



Rapport de stage : Intégration de Microdown dans la chaîne de compilation de Pillar

Delporte Gaylord

Dates du stage :
Du 11 avril 2022 au 11 août 2022

Tuteur professionnel : Ducasse Stephane
Tuteur universitaire : Rouvoy Romain

Entreprise d'accueil : Inria Lille Nord Europe, 40 avenue Halley, 59650 Villeneuve-d'Ascq
Université : Université de Lille Cité Scientifique, 59650 Villeneuve-d'Ascq

Intitulé de la formation : Licence 3 Informatique parcours Miage

Remerciements

Dans un premier temps je souhaite remercier mon tuteur professionnel qui est Ducasse Stephane responsable de l'équipe-projet R-mod pour m'avoir permis d'effectuer ce stage mais aussi pour tout ce qu'il a fait pour moi au cours du stage. Je tiens aussi à remercier mon tuteur universitaire qui est Rouvoy Romain avec qui j'ai communiqué chaque semaine sur les tâches effectuées et son soutien. Je tiens aussi à remercier Kasper Osterbye qui est une personne externe à l'équipe R-Mod travaillant sur Microdown avec qui j'ai conversé sur la mission de mon stage et qui fut une personne agréable et droit dans sa manière de travailler et communiquer. Je tiens aussi à remercier chacun des stagiaires qui ont effectué leurs stages dans la même équipe que moi et avec qui nous nous sommes entraidé à plusieurs reprises. Mais je tenais aussi à remercier l'ensemble de l'équipe R-Mod qui fut une équipe des plus agréables à travailler avec, chacun ayant été très accueillant et n'hésitant pas à parler et aider dans la mesure du possible.

Résumé du stage

Actuellement étudiant en Licence 3 Méthodes informatique appliquées à la gestion des entreprises, j'ai eu la chance de pouvoir effectuer un stage à Inria Lille Nord – Europe, qui est un centre de recherche en informatique situé à Villeneuve-d'Ascq. Pour ce stage, je suis dans l'équipe R-Mod, qui travaille sur l'analyse de code et l'évolution d'application. J'ai comme mission pour ce stage de travailler sur Microdown, un markdown plus simple extensible. Microdown est implémenté sous Pharo, un langage de programmation pure objet orienté.

Pendant mon stage, mon objectif principal était d'intégrer Microdown dans la chaîne de compilation de Pillar. Pillar est un langage de balisage qui permet aussi de générer du texte dans différents formats tel que LaTeX. La première tâche que j'ai réalisée était la conversion de Microdown vers LaTeX. Pour cela, j'ai commencé par la conversion des diaporamas car cela était plus simple que les livres à convertir et seulement après ça que j'ai donc travaillé sur la conversion des livres de Microdown vers LaTeX. Après cette tâche, la tâche suivante que j'ai réalisée fut la conversion de Pillar vers Microdown. Comme la tâche précédente, j'ai commencé par convertir les diaporamas puis après j'ai travaillé sur la conversion des livres. Mais depuis le début de mon stage j'ai aussi travaillé sur les tests faits dans Microdown et plus que ça j'ai aussi testé le rendu de la conversion des documents Microdown vers LaTeX utilisant la chaîne de compilation Pillar laquelle avec mon tuteur nous avons effectué plusieurs changements afin d'être capable d'utiliser des documents Microdown dans Pillar.

Grâce à ce stage, j'ai gagné en expérience professionnelle et j'ai aussi pu consolider mes connaissances acquises lors de ma formation. Cela m'a aussi permis de gagner en expérience dans l'utilisation d'outils tel que GitHub que j'ai utilisé pour Microdown pour travailler avec les autres personnes qui travaillent dessus. Aussi dans l'équipe RMod, à chaque fin de mois on effectue un sprint dans lequel on travaille par équipe de deux sur des problèmes de Pharo et cela m'a permis d'avoir le point de vue de plusieurs personnes avec qui je n'avais jamais travaillé avant et donc cela a contribué à changer mon point de vue sur ma façon de travailler sur un problème.

Internship resume

Currently in Licence 3 Méthodes informatique appliquées à la gestion de entreprises, i got the chance to do an internship in Inria Lille Nord – Europe, which is a research's center in computer sciences located in Villeneuve-D'ascq. For this internship, I am in the Rmod's team which work on the code's analyze and the application's evolution. I have for task for this internship to work on Microdown, a simplified markdown but more extensible. Microdown is implemented in Pharo, a programming language pure oriented object.

During my internship, my main goal was to implement Microdown in the compilation's chain of Pillar. Pillar is a markup syntax which also permit to generate text in different formats like LaTeX. The first task I have done was the conversion from Microdown to LaTeX. For that, I have begin by the conversion of slides because it was more simple than book to convert and only after that I worked on the conversion of the books from Microdown to LaTeX. After that, the task I have done was the conversion from Pillar to Microdown. Like the previous task, I have begin to convert the slide and after that I have work on convert the books. But from the beginning of my internship I also work on the tests done in Microdown and more than that I have work on testing the output of the conversion of documents Microdown to LaTeX using the compilation's chain of Pillar which with my tutor we have done some change for be able to use Microdown's documents in Pillar.

Thanks to this internship, I have gain more professional experience and I was able to consolidate my knowledge that I gain during my formation. It also permit to me to gain experience in the use of tools like GitHub which I use for Microdown to work with the others person who work on Microdown. Also in the team RMod, at the end of each month we do a sprint in which we work by team of two on an issues of Pharo and this give me the point of view of several persons who I had never work with before and this has contribute to change my view of the way I work on a problem.

Développement personnel et professionnel :

Lors de l'option égalité femmes-hommes, pour ma part les choses que j'ai découvertes sont les études faites sur le fait que lorsqu'on demande d'imaginer un profil en entendant un nom de métier que lorsqu'on ne cite pas le nom féminin de certains métiers les profils étaient majoritairement des hommes et que lorsqu'on cite un nom féminin pour un métier alors on obtient plus de profils féminins ce qui me semble normal du fait que selon si on dit par exemple docteur ou doctoresse on va plus être incité à imaginer un homme pour docteur et une femme pour doctoresse. Après j'ai aussi découvert pour le fait que dans certaines formations le fait que les femmes soient très peu nombreuses est un impact mais cela est fortement probable de la même manière que s'il y a très peu d'hommes dans une formation ceci les impactera et je peux donc tout à fait comprendre que cela peut être assez perturbant.

Concernant mon attitude envers mes collègues étudiants et étudiantes je ne pense pas la changer. Personnellement que la personne en face de moi soit une femme ou un homme n'a guère d'importance à mes yeux. Le plus important pour moi chez quelqu'un est certainement pas si c'est un homme ou une femme mais plutôt ce que vaut la personne. Je m'explique, par ce que vaut la personne j'entends si la personne est quelqu'un avec qui je peux m'entendre, quelqu'un avec qui je partage ou non le même point de vue, si on a des passions communes. Dans mon cas je ne faisais déjà pas de différence à ce sujet car pour moi tout ce qui importe sont nos actions et qui on est, ce n'est pas le fait d'être un homme ou une femme, d'être blond ou brun, d'être grand ou petit mais ce que chacun est. Pour moi tout le monde à un grand potentiel en une ou plusieurs choses, le tout est juste de trouver en quoi.

