Http2Java - Manuale Utente

Pellegrinelli Nico, Pellegrinelli Sean, Panzeri Federico  
Versione del 24/01/2024

Sommario

[Indice delle figure 2](#_Toc157459713)

[Introduzione 3](#_Toc157459714)

[Requisiti di Sistema 3](#_Toc157459715)

[Installazione 3](#_Toc157459716)

[Panoramica 4](#_Toc157459717)

[Descrizione elementi principali 4](#_Toc157459718)

[Funzioni principali 7](#_Toc157459719)

[Inserimento e visualizzazione del contenuto della chiamata HTTP da convertire 7](#_Toc157459720)

[Inserimento e visualizzazione di un file contenente la chiamata HTTP da convertire 8](#_Toc157459721)

[Traduzione da testo HTTP a codice Java 9](#_Toc157459722)

[Visualizzazione del codice Java, di errori e warnings 9](#_Toc157459723)

[Copia negli appunti del codice Java 10](#_Toc157459724)

[Grammatica 10](#_Toc157459725)

[Lessico 10](#_Toc157459726)

[Sintassi 11](#_Toc157459727)

[Conclusioni 15](#_Toc157459728)

# Introduzione

Benvenuto in Http2Java, la soluzione innovativa per semplificare l’integrazione delle chiamate HTTP nei progetti Java aziendali. Questa applicazione, sviluppata interamente in linguaggio Java, è progettata per rendere il processo di creazione delle chiamate HTTP più rapido, intuitivo ed efficiente.

Http2Java, tramite un compilatore progettato appositamente, genera il codice Java corrispondente alla chiamata HTTP che si vuole implementare. Questo progetto nasce dal desiderio di automatizzare e velocizzare un processo aziendale interno che consenta a chiunque, sia junior che senior, di risparmiare tempo prezioso ed aumentare la produttività.

## Requisiti di Sistema

Questo software viene distribuito sottoforma di file ***jar.***

Requisiti di sistema

Software:

* Java SE 1.8 o versioni successive installato sul PC. L’ultima versione di Java può essere scaricata ed installata da [qui](https://www.java.com/)

Sistema Operativo:

* Windows Server 2008 R2 SP1 (64 bit) e versioni successive
* Windows Vista SP2 e versioni successive
* Mac OS X 10.8.3+
* Ubuntu Linux 5.5+
* Oracle Solaris 10 Update 9+ (64 bit)

In tutti i casi sono richiesti i privilegi di amministratore

RAM:

* 128 MB per sistemi Windows
* 256 MB per tutti gli altri

Spazio di archiviazione

* > 300 MB disponibili

## Installazione

Una volta installata l’ultima versione di Java dal [sito ufficiale](https://www.java.com/), per eseguire l’applicativo è sufficiente scaricare e lanciare il file *jar* distribuito dall’azienda.

# Panoramica

L’applicazione, appena avviata, si presenta nel seguente modo:

