Pillar

One Format to Rule Them All!

Cyril Ferlicot-Delbecque, Damien Cassou

http://www.smalltalkhub.com/#!/~Pier/Pillar

What is Pillar ?

Problem

- We want more **documentation**
- We want many formats
  - *pdf*
  - *html*
  - *epub*
- We do not want to rewrite everything
- We want it simple

If you want them to RTFM,
make a better FM.

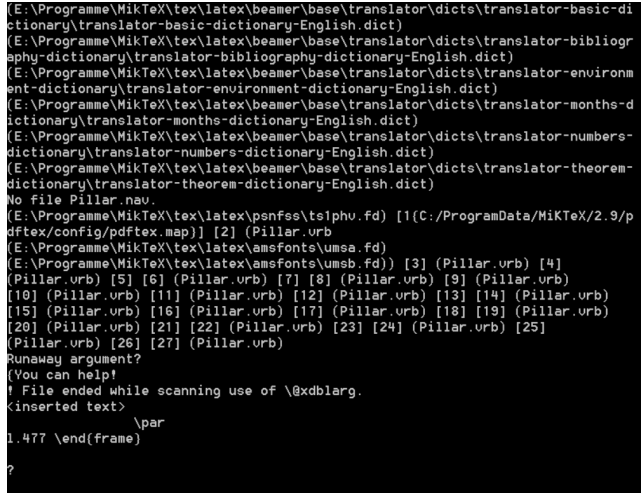
Currently

- LaTeX
  - Complicated
  - Hard to find an error
  - Hard to convert in HTML
- Markdown
  - Incomplete
  - Incompatible implementation

```
\begin{itemize}
\item  Replace \ct{Object} with \ct{SimpleSwitchMorph}.
\item  Replace \ct{NameOfSubClass} with \ct{LOCell}.
\item  Add \ct{mouseAction} to the list of instance variables.
\end{itemize}

The result should look like this:

% syntax highlighting = smalltalk

\begin{script}[scr:ClassLOCell]{Defining the classLOCell}
SimpleSwitchMorph subclass: #LOCell
    instanceVariableNames: 'mouseAction'
    classVariableNames: ''
    category: 'PBE-LightsOut'
\end{script}
```

Currently

- LaTeX
  - Complicated
  - Hard to find an error
  - Hard to convert in HTML
- Markdown
  - Incomplete
  - Incompatible implementations



Our Solution: Pillar

- One input, many outputs
  - *HTML*
  - *LaTeX*
  - *Text*
  - *Markdown*
- Textual syntax
  - No need for a specific editor

```
!Example
This is an example of a Pillar file.
*Link>http://www.smalltalkhub.com/*
- Unordered Item
- Unordered Item
# Ordered Item
# Ordered Item
|!Language |!Coolness
|Smalltalk | Hypra cool
|Java   | baaad
+ Figure !>file://path.png|width=80+
```

