SWEeneyThreads

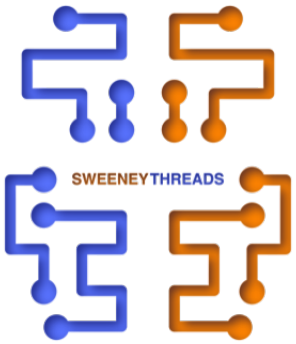
Actorbase

a NoSQL DB based on the Actor model

Definizione di prodotto

Redattori:
Maino Elia

Approvazione:
...
Verifica:
...



Versione 0.0.4

7 giugno 2016

Indice

Diario delle modifiche

Versione	Data	Autore	Descrizione
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

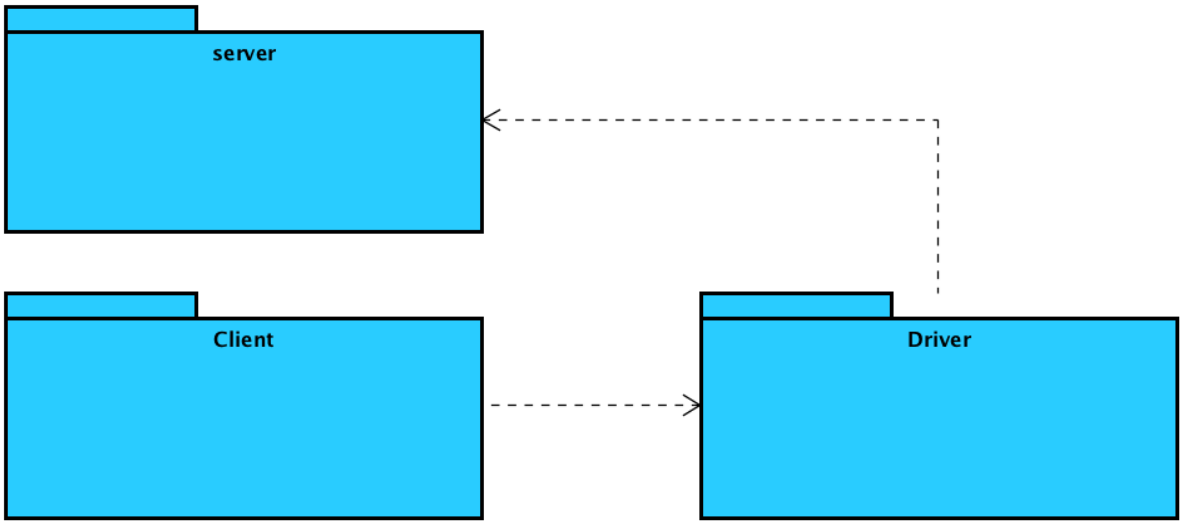


Figura 1: Actorbase architettura generale

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

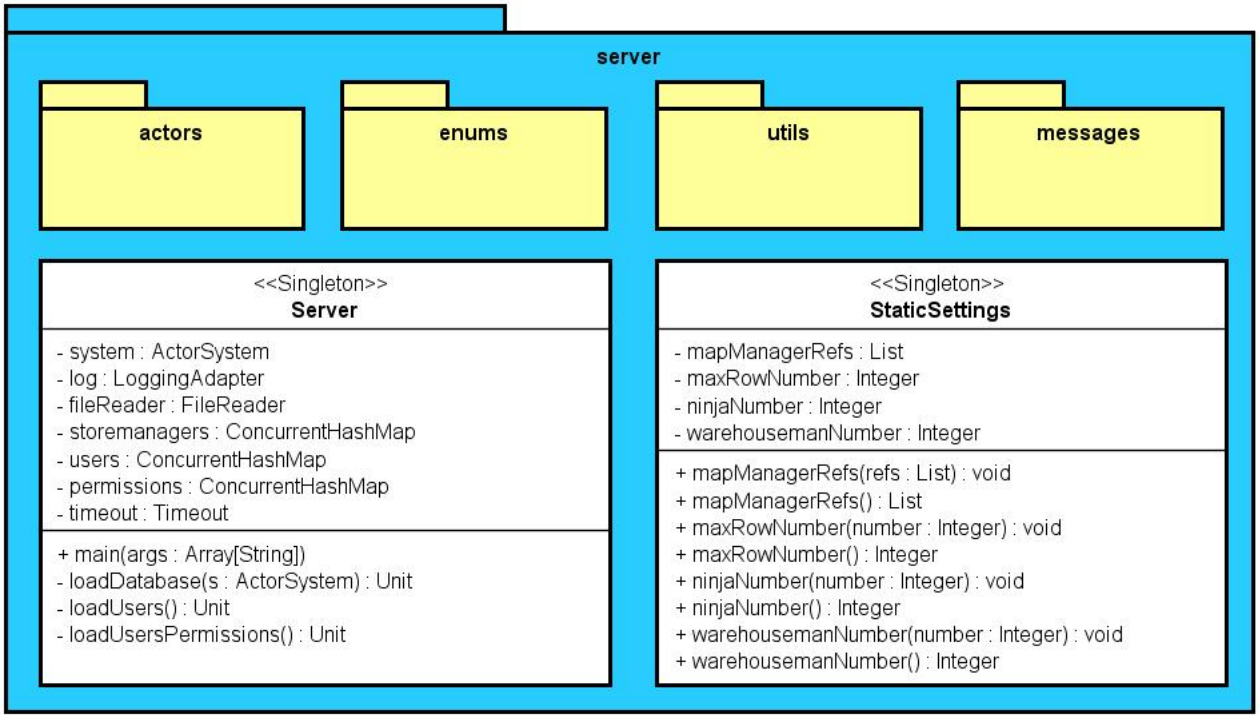


Figura 2: Componente Actorbase.server

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)

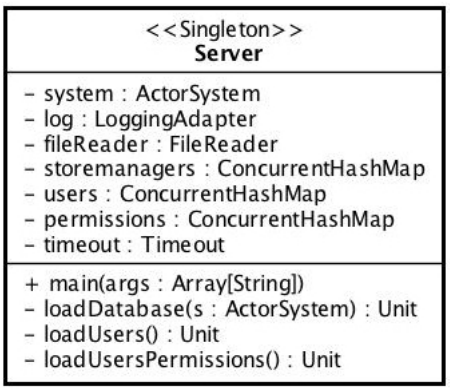


Figura 3: Classe Actorbase.server.Server

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)

Immagine UML.

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

- `var mapManagerRefs: ConcurrentHashMap[String, ActorRef]` - Riferimento ai `MapManger`.
- `var maxRowNumber: Integer` - Numero massimo di righe.
- `var ninjaNumber: Integer` - Numero di `Ninja`.
- `var warehousemanNumber: Integer` - Numero di `Warehousean`.

3.5 Actorbase.server.ClusterListener

Immagine UML.