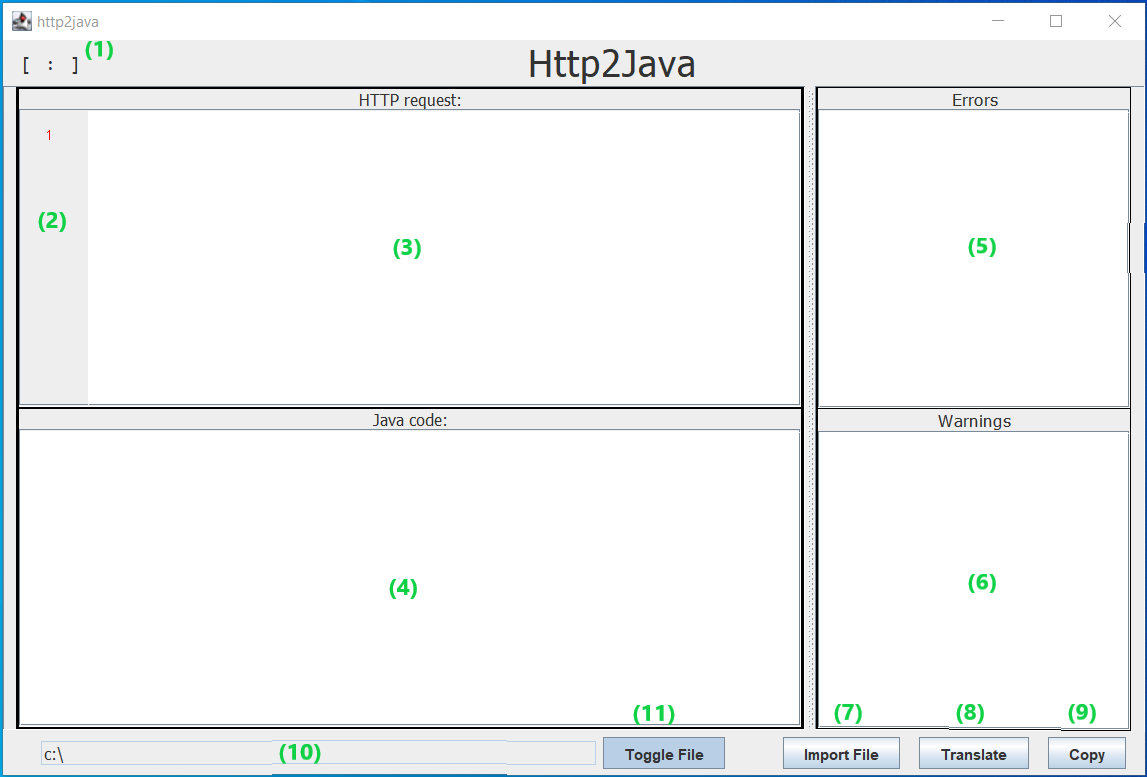


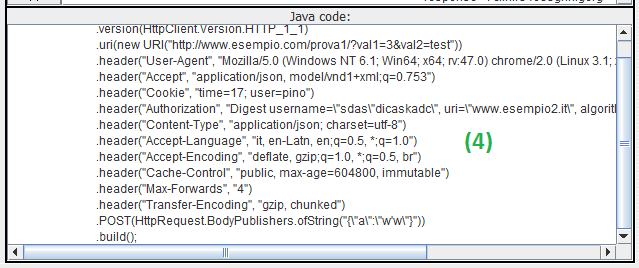
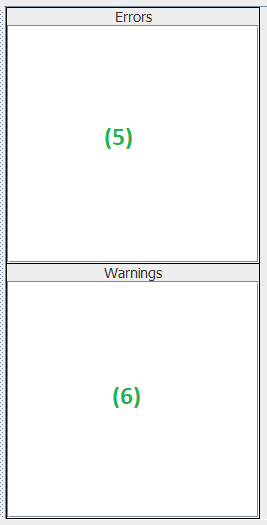
Figura : Schermata Principale

Questa schermata è di tipo responsive, nel senso che le dimensioni dei vari elementi si adattano dinamicamente in base alle dimensioni della finestra.

## Descrizione elementi principali

Per la numerazione degli elementi fare riferimento alla figura 1 qui sopra.

1. Immagine che contiene testo, schermata, schermo, numero

   Descrizione generata automaticamente(In verde) Indica, nel formato riga:colonna, la posizione del cursore all’interno della casella di testo nella figura qui di fianco. Quando la casella di testo perde il focus, l’indicatore non mostra alcun valore
2. (In blu) Indica, per ogni riga della casella di testo, il suo numero. Il numero della riga che attualmente contiene il cursore viene evidenziato in rosso (la riga attualmente selezionata è la 7, cerchiata in rosso).
3. È la casella di testo utilizzata per visualizzare ed eventualmente modificare il testo HTTP che si vuole tradurre. Lavora in accoppiata agli elementi 1) e 2). Qualora una riga occupi più spazio in orizzontale della larghezza della casella, i caratteri in eccesso vengono riportati a capo, ma per loro non viene aggiunto un nuovo numero di riga e pertanto il relativo spazio a sinistra nell’elemento 2) resta vuoto.
4. (Vedi a sx) È la casella di testo dove viene visualizzato il codice Java risultante dalla traduzione. L’utente può solo visualizzarne il contenuto, senza modificarlo. Se la traduzione ha esito negativo, ossia si verifica almeno un errore, questo spazio rimane vuoto in attesa che tutti gli errori vengano risolti.
5. È la casella di testo dedicata alla visualizzazione eventuale degli errori. Per ogni errore lanciato dal compilatore, viene generata una stringa di testo contenente il messaggio d’errore e prefissa dalla posizione (espressa in riga:colonna) del token che lo ha generato. Qualora si verifichi più di un errore, i relativi messaggi vengono mostrati ordinati secondo la relativa riga che li ha generati.

