorbase.server.messages.query.user.RowMessages.ListKeysMessage`
- `Actorbase.server.messages.query.user.RowMessages.StorefinderRowMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.103 Actorbase.server.messages.query.user.RowMessages.InsertRowMessage

Classe che rappresenta un messaggio di inserimento di una coppia chiave-valore all'interno di una mappa.

Utilizzo

Questa classe è un messaggio che riporta la chiave e il valore che l'utente vuole inserire.

Classi ereditate

- `Actorbase.server.messages.query.user.RowMessages.RowMessage`
- `Actorbase.server.messages.PermissionMessages.ReadWriteMessage`

Ereditata da

Nessuna.

Attributi

- `key: String` - Chiave da inserire.
- `value: Array[Byte]` - Valore da inserire.

Costruttore: `InsertRowMessage(key: String, value: Array[Byte])`

Metodo che costruisce un oggetto di tipo InsertRowMessage.

Lista parametri del metodo:

- `key: String` - Chiave da inserire.
- `value: Array[Byte]` - Valore da inserire.

Metodo: `key(): String`

Metodo che restituisce l'attributo key della classe.

Lista parametri del metodo:

Nessuno.

Metodo: `value: Array[Byte]`

Metodo che restituisce l'attributo value della classe.

Lista parametri del metodo:

Nessuno.

3.104 Actorbase.server.messages.query.user.RowMessages.UpdateRowMessage

Descrizione

Classe che rappresenta un messaggio di modifica di un valore all'interno di una mappa.

Utilizzo

Questa classe è un messaggio che riporta la chiave della coppia da modificare e il nuovo valore che l'utente vuole inserire.

Classi ereditate

- `Actorbase.server.messages.query.user.RowMessages.RowMessage`
- `Actorbase.server.messages.PermissionMessages.ReadWriteMessage`

Ereditata da

Nessuna.

Attributi

- `key: String` - Chiave della coppia da modificare.
- `value: Array[Byte]` - Nuovo valore da inserire.

Costruttore: `UpdateRowMessage(key: String, value: Array[Byte])`

Metodo che costruisce un oggetto di tipo UpdateRowMessage.

Lista parametri del metodo:

- `key: String` - Chiave della coppia da modificare.
- `value: Array[Byte]` - Nuovo valore da inserire.

Metodo: `key(): String`

Metodo che restituisce l'attributo key della classe.

Lista parametri del metodo:

Nessuno.

Metodo: `value: Array[Byte]`

Metodo che restituisce l'attributo value della classe.

Lista parametri del metodo:

Nessuno.

3.105 Actorbase.server.messages.query.user.RowMessages.RemoveRowMessage

Descrizione

Classe che rappresenta un messaggio di rimozione di una coppia chiave-valore da una mappa.

Utilizzo

Questa classe è un messaggio che riporta la chiave della coppia da rimuovere.

Classi ereditate

- `Actorbase.server.messages.query.user.RowMessages.RowMessage`
- `Actorbase.server.messages.PermissionMessages.ReadWriteMessage`

Ereditata da

Nessuna.

Attributi

- `key: String` - Chiave della coppia da rimuovere.

Costruttore: `RemoveRowMessage(key: String)`

Metodo che costruisce un oggetto di tipo RemoveRowMessage.

Lista parametri del metodo:

- `key: String` - Chiave della coppia da rimuovere.

Metodo: `key(): String`

Metodo che restituisce l'attributo key della classe.

Lista parametri del metodo:

Nessuno.

3.106 Actorbase.server.messages.query.user.RowMessages.FindRowMessage

Descrizione

Classe che rappresenta un messaggio di ricerca di una coppia chiave-valore di una mappa.

Utilizzo

Questa classe è un messaggio che riporta la chiave del valore richiesto.

Classi ereditate

- `Actorbase.server.messages.query.user.RowMessages.RowMessage`
- `Actorbase.server.messages.PermissionMessages.ReadMessage`

Ereditata da

Nessuna.

Attributi

- `key: String` - Chiave della coppia richiesta.

Costruttore: `FindRowMessage(key: String)`

Metodo che costruisce un oggetto di tipo FindRowMessage.

Lista parametri del metodo:

- `key: String` - Chiave della coppia richiesta.

Metodo: `key(): String`

Metodo che restituisce l'attributo key della classe.

Lista parametri del metodo:

Nessuno.

3.107 Actorbase.server.messages.query.user.RowMessages.ListKeysMessage

Descrizione

Classe che rappresenta un messaggio di richiesta della lista delle chiavi della mappa.

Utilizzo

Questa classe è un messaggio che serve a richiedere l'intera lista delle chiavi della mappa.

Classi ereditate

- `Actorbase.server.messages.query.user.RowMessages.RowMessage`
- `Actorbase.server.messages.PermissionMessages.ReadMessage`

Ereditata da

Nessuna.

Attributi

Nessuno.

Costruttore: `ListKeysMessage(key: String)`

Metodo che costruisce un oggetto di tipo ListKeysMessage.

Lista parametri del metodo:

Nessuno.

3.108 Actorbase.server.messages.query.user.RowMessages.KeyAlreadyExistInfo

Descrizione

Classe che rappresenta la risposta di chiave già esistente.

Utilizzo

Questa classe serve a comunicare all'utente che la chiave che si intende inserire è già presente nella mappa.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodi

Nessuno.

3.109 Actorbase.server.messages.query.user.RowMessages.KeyDoesNotExistInfo

Descrizione

Classe che rappresenta la risposta di chiave inesistente.

Utilizzo

Questa classe serve a comunicare all'utente che la chiave ricercata per operazioni di find, remove o update non esiste.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodi

Nessuno.

3.110 Actorbase.server.messages.query.user.RowMessages.ListKeyInfo

Descrizione

Classe che rappresenta la risposta alla richiesta di ricevere la lista delle chiavi della mappa.

Utilizzo

Questa classe serve a ritornare all'utente la lista delle chiavi della mappa.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

- `keys: List[String]` - Lista delle chiavi presenti nella mappa.

Costruttore: `ListKeyInfo(keys: List[String])`

Metodo che costruisce un oggetto di tipo ListKeyInfo.

Lista parametri del metodo:

- `keys: List[String]` - Lista delle chiavi presenti nella mappa.

Metodo: `keys(): List[String]`

Metodo che restituisce l'attributo key della classe.

Lista parametri del metodo:

Nessuno.

3.111 Actorbase.server.messages.query.user.RowMessages.NoKeyInfo

Descrizione

Classe che rappresenta la risposta di nessuna chiave nella mappa.

Utilizzo

Questa classe serve a comunicare all'utente, al seguito di una richiesta della lista delle chiavi, che non sono presenti chiavi nella mappa.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodi

Nessuno.

3.112 Actorbase.server.messages.query.user.RowMessages.FindInfo

Descrizione

Classe che rappresenta la risposta ad un ricerca di un utente.

Utilizzo

Questa classe serve a ritornare all'utente il valore ricercato tramite il comando find.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

- `value: Array[Byte]` - Valore ricercato dall'utente.

Costruttore: `FindInfo(value: Array[Byte])`

Metodo che costruisce un oggetto di tipo FindInfo.

Lista parametri del metodo:

- `value: Array[Byte]` - Valore ricercato dall'utente.

Metodo: `value: Array[Byte]`

Metodo che restituisce l'attributo value della classe.

Lista parametri del metodo:

Nessuno.

