Department of Computer Science
Technical University of Cluj-Napoca

Convertor din Markdown in HTML

Git Repository

Laboratory activity

Name: Polman Marian, Rus Maria Alina
Group: 30232
Email: marianpolman2000.mp@gmail.com, hexexec@gmail.com

Teaching Assistant: Timotei Molcut
timoteim98@gmail.com

Contents

Chapter 1

Convertor din Markdown in HTML

1.1 Introducere

Proiectul nostru constă în dezvoltarea unui convertor eficient și flexibil din limbajul Markdown în limbajul HTML. Markdown este un limbaj simplu și ușor de utilizat pentru formatarea textului, în timp ce HTML oferă posibilități mai avansate de structurare și stilizare a conținutului web. Convertorul nostru facilitează procesul de transformare a documentelor scrise în Markdown în pagini HTML bine structurate și vizualmente atractive.

Prin intermediul convertorului nostru, utilizatorii pot crea conținut în Markdown, utilizând sintaxa și convențiile specifice acestui limbaj. Apoi, convertorul preia documentele Markdown și le convertește în cod HTML corespunzător, utilizând etichete și structuri HTML adecvate pentru a reda corect formatarea, stilurile și elementele de structură din documentul original.

Funcționalități cheie ale convertorului nostru includ recunoașterea antetelor de diferite niveluri, inclusiv stilizarea acestora în etichetele corespunzătoare de antet HTML. De asemenea, convertorul detectează textul italic și boldat, îl formatează corespunzător în etichete HTML și menține structura paragrafelor și linii goale pentru o experiență de citire plăcută.

Prin intermediul acestui proiect, ne-am propus să oferim utilizatorilor o modalitate simplă și eficientă de a transforma conținutul scris în Markdown în pagini web HTML, reducând astfel nevoia de a efectua manual conversia și optimizând procesul de publicare a conținutului online.

Sursa limbaj Markdown

1.2 Descriere teoretica

1.2.1 Limbajul Mardown

Markdown este un limbaj simplu și intuitiv, folosit pentru a formata textul în mod structurat și ușor de citit. Este bazat pe utilizarea caracterelor speciale și convenții de formatare pentru a indica elemente precum antete, stiluri de text și structura paragrafelor. Markdown permite utilizatorilor să creeze conținut într-un format ușor de scris și de editat.

1.2.2 Limbajul HTML

HTML (HyperText Markup Language) este limbajul standard utilizat pentru crearea și afișarea paginilor web. HTML oferă etichete semantice și structurate care permit definirea ele-

mentelor și stilurilor pentru o prezentare adecvată a conținutului web. Prin utilizarea etichetelor HTML, este posibilă structurarea și stilizarea conținutului în mod flexibil și personalizat.

1.2.3 Scopul convertorului

Scopul convertorului nostru este de a facilita procesul de transformare a documentelor scrise în Markdown în pagini HTML bine structurate și vizualmente atractive. Convertorul preia conținutul scris în Markdown și îl traduce în cod HTML corespunzător, astfel încât să fie afișat într-un mod adecvat pe paginile web.

1.2.4 Functionalitati cheie

Convertorul nostru recunoaște și convertește următoarele elemente din Markdown în HTML:

Antete: Convertorul detectează antetele de diferite niveluri (de la h1 la h6) și le traduce în etichete HTML corespunzătoare.

Text italic și boldat: Convertorul identifică textul scris între caracterele "*" sau "_" pentru italic și textul scris între caracterele "**" sau "__" pentru boldat și îl formatează în etichete HTML corespunzătoare.

Paragrafe: Convertorul menține structura paragrafelor și linii goale pentru o afișare corectă a conținutului.

Alte elemente: Convertorul poate fi extins pentru a recunoaște și converti alte elemente specifice din Markdown, cum ar fi liste sau link-uri.

1.3 Implementare

Pentru acest proiect ne-am ajutat de parserul C dat ca exemplu in laborator in special pentru partea de transmitere a textului. Am folosit un union in fisierul yacc(p.y) pentru a stoca un element de tip char*.

1.3.1 Metoda de parsare

Am ales să implementăm un parser Markdown bazat pe expresii regulate pentru a recunoaște și a extrage elementele specifice din conținutul Markdown. Am dezvoltat o serie de reguli de analiză pentru a identifica antetele, stilurile de text și paragrafele în conținutul Markdown și pentru a le transforma în structuri de date interne reprezentând AST-ul.