Concernant l'option, je comprends le but et je suis plutôt d'accord avec ce que l'on m'a présenté mais une question reste cependant ancrée dans ma tête. Quand on parle d'égalité femme-homme ou homme-femme, veut on par là dire une égalité au sens que par exemple le milieu médical soit composé de 50% d'hommes et 50% de femmes ? Ou alors une égalité au sens de l'égalité des chances, par là je veux dire que l'on soit une femme ou un homme on ait les mêmes chances d'accéder à une formation, d'obtenir un travail ou même une hausse de salaire sans que le genre rentre en ligne de compte . Car ceci est une question qui me reste en tête de plus c'est un sujet qui m'intéresse et dont j'ai de temps en temps des discussions sur ce sujet avec diverses personnes. Cependant j'ai l'impression que tout le monde n'est pas d'accord sur la signification d'égalité femme-homme ou homme-femme et cela me m'intrigue.

Sommaire

Introduction.....	7
Contexte.....	8
L'entreprise.....	8
Objectif du stage.....	9
Contribution.....	11
Pillar.....	11
Microdown.....	15
Microdown vers LaTeX.....	17
Pillar vers Microdown.....	20
Tests.....	22
Parametrized tests pour windows.....	23
Tests Pillar.....	24
Conclusion.....	25
Travail réalisé et suite du stage.....	25
Enseignements/apports du stage.....	25
Bilan.....	26
Bibliographie.....	27

Introduction

J'ai eu la possibilité d'avoir ce stage par le biais de Ducasse Stephane lors d'un cours de Métaprogrammation. Il avait alors présenté plusieurs sujets de stage qui furent chacun intéressant mais mon choix fut finalement porté sur Microdown. J'ai choisi ce stage car le sujet est intéressant et à suscité mon intérêt mais aussi car la manière de travailler dans une équipe de recherche informatique m'intriguait. En effet, ayant effectué un DUT informatique précédemment, j'ai déjà eu l'occasion d'effectuer un stage et donc j'ai voulu effectuer un stage dans un environnement différent du précédent.

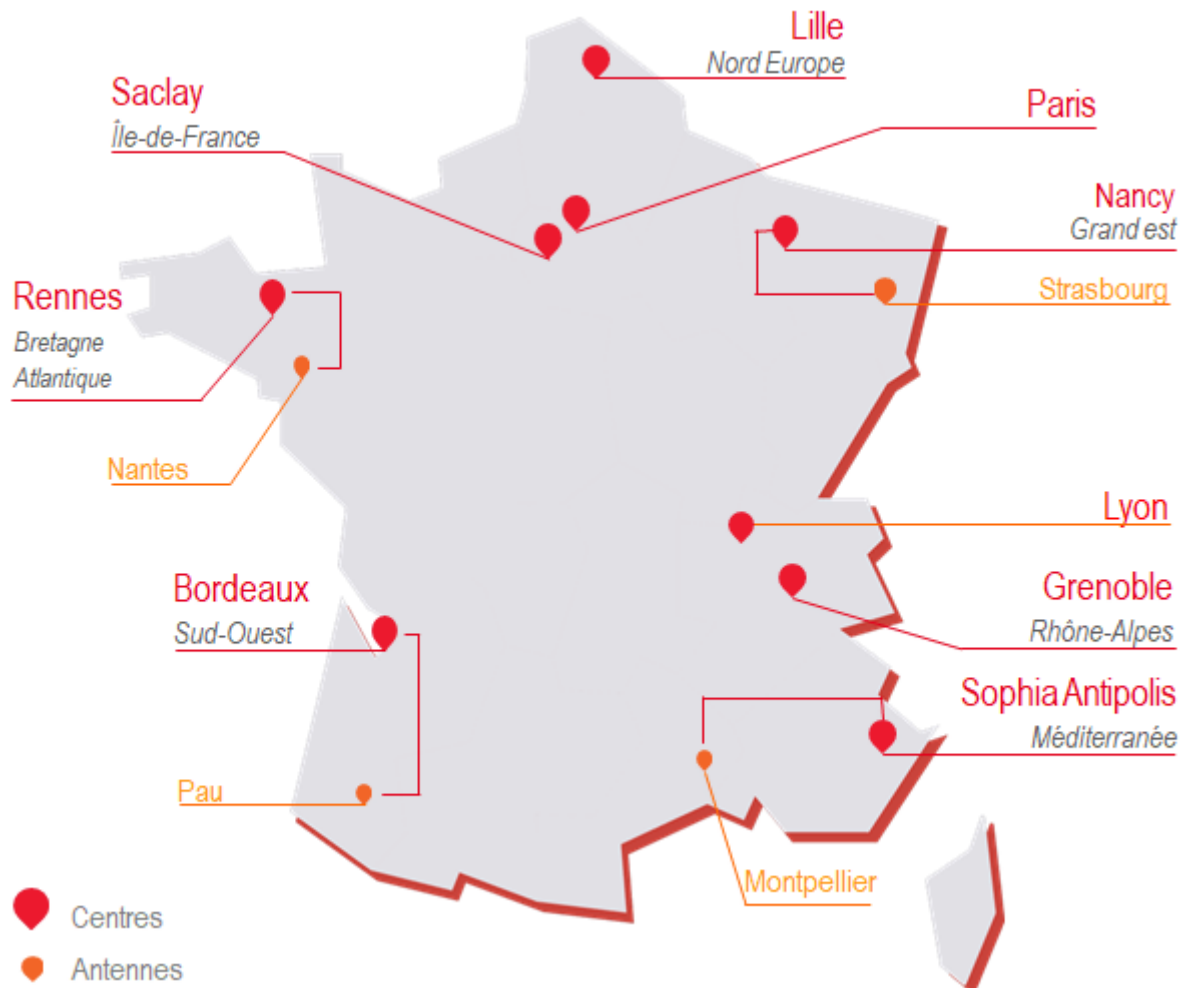
Mais parlons plus de ce qui nous intéresse qui est Microdown, et plus particulièrement de la mission que l'on m'a attribuée. Ma mission est l'intégration de Microdown dans Pillar. Mais plus exactement, l'intégration de Microdown doit être dans la chaîne de compilation de Pillar, de telle sorte à ce qu'on puisse utiliser aussi bien du texte Pillar que du texte Microdown pour générer du LaTeX ou HTML pour finalement passer que par du texte Microdown pour la conversion en d'autres formats.

Mais qui dit objectif à atteindre dit problème à résoudre, et dans notre cas le problème majeur concernait la conversion de documents Pillar à Microdown car nous n'avions pas la possibilité de convertir un document Pillar à Microdown mais aussi convertir les documents Microdown à LaTeX. En effet, si nous voulons pouvoir utiliser Microdown pour rédiger des livres et des diaporamas, il faut pour cela que nous puissions convertir des documents de Microdown à LaTeX.

Et donc pour résoudre ces problèmes et atteindre notre objectif, la marche à suivre fut de commencer par le plus simple, et cela était la conversion de diaporamas Microdown à LaTeX. En faisant cela, j'ai pu d'une pierre deux coups comprendre le code déjà existant de Microdown, et aussi pouvoir commencer à effectuer des ajouts et modifications au code existant. Pour la suite, nous allons passer au contexte de ce stage.