3.113 Actorbase.server.messages.query.user.MapMessages

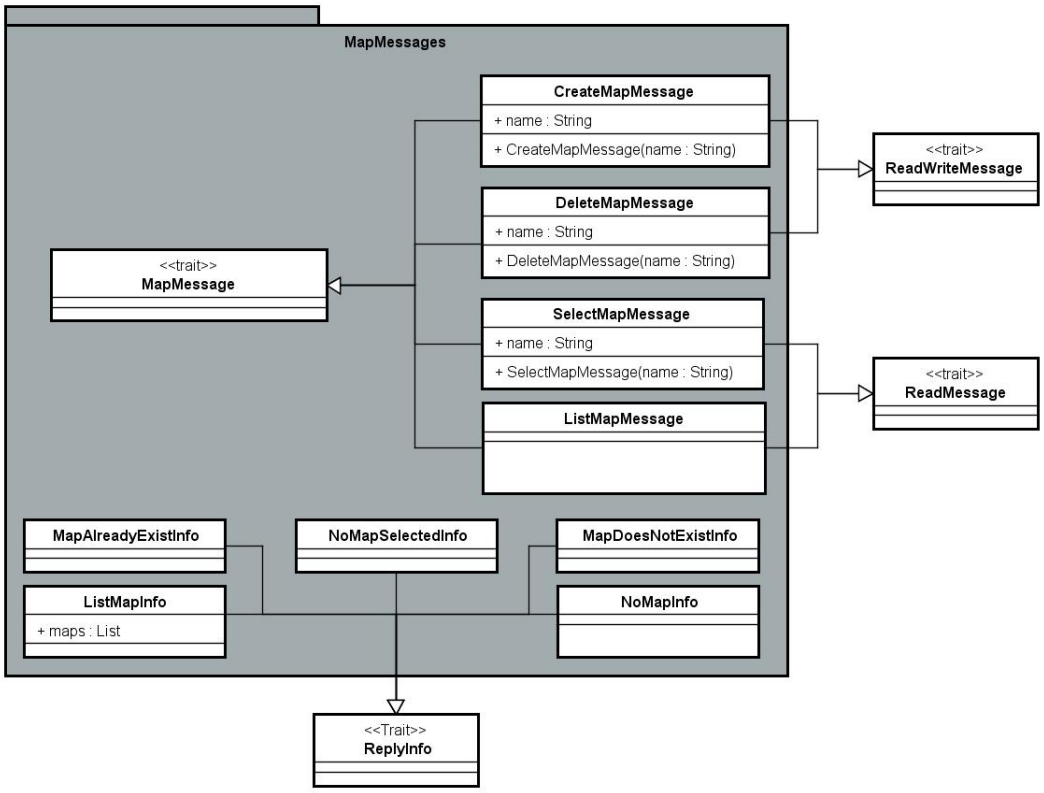


Figura 32: Componente Actorbase.server.messages.query.user.MapMessages

La componente MapMessages di *Actorbase* raccoglie tutti i messaggi che rappresentano richieste a livello di mappa di un utente. Raccoglie inoltre i rispettivi messaggi di risposta.

3.114 Actorbase.server.messages.query.user.MapMessages.MapMessage (trait)

Descrizione

Interfaccia comune a tutti i messaggi di query a livello di mappa.

Utilizzo

Questa interfaccia serve a dare un tipo comune a tutti i messaggi di query a livello di mappa.

Classi ereditate

- `Actorbase.server.messages.query.user.UserMessage`

Ereditata da

- `Actorbase.server.messages.query.user.MapMessages.CreateMapMessage`
- `Actorbase.server.messages.query.user.MapMessages.DeleteMapMessage`
- `Actorbase.server.messages.query.user.MapMessages.SelectMapMessage`
- `Actorbase.server.messages.query.user.MapMessages.ListMapMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.115 Actorbase.server.messages.query.user.MapMessages.CreateMapMessage

Descrizione

Classe che rappresenta la richiesta di creazione di una nuova mappa.

Utilizzo

Questa classe è un messaggio che riporta il nome della mappa che l'utente richiede di creare.

Classi ereditate

- `Actorbase.server.messages.query.user.MapMessages.MapMessage`
- `Actorbase.server.messages.PermissionMessages.ReadWriteMessage`

Ereditata da

Nessuna.

Attributi

- `name: String` - Nome della mappa da creare.

Costruttore: `CreateMapMessage(name: String)`

Metodo che costruisce un oggetto di tipo CreateMapMessage.

Lista parametri del metodo:

- `name: String` - Nome della mappa da creare.

Metodo: `name(): String`

Metodo che restituisce l'attributo name della classe.

Lista parametri del metodo:

Nessuno.

3.116 Actorbase.server.messages.query.user.MapMessages.DeleteMapMessage

Descrizione

Classe che rappresenta la richiesta di rimozione di una mappa.

Utilizzo

Questa classe è un messaggio che riporta il nome della mappa che l'utente richiede di rimuovere.

Classi ereditate

- `Actorbase.server.messages.query.user.MapMessages.MapMessage`
- `Actorbase.server.messages.PermissionMessages.ReadWriteMessage`

Ereditata da

Nessuna.

Attributi

- `name: String` - Nome della mappa da eliminare.

Costruttore: `DeleteMapMessage(name: String)`

Metodo che costruisce un oggetto di tipo DeleteMapMessage.

Lista parametri del metodo:

- `name: String` - Nome della mappa da eliminare.

Metodo: `name(): String`

Metodo che restituisce l'attributo name della classe.

Lista parametri del metodo:

Nessuno.

3.117 Actorbase.server.messages.query.user.MapMessages.SelectMapMessage

Descrizione

Classe che rappresenta la richiesta di selezione di una mappa.

Utilizzo

Questa classe è un messaggio che riporta il nome della mappa che l'utente richiede di selezionare.

Classi ereditate

- `Actorbase.server.messages.query.user.MapMessages.MapMessage`

- `Actorbase.server.messages.PermissionMessages.ReadMessage`

Ereditata da

Nessuna.

Attributi

- `name: String` - Nome della mappa da selezionare.

Costruttore: `SelectMapMessage(name: String)`

Metodo che costruisce un oggetto di tipo SelectMapMessage.

Lista parametri del metodo:

- `name: String` - Nome della mappa da selezionare.

Metodo: `name(): String`

Metodo che restituisce l'attributo name della classe.

Lista parametri del metodo:

Nessuno.

3.118 Actorbase.server.messages.query.user.MapMessages.ListMapMessage

Descrizione

Classe che rappresenta la richiesta di visualizzare la lista delle mappe del database.

Utilizzo

Questa classe è un messaggio che rappresenta la richiesta dell'utente di visualizzare la lista delle mappe del database selezionato.

Classi ereditate

- `Actorbase.server.messages.query.user.MapMessages.MapMessage`
- `Actorbase.server.messages.PermissionMessages.ReadMessage`

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodi

Nessuno.

3.119 Actorbase.server.messages.query.user.MapMessages.MapAlreadyExistInfo

Descrizione

Classe che rappresenta la risposta di mappa già esistente.

Utilizzo

Questa classe serve a comunicare all'utente che esiste già una mappa con lo stesso nome nel database selezionato.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodi

Nessuno.

3.120 Actorbase.server.messages.query.user.MapMessages.MapDoesNotExistInfo

Descrizione

Classe che rappresenta la risposta di mappa inesistente.

Utilizzo

Questa classe serve a comunicare all'utente che la mappa richiesta tramite le operazioni di select o delete non esiste.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodi

Nessuno.

3.121 Actorbase.server.messages.query.user.MapMessages.ListMapInfo

Descrizione

Classe che rappresenta la risposta alla richiesta di ricevere la lista delle mappe del database selezionato.

Utilizzo

Questa classe serve a ritornare all'utente la lista dei nomi delle mappe del database selezionato.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

- `maps: List[String]` - Lista dei nomi delle mappe del database.

Costruttore: `ListMapInfo(maps: List[String])`

Metodo che costruisce un oggetto di tipo ListMapInfo.