1.3.2 Structura de date pentru stocarea conținutului

Pentru a stoca și a manipula conținutul într-un mod eficient, am implementat o structură de date în formă de arbore pentru reprezentarea AST-ului. Fiecare nod din arbore reprezintă un element specific din Markdown, cum ar fi un antet sau un paragraf, și conține informații relevante, cum ar fi nivelul antetului sau textul paragrafului.

1.3.3 Traducerea elementelor Markdown în HTML

Am implementat o funcție de generare a codului HTML, care traversează AST-ul și transformă fiecare nod în cod HTML corespunzător. De exemplu, pentru un antet de nivel 1, generăm eticheta ‘¡h1¿‘ în HTML, iar pentru stilul de text italic, generăm eticheta ‘¡em¿‘. Am asigurat că sintaxa și convențiile specifice ale limbajului Markdown sunt respectate în rezultatul HTML generat.

1.3.4 Gestionarea stilizării

Pentru gestionarea stilizării textului în HTML, am implementat o funcție care recunoaște caracteristicile specifice ale Markdown pentru italic și boldat și le convertește în etichete HTML adecvate. Astfel, textul scris între asteriscuri sau sublinieri simple (‘*text*‘ sau ‘_text_‘) este încapsulat în eticheta ‘¡em¿‘, iar textul scris între dublu asteriscuri sau duble sublinieri (‘**text**‘ sau ‘__text__‘) este încapsulat în eticheta ‘¡strong¿‘.

1.3.5 Extensibilitate și configurabilitate

Pentru a permite utilizatorilor să personalizeze convertorul, am adăugat opțiuni de configurare suplimentare. Utilizatorii pot specifica adăugarea de stiluri CSS personalizate sau pot activa/dectiva anumite opțiuni de conversie prin intermediul unui fișier de configurare. Am implementat un sistem flexibil de gestionare a configurării, care permite adaptabilitate și extensibilitate.

```
%{
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include "y.tab.h"
%}
    %option noyywrap
    string ([a-zA-Z1-9.?! ])*
    heading1 ("#")
    heading2 ("##")
    heading3 ("###")
    heading4 ("####")
    heading5 ("#####")
    heading6 ("######")
    bold ("_")
    bold2 ("**")
    italic ("")
    italic2 ("*")
    br  (" ")
    NL \n
%%
    {italic}    return ITALIC;
    {italic2}    return ITALIC;
    {NL} return NL;
    {br}+    return BR;
    {bold}|{bold2} return BOLD;
    {string} {yylval.string = yytext; return STRING;};
    {heading1} return HEADING1;
    {heading2} return HEADING2;
    {heading3} return HEADING3;
    {heading4} return HEADING4;
    {heading5} return HEADING5;
    {heading6} return HEADING6;
```

```
%%
```

Listing 1.1: p.l

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
char* removeChar(char* s,char c);
%}
%union{
  char* string;
}
%token <string> STRING
%token HEADING1 HEADING2 HEADING3 HEADING4 HEADING5 HEADING6 BR NL BOLD
    ITALIC
%%
FISIER: INTRO BODYPART OUTRO;
BODYPART: BREAK | HEADING | P | BODYPART BREAK | BODYPART HEADING | BODYPART
     P | ITA| BODYPART ITA |  BLD | BODYPART BLD;
BREAK: BR NL  {printf("   <br>\n");};
P: STRING NL NL {printf("   <p>%s</p> \n",removeChar($1,'\n'));};
ITA: ITALIC STRING ITALIC NL    {char* s=removeChar($2,'\n');
                                 printf("   <em>%s</em>\n",removeChar(s,s[
    strlen(s)-1]));};
BLD: BOLD STRING BOLD NL    {char* s=removeChar($2,'\n');
                printf("   <strong>%s</strong>\n",removeChar(s,s[strlen(s)
    -1]));};
HEADING: HEADING1 STRING NL {printf("   <h1>%s</h1> \n",removeChar($2,'\n'))
    ;}
| HEADING2 STRING NL {printf("   <h2>%s</h2> \n", removeChar($2,'\n'));}
| HEADING3 STRING NL {printf("   <h3>%s</h3> \n", removeChar($2,'\n'));}
| HEADING4 STRING NL {printf("   <h4>%s</h4> \n", removeChar($2,'\n'));}
| HEADING5 STRING NL {printf("   <h5>%s</h5> \n", removeChar($2,'\n'));}
| HEADING6 STRING NL {printf("   <h6>%s</h6> \n", removeChar($2,'\n'));};
INTRO: {printf("<!DOCTYPE html>\n");
        printf("<html> \n");
        printf("<body> \n");
        };
OUTRO: {
    printf("</body> \n");
    printf("</html> \n");
};
%%