Contexte

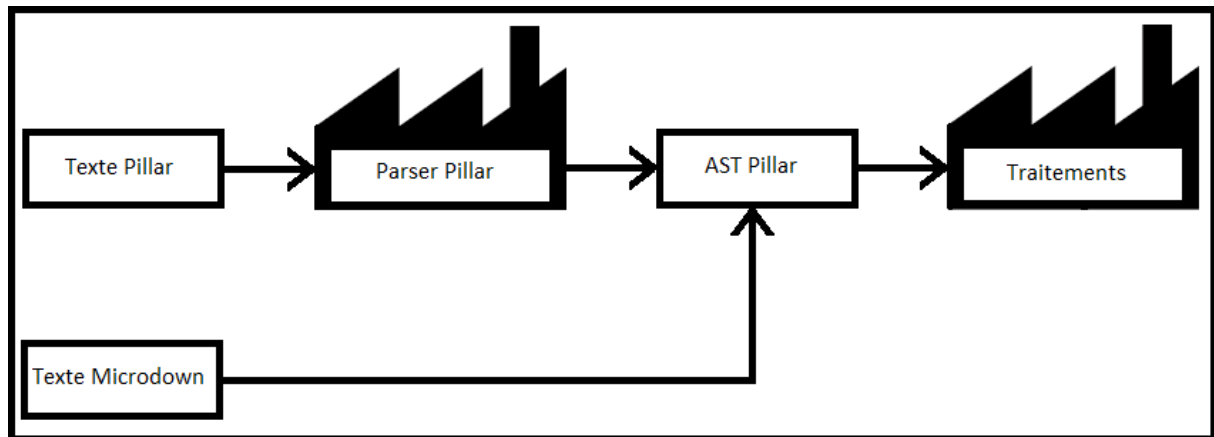
L'entreprise



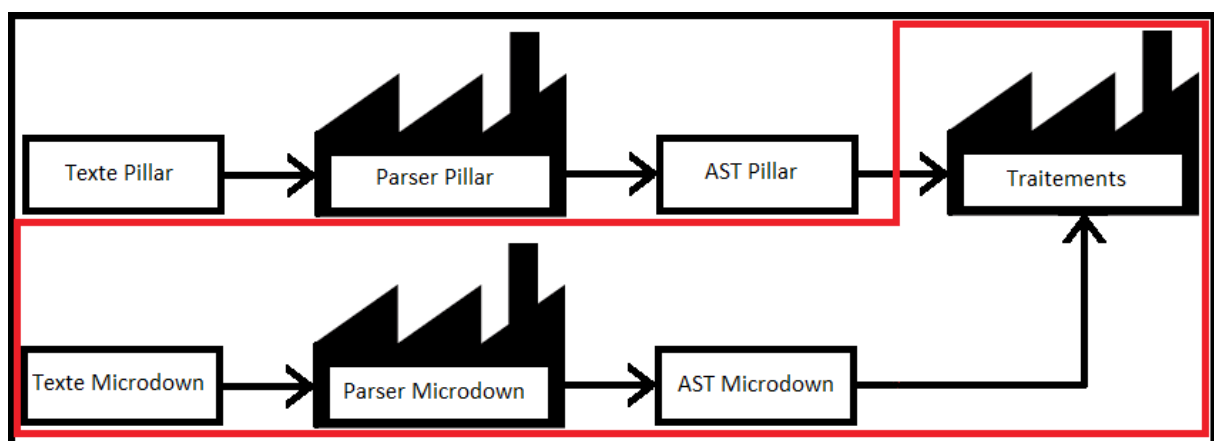
L'entreprise qui m'a accueilli pour ce stage est INRIA qui signifie institut national de recherche en sciences et technologie du numérique et qui est constitué de 10 centres répartis dans toute la France et j'ai effectué mon stage dans celui situé dans le département du Nord à Villeneuve-d'Ascq. INRIA compte plus de 3900 chercheurs et ingénieurs réparti dans plus de 200 équipes. Et pour ce stage j'ai effectué mon stage au sein de l'équipe RMod qui est spécialisée dans l'analyse de code et la remodularisation d'application.

Objectif du stage

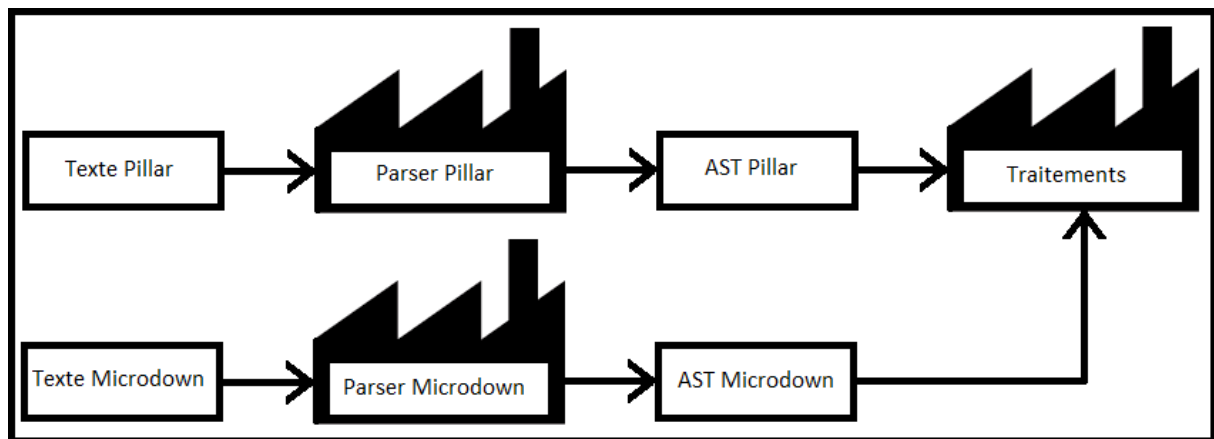
À présent je vais vous présenter l'objectif de mon stage que j'ai effectué. Pour ce stage j'ai eu comme mission l'intégration de Microdown dans Pillar et plus exactement de l'intégration de Microdown dans la chaîne de compilation de Pillar. Car avant mon stage la chaîne de compilation de Pillar ressemblait plutôt à ceci :



Comme vous pouvez le voir, avant pour avoir un rendu de ce que l'on pouvait produire en Microdown on passait du texte Microdown à un arbre de syntaxe abstraite Pillar qui ensuite passé par plusieurs traitements et qui donne en sortie le rendu dans un autre format tel que LaTeX ou HTML. Mais à présent mon objectif est de passer de cela à ce schéma ci-dessous :



Et ainsi au lieu de passé d'un texte Microdown à un arbre de syntaxe abstraite Pillar, on passe du texte Microdown à un arbre de syntaxe abstraite Microdown qui diffère d'un venant de Pillar car chacun est fait selon le langage dont il représente une représentation de leur langage associé. Et donc la partie sur laquelle je dois travailler sur la partie encadrée en rouge. Mais ce n'est pas tout car la finalité est de faire passer le pivot principal de Pillar, qui actuellement est sur le passage du texte Pillar à un arbre de syntaxe abstraite Pillar, sur l'utilisation de Microdown et donc cela donnerait ce schéma :

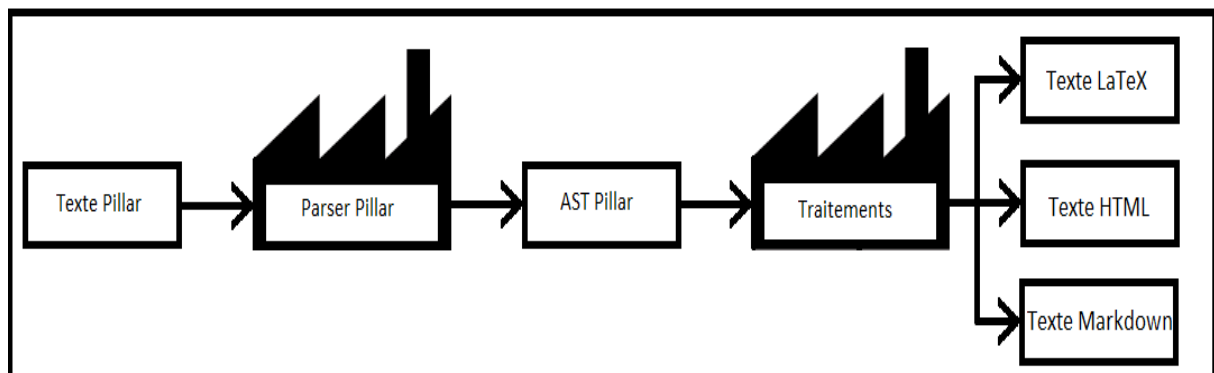


Et donc Microdown deviendrait le pivot principal pour la génération de textes dans la chaîne de compilation de Pillar. À présent que l'on a vu l'objectif de mon stage, nous allons à présent passé sur ma contribution à ce stage qui commencera par l'explication de Pillar et Microdown puis qui sera sur ce que j'ai fait durant mon stage qui sera en trois parties. Une partie sur le passage de Microdown vers LaTeX, une deuxième partie sur le passage de Pillar vers Microdown et pour finir une troisième partie sur les tests effectués.

