Mathematical Expressions Evaluator Server

di Francesco Benincasa

Introduzione

Il progetto si poneva come obiettivo la realizzazione di un server per la valutazione di espressioni matematiche, come descritto nelle specifiche del progetto fornite dal docente del corso. Sono apportate le seguenti aggiunte rispetto alle specifiche originarie:

- Gli operatori usati nelle espressioni matematiche seguono l'ordine "normale" di esecuzione: `^, */, +-`
- È possibile inserire degli spazi tra i token della grammatica (è sempre possibile evitare questo comportamento commentando la riga nella definizione della grammatica `WS: [ \t]+ -> channel(HIDDEN);`
  (per rendere operativa la modifica è necessario ricompilare il progetto).
- È possibile utilizzare espressioni del tipo `-x0`.

La grammatica usata nel progetto è definita mediante il file:

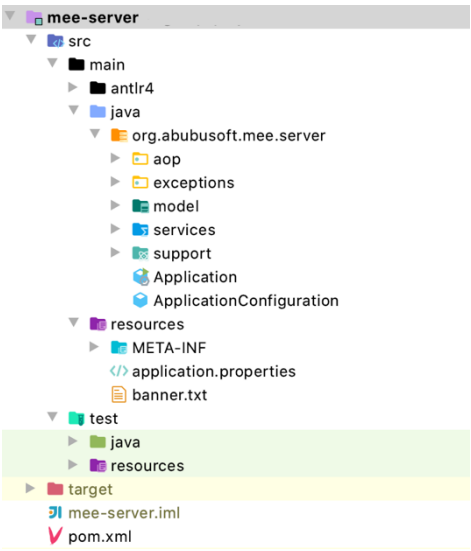
`src/main/antlr4/org/abubusoft/mee/server/grammar/Commands.g4`

Contenuta nei sorgenti del progetto.

Piattaforma tecnologica

La soluzione è stata realizzata con e per OpenJDK 11.

Tecnologie utilizzate



Sono state utilizzate diverse tecnologie per realizzare il progetto. Di seguito sono riportate quelle principali:

Si è stato utilizzato Maven (versione 3.6.3) per la gestione del progetto. Come conseguenza di ciò, i sorgenti del progetto sono organizzati seguendo la convenzione di Maven stesso.

Come piattaforma di source version control e di issue management si è utilizzata la piattaforma GitHub mediante un progetto privato.

Si riportano di seguito un elenco dei principali framework e le librerie utilizzati nel progetto:

- Spring framework: framework scelto come container Inversion of Control (IoC). Lo stesso è stato utilizzato per realizzare il componente AOP[1] definito per monitorare le statistiche di esecuzione dei comandi.
- Spring boot: utilizzato per semplificare l'utilizzo di Spring.
- ANTLR[2]: libreria utilizzata per generare l'analizzatore sintattico lessicale necessario a gestire i comandi gestiti dal server.
- JUnit: framework utilizzato per testare i componenti "core" del progetto.
- Varie librerie di utilità: Google Guava, Apache commons IO, Apache commons lang.

Gestione del progetto e dei sorgenti

- Maven
- Git
- GitHub
- Travis[3]

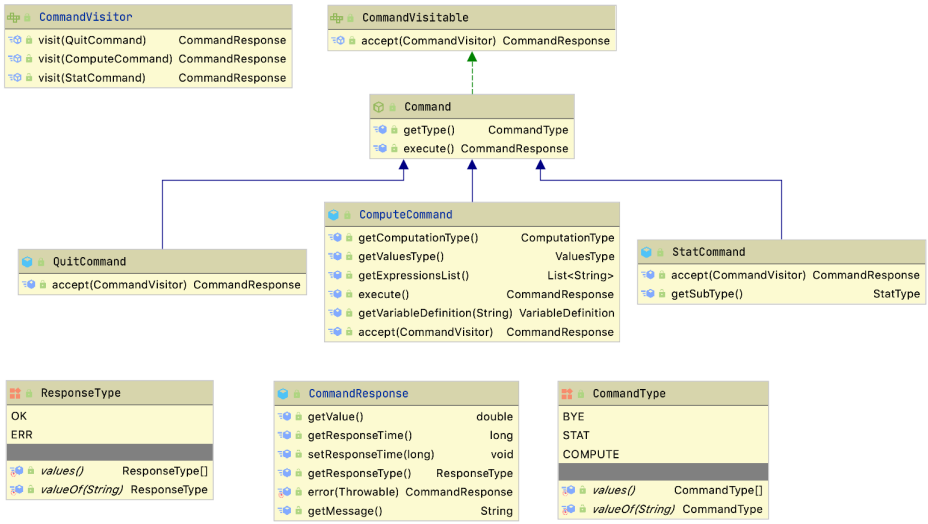
[1] https://it.wikipedia.org/wiki/Programmazione_orientata_agli_aspetti
[2] https://www.antlr.org/
[3] https://travis-ci.com/