Lista parametri del metodo:

- `maps: List[String]` - Lista dei nomi delle mappe del database.

Metodo: `maps: List[String]`

Metodo che restituisce l'attributo maps della classe.

Lista parametri del metodo:

Nessuno.

3.122 Actorbase.server.messages.query.user.MapMessages.NoMapInfo

Descrizione

Classe che rappresenta la risposta di nessuna mappa presente nel database.

Utilizzo

Questa classe serve a comunicare all'utente, in risposta alla richiesta di visualizzazione della lista delle mappe, che non ci sono mappe nel database.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodi

Nessuno.

3.123 Actorbase.server.messages.query.user.MapMessages.NoMapSelectedInfo

Descrizione

Classe che rappresenta la risposta di nessuna mappa selezionata.

Utilizzo

Questa classe serve a comunicare all'utente che ha inviato una richiesta a livello di riga che non ha selezionato nessuna mappa.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodi

Nessuno.

3.124 Actorbase.server.messages.query.user.DatabaseMessages

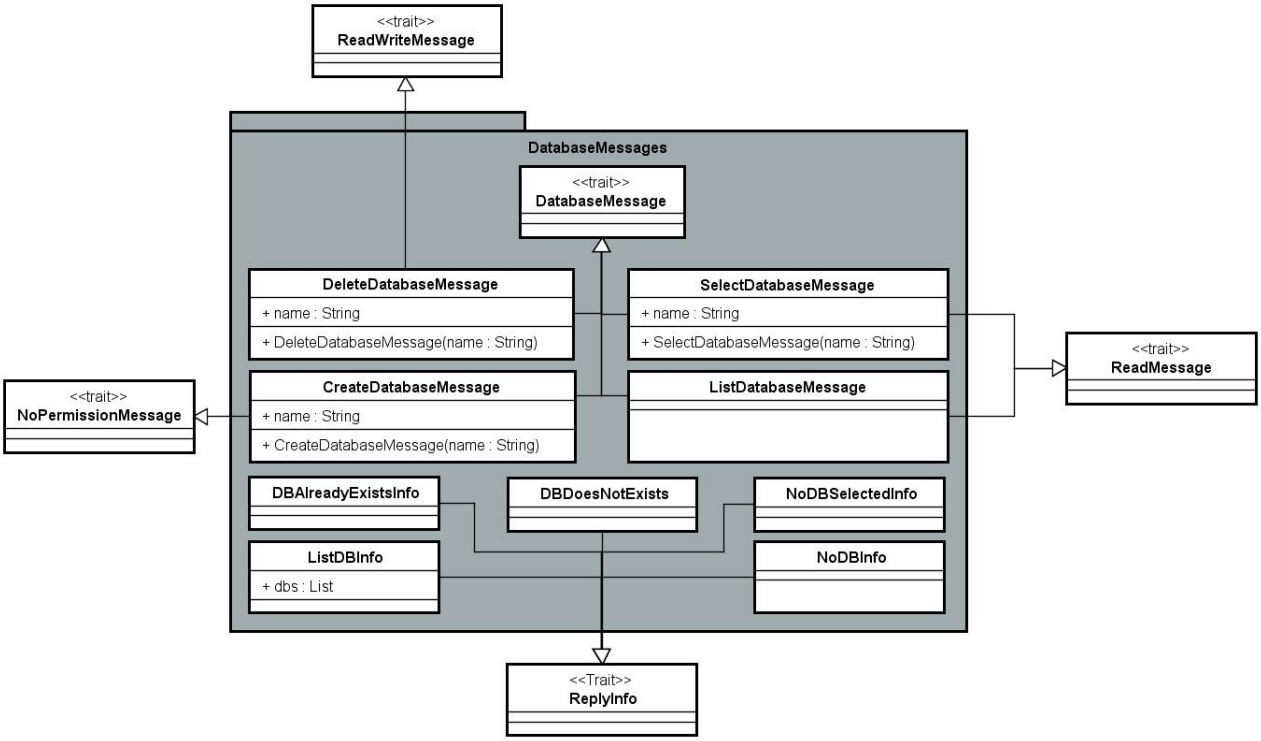


Figura 33: Componente Actorbase.server.messages.query.user.DatabaseMessages

La componente DatabaseMessages di *Actorbase* raccoglie tutti i messaggi che rappresentano richieste a livello di database di un utente. Raccoglie inoltre i rispettivi messaggi di risposta.

3.125 Actorbase.server.messages.query.user.DatabaseMessages.DatabaseMessage (trait)

Descrizione

Interfaccia comune a tutti i messaggi di query a livello di database.

Utilizzo

Questa interfaccia serve a dare un tipo comune a tutti i messaggi di query a livello di database.

Classi ereditate

- `Actorbase.server.messages.query.user.UserMessage`

Ereditata da

- `Actorbase.server.messages.query.user.DatabaseMessages.CreateDatabaseMessage`
- `Actorbase.server.messages.query.user.DatabaseMessages.DeleteDatabaseMessage`
- `Actorbase.server.messages.query.user.DatabaseMessages.SelectDatabaseMessage`
- `Actorbase.server.messages.query.user.DatabaseMessages.ListDatabaseMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.126 Actorbase.server.messages.query.user.DatabaseMessages.CreateDatabaseMessage

Descrizione

Classe che rappresenta la richiesta di creazione di un nuovo database.

Utilizzo

Questa classe è un messaggio che riporta il nome del database che l'utente richiede di creare. Non sono necessari permessi particolari per creare un nuovo database.

Classi ereditate

- `Actorbase.server.messages.query.user.DatabaseMessages.DatabaseMessage`
- `Actorbase.server.messages.PermissionMessages.NoPermissionMessages`

Ereditata da

Nessuna.

Attributi

- `name: String` - Nome del database da creare.

Costruttore: `CreateDatabaseMessage(name: String)`

Metodo che costruisce un oggetto di tipo CreateDatabaseMessage.

Lista parametri del metodo:

- `name: String` - Nome del database da creare.

Metodo: `name(): String`

Metodo che restituisce l'attributo name della classe.

Lista parametri del metodo:

Nessuno.

3.127 Actorbase.server.messages.query.user.DatabaseMessages.DeleteDatabaseMessage

Descrizione

Classe che rappresenta la richiesta di rimozione di un database.

Utilizzo

Questa classe è un messaggio che riporta il nome del database che l'utente richiede di rimuovere.

Classi ereditate

- `Actorbase.server.messages.query.user.DatabaseMessages.DatabaseMessage`
- `Actorbase.server.messages.PermissionMessages.ReadWriteMessage`

Ereditata da

Nessuna.

Attributi

- `name: String` - Nome del database da rimuovere.

Costruttore: `DeleteDatabaseMessage(name: String)`

Metodo che costruisce un oggetto di tipo DeleteDatabaseMessage.

Lista parametri del metodo:

- `name: String` - Nome del database da rimuovere.

Metodo: `name(): String`

Metodo che restituisce l'attributo name della classe.

Lista parametri del metodo:

Nessuno.

3.128 Actorbase.server.messages.query.user.DatabaseMessages.SelectDatabaseMessage

Descrizione

Classe che rappresenta la richiesta di selezione di un database.

Utilizzo

Questa classe è un messaggio che riporta il nome del database che l'utente richiede di selezionare.

Classi ereditate

- `Actorbase.server.messages.query.user.DatabaseMessages.DatabaseMessage`
- `Actorbase.server.messages.PermissionMessages.ReadMessage`

Ereditata da

Nessuna.

Attributi

- `name: String` - Nome del database da selezionare.

Costruttore: `SelectDatabaseMessage(name: String)`

Metodo che costruisce un oggetto di tipo SelectDatabaseMessage.