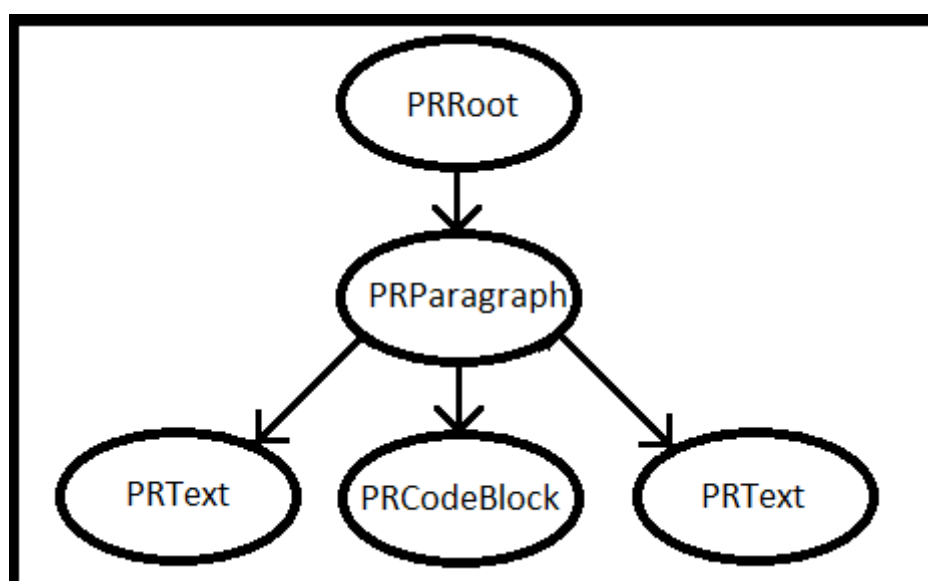
Contribution

Pillar

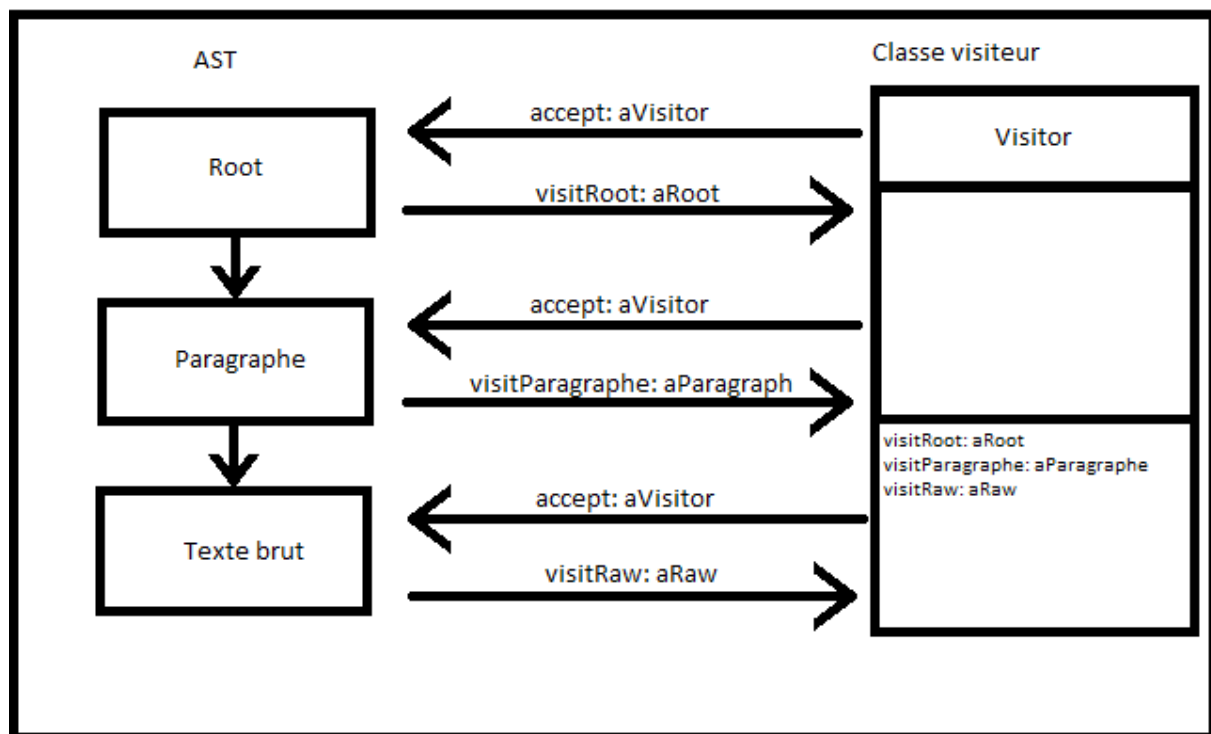
Avant de commencer d'expliquer ce que j'ai accompli durant mon stage, je vais parler des éléments principaux de ce stage. Dans un premier temps je vais vous parler de Pillar qui est un système de balisage mais bien plus encore il est aussi un générateur de texte composé donc d'une chaîne de compilation.



Comme on peut voir sur le schéma ci-dessus, la chaîne de compilation Pillar prend du texte Pillar qu'il convertit en un AST Pillar mais qu'est ce donc un AST ? Un AST signifie arbre de syntaxe abstraite c'est-à-dire un ensemble de nœuds qui contiennent chacun une information et qui sont organisés sous la forme d'un arbre, cela part de la racine (donc le nœud principale) jusqu'aux extrémités en passant par ces fils qui eux-même peuvent avoir des fils. Voici un exemple d'arbre de syntaxe abstraite Pillar simple ci-dessous.



Comme on peut le voir, l'arbre commence par un nœud qui est la racine puis il peut avoir comme enfant un nœud qui lui-même peut avoir de zéro à plusieurs nœuds comme enfants qui eux-même pourraient avoir des enfants et tout cela permet de constituer un arbre de syntaxe abstraite. Et sur ce même arbre des traitements sont faits par des visiteurs. Ces visiteurs utilisent le pattern design du même nom qui permet de visiter l'AST et que selon l'élément on pourra le traiter différemment. Cependant, un visiteur ne modifie en aucun cas l'AST de base, il récupère les informations qu'il traite selon la nature du nœud de l'arbre qu'il visite. Voici un schéma montrant comment procède un visiteur pour visiter un AST :




Comme on peut le voir ci-dessus, le visiteur va demander au nœud de l'arbre de l'accepter, et celui-ci va lui dire de le visiter selon sa nature. Par exemple, si le nœud est du texte brut, il va dire au visiteur de le visiter comme tel. Ceci reprend le principe du double dispatch, une notion que j'ai apprise lors du cours de Métaprogrammation ou pour l'échange de messages, au lieu de demander la classe de l'élément, on va lui dire d'effectuer son action non seulement en fonction du récepteur mais aussi avec des arguments.

