Horia Mut, Lucas Bitter, Steve Visinand

HE-ARC

Générateur de pages HTML

Projet compilateur

# Introduction

Ce projet s’inscrit dans le cadre du cours de compilateur de troisième année de la branche DLM de l’He-arc.

**Langage utilisé** : Python avec PLY

## Etat de l’art

Ce projet s’inscrit dans la ligné des langage de description visant à améliorer la productivité du programmeur et à lui éviter des tâches redondantes.

Dans le même type que ce projet nous pouvant citer les langages de « pre-processing » devenus très utilisés telle que :

* **SASS**  : <http://sass-lang.com>

Syntactically Awesome Style Sheets est un langage qui vise à donner des « super-pouvoirs » aux feuilles de style CSS.

* **LESS** : <http://lesscss.org>

Même utilité que SASS.

* **HAML** : <http://haml.info>

HAML sert à accélérer l’écriture d’HTML.

Fondamentalement notre projet aborde une syntaxe plus courte et produisant de plus grandes portions de code, il a plutôt comme but d’initier un projet web rapidement avec une structure initiale de page auto-générée.

## Suivi de projet

**Wiki** : <https://github.com/stevevisinand/CompilerToHTML/wiki>

**Git** : <https://github.com/stevevisinand/CompilerToHTML>

Ce projet étant réalisé à trois, nous avons pris la peine de créer un Git, un suivi de projet ainsi qu’une spécification dans un wiki commun en plus de ce document.

# But

Le but de notre programme est de générer des pages HTML ainsi que leur style avec un langage simplifié et beaucoup plus court que de l’HTML.

Ainsi un utilisateur sans connaissance de l’HTML pourra facilement générer plusieurs pages avec une structure et des liens entre elles.

# Spécifications du langage

## Eléments

Un **élément** représente une partie de la page HTML une fois générée. L’utilisateur paramètre cet élément sous forme de variable avec des attributs spécifiés pour le personnaliser comme il le souhaite.

**"element"** est un type abstrait et ne peu pas être créé seul, il lui faut un type spécifique.

On peu passer des variables à un "element" entre les parenthèses. On pourra ensuite utiliser ces variables pour les concaténer à des "attributs" de notre element.

Problème : PLY ne permet pas de définir une grammaire contenant plusieurs "literals" à la suite. (){} devient {}

**Déclaration :**

**element** **TYPE\_SPECIFIQUE NOM\_ELEMENT**{

ATTRIBUTES

}

« TYPE\_SPECIFIQUE » est le type réel de l’élément : « header », « footer », « nav », …

« NOM\_ELEMENT » est le nom de la variable contenant l’élément.

« ATTRIBUTES » sont les paramètres possibles de l’élément.

### Header

Entête du site.

|  |  |
| --- | --- |
| **TYPE\_SPECIFIQUE** | header |
| **ATTRIBUTES**  **(possibles)** | title, color, text\_color |

**Exemple :**

**element** **header** **monHead**{

title : "Mon site";

color : "blue";

text\_color : "red";

}

**Résultat (print dans une page) :**



### Nav

Menu du site web.

|  |  |
| --- | --- |
| **TYPE\_SPECIFIQUE** | nav |
| **ATTRIBUTES**  **(possibles)** | color, text\_color, menu |

"menu" décrit l'apparence et le contenu du menu du site internet, le texte des cases éléments du menu et la page cible.

**Exemple :**

**element nav monNav**{

color : "#FFFFF";

text\_color : "#FAFAFA";

menu : [

"home" : index, //index contient la page "index"

"gallerie" : gallery,

];

}

**Résultat (print dans une page) :**



Dans ce cas gallery contient deux sous pages, gal1 et gal2

### footer

Pied du site web.

|  |  |
| --- | --- |
| **TYPE\_SPECIFIQUE** | footer |
| **ATTRIBUTES**  **(possibles)** | title, paragraph (text), copyright (text), color, text\_color |

**Exemple :**

**element footer myFooter**{  
 title: "Title";

paragraph: "paragraphe";

copyright: "blue";  
 color: "red";

text\_color: "white";  
};

**Résultat (print dans une page) :**



## Pages

La page est un type défini par l’utilisateur qui génèrera une page HTML, dans cette page on ajoute nos éléments HTML déjà crées.

Une page est décrite par un nom de variable et des attributs obligatoire.

**Elle doit définir les attributs :**

* Name : Nom de la page
* Adress : Adresse relative de la page

On peu appeler la méthode **"print"** dans une page afin d'imprimer des **"element"** dans la page.