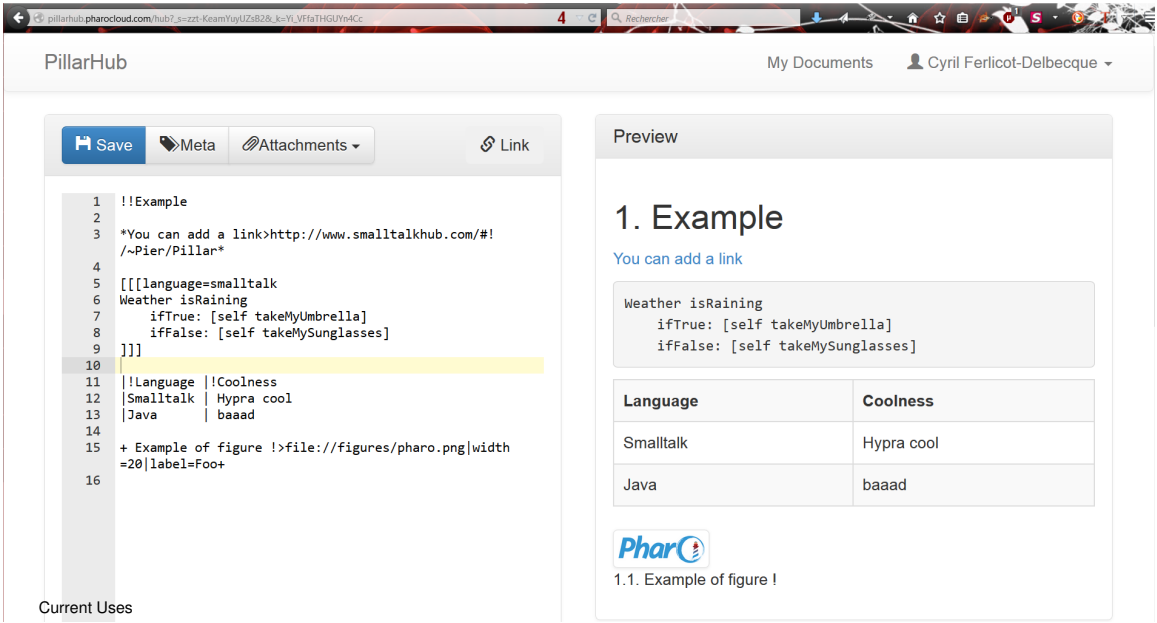
Our Solution: Pillar

- Extracted from Pier
- Now Pier needs Pillar


```
!Example
This is an example of a Pillar file.
*Link>http://www.smalltalkhub.com/*
- Unordered Item
- Unordered Item
# Ordered Item
# Ordered Item
|!Language |!Coolness
|Smalltalk | Hypra cool
|Java   | baaad
+ Figure !>file://path.png|width=80+
```

Current Uses

PillarHub by **Mike Filonov**.



http://pillarhub.pharocloud.com/

Books

1.3. Create a repository

In Voyage, all persistent objects are stored in a repository. The kind of repository that is used determines the storage backend for the objects.

To use the in-memory layer for Voyage, an instance of `VOMemoryRepository` needs to be created, as follows:

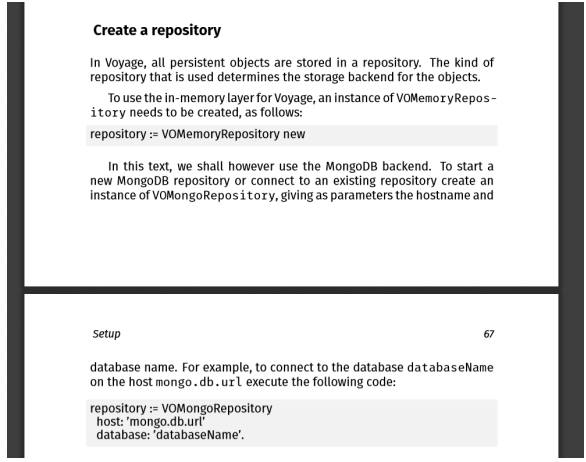
```
repository := VOMemoryRepository new
```

In this text, we shall however use the MongoDB backend. To start a new MongoDB repository or connect to an existing repository create an instance of `VOMongoRepository`, giving as parameters the hostname and database name. For example, to connect to the database `databaseName` on the host `mongo.db.url` execute the following code:

```
repository := VOMongoRepository
  host: 'mongo.db.url'
  database: 'databaseName'.
```

Alternatively, the port to connect to is specified by using the message `host:port:database:`. Lastly, if authentication is required, the message `host:database:username:password:` or `host:port:database:username:password:` needs to be used.

1.4. Singleton Mode and Instance Mode



Create a repository

In Voyage, all persistent objects are stored in a repository. The kind of repository that is used determines the storage backend for the objects.

To use the in-memory layer for Voyage, an instance of `VOMemoryRepository` needs to be created, as follows:

```
repository := VOMemoryRepository new
```

In this text, we shall however use the MongoDB backend. To start a new MongoDB repository or connect to an existing repository create an instance of `VOMongoRepository`, giving as parameters the hostname and

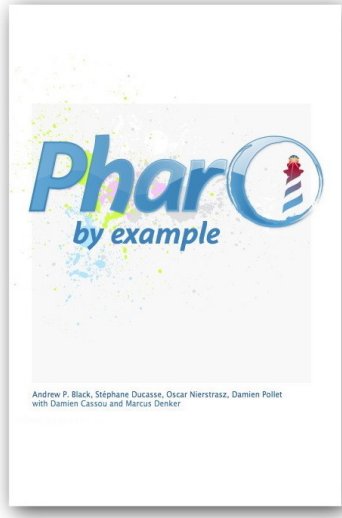
Setup 67

database name. For example, to connect to the database `databaseName` on the host `mongo.db.url` execute the following code:

```
repository := VOMongoRepository
  host: 'mongo.db.url'
  database: 'databaseName'.
```

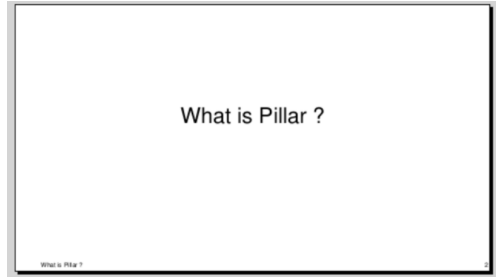
Current Books

- Updated Pharo by Example
- Pharo for the Enterprise



Presentations

```
! What is Pillar ?

${slide:Problem}$

${columns}$

${column:50}$

- We want more ""documentation""
- We want many formats
-- ''pdf''
-- ''html''
-- ''epub''
- We do not want to rewrite everything
- We want it simple

${column:50}$

''If you want them to RTFM,''

''make a better FM.''

${endColumns}$
```