Et donc après avoir fait cela, on passe le résultat à l'output qui effectue diverses actions et qui permet ensuite de passer à du texte qui peut être sous différents formats comme le montre le schéma. Cela peut être du LaTeX, de l'HTML ou encore du Markdown. Voici ci-dessous la syntaxe de Pillar :

Headers <ul style="list-style-type: none"> !Header 1 !!Header 2 !!!Header 3 	Tables <table> <tr> <td> Centered Cell</td><td> Centered Title</td></tr> <tr> <td>{ Left-Aligned Cell</td><td>} Right-Aligned Cell</td></tr> </table>	Centered Cell	Centered Title	{ Left-Aligned Cell	} Right-Aligned Cell
Centered Cell	Centered Title				
{ Left-Aligned Cell	} Right-Aligned Cell				
Special Paragraphs <ul style="list-style-type: none"> Annotation @@note this is a note Todo @@todo this is todo 	Links <ul style="list-style-type: none"> Anchor @anchor Internal Link *@anchor* External Link *Google>www.google.fr* 				
Lists <ul style="list-style-type: none"> - Unordered List # Ordered List ; Description Term : Description Definition 	Figures <ul style="list-style-type: none"> +Caption>file://pic.png width=50 label=fig+ 				
Text Formats <ul style="list-style-type: none"> ""bold"" "italic" --strikethrough-- __underline__ ==monospace== @@subscript@@ ^^superscript^^ 	Scripts <ul style="list-style-type: none"> [[[label=hello language=Smalltalk Transcript show: 'Hello World'.]]] 				
Comment <ul style="list-style-type: none"> % each line starting with % is ignored 	Raw <ul style="list-style-type: none"> {{{latex: injects raw \LaTeX in your output file }}} {{{html: injects raw html in your output file }}} 				

Pour vous montrer un exemple de rendu on peut voir ci-dessous un rendu en HTML :



Introduction

- > Scheme in a (super) nutshell
- > A simple parser for Phayche
- > Limited Phayche
- > Function definition and application
- > Adding closures to Phayche
- > Phaycoo

PHYSICHE: A LITTLE SCHEME IN PHARO

Introduction

In this booklet we will build together a little interpreter for a subset of the Scheme language, that we called Physche. The idea is to implement it as simply as possible to illustrate the key aspects and share with you the fun of building language interpreters. Doing so we will explore several concepts:

- limited parsing
- basic interpreter, and
- closure concepts and implementation.

As future readings, I suggest *Structure and Interpretation of Computer Programs* by Abelson, Sussman and Sussman. I simply love it. There is also the excellent book of Jacques Chazarain (which is one of the persons who taught me Lisp) "Programmer avec Scheme" by International Thomson Publishing.

A more personal note. We will implement a subset of Scheme because the language is simple but not trivial, really powerful and also because I love it. And while Pharo is my favorite language, it always has the taste of a Lisp language but with lovely objects. Since I implemented several mini Schemes in Scheme, I got inspired by the (How to Write a (Lisp) Interpreter (in Python)) post of Peter Norvig to write one in Pharo for fun.

Thanks to Clément Béra for feedback on the early version and special thanks for Quentin D. for his great feedback on the understandability and the copy edit suggestions.

Please contact me if you noticed I wrote something wrong or not fully precise.

S. Ducasse (stephane.ducasse@inria.fr) 7 June 2018

Scheme in a (super) nutshell

We will start with a limited version of Physche, our small functional programming language inspired from Scheme. In the first version, we will not support the definition of new functions (also called procedures). In the second version, we will support function definition and closures, in particular. We start by presenting the subset of Scheme that we will implement.

Our objective here is not to write a Scheme following the latest language specification. We will just cover a tiny subset. Purists may not like what I will write but I take this tiny subset as a pretext for a first exploration. I will only present the parts that we will implement. Phayche does not support vectors, dotted pairs, continuations, macros.

For a fast yet more complete description of Scheme I like *Teach yourself Scheme in fixnum days* by Dorai Sitaram <http://ds2fgte.github.io/tyscheme/index.html>.

Here is simple function expressing the length of a list and one example.

Pour voir le contenu complet voici le lien du site :

books.pharo.org/booklet-AMiniSchemeInPharo/html/book.html

Microdown

À présent, je vais vous parler de Microdown, un markdown plus simple mais extensible.

Microdown est un sous-ensemble de Markdown qui est plus simple mais qui comparé à Markdown supporte plus de fonctionnalité telle que ;

- Les environnements avec arguments
- Les ancrages
- Les paragraphes annotés
- Autre éléments intra-bloc (Brut, math et référence)

Microdown a pour objectif d'avoir une syntaxe compatible avec la plupart des implémentations existante de Markdown, d'être utilisable pour l'écriture de livre, d'être extensible et aussi de ne pas être ambigu. Comme indiqué précédemment, Microdown a donc une syntaxe se rapprochant de Markdown mais qui est plus simple afin de pouvoir être plus familier même aux nouveaux utilisateurs et donc d'en faciliter l'utilisation. Voici ci-dessous un exemple de la syntaxe de Microdown ainsi que le rendu de celui-ci :

```
#Header

```language=Pharo&cpation=Beautiful&label=Fig1
 1000 factorial / 999 factorial
```

![Pharologo](https://files.pharo.org/media/logo/logo.png|size=80)

A link: [http://pharo.org](http://pharo.org)

- item 1
  1. sub item 1
  2. sub item 2
- item 2

Bold, italic, monospace

In Pharo hyperlinks to:
- class Point
- method Point class`, `Point>>setX:setY:`, and
- package #'Microdown-Tests' (for packages).

You can edit this file clicking on `ClySyntaxJelpMorph>>#rawMicrodownSyntax`.
```

Header

1000 factorial / 999 factorial



A link: <http://pharo.org>

- item 1
 - a) sub item 1
 - b) sub item 2
- item 2

Bold, *italic*, monospace

In Pharo hyperlinks to:

- class `Point`,
- method `Point class`, `Point>>#setX:setY:`, and
- package ``#'Microdown-Tests'` (for packages).

You can edit this file clicking on `ClySyntaxHelpMorph>>#rawMicrodownSyntax`.

Mais Microdown est aussi un élément important pour Microdown car il prend en charge l'écriture des commentaires des classes et des packages. De plus, un des objectifs de Microdown est de pouvoir écrire en Microdown la documentation et donc de pouvoir l'afficher dans Pharo directement. À présent que j'ai fini d'expliquer Pillar et Microdown nous allons passer sur ce que j'ai réalisé lors de mon stage qui commence par la conversion de Microdown vers LaTeX.