Tools ed ambienti di sviluppo utilizzati

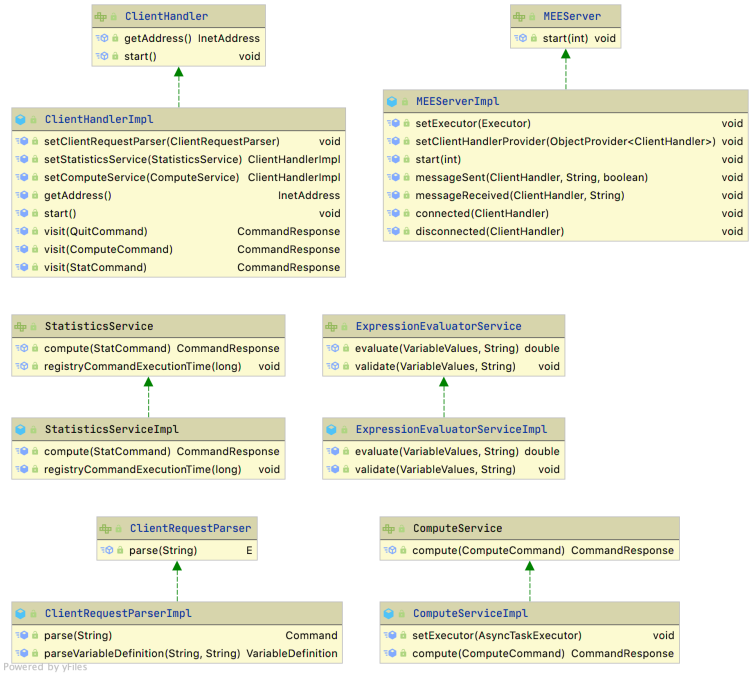
- IntelliJ
- IntellJ ANTLR v4 Grammar Plugin

Struttura del progetto

Dal punto di vista progettuale, MEE Server può essere riassunto mediante due class diagram, uno per le classi che rappresentano il modello dei dati e delle classi che rappresentano i servizi.



Segue il class diagram dei servizi e delle relative interfacce.



Compilazione

Si assume che il codice sorgente sia stato scompattato in una cartella. Per compilare il progetto è necessario aver installato un JDK 11 (OpenJDK) e Maven (versione utilizzata 3.6.3). Una volta aperta una CLI sulla cartella con i sorgenti scompattati, eseguire il seguente comando:

```
mvn clean package
```

L'esecuzione di tale comando produce l'artifact `./target/BenincasaFrancesco.jar`. Il jar contiene, oltre alle classi del progetto vero e proprio, tutte le classi delle librerie e dei framework utilizzate (escluse ovviamente quelle appartenenti al JDK).