Lista parametri del metodo:

- `name: String` - Nome del database da selezionare.

Metodo: `name(): String`

Metodo che restituisce l'attributo name della classe.

Lista parametri del metodo:

Nessuno.

3.129 Actorbase.server.messages.query.user.DatabaseMessages.ListDatabaseMessage

Descrizione

Classe che rappresenta la richiesta di visualizzare la lista dei database disponibili.

Utilizzo

Questa classe è un messaggio che riporta la richiesta dell'utente di visualizzare la lista dei database disponibili.

Classi ereditate

- `Actorbase.server.messages.query.user.DatabaseMessages.DatabaseMessage`
- `Actorbase.server.messages.PermissionMessages.ReadMessage`

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodi

Nessuno.

3.130 Actorbase.server.messages.query.user.DatabaseMessages.DBAlreadyExistInfo

Descrizione

Classe che rappresenta la risposta di database già esistente.

Utilizzo

Questa classe serve a comunicare all'utente che esiste già un database con lo stesso nome.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodi

Nessuno.

3.131 Actorbase.server.messages.query.user.DatabaseMessages.DBDoesNotExistInfo

Descrizione

Classe che rappresenta la risposta di database inesistente.

Utilizzo

Questa classe serve a comunicare all'utente che il database richiesto tramite le operazione di select o delete non esiste.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodi

Nessuno.

3.132 Actorbase.server.messages.query.user.DatabaseMessages.ListDBInfo

Descrizione

Classe che rappresenta la risposta alla richiesta di ricevere la lista dei database disponibili.

Utilizzo

Questa classe serve a ritornare all'utente la lista dei nomi dei database disponibili.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

- `dbs: List[String]` - Lista dei nomi dei database disponibili.

Costruttore: `ListDBInfo(maps: List[String])`

Metodo che costruisce un oggetto di tipo ListDBInfo.

Lista parametri del metodo:

- `dbs: List[String]` - Lista dei nomi dei database disponibili.

Metodo: `dbs: List[String]`

Metodo che restituisce l'attributo dbs della classe.

Lista parametri del metodo:

Nessuno.

3.133 Actorbase.server.messages.query.user.DatabaseMessages.NoDBInfo

Descrizione

Classe che rappresenta la risposta di nessun database disponibile.

Utilizzo

Questa classe serve a comunicare all'utente, in risposta alla richiesta di visualizzazione della lista dei database, che non ci sono database disponibili.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodi

Nessuno.

3.134 Actorbase.server.messages.query.user.DatabaseMessages.NoDBSelectedInfo

Descrizione

Classe che rappresenta la risposta di nessun database selezionato.

Utilizzo

Questa classe serve a comunicare all'utente che ha inviato una richiesta a livello di mappa o riga che non ha selezionato nessuna database.

Classi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodi

Nessuno.

3.135 Actorbase.server.messages.query.user.HelpMessages

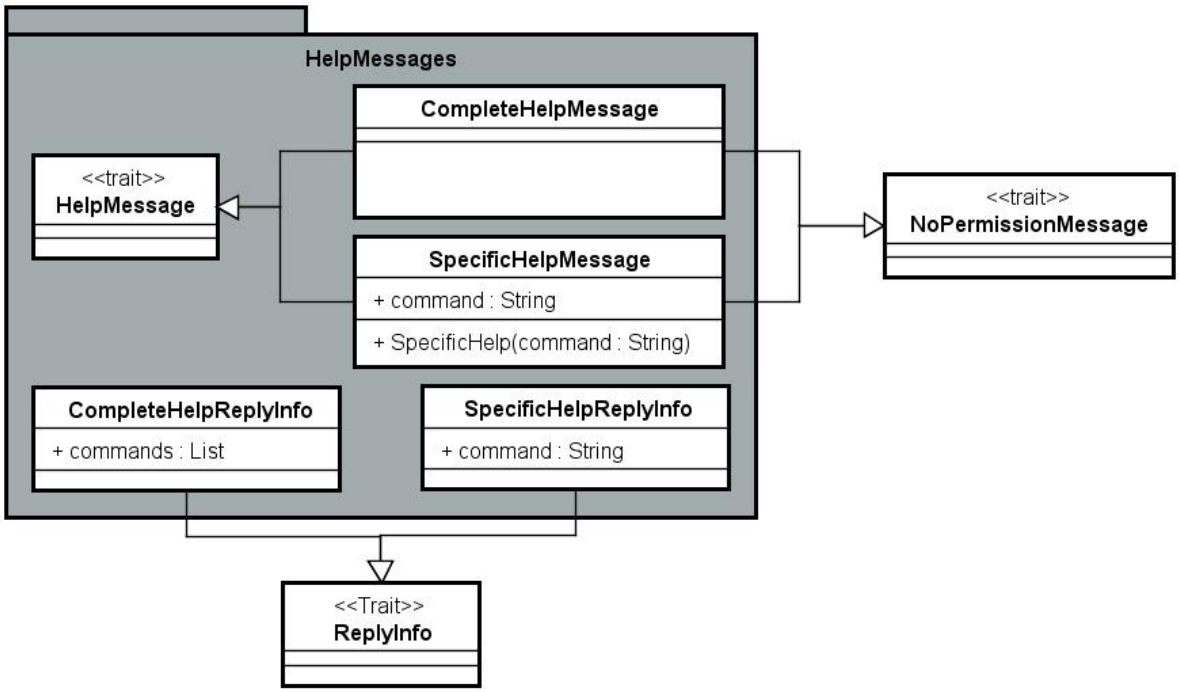


Figura 34: Componente Actorbase.server.messages.query.user.HelpMessages

Questo package contiene tutte le classi e interfacce che rappresentano messaggi per operazioni di richiesta aiuto.

3.136 Actorbase.server.messages.query.user.HelpMessages.HelpMessage (trait)

Descrizione

Interfaccia di base per i messaggi di tipo HelpMessage.

Utilizzo

Questa interfaccia fornisce un tipo comune per tutti i messaggi che rappresentano dei comandi di aiuto.

Classi ereditate

- `Actorbase.server.messages.query.user.UserMessage`

Ereditata da

- `Actorbase.server.messages.query.user.HelpMessages.CompleteHelpMessage`
- `Actorbase.server.messages.query.user.HelpMessages.SpecificHelpMessage`

Attributi

Nessuno.

Metodi

Nessuno.

3.137 Actorbase.server.messages.query.user.HelpMessages.CompleteHelp

Descrizione

Messaggio per richiedere l'aiuto completo.

Utilizzo

Viene usato per richiedere l'aiuto completo.

Classi ereditate

- `Actorbase.server.messages.query.user.HelpMessages.HelpMessage`
- `Actorbase.server.messages.query.PermissionMessages.NoPermissionMessage`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.138 Actorbase.server.messages.query.user.HelpMessages.SpecificHelp

Descrizione

Messaggio per richiedere l'aiuto specifico.

Utilizzo

Viene usato per richiedere l'aiuto specifico.

Classi ereditate

- `Actorbase.server.messages.query.user.HelpMessages.HelpMessage`
- `Actorbase.server.messages.query.PermissionMessages.NoPermissionMessage`

Ereditata da

Nessuno.

Attributi

Nessuno.

Metodi

Nessuno.

3.139 Actorbase.server.messages.query.user.HelpMessages.CompleteHelpReplyInfo

Descrizione

Questa classe rappresenta le informazioni riguardanti la richiesta di aiuto completo.

Utilizzo

Viene usata per aggiungere le informazioni riguardanti la richiesta di aiuto completo.

textbfClassi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuno.

Attributi

- `commands: String` - Le informazioni.