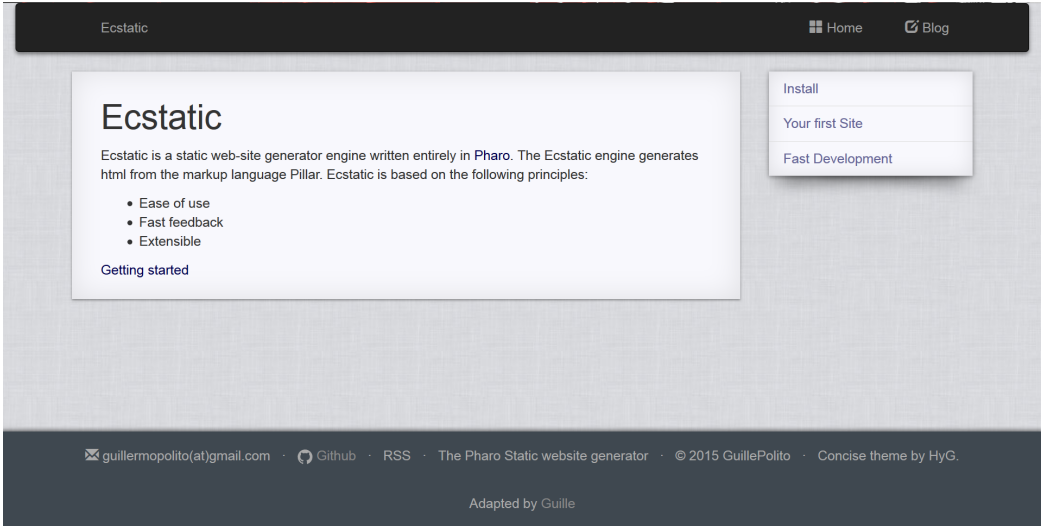
Presentations

This presentation itself is written in Pillar.

Ecstatic: Static Websites

```
!Ecstatic

Ecstatic is a static web-site generator engine written entirely in
+Pharo>http://www.pharo.org+.
The Ecstatic engine generates html from the markup language Pillar.
Ecstatic is based on the following principles:

- Ease of use
- Fast feedback
- Extensible

+Getting started>getting_started.pillar+
```

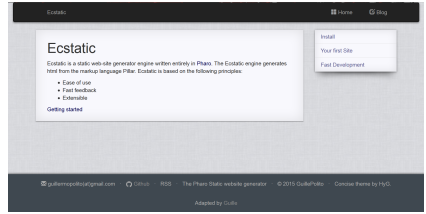
Ecstatic: Static Websites



Current Uses http://pillarhub.pharocloud.com/

Ecstatic: Static Websites

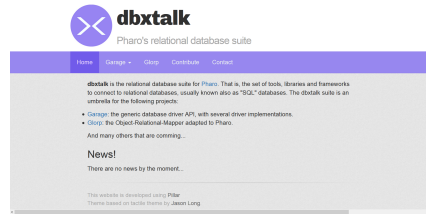
- Created by **Guillermo Polito** and **Stephane Ducasse**
- Generate static website with Pillar
- Many templates
- Examples
  - Ecstatic (guillep.github.io/ecstatic)
  -
  -



Current Uses http://pillarhub.pharocloud.com/

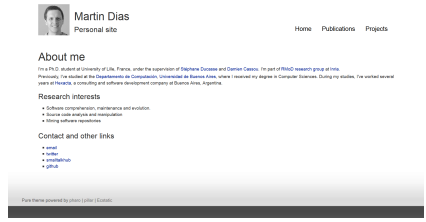
Ecstatic: Static Websites

- Created by **Guillermo Polito** and **Stephane Ducasse**
- Generate static website with Pillar
- Many templates
- Examples
  - Ecstatic (guillep.github.io/ecstatic)
  - DBXTalk (guillep.github.io/DBXTalk)
  -



Ecstatic: Static Websites

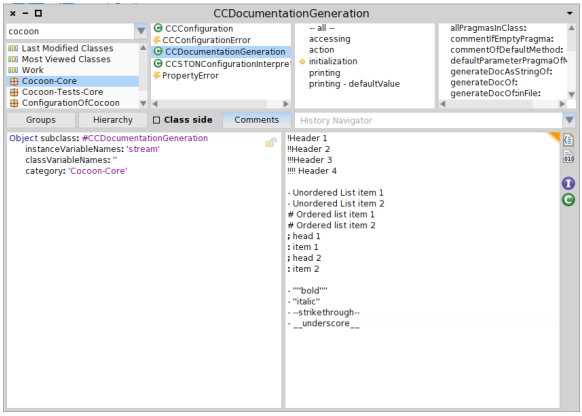
- Created by **Guillermo Polito** and **Stephane Ducasse**
- Generate static website with Pillar
- Many templates
- Examples
  - Ecstatic (guillep.github.io/ecstatic)
  - DBXTalk (guillep.github.io/DBXTalk)
  - Personal pages as tinchodias.github.io