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

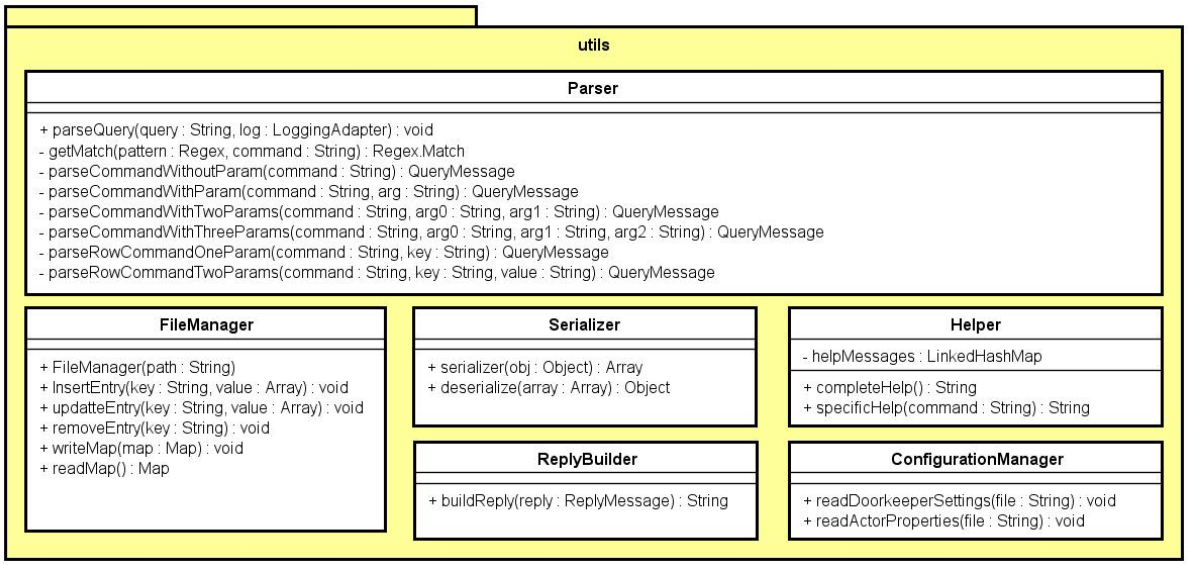


Figura 4: Componente Actorbase.server.utils

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

Immagine UML.

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: `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: `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: `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: `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: `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: `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: `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.FileManager

Immagine UML.

Descrizione

Classe che definisce 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

Nessuno.

Attributi

- `path: String` - Il percorso su cui effettuare le letture e le scritture.
- ...

Metodi

`Firma del metodo`

Descrizione del metodo.

Lista parametri del metodo:

- `Nome parametro: tipo parametro` - Descrizione parametro

3.9 Actorbase.server.utils.Helper

Immagine UML.

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

- `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.

3.10 Actorbase.server.utils.ConfigurationManager

Immagine UML.

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.

3.11 Actorbase.server.utils.ReplyBuilder

Immagine UML.

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: `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: `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: `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: `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: `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: `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: `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: `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: `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

3.12 Actorbase.server.utils.Serializer

Immagine UML.

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.

3.13 Actorbase.server.actors

Immagine UML del package e breve descrizione.

3.14 Actorbase.server.actors.Doorkeeper

Immagine UML.

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.

3.15 Actorbase.server.actors.Usermanager

Immagine UML.

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

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

3.16 Actorbase.server.actors.Main

Immagine UML.

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

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

3.17 Actorbase.server.actors.MapManager

Immagine UML.

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 def 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 def 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.

3.18 Actorbase.server.actors.IndexManager

Immagine UML.

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 def 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.

3.19 Actorbase.server.actors.Storemanager

Immagine UML.

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

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

3.20 Actorbase.server.actors.ReplyActor (trait)

Immagine UML.

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.

3.21 Actorbase.server.actors.ClusterAwareActor (trait)

Immagine UML.

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.

3.22 Actorbase.server.actors.Warehouseman

TODO

3.23 Actorbase.server.enums

Immagine UML del package e breve descrizione.

3.24 Actorbase.server.enums.Permission (trait)

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.25 Actorbase.server.enums.ReplyResult (trait)

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.26 Actorbase.server.enums.Read

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.27 Actorbase.server.enums.Write

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.
Immagine UML. **Descrizione** Descrizione testuale. **Utilizzo** Descrizione testuale. **Classi ereditate**

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi `Firma del metodo`
Descrizione del metodo.
Parametri

- `Nome parametro: tipo parametro` - Descrizione parametro

3.28 Actorbase.server.enums.Done

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.29 Actorbase.server.enums.Error

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.30 Actorbase.server.enums.EnumPermission (enumeration)

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.31 Actorbase.server.enums.EnumReplyResult (enumeration)

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.32 Actorbase.server.messages

Immagine UML del package e breve descrizione.

3.33 Actorbase.server.messages.internal

Immagine UML del package e breve descrizione.

3.34 Actorbase.server.messages.internal.AskMapMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe

- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.35 Actorbase.server.messages.internal.BecomeAStorekeeperMsg

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.36 Actorbase.server.messages.internal.SendMapMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.37 Actorbase.server.messages.internal.LinkMessages

Immagine UML del package e breve descrizione.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.39 Actorbase.server.messages.internal.LinkMessages.AddNinjaMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.40 Actorbase.server.messages.internal.LinkMessages.AddWarehousemanMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.41 Actorbase.server.messages.internal.LinkMessages.RemoveNinjaMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.42 Actorbase.server.messages.internal.LinkMessages.RemoveWarehousemanMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.43 Actorbase.server.messages.query

Immagine UML del package e breve descrizione.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.45 Actorbase.server.messages.query.LoginMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.46 Actorbase.server.messages.query.ReplyMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.47 Actorbase.server.messages.query.ErrorMessages

Immagine UML del package e breve descrizione.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo

- ...