Metodi

Nessuno.

3.140 Actorbase.server.messages.query.user.HelpMessages.SpecificHelpReplyInfo

Descrizione

Questa classe rappresenta le informazioni riguardanti la richiesta di aiuto specifico.

Utilizzo

Viene usata per aggiungere le informazioni riguardanti la richiesta di aiuto specifico.

textbfClassi ereditate

- `Actorbase.server.messages.ReplyInfo`

Ereditata da

Nessuno.

Attributi

- `commands: String` - Le informazioni.

Metodi

Nessuno.

3.141 Actorbase.client

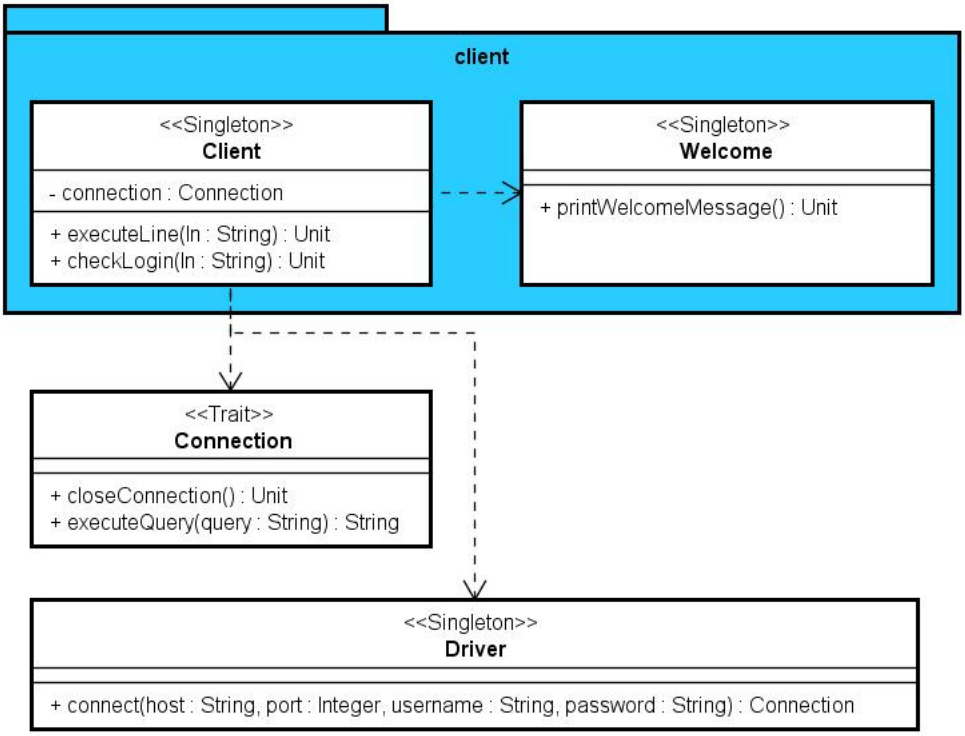


Figura 35: Componente Actorbase.client

La componente client di *Actorbase* è un semplice client da riga di comando per interagire con il server attraverso il driver, è composta dalle classi Client e Welcome.

3.142 Actorbase.client.Client

Descrizione

Classe che rappresenta il client a riga di comando di *Actorbase*.

Utilizzo

Questa classe viene utilizzata per inviare le stringhe inserite dall'utente al driver di *Actorbase* e per visualizzare le risposte del server.

Classi ereditate

Nessuna.

Ereditata da

Nessuna.

Attributi

- `private val connection: Connection` - attributo di tipo Connection, definito nel driver, che rappresenta la connessione con il server. Inizialmente nullo, la connessione viene fornita dal driver dopo aver eseguito l'accesso al server correttamente.

Metodo: `def main(args: Array[String])`

Metodo main del client che avvia il programma. All'avvio stampa il messaggio di benvenuto della classe Welcome, poi rimane in attesa di input da parte dell'utente.

Lista parametri del metodo:

- `args: Array[String]` - Parametro standard del metodo main di *Scala*.

Metodo: `def executeLine(ln: String)`

Metodo executeLine del client che ha il compito di inviare al driver le stringhe inserite dall'utente tramite l'omonimo metodo della connection. Deve poter riconoscere la stringa di disconnessione, che chiama un metodo diverso della connection, e la stringa di chiusura, che termina il programma. Inoltre è necessario che le altre stringhe siano processate solo se esiste una connessione con il server.

Lista parametri del metodo:

- `ln: String` - Stringa fornita in input dall'utente.

Metodo: `def checkLogin(ln:String)`

Metodo checkLogin che riconosce tramite un'espressione regolare il comando di connessione. Una volta riconosciuto il comando di connessione chiama il metodo connect del Driver per ottenere una Connection nel caso in cui i dati siano corretti.

Lista parametri del metodo:

- `ln: String` - Stringa fornita in input dall'utente.

3.143 Actorbase.client.Welcome

Descrizione

Classe che stampa un messaggio di benvenuto sulla console del client e alcuni dati riguardanti la macchina che si sta utilizzando.

Utilizzo

Stampa di un messaggio di benvenuto per l'utente.

Classi ereditate

Nessuna.

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodo: `def printWelcomeMessage(): Unit`

Stampa una stringa di benvenuto con il nome del prodotto e la dichiarazione di essere open-source. Inoltre recupera e stampa il nome e la versione del sistema operativo dell'utente, il nome dell'utente e la versione della JVM installata nella macchina dell'utente.

Lista parametri del metodo:

Nessuno.

3.144 Actorbase.driver

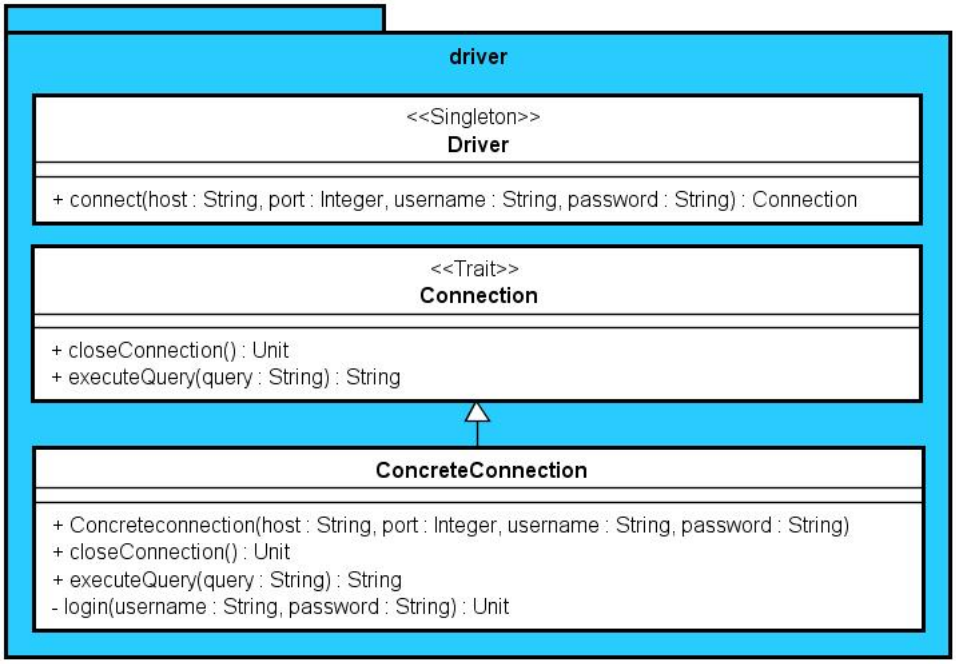


Figura 36: Componente Actorbase.driver

La componente driver di *Actorbase* è un semplice driver per gestire le comunicazioni con il server. É composta dalle classi Driver, Connection e ConcreteConnection.