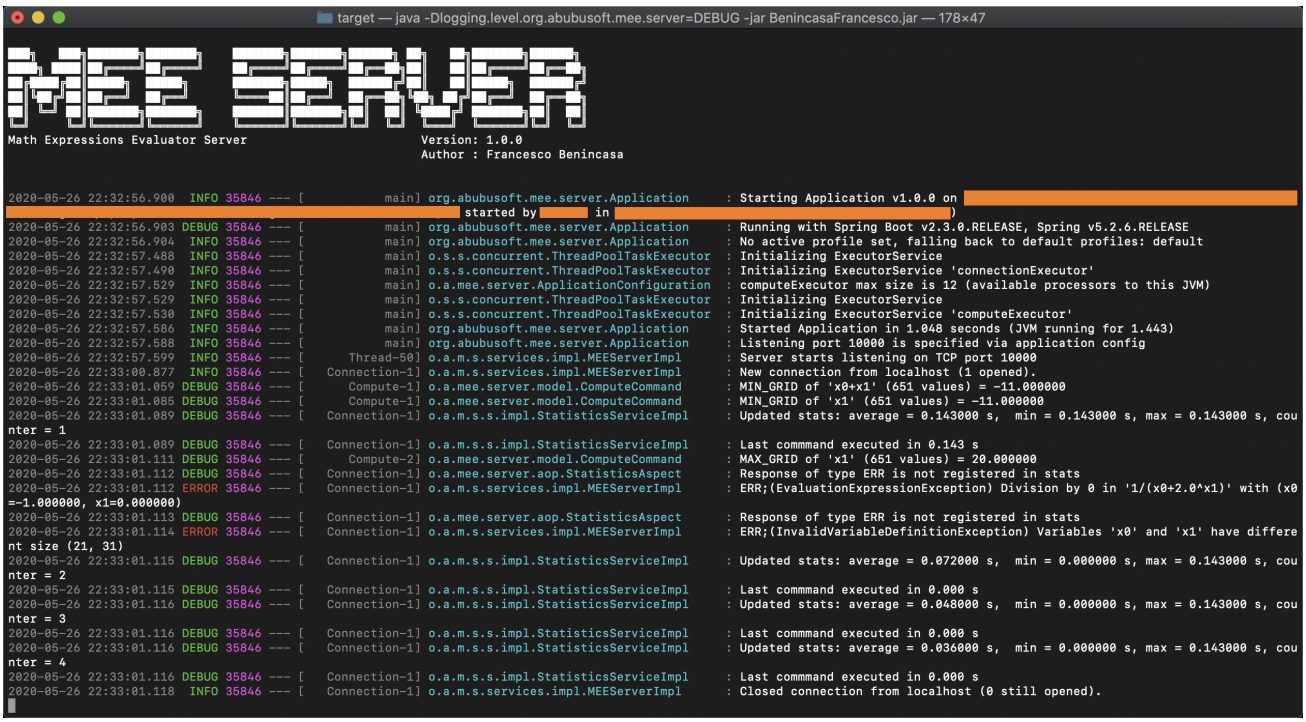
Esecuzione

Assieme a questo documento è allegato il jar prodotto dall'ultima build eseguita dal progetto. È possibile eseguire l'applicativo con il comando `java -jar ./BenincasaFrancesco.jar`.

Non serve specificare la porta, che di default è impostata a `10000`. Nel caso si desideri mettere in ascolto l'applicativo su un'altra porta è sufficiente eseguire il comando

```
java -jar ./BenincasaFrancesco.jar ${p}
```

Dove `${p}` rappresenta il numero di porta di ascolto del server.



Log

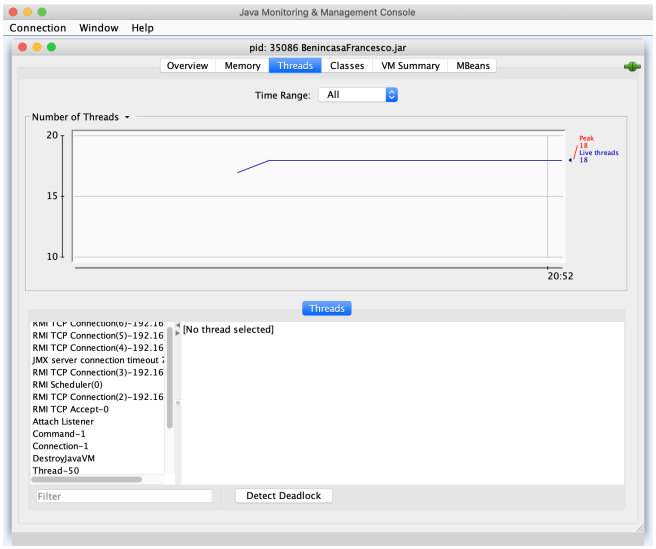
Il livello di log impostato durante la build è a livello `INFO`. A questo livello vengono visualizzate le informazioni inerenti informazioni sull'avvio dell'app, delle nuove connessioni, delle connessioni chiuse e degli eventuali comandi con errori. Per impostare un maggiore livello di dettaglio dei log, è possibile eseguire l'applicazione impostando il livello di log a `DEBUG` e `TRACE`:

```
java -Dlogging.level.org.abubusoft.mee.server=DEBUG -jar BenincasaFrancesco.jar
```

Ogni voce del log contiene l'informazione relativa al thread con il quale si è invocata l'operazione stessa. Questa informazione può risultare utile per verificare il numero di thread attivi per la valutazione delle espressioni matematiche (sono i thread con prefisso `Compute`).

Monitoraggio Thread

È possibile monitorare diversi aspetti dell'applicativo e della relativa Java Virtual Machine eseguendo a linea di comando l'applicativo `jconsole` (distribuito assieme al JDK).



Client

È possibile collegarsi al server ovviamente mediante telnet. Al fine testare il comportamento del server durante l'utilizzo di più client, è stato realizzato un client che consente di mandare al server tutti i comandi presenti su un file. Nel file allegato è presente un file `commands.txt` contenente alcuni comandi di test. I file possono contenere righe vuote e righe commenti (che iniziano con il carattere `#`). L'artifact `mee-client-1.0.0-jar-with-dependencies.jar` (ed i relativi sorgenti) sono inclusi nel secondo allegato `client.zip` (sono stati messi a parte in quanto non richiesti al fine valutativo del progetto). Sono stati inclusi ad eventuale beneficio del Professore nel caso in cui voglia utilizzarlo per testare l'app.

```
target — -zsh — 94×18
java -jar mee-client-1.0.0-jar-with-dependencies.jar -h
Usage: MEE Client Application [-hV] [-d=<delay>] [-p=<port>] [-r=<repeat>]
                              [-s=<host>] [-t=<threadPoolSize>] <file>
Generate somes calls to the specified MEE-server.
      <file>                File containg commands
  -d, --delay=<delay>       wait time (ms) before client send another command (in
                              the same connection)
  -h, --help                Show this help message and exit.
  -p, --port=<port>         port used by the MEE server. Default value is 10000.
  -r, --repeat=<repeat>     how time time the sequence of command is repated.
                              Default value is 1.
  -s, --server=<host>       IP or name of the MEE server. Default value is
                              'localhost'.
  -t, --threads=<threadPoolSize>
                            used threads. Default value is 1.
  -V, --version             Print version information and exit.
%
```

```
java -jar mee-client-1.0.0-jar-with-dependencies.jar commands.txt
```