La presenza di anche solo un errore blocca il processo di traduzione fino a che tutti gli errori non sono stati risolti

1. È la casella di testo dedicata alla visualizzazione di eventuali warnings.

La presenza di anche un solo warning non influisce sul buon esito della procedura di traduzione. Qualora il compilatore restituisca più di un warning, i relativi messaggi vengono stampati in ordine di generazione, senza mostrare la posizione all’inizio della riga.



1. (“Import File”) È il pulsante utilizzato per importare un file. Alla pressione, viene aperta una nuova finestra in cui si chiede all’utente di scegliere il file da importare. Se l’importazione avviene correttamente, il path del file viene mostrato nella barra di testo a sinistra, altrimenti se si verifica un errore o se l’operazione viene annullata viene mostrato un messaggio di errore nella barra di testo a sinistra.
2. (“Translate”) È il pulsante cha avvia la traduzione vera e propria
3. (“Copy”) È il pulsante che copia negli appunti il codice Java generato dalla traduzione. Se la traduzione non è andata a buon fine, viene stampato un messaggio di errore nella barra di testo a sinistra.
4. (Barra di testo) Barra di testo utilizzata per mostrare il path del file importato o in alternativa alcuni messaggi di errore. Il suo contenuto non può essere modificato manualmente dall’utente
5. (“Toggle File”) È il pulsante che specifica se il testo da tradurre deve essere prelevato dalla casella di testo 3) oppure dal file importato. Può assumere due stati differente: “attivo” e “passivo”. Se si trova nello stato “attivo”, il programma preleverà il testo da tradurre dal file specificato nell’operazione di import, altrimenti, nel caso contrario, dalla casella di testo 3).  
   All’avvio dell’applicazione questo pulsante è impostato di default nello stato “attivo”.

# Funzioni principali

Le funzioni principali implementate sono:

1. Inserimento e visualizzazione del contenuto della chiamata HTTP da convertire
2. Importazione e visualizzazione di un file contenente la chiamata HTTP da convertire
3. Traduzione da testo HTTP a codice Java
4. Visualizzazione del codice Java, di errori e warnings
5. Copia negli appunti del codice Java

### Inserimento e visualizzazione del contenuto della chiamata HTTP da convertire

Questa operazione consiste nell’inserire nella casella di testo (segnaposto 3 in figura 1) il testo che si vuole tradurre in codice Java.

Le operazioni da seguire sono le seguenti:

1. Se necessario, de-selezionare il pulsante *“Toggle File”* (indicatore 11 in figura 1).
2. Immagine che contiene schermata, testo, Rettangolo, linea

   Descrizione generata automaticamenteUna volta de-selezionato, la casella di testo in alto a sinistra (segnaposto 3 figura 1) si sbloccherà, consentendo all’utente di incollare il testo oggetto di traduzione

Figura : Pulsante Toggle File

1. Una volta incollato, il testo verrà mostrato sempre nella stessa casella di testo. Allo stesso tempo, gli indicatori dinamici (segnaposti 1 e 2 in figura 1) si aggiorneranno. Il numero della riga contenente il cursore verrà evidenziato di rosso.

Immagine che contiene testo, schermata, schermo, numero

Descrizione generata automaticamente

Figura : Casella di testo HTTP

### Inserimento e visualizzazione di un file contenente la chiamata HTTP da convertire

Questa procedura consente di salvare in memoria il contenuto di un file con il testo che si vuole convertire.