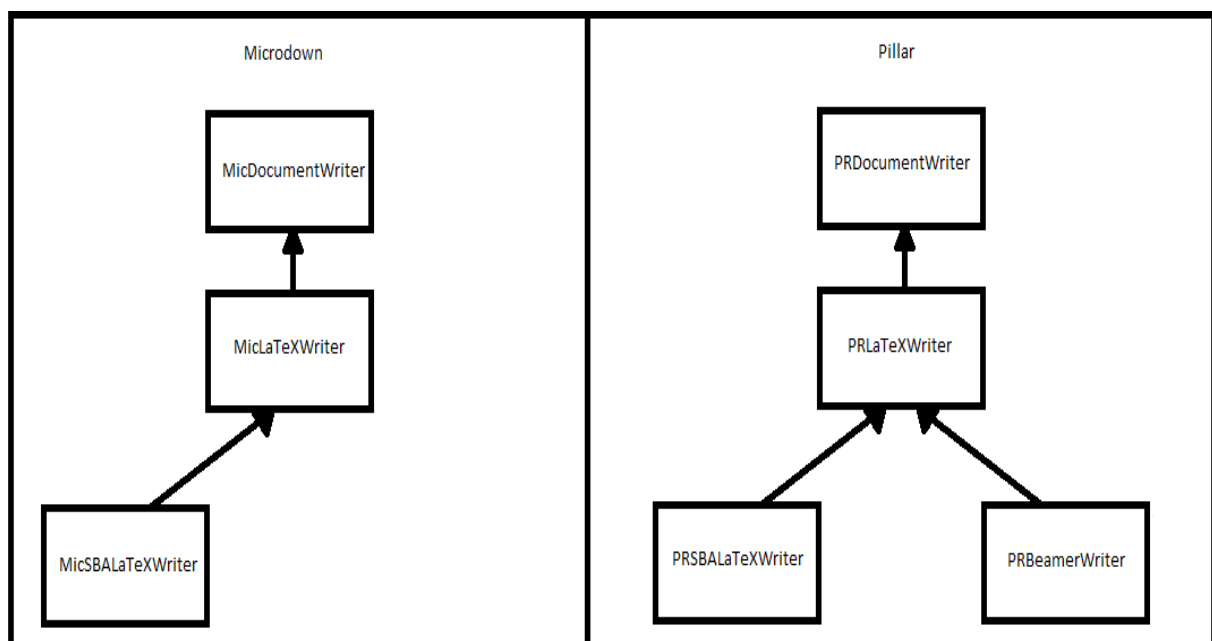
Microdown vers LaTeX

Comme indiqué précédemment, la 1^{re} partie de mon travail se porte sur la conversion de Microdown vers LaTeX. Microdown étant un projet ouvert qui avait déjà commencé avant le début de mon stage, j'ai donc dû démarrer par la compréhension du code déjà existant et pour cela avec mon tuteur nous avons effectué un brainstorming.

Nous en avons conclu que commencé par la conversion de Microdown vers LaTeX permettrait d'avoir un retour plus rapide de ce que nous arriverions à produire mais aussi me permettrait de pouvoir plus facilement comprendre le fonctionnement du code de Microdown.

Mais aussi, nous avons décidé que je commencerais par la conversion des diaporamas car comparé aux livres, elles contiennent moins d'éléments et sont donc plus simple surtout pour moi qui commencé mon stage. Mon tuteur m'a ensuite indiqué plusieurs classes qui étaient importantes pour la bonne compréhension du code de Microdown.

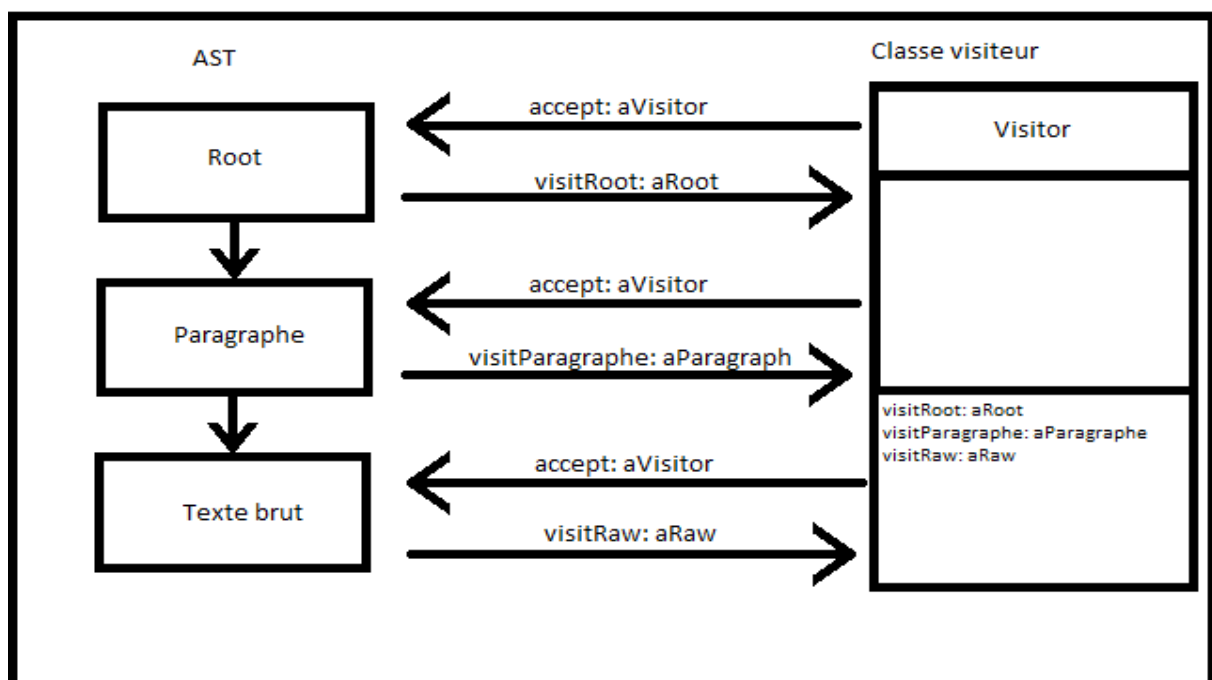
De plus, je me suis aidé de la structure du code de Pillar concernant la conversion de Pillar vers LaTeX car attendant le même résultat, la seule chose à modifier était la manière de récupérer mais aussi la manière de traiter les données car sachant qu'un AST Pillar diffère d'un AST Microdown, les données ne sont donc pas les mêmes et nécessite d'être pris en charge différemment. Ci-dessous un schéma montrant les structure de Pillar et Microdown concernant la conversion vers LaTeX :



Suite à cela, j'ai commencé par regarder les différentes classes que mon tuteur m'avait conseillé de regarder afin de mieux comprendre le code existant. La première classe était `MicMicrodownTextualBuilder`, qui permet de construire du texte Microdown en lui donnant les informations que l'on souhaite insérer tel que si on veut créer un ancrage, il faut donner en argument la donnée à la méthode *anchor* : *aLabel* qui permettra donc de créer un ancrage en Microdown. Cette classe m'a permis de mieux comprendre la syntaxe de Microdown et ainsi de pouvoir savoir d'où je partais pour la conversion de Microdown vers LaTeX.

La deuxième classe était `MicTextualMicrodownExporter` qui permet de visiter un AST Microdown pour écrire du texte Microdown. Cette classe m'a permis de mieux comprendre comment étaient faits les AST Microdown et donc de savoir les équivalents entre un AST Microdown et du texte Microdown.