3.145 Actorbase.driver.Connection (trait)

Descrizione

Interfaccia che definisce una connessione con il server di *Actorbase*.

Utilizzo

Questa interfaccia espone i metodi che deve implementare una classe la concretizza.

Classi ereditate

Nessuna.

Ereditata da

- `Actorbase.driver.ConcreteConnection`

Attributi

Nessuno.

Metodo: `def closeConnection(): Unit`

Metodo astratto per chiudere la connessione con il server.

Lista parametri del metodo:

Nessuno.

Metodo: `def executeQuery(query: String): String`

Metodo astratto per inviare una stringa al server. Ritorna una stringa che rappresenta la risposta del server.

Lista parametri del metodo:

- `query: String` - Stringa da inviare al server.

3.146 Actorbase.driver.ConcreteConnection

Descrizione

Concretizzazione della classe Connection.

Utilizzo

Questa classe riceve delle stringhe tramite il metodo executeLine. Queste stringhe vengono modificate ed inviate al Server. La classe ritorna le risposte alle query in formato di stringa.

Classi ereditate

- `Actorbase.driver.Connection`

Ereditata da

Nessuna.

Attributi

- `private val socket: Socket` - Socket di java.net usato per la connessione con il server.
- `private val out: PrintStream` - PrintStream di java.io usato per scrivere sul socket.
- `private val in: InputStream` - InputStream di java.io usato per leggere dal socket.
- `val host: String` - Nome dell'host.
- `val port: Integer` - Numero della porta su cui impostare la connessione.
- `val username: String` - Nome utente per effettuare il login al server.
- `val password: String` - Password dell'utente per effettuare il login al server.

Costruttore: `ConcreteConnection(val host: String, val port: Integer, val username: String, val password: String)`

Costruisce un oggetto di tipo Concrete connection.

Lista parametri del metodo:

- `host: String` - Nome dell'host.
- `port: Integer` - Numero della porta su cui impostare la connessione.
- `username: String` - Nome utente per effettuare il login al server.
- `password: String` - Password dell'utente per effettuare il login al server.

Metodo: `def closeConnection(): Unit`

Chiude la connessione con il server chiudendo il socket e tutti gli stream su di esso.

Lista parametri del metodo:

Nessuno.

Metodo: `def executeQuery(query: String): String`

Prepara la stringa per essere inviata al server, aggiungendo i byte per il protocollo e per l'identificazione della richiesta. Dopo aver inviato la richiesta al server resta in attesa di una risposta per un tempo determinato.

Lista parametri del metodo:

- `query: String` - Query da inviare al server.

Metodo: `private def login(username: String, password: String): Unit`

Questo metodo deve essere eseguito alla costruzione della classe. Il metodo prova a connettersi al server mandando il comando di login e attende una risposta. In caso di risposta negativa chiude la connessione.

Lista parametri del metodo:

- `username: String` - Nome utente per autenticarsi nel server.
- `password: String` - Password dell'utente per autenticarsi nel server.

3.147 Actorbase.driver.Driver

Descrizione

Driver di *Actorbase*.

Utilizzo

La classe Driver crea un oggetto di tipo Connection e lo restituisce.

Classi ereditate

Nessuna.

Ereditata da

Nessuna.

Attributi

Nessuno.

Metodo: `def connect(host: String, port: Integer, username: String, password: String): Connection`

Questo metodo crea un oggetto di tipo Connection con i parametri passati e lo ritorna al chiamante. Se la connessione non è stata effettuata il metodo ritorna il valore nullo. Il metodo deve anche gestire le eccezioni InterruptedException ed Exception.

Lista parametri del metodo:

- `host: String` - Nome dell'host da connettere al server.
- `port: Integer` - Porta sulla quale aprire la connessione.
- `username: String` - Nome dell'utente che vuole accedere al sistema.

- `password: String` - Password dell'utente che vuole accedere al sistema.

4 Diagrammi di sequenza

In questa sezione verranno illustrati e descritti i principali diagrammi di sequenza realizzati.

Per esplicitare l'invio di un messaggio tra due attori, nei diagrammi viene chiamato il metodo *sendMessage()* dell'attore che manda il messaggio. Questo perché per mandare messaggi con Akka si usano punti esclamativi e di domanda

Trattandosi di diagrammi di sequenza non molto specifici, alcuni attori effettuano operazioni senza chiamare dei metodi specifici già dichiarati.

4.1 Ricezione messaggio

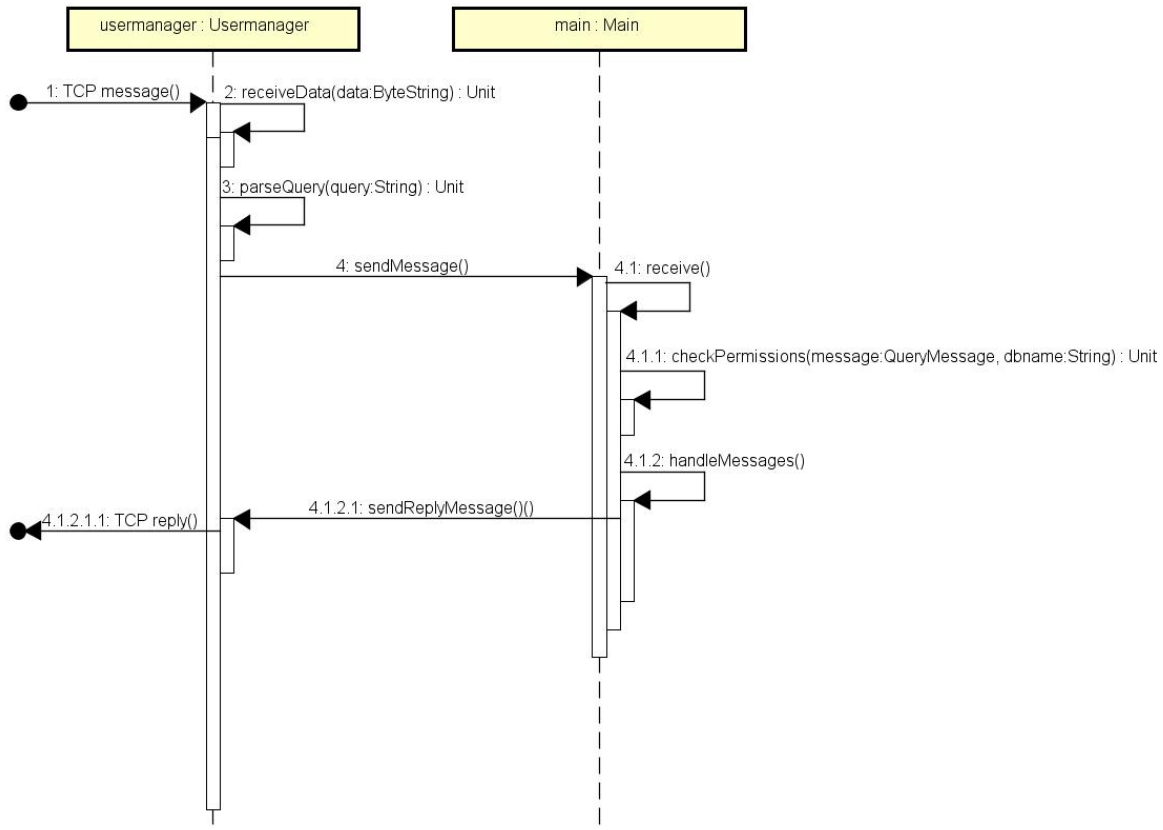


Figura 37: Diagramma di sequenza per la ricezione di un messaggio

Nel diagramma è possibile visualizzare quali sono le operazioni che vengono eseguite dagli attori Usermanager e Main alla ricezione di una richiesta dall'utente. Usermanager, dopo aver convertito i byte bufferizzati ricevuti in una stringa, crea un messaggio che rappresenta la richiesta dell'utente ed lo invia al Main. Il Main controlla che l'utente abbia i permessi per il tipo di richiesta, in caso affermativo gestisce, anche indirettamente, il messaggio e risponde.