Operazioni da seguire:

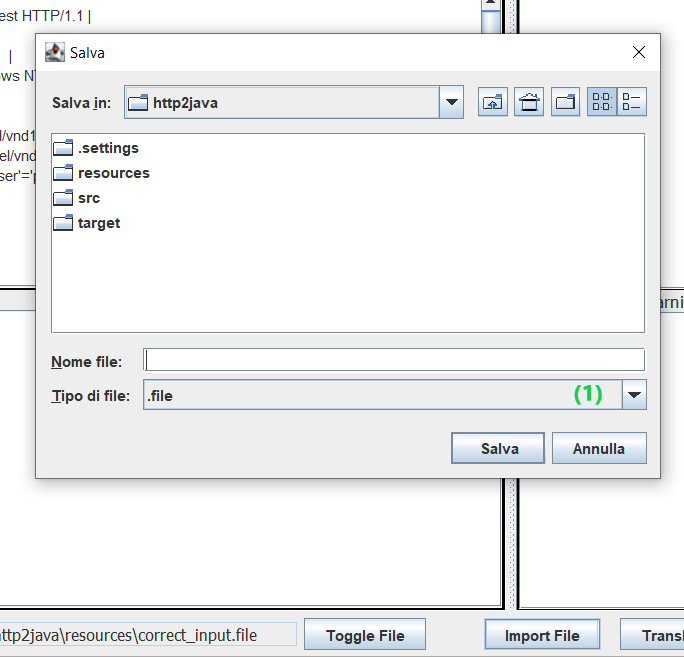
1. Dalla barra in basso, premere sul pulsante *“Import File”* (segnaposto 7 figura 1). La pressione di questo pulsante avvierà una nuova finestra in cui verrà chiesto all’utente di selezionare un file da importare

Figura : Finestra di selezione file

1. Di default la nuova schermata mostrerà soltanto i file con estensione “*file*”. Questo filtro può essere rimosso o modificato premendo sul selettore 1 in figura 4.

Immagine che contiene testo, schermata, Rettangolo, linea

Descrizione generata automaticamente

1. Una volta importato correttamente il file, se necessario, selezionare e portare nello stato “attivo” il pulsante *“Toggle File”* (segnaposto 11 in figura 1). Quando viene selezionato, l’applicazione ignora il testo della casella 3 rendendola inattiva ed estrae il testo oggetto di conversione dal file importato.
2. Immagine che contiene testo, schermata, schermo, Rettangolo

   Descrizione generata automaticamenteInfine, nella casella di testo 3 (figura 1) verrà mostrato il testo estratto dal file importato

### Traduzione da testo HTTP a codice Java

1. Assicurarsi di avere inserito correttamente il testo da tradurre secondo una delle due procedure indicate in questa stessa sezione. La corretta importazione del testo può essere verificata osservando la casella 3). Se questa è popolata, allora Http2Java ha salvato in memoria il testo da convertire.
2. Premere il pulsante “Translate” per avviare la procedura. Se non si verificano errori, il codice Java risultante verrà stampato nella casella di testo 4); al contrario, se si è verificato almeno un errore, la procedura si blocca e nella casella 5) verranno stampati tutti i messaggi degli errori rilevati.

### Visualizzazione del codice Java, di errori e warnings

1. Dopo aver avviato il processo di traduzione secondo le istruzioni di cui appena sopra, possono verificarsi due scenari:
   1. Immagine che contiene testo, schermata, software, numero

      Descrizione generata automaticamenteScenario 1: la traduzione viene completata con successo. In questo caso il codice Java risultante verrà mostrato nell’apposita casella di testo (elemento 4 figura 1).  
      Se il compilatore ha prodotto dei warnings, questi verranno stampati nell’apposita sezione (vedi punto 6 figura 1). A questo punto sarà possibile procedere con la copiatura del testo prodotto o con la traduzione di un nuovo testo

Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamente

* 1. Scenario 2: la traduzione non viene portata a termine. In questo caso il codice Java non viene prodotto, la relativa casella di testo rimane vuota mentre nella casella dedicata agli errori vengono stampati i messaggi di tutti gli errori rilevati. Se il compilatore ha prodotto dei warnings, questi verranno stampati nell’apposita sezione. A questo punto, per poter procedere con la traduzione, sarà necessario modificare il testo in input affinché tutti gli errori vengano risolti.

Per maggiori informazioni sugli errori rilevati dall’applicazione si faccia riferimento alla sezione Errori e Warning.

