Mathematical Expressions Evaluator Server

di Francesco Benincasa

Introduzione

Il progetto si pone l'obiettivo di implementare un server per la valutazione di espressioni matematiche, come descritto nei requisiti forniti per progetto finale del corso. La soluzione è stata realizzata con e per la piattaforma Java OpenJDK 11. Per la gestione del progetto è stato utilizzato Maven. L'IDE utilizzato è IntelliJ.

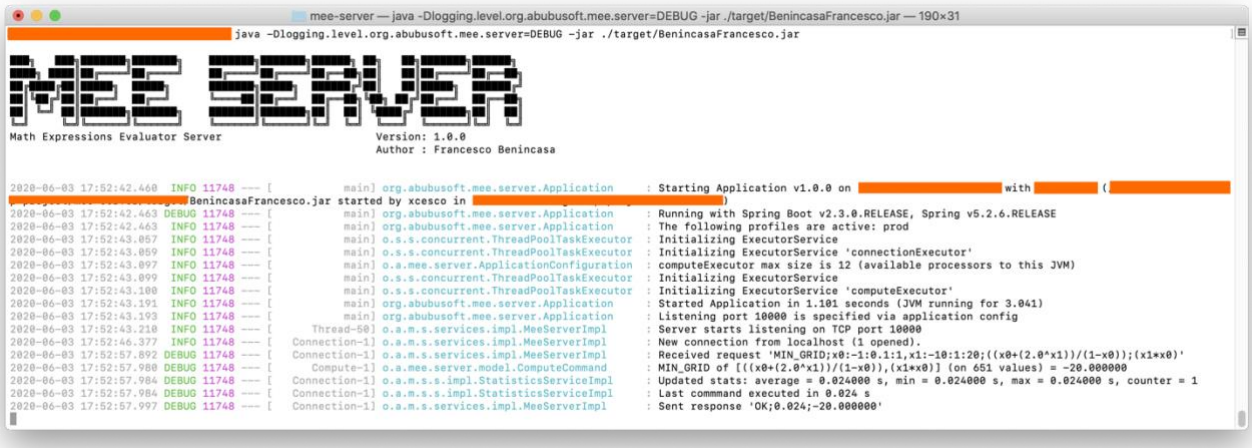
Esecuzione

Il JAR prodotto dall'ultima build del progetto è incluso nello ZIP assieme al presente documento. È possibile eseguire l'applicativo con il comando `java -jar BenincasaFrancesco.jar`.

La porta 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}
```

Il placeholder `${p}` rappresenta il numero di porta di ascolto del server. Alla prima esecuzione, il programma scaricherà dal repository centrale di maven gli artifact necessari alla sua esecuzione (scelta fatta per avere una JAR di dimensioni ridotte). L'operazione risulta essere trasparente e non comporta alcun intervento, se non quello di eseguire il programma da un computer da cui risulti accessibile il repository centrale di Maven.



Profili e log

L'esecuzione dell'applicazione senza parametri consente di eseguire il server in modalità `prod (produzione)`: in questa configurazione, nel log applicativo, vengono visualizzate solo le informazioni sull'avvio, delle connessioni aperte, di quelle chiuse, e degli eventuali comandi con errori (il livello di log impostato è il livello `INFO`). Qualora si desideri aumentare il livello di dettaglio dei log, è possibile agire su due parametri della JVM usata per eseguire il programma: quello per selezionare il livello dei log dell'applicativo e quello per definire il profilo Spring da utilizzare (oltre a `prod`, usato di default, è stato definito il profilo `dev`). Si riportano alcuni esempi di esecuzione dell'applicazione con l'utilizzo dei due parametri citati:

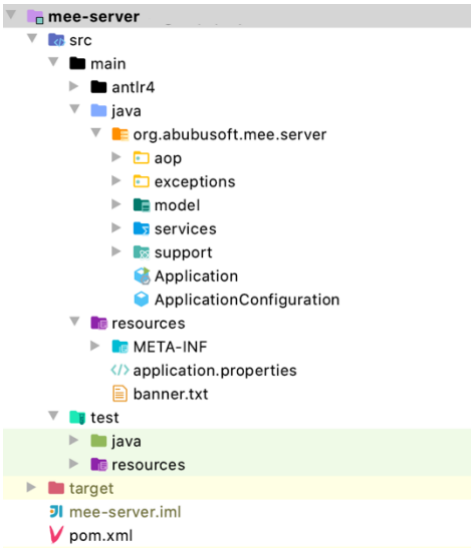
```
java -Dlogging.level.org.abubusoft.mee.server=DEBUG -jar BenincasaFrancesco.jar
```

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