Et la troisième et dernière classe était `PRBeamerWriter` qui visite un AST Pillar pour ensuite générer le corps des diaporamas en LaTeX, et plus exactement en Beamer qui est une extension de LaTeX utilisé dans la création de diaporamas. Cette classe m'a permis de voir comment les données récupérées d'un AST Pillar étaient traitées puis converties en Beamer et cela m'a aidé à comprendre le point d'arrivée que je devais atteindre pour ma tâche. Pour mieux comprendre le principe du visiteur, voici ci-dessous un schéma pour montrer le principe :



Comme montré sur ce schéma, la classe visitant un AST va demander à chaque nœud dont il est composé sa nature et donc ainsi l'AST indiquera à la classe comment elle doit le visiter si c'est un paragraphe, un bloc de code, du texte en gras ou même du texte brut. Et de cette manière, cela permet d'une part de ne pas modifier la structure et les données de l'AST et d'une autre part de pouvoir récupérer les données et de les utiliser pour créer par exemple un autre AST ou alors du texte. Grâce à toutes ces classes et ce qu'elles m'ont apporté, j'ai donc pu créer une classe nommée `MicBeamerWriter` qui permet de visiter un AST Microdown pour générer le corps des diaporamas en Beamer. Voici le code d'une méthode de `MicBeamerWriter` qui permet de visiter un nœud correspondant à une diapositive :

```
visitSlide: aSlide
  canvas newLine.
  canvas environment
    name: 'frame';
    optParameter: 'fragile';
    with: [
      canvas newLine.
      self createFrameTitle: aSlide title.
      canvas newLine.
      super visitSlide: aSlide ].
  canvas newLine
```

Comme on peut voir, dans cette méthode on utilise un canvas qui est un stream sur lequel on écrit le contenu de l'AST que l'on visite. De plus on a des méthodes telles que `newLine` qui permet de faire un saut de ligne ou `environment` qui permet de créer des blocs LaTeX commençant par `\begin{something}` et qui finit par `\end{something}` tout en incluant à l'intérieur le contenu du diaporama. Mais ce n'est pas tout car après j'ai aussi travaillé sur la conversion des livres Microdown vers Pillar dont une classe avait déjà été créée et qui est nommée `MicSBA LaTeXWriter`. Cette classe était déjà plutôt complète et a donc nécessité peu de changements. Suite à cela, je suis donc passé sur la deuxième partie de ma contribution qui est sur la conversion de Pillar vers Microdown.

Pillar vers Microdown

Dans cette deuxième partie, j'ai donc suite à la conversion de Microdown vers LaTeX passé sur la conversion de Pillar vers Microdown. Pour cela, j'ai donc commencé par la conversion des diaporamas et pour cela j'ai commencé par regarder différentes classes pour m'aider à mieux comprendre comment j'allais procéder et aussi m'aider à réaliser cette tâche. J'ai donc commencé par regarder la classe `PRMicrodownWriter` qui permet de passer d'un AST Pillar à du texte Microdown. J'ai aussi regardé la classe `PRPillarParser` qui permet de passer d'un texte Pillar à un AST Pillar. Avec ces deux classes, j'avais donc à ma disposition tous les éléments nécessaires à la conversion de Pillar vers Microdown car avec `PRPillarParser` je peux à partir d'un texte Pillar générer un AST Pillar qui avec `PRMicrodownWriter` je peux convertir en texte Microdown. Et pour cela j'ai donc créé une classe nommée `PRConverterToMic` qui utilise ces deux classes afin de permettre de passer d'un texte Pillar à un texte Microdown. Voici ci-dessous le code de la méthode `go` : qui permet la conversion d'un diaporama Pillar vers Microdown :

```
go: aPath
| input output fileReference firstParagraph dict metadata |
input := PRSlideTransformer new start: (PRPillarParser parse: aPath readStream contents).
firstParagraph := input children first.
dict := STONJSON fromString: firstParagraph text.
metadata := PRMetadata new metadata: dict.
input children at: 1 put: metadata.

fileReference := FileSystem workingDirectory / 'MyPresentation.md'.
fileReference ensureDelete.
output := PRMicrodownWriter new
    setStream: fileReference writeStream;
    start: input.
```

Comme on peut le voir, les deux classes mentionnées plus tôt sont utilisées mais ce n'est pas tout. Pour la conversion du texte Pillar vers un AST Pillar j'utilise aussi la classe `PRSlideTransformer` qui est nécessaire car pour la conversion en un AST Pillar, le parser passe par un autre AST Pillar que celui que `PRMicrodownWriter` convertit et qui consiste en des annotations. Ceci est dû à la manière dont les AST Pillar étaient construits et convertit auparavant mais cela reste nécessaire pour certaines parties de l'AST. Mais concernant les diaporamas, j'utilise aussi la classe `STONJSON` pour convertir le premier bloc de données en un bloc de métadonnées car celui-ci contient des informations importantes comme par exemple l'auteur du diaporama.

Et donc suite à cela on s'assure que le fichier qui contiendra le texte Microdown est bien initialisé ou réinitialisé. Et on finit par l'utilisation de `PRMicrodownWriter` pour écrire le résultat de la conversion de l'AST Pillar en texte Microdown. Mais bien évidemment, plusieurs problèmes sont apparus comme par exemple les colonnes qui n'étaient pas prises en compte et cela était dû au fait que la méthode permettant de les construire ainsi que de les visiter n'avait pas été faite. Pour la méthode servant à construire les colonnes j'ai donc dû m'intéresser à une autre classe qui était `MicMicrodownTextualBuilder` dans laquelle j'ai donc créé des méthodes permettant la création d'une colonne mais aussi de plusieurs colonnes. Et ainsi, j'ai donc après créé la méthode permettant de visiter les colonnes et ainsi de les construire à l'aide de la méthode précédemment créée dans `MicMicrodownTextualBuilder`. Voici la méthode permettant de créer une colonne ci-dessous :

```
columnWidth: aString withBody: aBodyBlock
  self raw: EnvironmentOpeningBlockMarkup.
  self raw: 'column|width='.
  self raw: aString.
  self newLine.
  self newLine.
  aBodyBlock value.
  self newLine.
  self raw: EnvironmentClosingBlockMarkup.
  self newLine.
  self newLine.
```

Comme on peut le voir ci-dessus, on utilise pour le début et la fin de la colonne deux variables qui proviennent toutes les deux de `MicMicrodownSharedPool` qui contient l'ensemble des délimiteurs de Microdown tel que ceux pour les environnements ou alors les blocs de maths et aussi pour les textes tels que le texte en gras. Pour la suite, nous allons passer à la troisième partie qui traite les tests.

Tests

Pour cette troisième partie, je vais vous présenter les tests réalisés durant mon stage. Mais avant cela je vais vous expliquer l'importance de ces tests et pourquoi ils sont nécessaires. Effectuer des tests permet d'avoir une valeur sûre de ce qu'on écrit et aussi permet de mieux réfléchir au résultat que l'on souhaite atteindre, ainsi il est plus aisé d'arriver à écrire notre code correspondant au test car on sait déjà quel résultat on doit obtenir. Mais aussi les tests permettent plusieurs choses, ils permettent une validation de notre code. En effet, si nos tests passent avec le code que l'on a écrit, en supposant aussi que les tests sont correctement écrits, alors cela indique que notre code est valide et effectue bien ce à quoi il doit répondre. Les tests permettent aussi une non-régression du code car si on viendrait à changer une partie du code, on peut grâce aux tests savoir si cela n'a pas affecté les résultats que l'on obtenait précédemment et donc savoir si on ne perd pas de fonctionnalités. Mais ce n'est pas tout, les tests permettent aussi une bonne intégration et aussi un contrôle de la collaboration car lorsqu'on travaille en équipe sur un projet, si on intègre des changements et que les tests passent alors ça permet de valider l'intégration. Du moins par rapport au résultat attendu défini dans les tests mais aussi cela permet de faire savoir à chaque autre personne de l'équipe que le code que l'on ajoute est correcte et aussi cela permet de vérifier que le travail de l'un n'affecte pas le travail de quelqu'un d'autre.

Et donc pour ce projet qui avait déjà démarré avant que je travaille dessus et donc en plus du code déjà présent il y a aussi son lot de tests pour les différentes raisons que j'ai citées plus tôt. Et donc pour chaque code que j'ai ajouté, chaque classe que j'ai créée et donc ajouté au code de Microdown j'ai donc fait aussi l'ajout de tests afin de savoir quel résultat je devais atteindre. Pour quantifier cela, le projet Microdown avant mon stage il y avait un peu moins de 1000 tests et actuellement il y a exactement 1609 tests. Les tests ajoutés représentent tous les tests que j'ai ajoutés pour les classes que j'ai créées mais aussi une tâche que j'ai faite qui consistait à passer des `testCase` à des `ParametrizedTest` qui permettent d'écrire des cas de tests en permettant d'avoir plusieurs valeurs pour une même variable donnée et donc vérifié si pour chaque valeur le test passe ou non.