### Copia negli appunti del codice Java

1. Dopo aver portato a termine con successo l’operazione di traduzione, la casella dedicata alla visualizzazione sarà popolata con il testo risultante. Lo stesso testo può essere copiato in due modi:
   1. Evidenziandolo per intero e premendo la combinazione di tasti *ctrl + C*
   2. Premendo il pulsante “Copy” in basso a destra

# Grammatica

Per quanto riguarda il testo in input con le informazioni della chiamata HTTP, sono state definite delle regole ben precise che definiscono il lessico e la sintassi accettati. Pertanto, per evitare di incappare in errori, è opportuno seguire le regole di seguito elencate.

## Lessico

Con lessico si intendono tutte le regole che definiscono i Token (ovvero “parole”, sequenze di caratteri ammessi dalla grammatica) riconosciuti dal linguaggio. Di seguito viene specificata la lista completa di Token che costituiscono il linguaggio, e che quindi vengono riconosciuti dal compilatore. Per ciascun token è riportata la definizione della regola e il relativo railroad (rappresentazione grafica della regola). Per le *keywords* il railroad (banale) non viene riportato.

Nota: Ciascun diagramma (railroad) è da leggere da sinistra verso destra. Ogni freccia che collega due elementi indica che l’elemento da cui parte la freccia è seguito dall’elemento di arrivo. Eventuali sdoppiamenti delle frecce sono da considerarsi al pari di “OR” logiche, nel senso che rappresentano un’alternativa.

GET : 'GET';  
POST : 'POST' ;  
HOST : 'Host';  
USER\_AGENT : 'User-Agent';  
CONTENT\_TYPE : 'Content-Type';  
ACCEPT : 'Accept';  
COOKIE : 'Cookie';  
AUTHORIZATION : 'Authorization';  
ACCEPT\_LANGUAGE : 'Accept-Language';  
ACCEPT\_ENCODING : 'Accept-Encoding';  
CACHE\_CONTROL : 'Cache-Control';  
MAX\_FORWARDS : 'Max-Forwards';  
Q : 'q';  
EQUALS : '=';  
COMMA : ',' ;  
COLUMN : ':';  
SEMI\_COLUMN : ';';  
TERMINAL : '|';  
STAR : '\*';

ID  
 : ALPHA\_CHAR ALPHA\_CHAR+  
 ;



INT\_NUM  
 : NUM  
 ;



HTTP\_VERSION  
 : 'HTTP/' ('1.1'|'2')  
 ;



PATH  
 : '/' | (('/' PATH\_ELEMENT)+ '/'?)  
 ;

Immagine che contiene schermata, linea, testo, diagramma

Descrizione generata automaticamente

QUERY  
 : '?' QUERY\_ELEMENT ('&' QUERY\_ELEMENT)\*  
 ;



DNS  
 : DNS\_ELEMENT ('.' DNS\_ELEMENT)\* ('.' ALPHA\_CHAR ALPHA\_CHAR+)  
 ;



IPV4  
 : NUM '.' NUM '.' NUM '.' NUM  
 ;



MIME  
 : MIME\_TYPE '/' MIME\_SUBTYPE  
 | '\*/\*'  
 ;



MULTIPART\_MIME  
 : 'multipart/' MIME\_SUBTYPE  
 ;



Q\_VAL  
 : '0' '.' (('0'..'9')   
 |('0'..'9') ('0'..'9')  
 |('0'..'9') ('0'..'9') ('0'..'9'))  
 | '1.0'  
 ;

Immagine che contiene schermata, linea, diagramma, Parallelo

Descrizione generata automaticamente

PRODUCT  
 : ALPHA\_NUM\_CHAR+ '/' VERS\_NUM  
 ;



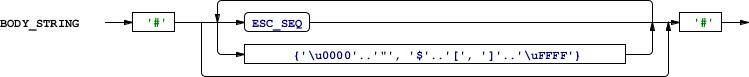
PRODUCT\_INFO  
 : '(' (~('('|')'))\* ')'  
 ;



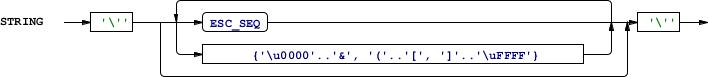
LANGUAGE\_ELEMENT  
 : ALPHA\_CHAR ALPHA\_CHAR ALPHA\_CHAR? ('-' ALPHA\_CHAR+)  
 ;