char* removeChar(char* s, char c)
{

    int j, n = strlen(s);
    for (int i = j = 0; i < n; i++)
        if (s[i] != c)
            s[j++] = s[i];

    s[j] = '\0';
    return s;
}


#include"lex.yy.c"
#include<ctype.h>
```

```
int main()
{
    yyparse();
}
int yyerror() {;}
```

Listing 1.2: p.y

1.3.6 Arborele de parsare

Utilizarea site-ului ANTLR pentru afișarea arborelui de parsare este o modalitate excelentă de a vizualiza structura internă a conținutului Markdown și modul în care este interpretat de parser. Arborele de parsare oferă o reprezentare grafică a ierarhiei elementelor și relațiilor dintre acestea.

Site-ul ANTLR oferă o interfață ușor de utilizat pentru a încărca fișierul de gramatică și a genera arborele de parsare. După ce arborele este generat, acesta poate fi explorat, extins și restrâns pentru a vizualiza diferitele niveluri ale ierarhiei. Aceasta permite o înțelegere mai clară a modului în care regulile de parsare din gramatică sunt aplicate și cum sunt identificate elementele specifice din conținutul Markdown.

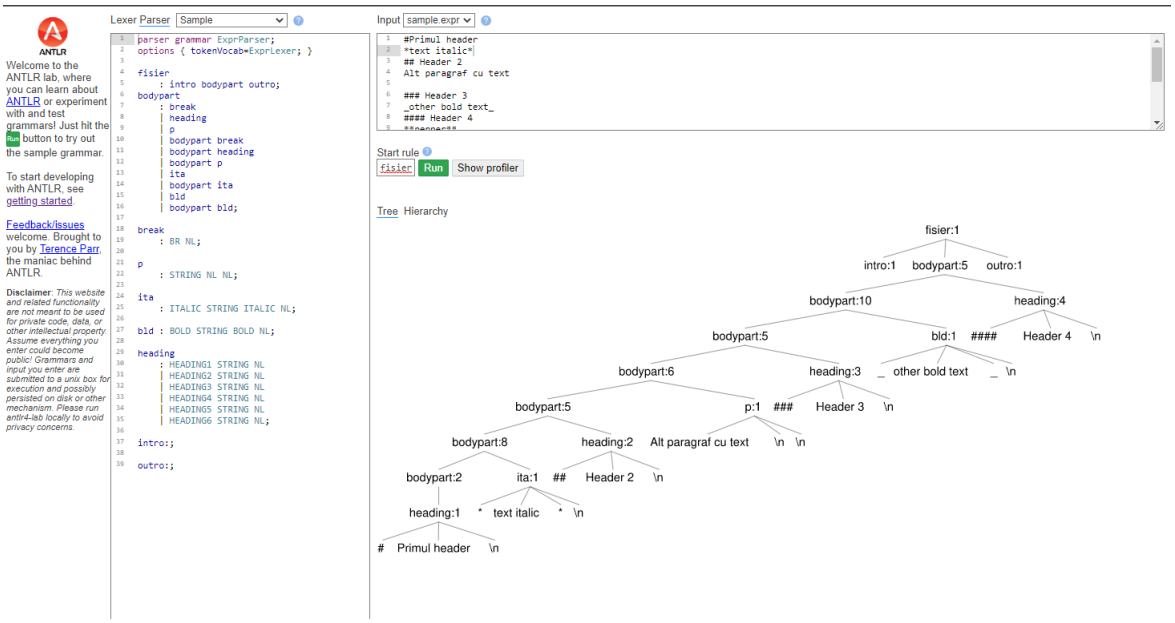
```
lexer grammar ExprLexer;

STRING: [a-zA-Z1-9.?! ]+;
HEADING1: '#';
HEADING2: '##';
HEADING3: '###';
HEADING4: '####';
HEADING5: '#####';
HEADING6: '######';