4.2 Ricezione comando a livello di riga

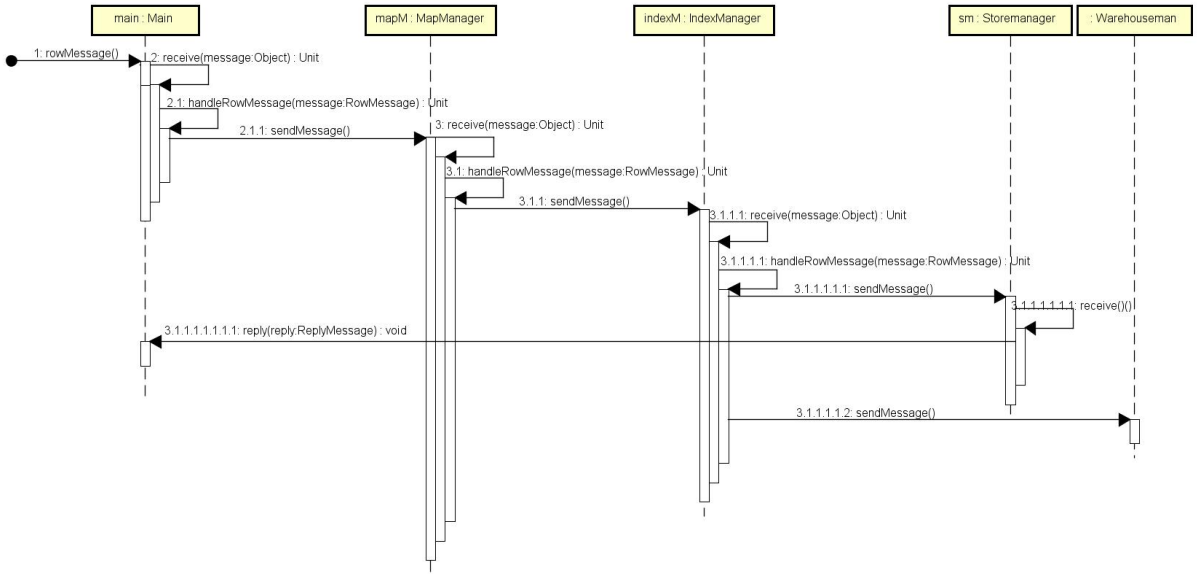


Figura 38: Diagramma di sequenza per la ricezione di un comando a livello riga

Nel diagramma è possibile visualizzare quali sono le operazioni che vengono eseguite dagli attori principali alla ricezione di una richiesta a livello di riga. Ogni attore capisce che è una richiesta a livello di riga e la gestisce mandandola all'attore figlio che potrebbe gestirla. In base a database selezionato, mappa selezionata e chiave, il messaggio arriva al corretto Storemanager. Il messaggio viene inoltra mandato ad ogni Warehouseman responsabile della mappa selezionata.

4.3 Scalabilità Storemanager

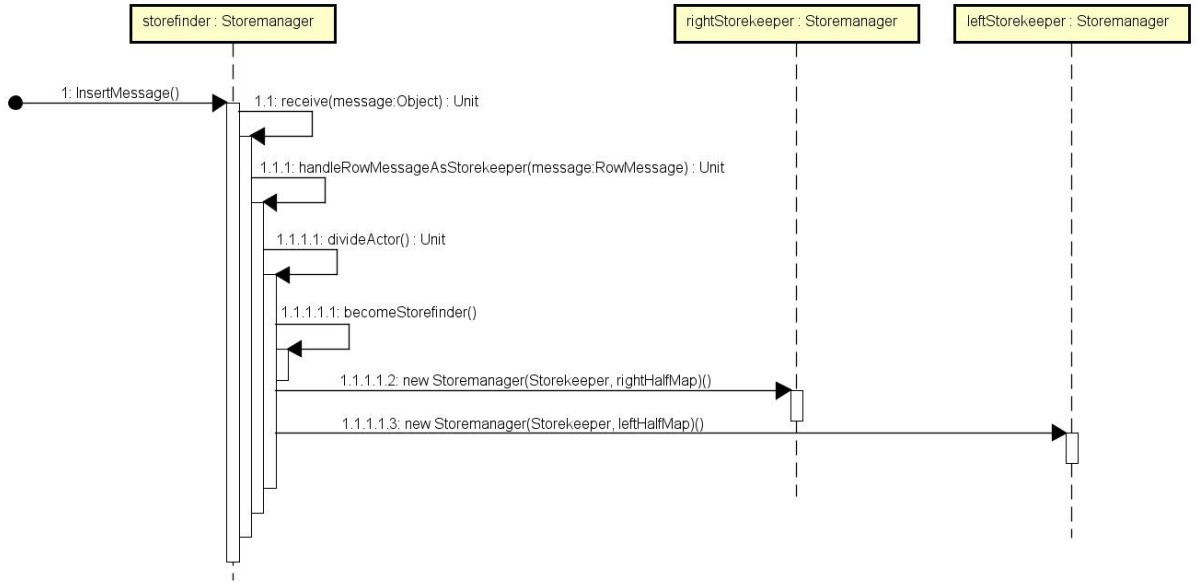


Figura 39: Diagramma di sequenza per la scalabilità degli Storemanager

Nel diagramma è possibile visualizzare quali sono le operazioni che vengono eseguite da un attore Storemanager con comportamento Storekeeper quando la propria mappa raggiunge la grandezza massima

impostata. Per prima cosa cambia comportamento, assumendo quello di uno Storefinder, crea due attori figli di tipo Storemanager con comportamento Storekeeper ed assegna a ciascuno una metà della propria mappa.1

5 Tracciamento

5.1 Tracciamento requisiti-classi

Requisiti	Classi
R[1.1][N][F] e figli	Actorbase.server.Server
R[1.1.3][N][F]	Actorbase.server.ClusterListener, Actorbase.server.actors.ClusterAwareActor
R[1.2][N][F] e figli	Actorbase.server.Server
R[1.3][N][F] e figli	Actorbase.server.actors.Usermanager, Actorbase.server.utils.Parser, Actorbase.server.messages.query.LoginMessage
R[1.4][N][F] e figli	Actorbase.server.actors.Main, Actorbase.server.messages.query.user.DatabaseMessage, Actorbase.server.utils.Parser
R[1.4.1.3][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.4.4.3][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.4.4.4][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.4.5.3][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.4.5.4][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.4.7.3][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.4.7.4][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.5][N][F] e figli	Actorbase.server.actors.MapManager, Actorbase.server.messages.query.user.MapMessage, Actorbase.server.utils.Parser
R[1.5.1.2][D][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.5.2.3][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.5.2.4][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.5.3.3][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.5.3.4][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.5.5.3][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor