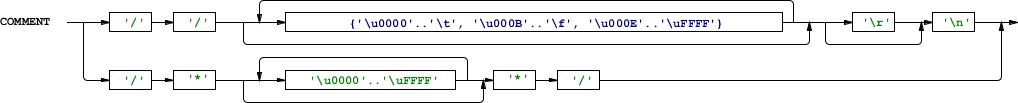
BODY\_STRING  
 : '#' (ESC\_SEQ | ~('\\' | '#'))\* '#'  
 ;



STRING  
 : '\'' (ESC\_SEQ | ~('\\'|'\''))\* '\''  
 ;



COMMENT  
 : '//' ~('\n'|'\r')\* '\r'? '\n' {$channel=HIDDEN;}  
 | '/\*' ( options {greedy=false;} : . )\* '\*/' {$channel=HIDDEN;}  
 ;



WS  
 : (' '  
 | '\t  
 | '\r'  
 | '\n'  
 ) {$channel=HIDDEN;}  
 ;



fragment  
VERS\_NUM

: ALPHA\_NUM\_CHAR+ ('.' ALPHA\_NUM\_CHAR+)\*

;



fragment  
DNS\_ELEMENT  
 : (ALPHA\_NUM\_CHAR | '-') (ALPHA\_NUM\_CHAR | '-')+  
 ;



fragment  
PATH\_ELEMENT  
 : (ALPHA\_NUM\_CHAR  
 | '.' | '~' | '-' | '\_'  
 | '%' HEX\_DIGIT HEX\_DIGIT)+  
 ;

Immagine che contiene schermata, testo, linea, diagramma

Descrizione generata automaticamente

fragment  
QUERY\_ELEMENT  
 : PATH\_ELEMENT '=' PATH\_ELEMENT  
 ;

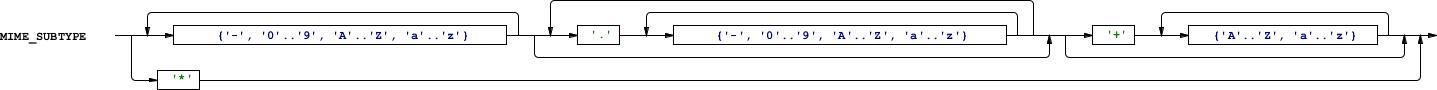


fragment  
MIME\_TYPE  
 : 'application'  
 | 'audio'  
 | 'chemical'  
 | 'font'  
 | 'image'  
 | 'message'  
 | 'model'  
 | 'text'  
 | 'video'  
 ;

Immagine che contiene testo, linea, ricevuta, numero

Descrizione generata automaticamente

fragment  
MIME\_SUBTYPE  
 : (ALPHA\_NUM\_CHAR | '-')+ ('.' (ALPHA\_NUM\_CHAR | '-')+)\* ('+' ALPHA\_CHAR+)?  
 | '\*'  
 ;



fragment  
NUM  
 : '0'  
 | ('1'..'9')('0'..'9')\*  
 ;

Immagine che contiene schermata, linea, diagramma, Rettangolo

Descrizione generata automaticamente

fragment  
ALPHA\_CHAR  
 : ('a'..'z')  
 | ('A'..'Z')  
 ;



fragment  
ALPHA\_NUM\_CHAR  
 : ('0'..'9')  
 | ALPHA\_CHAR  
 ;



fragment  
ESC\_SEQ  
 : '\\' ('b'|'t'|'n'|'f'|'r'|'\"'|'\''|'\\')  
 | UNICODE\_ESC  
 | OCTAL\_ESC  
 ;

Immagine che contiene testo, schermata, linea, Parallelo

Descrizione generata automaticamente

fragment  
OCTAL\_ESC  
 : '\\' ('0'..'3') ('0'..'7') ('0'..'7')  
 | '\\' ('0'..'7') ('0'..'7')  
 | '\\' ('0'..'7')  
 ;

Immagine che contiene schermata, linea, testo, diagramma

Descrizione generata automaticamente

fragment  
UNICODE\_ESC  
 : '\\' 'u' HEX\_DIGIT HEX\_DIGIT HEX\_DIGIT HEX\_DIGIT  
 ;



fragment  
HEX\_DIGIT  
 : ('0'..'9'|'a'..'f'|'A'..'F')  
 ;



ERROR\_TOKEN  
 : .  
 ;



Nota: Le regole precedute da “fragment” non definiscono Token veri e propri, ma “frammenti” di Token utili alla definizione di altri Token così da rendere la grammatica più leggibile. Di conseguenza questi “frammenti” non vengono riconosciuti dal compilatore come Token.