Parametrized tests pour windows

Pour ma part j'ai utilisé les parametrizedTests pour l'utilisation des sauts de ligne car on faisait les sauts de lignes manuellement pour les tests mais cela posait un problème car si on changeait le saut de ligne de la configuration de Microdown, on pouvait alors avoir tous les tests incluant un saut de ligne ne plus passer dû à cela. Et donc c'est là que les ParametrizedTest m'ont permis de résoudre cela en ajoutant une méthode de classe nommée testParameters qui permet de configurer pour chaque variable d'instances une ou plusieurs valeurs qui permet donc d'effectuer chaque tests avec chaque combinaison possible. Voici ci-dessous une capture d'écran de la méthode testParameters que j'ai ajoutée dans la classe MicLaTeXWriter :

testParameters

```

^ ParametrizedTestMatrix new
  forSelector: #writer addOptions: { MicLaTeXWriter };
  forSelector: #factory addOptions: { MicMicrodownSnippetFactory };
  forSelector: #parser addOptions: { Microdown };
  forSelector: #newLine addOptions: { String cr . String lf . String crlf};
  yourself

```

Et donc de cette manière cela permet de vérifier que peu importe le saut de ligne utilisé, on aura un test confirmant que le code donne le résultat que l'on souhaite avoir. Mais aussi les tests permettent comme je l'ai indiqué précédemment de vérifier le code ajouté pour sa bonne intégration. Et en effet cela fut aussi un point important concernant la collaboration avec les autres personnes travaillant sur Microdown car sur le dépôt GitHub du projet il y a deux systèmes qui vont lancer l'ensemble des tests et donc vérifier qu'ils passent tous comme on peut le voir ci-dessous :

| | | | |
|--|--|----------------------|-----|
| Fixes #437 - isMicrodownOrFolder: is dead code
currentDev #694: Pull request #494 opened by kasperosterbye | kasperosterbye:437-MicGitHu... | yesterday
1m 14s | ... |
| Merge pull request #491 from DelGaylord/dev
currentDev #693: Commit bb12092 pushed by kasperosterbye | dev | 2 days ago
1m 15s | ... |
| Creation of MicInputFileResolver + test
currentDev #692: Pull request #491 synchronize by DelGaylord | DelGaylord:dev | 6 days ago
1m 22s | ... |
| Fixing parserClass to return the class and not microdown.
currentDev #691: Commit 51ae649 pushed by Ducasse | dev | 7 days ago
1m 51s | ... |
| Remove Microdown-Converter since it was empty and m...
currentDev #690: Commit 817590c pushed by Ducasse | dev | 7 days ago
1m 29s | ... |
| Creation of MicInputFileResolver + test
currentDev #689: Pull request #491 opened by DelGaylord | DelGaylord:dev | 7 days ago
1m 25s | ... |

Tests Pillar

Mais ce n'est pas tout pour les tests que j'ai fait car étant donné que Microdown doit être intégré dans la chaîne de compilation de Pillar, cela inclut des tests que l'on doit faire en vérifiant le rendu qu'on obtient quand on convertit de Pillar vers Microdown puis de Microdown vers Latex et donc pour cela on a utilisé Pillar et plus exactement le build de Pillar qui permet d'utiliser Pillar pour convertir des documents Pillar en d'autres formats.

Pour ces tests-là on a procédé de manière croissant en commençant par les diaporamas puis des livres simples jusqu'à des livres plus complexes. Effectuer cela nous a permis de voir des problèmes que l'on n'aurait pas pu détecter par des tests classiques par exemple le fait que les chemins pour les images n'étaient pas corrects ce qui a nécessité de corriger cela pour avoir le chemin correct et donc l'affichage de l'image.

Pour pouvoir effectuer ces tests, nous avons donc dû faire plusieurs changements. Pour pouvoir effectuer ces tests, nous avons donc dû faire plusieurs changements. D'une part nous avons dû permettre de convertir des documents dans un autre format que Pillar. Pour cela, nous avons donc regardé puis modifié la méthode utilisée pour définir les extensions acceptées et donc ajouté les extensions correspondant à Microdown. Suite à cela nous avons modifié la recherche des writers car celle-ci n'était faite que dans Pillar et donc les writers dans Microdown n'étaient pas pris en compte.

Pour cela nous avons donc basé la recherche sur les classes possédant une méthode de classe qui est `writerName` qui indique donc le nom pour le fichier de configuration de Pillar quel writer utilisé. Mais ce n'est pas tout, nous avons aussi modifié le type de document ainsi créé par la conversion qui ne créait auparavant que des documents Pillar et donc on a ajouté aussi les documents Microdown lorsqu'on utilise un writer issu de Microdown. Et donc grâce à tout cela nous étions enfin capables de tester la conversion des documents Microdown vers LaTeX afin de voir si cela s'était correctement fait ou non.

Conclusion

Travail réalisé et suite du stage

Pour conclure le travail réalisé permet désormais la conversion de diaporamas et de livres simple dans Pillar à partir de documents Microdown et nous avons aussi la possibilité de convertir des diaporamas et livres simples de Pillar vers Microdown. Le nombre de tests pour Microdown a aussi augmenté suite aux ajouts de nouvelles classes et aussi le changement de certaines classes de tests de `testCase` à `parametrizedTest`.

Cependant il reste encore du travail concernant la conversion des livres complet Microdown vers LaTeX et de Pillar vers Microdown mais aussi de la conversion de Microdown vers d'autres formats tel que HTML qui est sur la liste des tâches que j'ai pour la suite de mon stage. Il y a aussi l'inclusion de fichier qui est une tâche essentielle pour la conversion des livres entiers. Mais comme dit précédemment, il reste encore des tâches à réaliser durant le temps restant de mon stage.

Enseignements/apports du stage

Ce stage m'aura permis d'acquérir des compétences principalement dans la reprise d'un projet existant car Microdown est un projet qui était déjà démarré avant mon stage et j'ai donc dû apprendre à gérer cela par la compréhension du code existant afin de pouvoir ajouter du code pour réaliser mes tâches. J'ai aussi beaucoup appris et pus mettre en pratique ce que j'ai appris durant ma formation concernant les tests avec par exemple le TDD.

Mais aussi cela m'a permis de gagner en expérience professionnelle et permis de travailler avec des personnes qui ont des années d'expérience et qui m'ont beaucoup apporté en partageant leur point de vue sur leurs façons de programmer par exemple lors des sprints de fin de mois où on travaille par paires sur la résolution de problèmes divers et variés.

Bilan

Ce stage m'a permis de consolider mes connaissances et d'acquérir plus d'expérience ce qui me sera bénéfique autant personnellement que professionnellement. En ce qui concerne ma formation et la suite de celle-ci, ce stage s'inscrit parfaitement dans ce cadre car effectuant une licence 3 MIAGE et comptant aller en master MIAGE l'année prochaine, le fait d'avoir travaillé sur un projet qui avait déjà commencé me sera très utile pour la suite mais aussi la manière de travailler en équipe et la communication sont des éléments importants et même clé pour la gestion d'une équipe. Et aussi bien personnellement que professionnellement, ce stage m'aura beaucoup apporté aussi sur ma vision de gérer un projet et de mener à bien celui-ci.

Bibliographie

Lien du dépôt de Microdown :

<https://github.com/pillar-markup/Microdown>

Lien du dépôt de Pillar :

<https://github.com/pillar-markup/pillar>