Requisiti	Classi
R[1.5.5.4][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.6][N][F] e figli	Actorbase.server.actors.Storemanager, Actorbase.server.messages.query.user.RowMessage, Actorbase.server.utils.Parser
R[1.6.1.2][D][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.6.2.3][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.6.2.4][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.6.3.3][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.6.3.4][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.6.4.3][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.6.4.4][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.6.6.3][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.6.6.4][N][F]	Actorbase.server.messages.query.ReplyMessage, Actorbase.server.utils.ReplyBuilder, Actorbase.server.actors.ReplyActor
R[1.7][N][F] e figli	Actorbase.server.actors.Main
R[2][N][F] e figli	Actorbase.client.Client
R[3.1][N][F] e figli	Actorbase.driver.Driver
R[3.2][N][F] e figli	Actorbase.driver.ConcreteConnection
R[3.3][N][F] e figli	Actorbase.driver.Driver
R[1.4.6.2][N][F] e figli	Actorbase.server.utils
R[1.4.7.1][N][F] e figli	Actorbase.server.utils
R[7][N][V] e figli	Actorbase.server.Server
R[8][N][V] e figli	Actorbase.server.actors.Storemanager

Tabella 2: Tracciamento requisiti-classi

5.2 Tracciamento classi-requisiti

Classi	Requisiti
Actorbase.client.Client	R[2][N][F] e figli
Actorbase.driver.ConcreteConnection	R[3.2][N][F] e figli
Actorbase.driver.Driver	R[3.1][N][F] e figli, R[3.3][N][F] e figli
Actorbase.server.ClusterListener	R[1.1.3][N][F]
Actorbase.server.Server	R[1.1][N][F] e figli, R[1.2][N][F] e figli, R[7][N][V] e figli

Classi	Requisiti
Actorbase.server.actors.ClusterAwareActor	R[1.1.3][N][F]
Actorbase.server.actors.Main	R[1.4][N][F] e figli, R[1.7][N][F] e figli
Actorbase.server.actors.MapManager	R[1.5][N][F] e figli
Actorbase.server.actors.ReplyActor	R[1.4.1.3][N][F], R[1.4.4.3][N][F], R[1.4.4.4][N][F], R[1.4.5.3][N][F], R[1.4.5.4][N][F], R[1.4.7.3][N][F], R[1.4.7.4][N][F], R[1.5.1.2][D][F], R[1.5.2.3][N][F], R[1.5.2.4][N][F], R[1.5.3.3][N][F], R[1.5.3.4][N][F], R[1.5.5.3][N][F], R[1.5.5.4][N][F], R[1.6.1.2][D][F], R[1.6.2.3][N][F], R[1.6.2.4][N][F], R[1.6.3.3][N][F], R[1.6.3.4][N][F], R[1.6.4.3][N][F], R[1.6.4.4][N][F], R[1.6.6.3][N][F], R[1.6.6.4][N][F]
Actorbase.server.actors.Storemanager	R[1.6][N][F] e figli
Actorbase.server.actors.Usermanager	R[1.3][N][F] e figli
Actorbase.server.messages.query.LoginMessage	R[1.3][N][F] e figli
Actorbase.server.messages.query.ReplyMessage	R[1.4.1.3][N][F], R[1.4.4.3][N][F], R[1.4.4.4][N][F], R[1.4.5.3][N][F], R[1.4.5.4][N][F], R[1.4.7.3][N][F], R[1.4.7.4][N][F], R[1.5.1.2][D][F], R[1.5.2.3][N][F], R[1.5.2.4][N][F], R[1.5.3.3][N][F], R[1.5.3.4][N][F], R[1.5.5.3][N][F], R[1.5.5.4][N][F], R[1.6.1.2][D][F], R[1.6.2.3][N][F], R[1.6.2.4][N][F], R[1.6.3.3][N][F], R[1.6.3.4][N][F], R[1.6.4.3][N][F], R[1.6.4.4][N][F], R[1.6.6.3][N][F], R[1.6.6.4][N][F]
Actorbase.server.messages.query.user.DatabaseMessage	R[1.4][N][F] e figli
Actorbase.server.messages.query.user.MapMessage	R[1.5][N][F] e figli
Actorbase.server.messages.query.user.RowMessage	R[1.6][N][F] e figli

Classi	Requisiti
Actorbase.server.utils	R[1.4.6.2][N][F] e figli, R[1.4.7.1][N][F] e figli
Actorbase.server.utils.Parser	R[1.3][N][F] e figli, R[1.4][N][F] e figli, R[1.5][N][F] e figli, R[1.6][N][F] e figli
Actorbase.server.utils.ReplyBuilder	R[1.4.1.3][N][F], R[1.4.4.3][N][F], R[1.4.4.4][N][F], R[1.4.5.3][N][F], R[1.4.5.4][N][F], R[1.4.7.3][N][F], R[1.4.7.4][N][F], R[1.5.1.2][D][F], R[1.5.2.3][N][F], R[1.5.2.4][N][F], R[1.5.3.3][N][F], R[1.5.3.4][N][F], R[1.5.5.3][N][F], R[1.5.5.4][N][F], R[1.6.1.2][D][F], R[1.6.2.3][N][F], R[1.6.2.4][N][F], R[1.6.3.3][N][F], R[1.6.3.4][N][F], R[1.6.4.3][N][F], R[1.6.4.4][N][F], R[1.6.6.3][N][F], R[1.6.6.4][N][F]

Tabella 3: Tracciamento classi-requisiti

5.3 Tracciamento classi-test

Elenco delle figure

1 Actorbase architettura generale . . . 8
2 Componente Actorbase.server . . . 9
3 Classe Actorbase.server.Server . . . 10
4 Classe Actorbase.server.StaticSettings . . . 11
5 Classe Actorbase.server.ClusterListener . . . 13
6 Componente Actorbase.server.utils . . . 15
7 Componente Actorbase.server.utils.Parser . . . 15
8 Componente Actorbase.server.utils.Helper . . . 17
9 Componente Actorbase.server.utils.ConfigurationManager . . . 18
10 Componente Actorbase.server.utils.ReplyBuilder . . . 19
11 Componente Actorbase.server.utils.Serializer . . . 22
12 Componente Actorbase.server.utils.FileManager . . . 23
13 Componente Actorbase.server.actors . . . 29
14 Componente Actorbase.server.actors.Doorkeeper . . . 29
15 Componente Actorbase.server.actors.Usermanager . . . 31
16 Componente Actorbase.server.actors.Main . . . 33
17 Componente Actorbase.server.actors.MapManager . . . 37
18 Componente Actorbase.server.actors.IndexManager . . . 39
19 Componente Actorbase.server.actors.Storemanager . . . 40
20 Componente Actorbase.server.actors.ReplyActor . . . 44
21 Componente Actorbase.server.actors.ClusterAwareActor . . . 45
22 Componente Actorbase.server.actors.Warehouseman . . . 46
23 Actorbase.server.enums . . . 46
24 Componente Actorbase.server.messages . . . 53
25 Componente Actorbase.server.messages.internal . . . 54
26 Componente Actorbase.server.messages.query . . . 62
27 Componente Actorbase.server.messages.query.ErrorMessages . . . 67
28 Componente Actorbase.server.messages.query.PermissionMessages . . . 69
29 Componente Actorbase.server.messages.query.admin . . . 73
30 Componente Actorbase.server.messages.query.user . . . 81
31 Componente Actorbase.server.messages.query.user.RowMessages . . . 82
32 Componente Actorbase.server.messages.query.user.MapMessages . . . 89
33 Componente Actorbase.server.messages.query.user.DatabaseMessages . . . 95
34 Componente Actorbase.server.messages.query.user.HelpMessages . . . 101
35 Componente Actorbase.client . . . 104
36 Componente Actorbase.driver . . . 106
37 Diagramma di sequenza per la ricezione di un messaggio . . . 110
38 Diagramma di sequenza per la ricezione di un comando a livello riga . . . 111
39 Diagramma di sequenza per la scalabilità degli Storemanager . . . 111

Elenco delle tabelle

1 Diario delle modifiche 5
2 Tracciamento requisiti-classi 114
3 Tracciamento classi-requisiti 116