## Sintassi

Oltre al lessico, l’utente deve prestare attenzione anche alle regole che governano le strutture delle frasi del linguaggio (regole sintattiche). Anche in questo caso di seguito sono riportate tali regole seguite dal relativo railroad. In queste regole le parole in maiuscolo sono i Token (definiti nel lessico appena riportato) mentre le parole che iniziano con una lettera minuscola rappresentano le regole sintattiche.

request  
 : requestLine  
 header\*  
 body?  
 EOF  
 ;



requestLine  
 : method  
 pathRule  
 HTTP\_VERSION  
 TERMINAL  
 ;



pathRule  
 : PATH QUERY?  
 | STRING  
 ;

Immagine che contiene testo, schermata, linea, diagramma

Descrizione generata automaticamente

method  
 : GET  
 | POST  
 ;

Immagine che contiene testo, schermata, linea, Carattere

Descrizione generata automaticamente

header  
 : hostRule  
 | userAgentRule  
 | contentTypeRule  
 | acceptRule  
 | cookieRule  
 | authorizationRule  
 | acceptLanguageRule  
 | acceptEncodingRule  
 | chacheControlRule  
 | maxForwardsRule  
 | genericHeaderRule  
 ;

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

hostRule  
 : HOST COLUMN  
 (DNS | IPV4)  
 COLUMN INT\_NUM?  
 TERMINAL  
 ;



userAgentRule  
 : USER\_AGENT COLUMN  
 productRule  
 (productRule  
 (extensionRule)\*)?  
 TERMINAL  
 ;



productRule  
 : PRODUCT  
 PRODUCT\_INFO?  
 ;



extensionRule  
 : PRODUCT  
 ;



acceptRule  
 : ACCEPT COLUMN  
 mimeList  
 TERMINAL  
 ;



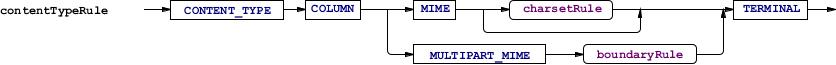
mimeList  
 : mimeElement  
 (COMMA mimeElement)\*  
 ;



mimeElement  
 : MIME  
 qValueRule?   
 ;



contentTypeRule  
 : CONTENT\_TYPE COLUMN  
 (MIME charsetRule?  
 |MULTIPART\_MIME boundaryRule)  
 TERMINAL  
 ;



charsetRule  
 : SEMI\_COLUMN ID EQUALS STRING  
 ;



boundaryRule  
 : SEMI\_COLUMN ID EQUALS STRING  
 ;



cookieRule  
 : COOKIE COLUMN  
 cookieList  
 TERMINAL  
 ;



cookieList  
 : cookieElement  
 (SEMI\_COLUMN cookieElement)\*  
 ;



cookieElement  
 : STRING EQUALS STRING  
 ;



qValueRule  
 : SEMI\_COLUMN Q EQUALS Q\_VAL  
 ;



authorizationRule  
 : AUTHORIZATION COLUMN  
 ID  
 (basicAuthRule  
 | digestAuthRule)  
 TERMINAL  
 ;



basicAuthRule  
 : STRING  
 ;



digestAuthRule  
 : authRule  
 (COMMA authRule)\*  
 ;



authRule  
 : ID  
 EQUALS STRING  
 ;



acceptLanguageRule  
 : ACCEPT\_LANGUAGE COLUMN  
 languageList  
 TERMINAL  
 ;



languageList  
 : languageElement  
 (COMMA languageElement)\*   
 ;



languageElement  
 : (ID  
 |LANGUAGE\_ELEMENT  
 |STAR)  
 qValueRule?  
 ;