On peu ajouter des sous-pages à une page pour créer une hiérarchie dans la navigation avec l'opérateur "+=".

**exemple :**

page gallery{

//paramètres obligatoires

name : "home";

address : "/index.htm";

print monHead();

print monNav();

print "contenu de la page";

print monFoot(1);

}

gallery += galleryPaysages;

## Boucles

TODO

## Variables

Un utilisateur peu définir une variable contenant des éléments, des pages mais également des variables au contenu plus classique telle que des nombres ou des valeurs texte.

La portée d’une variable est délimitée par « {} » et sont déclarées ainsi :

VAR\_NAME = VALUE;

# Implémentation

## Mots réservés

Voici la liste des mots réservés à l’exécution du programme. L’utilisateur ne peut pas les utiliser hors de leur fonction prévue.

|  |  |
| --- | --- |
| **Mot clé** | **Type** |
| page | "VARIABLE" |
|  |  |
| name | PAGE ATTRIBUT |
| adress | PAGE ATTRIBUT |
| content | PAGE ATTRIBUT |
|  |  |
| element | "ABSTRACT" |
|  |  |
| nav | "VARIABLE" |
| header | "VARIABLE" |
| footer | "VARIABLE" |
|  |  |
| titile | HEADER ATTRIBUT |
| color | ELEMENT ATTRIBUT |
| text | HEADER, FOOTER ATTRIBUT |
| text\_color | HEADER, FOOTER ATTRIBUT |
| paragraph | FOOTER ATTRIBUT |
| copyright | FOOTER ATTRIBUT |

## Opérateurs

Voici la liste des symboles réservés au programme avec leur utilisation.

|  |  |
| --- | --- |
| **Opérateur** | **Fonction** |
| ; | Fin de ligne |
| = | Affectation de variable |
| += | Ajout d'une sous-page à une page |
| [] | Définition d'un tableau associatif pour l'élément menu |
| {} | Block de code, délimitation de la portée de variables |
| () | Paramètres élément, for |
| "" | Délimitation chaine de caractère |
| . | Concaténation de deux chaines de caractères |
| : | Affectation d'attributs "réservés" des éléments |
| / | Opérateur de division sur variable nombre |
| \* | Opérateur de multiplication sur variable nombre |
| + | Opérateur d'addition sur variable nombre |
| - | Opérateur de soustraction sur variable nombre |

## Structure

* **compilerHTML.py**  : Compilateur
  + **generetorHTML.py** : Ensemble de fonctions utiles à la génération de code HTML
  + **AST.py**  : Classe de description d’un noeud
* **syntaxeHTML.py** : Analyse syntaxique
* **lexHTML.py**  : Analyse lexicale

Les fichiers HTML sont générés dans le dossier « generatedSite », attention le fichier « styles.css » doit obligatoirement être présent à l’intérieur.

# Fonctionnalités implémentés

# Exemple d’application

# Remarques et conclusion

Lors de l’implémentation nous avons rapidement rencontrer les limitations de PLY, ce qui nous a compliqué la tâche.

En effet, notre langage particulier oblige l’utilisateur à respecter une syntaxe composé d’une suite de mots réservés dans un ordre précis et pouvant être utilisés que dans certaines situations (dans une page il faut au minimum les deux attributs, etc…).

Avec du recule, la syntaxe de notre langage s’approche trop d’un langage de description que d’un réel langage de programmation, s’il fallait refaire le projet nous choisirions une syntaxe plus proche de l’utilisation de PLY.

Nous avons donc dû adapter notre langage lors du déroulement du projet et faire avec les limitations de cet utilitaire.

Dans notre cas l’utilisation d’un arbre syntaxique était alors moins évidant. Le language étant plutôt descriptif, nous avions directement la possibilité de générer notre propre structure de donnée a partir de l’analyse syntaxique et de la parser plus simplement par la suite.

Nous avons donc commencé par implémenter cette structure mais après discution la génération d’un AST est une contrainte de ce cours. Nous avons en conséquence revu notre code.

Vous pouvez retrouver cette version (non complète) sur notre Git.

Autre difficulté, nous générons directement du code HTML, il a donc été nécessaire de créer une structure HTML et CSS qui se prête à la génération de code, notre langage est donc restreint par ces contraintes et le temps nécessaire à prévoir le code que l’utilisateur pourra générer et donc nos types d’éléments de page sont limités.