BOLD: '_';
BOLD2: '**';
ITALIC: '*';
BR: ' ';
NL: '\n';
```

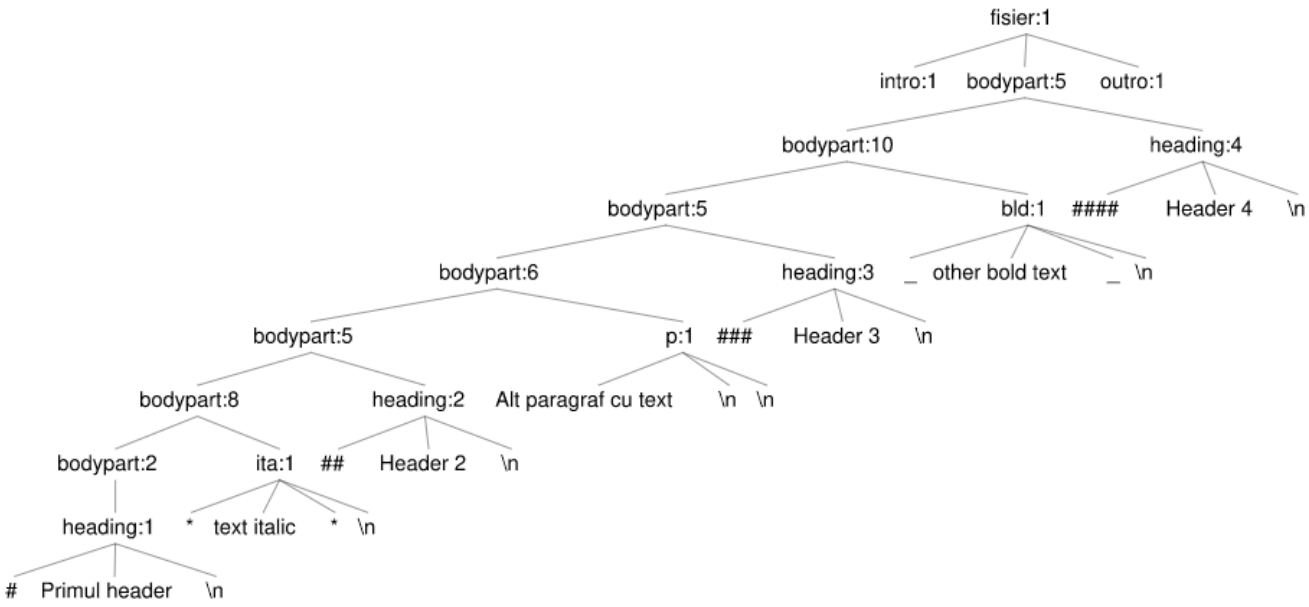
Listing 1.3: antlr.lexer

```
parser grammar ExprParser;
options { tokenVocab=ExprLexer; }

fisier
    : intro bodypart outro;
bodypart
    : break
    | heading
    | p
    | bodypart break
    | bodypart heading
    | bodypart p
    | ita
    | bodypart ita
    | bld
    | bodypart bld;

break
    : BR NL;

```

```
p
     : STRING NL NL;

ita
     : ITALIC STRING ITALIC NL;

bld : BOLD STRING BOLD NL;

heading
     : HEADING1 STRING NL
     | HEADING2 STRING NL
     | HEADING3 STRING NL
     | HEADING4 STRING NL
     | HEADING5 STRING NL
     | HEADING6 STRING NL;

intro:;

outro:;
```

Listing 1.4: antlr.parser





1.4 Rezultate

```
#Primul header
*text italic*
## Header 2
Alt paragraf cu text

### Header 3
_other bold text_
#### Header 4
**pepper**
asta e un paragraf

Un alt paragraf care contine text

##### Header 5
Paragraf care contine descrierea documentului html
```

Listing 1.5: text.y

```
<!DOCTYPE html>
<html>
<body>
   <h1>Primul header</h1>
   <em>text italic</em>
   <h2> Header 2 </h2>
   <p>Alt paragraf cu text</p>
   <h3> Header 3 </h3>
   <strong>other bold text</strong>
   <h4> Header 4 </h4>
   <strong>pepper</strong>
   <p>asta e un paragraf</p>
   <p>Un alt paragraf care contine text</p>
   <h5> Header 5</h5>
   <p>Paragraf care contine descrierea documentului html</p>
</body>
```

```
177 </html>
```

Listing 1.6: text.y