Immagine che contiene testo, schermata, linea, Carattere

Descrizione generata automaticamente

acceptEncodingRule  
 : ACCEPT\_ENCODING COLUMN  
 encodingList  
 TERMINAL  
 ;



encodingList  
 : encodingElement  
 (COMMA encodingElement)\*   
 ;



encodingElement  
 : (ID  
 |STAR)   
 qValueRule?  
 ;



chacheControlRule  
 : CACHE\_CONTROL COLUMN  
 STRING  
 (COMMA STRING)\*  
 TERMINAL  
 ;



maxForwardsRule  
 : MAX\_FORWARDS COLUMN  
 INT\_NUM   
 TERMINAL  
 ;



genericHeaderRule  
 : STRING COLUMN STRING  
 TERMINAL  
 ;



body  
 : BODY\_STRING  
 TERMINAL  
 ;



# Errori e Warning

Nell’applicazione è possibile utilizzare tutti gli header possibili (utilizzando una stringa tra singoli apici sia per il nome che per il contenuto dell’header, come si può notare dalla regola sintattica “genericHeaderRule”), ma i controlli di correttezza principali vengono eseguiti solo sui seguenti header:

* Host
* User-Agent
* Content-Type
* Accept
* Cookie
* Authorization
* Accept-Language
* Accept-Encoding
* Cache-Control
* Max-Forwards

Sul body non viene fatto nessun controllo se non che sia una stringa delimitata da due caratteri #. Sulla request line e sugli header definiti tra singoli apici si controlla solamente che non si discostino dalla grammatica sopra riportata.

La stringa mostrata in caso di errore ha la seguente struttura:

* **posizione**, nel formato [riga : colonna], del token che ha generato l’errore. Se l’errore non è generato da un token univocamente identificabile la posizione viene omessa;
* **tipo** dell’errore; sono possibili tre tipi:
  1. duplicate information;
  2. missing information;
  3. wrong information;
* **messaggio** specifico dell’errore.

Di seguito vengono indicate le cause di errore per ognuno dei tre tipi di errore:

1. **Duplicate information**: informazione già specificata; si verifica nei seguenti casi:
   1. inserimento di due o più header con lo stesso nome;
   2. nel caso di header con nome “Authorization” di tipo “Digest”, inserimento di due parametri con lo stesso nome;
2. **Missing information**: mancanza di un’informazione necessaria alla traduzione; si verifica nei seguenti casi:
   1. nessun header con nome “Host” specificato (sia tra singoli apici che non).
3. **Wrong information**: informazione errata che non consente; si verifica nei seguenti casi:
   1. inserimento di sequenze di caratteri non appartenenti al lessico (riportato sopra);
   2. inserimento di sequenze di token che non rispettano la sintassi (riportata sopra);
   3. nel caso di header con nome “Content-Type”, inserimento, a seguito di un token MIME, di un ID diverso da “charset”;
   4. nel caso di header con nome “Content-Type”, inserimento, a seguito di un token MULTIPART\_MIME, di un ID diverso da “boundary”;
   5. nel caso di header con nome “Authorization”, inserimento di un ID diverso da “Basic” e seguito da una stringa;
   6. nel caso di header con nome “Authorization”, inserimento di un ID diverso da “Digest” e seguito da una lista di parametri (del tipo ID = STRING);
   7. nel caso di header con nome “Authorization” seguito dall’ID “Digest”, inserimento di un parametro (del tipo ID = STRING) con ID diverso da "username", "realm", "uri", "algorithm", "nonce", "nc", "cnonce", "qop", "response" o "opaque";
   8. nel caso di header con nome “Accept-Language”, inserimento di un ID con un solo carattere o più di tre caratteri.
   9. nel caso di header con nome “Accept-Encoding”, inserimento di un ID diverso da "gzip", "compress", "deflate", "br" o "identity".

Per quanto riguarda gli warning, la struttura della stringa mostrata è la seguente:

* la stringa “Warning: ”;
* messaggio specifico dello warning.

Gli warning, che non precludono la generazione della traduzione della richiesta http in codice java, sono mostrati nei seguenti casi:

1. definizione del body in una richiesta GET;
2. definizione dell’header “Content-Type” in una richiesta GET;
3. mancata definizione del body in una richiesta POST;
4. mancata definizione dell’header “Content-Type” in una richiesta POST.