```
java -Dspring.profiles.active=dev -jar BenincasaFrancesco.jar
```

Tra le informazioni presenti nelle voci di log è presente il thread usato. I thread utilizzati per calcolare il valore delle espressioni matematiche sono quelli con il nome con prefisso `Compute`.

Tecnologie utilizzate



Per la gestione del progetto si è utilizzato Maven (versione 3.6.3). 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.

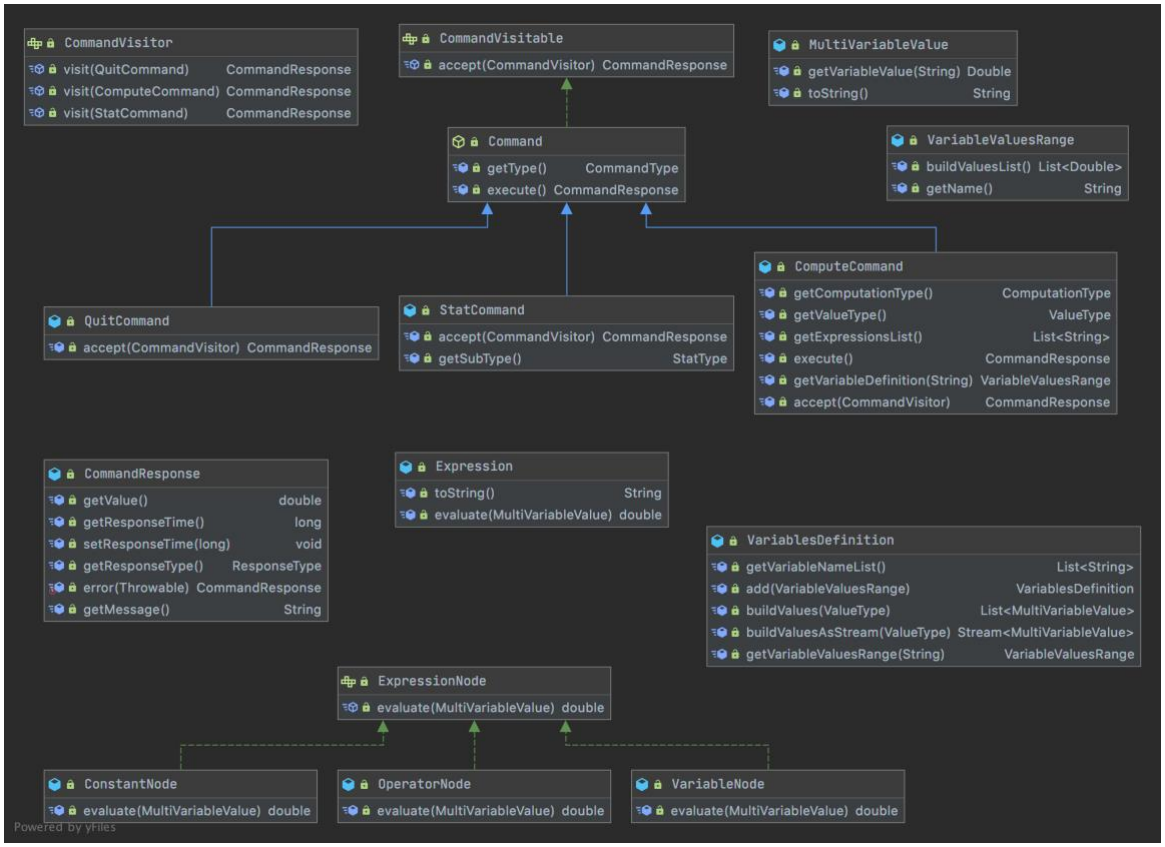
Di seguito sono elencati i principali framework e librerie utilizzati nel progetto:

- **Spring framework**: framework utilizzato 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 ad analizzare le richieste effettuate dai client. La grammatica definita per il progetto con le convenzioni ANTLR è definita nel file compreso nei sorgenti dell'applicativo `src/main/antlr4/org/abubusoft/mee/server/grammar/Commands.g4`.
- **JUnit**: framework utilizzato per testare i componenti "core" del progetto.
- Google Guava.

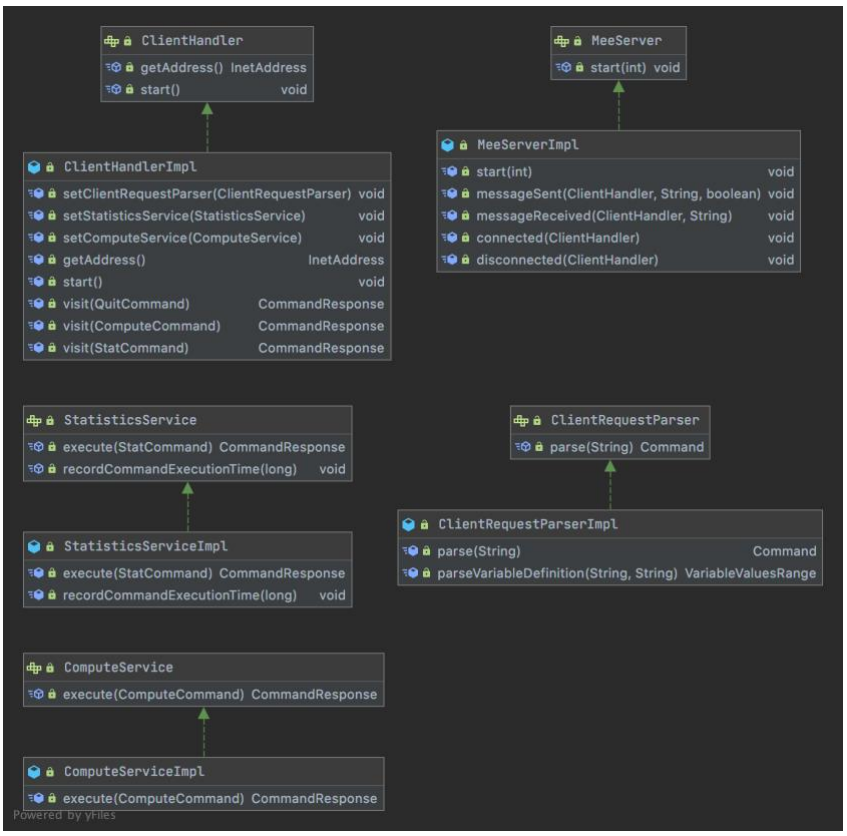
Struttura del progetto

Si riportano le principali entità definite nel progetto mediante due diagrammi delle classi; uno per il data model ed un altro per le classi definiscono la logica di business. Si riporta di seguito il class diagram del modello dati:

1 https://it.wikipedia.org/wiki/Programmazione_orientata_agli_aspetti
2 https://www.antlr.org/



Il class diagram della logica di business:



Compilazione del progetto

Si assume che il codice sorgente sia stato scompattato in una cartella. Per compilare il progetto è necessario aver installato il JDK 11 o superiore (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`. L'esecuzione di questo artifact mediante Java, al momento del primo avvio, comporta il download di tutte le dipendenze del progetto necessarie (mediante Maven) escluse ovviamente quelle appartenenti al JDK.

Qualora si desideri eseguire ottenere l'artifact `BenincasaFrancesco.jar` comprensivo di tutte le classi utilizzate dall'applicativo (incluse quelle delle librerie e dei framework utilizzati), è necessario effettuare la build del progetto con il comando:

```
mvn -Pflat clean package
```

L'artifact generato si chiamerà anche in questo caso `BenincasaFrancesco.jar`; in questo caso, invece di avere una dimensione di circa 120KB, avrà una dimensione di circa 13MB.