Metodi

Nessuno.

3.49 Actorbase.server.messages.query.ErrorMessages.InvalidQueryMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.50 Actorbase.server.messages.query.PermissionMessages

Immagine UML del package e breve descrizione.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.55 Actorbase.server.messages.query.admin

Immagine UML del package e breve descrizione.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.57 Actorbase.server.messages.query.admin.ActorPropertiesMessages

Immagine UML del package e breve descrizione.

3.58 Actorbase.server.messages.query.admin.ActorPropertiesMessages.ActorPropertiesM (trait)

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.59 Actorbase.server.messages.query.admin.ActorPropertiesMessages.MaxRowMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.60 Actorbase.server.messages.query.admin.ActorPropertiesMessages.MaxRowMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe

- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.61 Actorbase.server.messages.query.admin.ActorPropertiesMessages.SetNinjaMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.62 Actorbase.server.messages.query.admin.ActorPropertiesMessages.MaxNinjaMessag

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.63 Actorbase.server.messages.query.admin.ActorPropertiesMessages.SetWarehousemanMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.64 Actorbase.server.messages.query.admin.ActorPropertiesMessages.MaxWarehousemanMessage

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.65 Actorbase.server.messages.query.admin.ActorPropertiesMessages.MaxStorekeeperM

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.66 Actorbase.server.messages.query.admin.ActorPropertiesMessages.MaxStorefinderM

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.67 Actorbase.server.messages.query.admin.PermissionsManagementMessages

Immagine UML del package e breve descrizione.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.72 Actorbase.server.messages.query.admin.UserManagementMessages

Immagine UML del package e breve descrizione.

3.73 Actorbase.server.messages.query.admin.UserManagementMessages.UserManageme (trait)

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.74 Actorbase.server.messages.query.admin.UserManagementMessages.AddUserMessag

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.75 Actorbase.server.messages.query.admin.UserManagementMessages.RemoveUserMe

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.76 Actorbase.server.messages.query.admin.UserManagementMessages.ListUserMessag

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.77 Actorbase.server.messages.query.user

Immagine UML del package e breve descrizione.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.79 Actorbase.server.messages.query.user.RowMessages

Immagine UML del package e breve descrizione.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.86 Actorbase.server.messages.query.user.MapMessages

Immagine UML del package e breve descrizione.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo

- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.92 Actorbase.server.messages.query.user.DatabaseMessages

Immagine UML del package e breve descrizione.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.98 Actorbase.server.messages.query.user.HelpMessages

Immagine UML del package e breve descrizione.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

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

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

Nessuno.

3.102 Actorbase.client

Immagine UML del package e breve descrizione.

3.103 Actorbase.client.Client

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo

- ...

Metodi

`Firma del metodo`

Descrizione del metodo.

Lista parametri del metodo:

- `Nome parametro: tipo parametro` - Descrizione parametro

3.104 Actorbase.client.Welcome

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

`Firma del metodo`

Descrizione del metodo.

Lista parametri del metodo:

- `Nome parametro: tipo parametro` - Descrizione parametro

3.105 Actorbase.driver

Immagine UML del package e breve descrizione.

3.106 Actorbase.driver.Connection (trait)

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

`Firma del metodo`

Descrizione del metodo.

Lista parametri del metodo:

- `Nome parametro: tipo parametro` - Descrizione parametro

3.107 Actorbase.driver.ConcreteConnection

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

`Firma del metodo`

Descrizione del metodo.

Lista parametri del metodo:

- `Nome parametro: tipo parametro` - Descrizione parametro

3.108 Actorbase.driver.Driver

Immagine UML.

Descrizione

Descrizione testuale.

Utilizzo

Descrizione testuale.

Classi ereditate

- Classe
- ...

Ereditata da

- Classe
- ...

Attributi

- `Nome attributo: tipo attributo` - Descrizione attributo
- ...

Metodi

`Firma del metodo`

Descrizione del metodo.

Lista parametri del metodo:

- `Nome parametro: tipo parametro` - Descrizione parametro

4 Diagrammi di sequenza

5 Tracciamento

5.1 Tracciamento requisiti-classi

5.2 Tracciamento classi-requisiti

5.3 Tracciamento classi-test

Elenco delle figure

Elenco delle tabelle