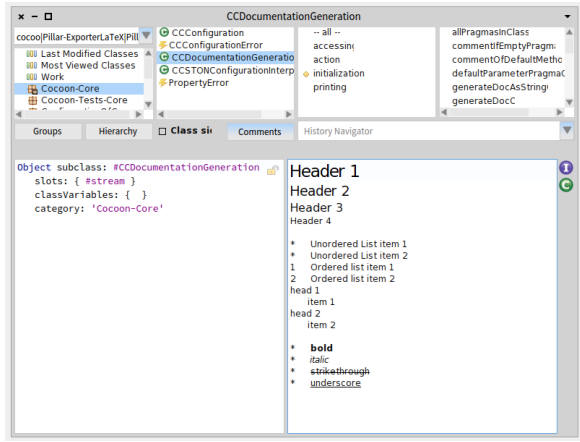


http://guillep.github.io/ecstatic

Documentation renderer

Prototype by **Kasper Osterbye**





Begin a Documentation with Pillar

Simple Example

1. `https://raw.githubusercontent.com/pillar-markup/book-skeleton/master/download.sh`
2. echo "!Hello World" > Example.pillar
3. ./pillar export - -to=html - -outputFile=Example.html Example.pillar

Use Your Own Templates

- One default template by exporter
- Possibility to create your own template
- Possibility to use *meta-data* in the template

```
'<!DOCTYPE html>
<html lang="en">
  <head>
    <title>{{{title}}}</title>
    <link rel="stylesheet" href="http://yandex.st/
        highlightjs/8.0/styles/default.min.css">
  </head>
  <body>
   <h1>{{{subtitle}}}</h1>
    <div class="container">
      {{{content}}}
    </div>
  </body>
</html>'
```

Configuration: pillar.conf

- Configure your exports with a simple file
- STON syntax
- Can contain
  - Export configuration
  - Sub configurations
  - Meta-data

```
'{
  "inputFiles": [ "Pillar.pillar" ],
  "outputFile": "Pillar.tex",
  "author":"Damien Cassou, Cyril Ferlicot",
  "title": "Pillar",
  "configurations": {
    "beamer":{
      "template" : "slides.beamer.template",
    }
    "beamer2":{
      "outputType": #beamer,
      "template":"slides.beamer2.template"
    }
  }
}'
```

Text Editor Plugins

- You can find Pillar plugin for:
  - Emacs
  - Vim
  - TextMate
  - Atom

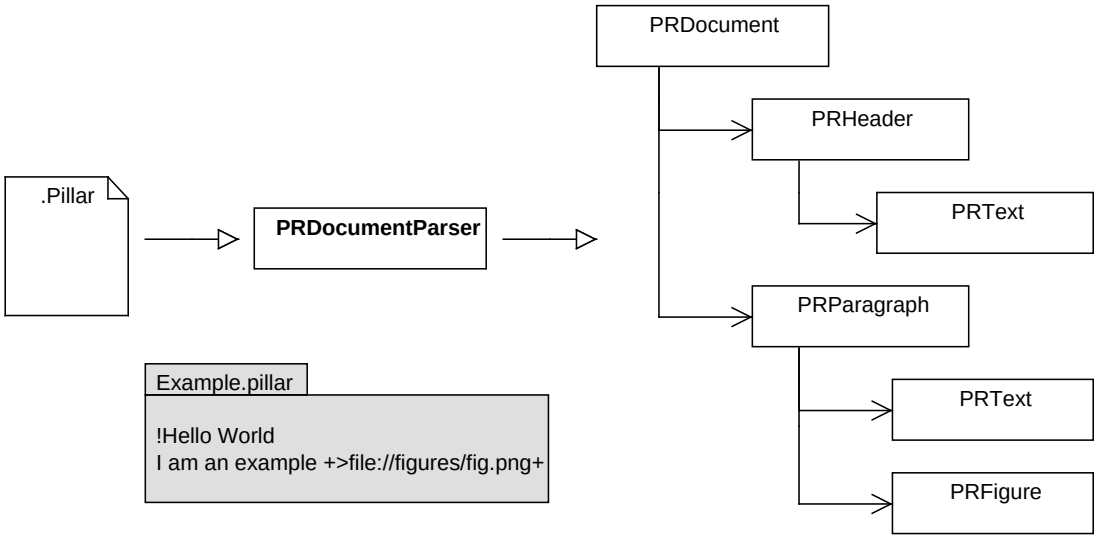
Text Editor Plugins

- You can find Pillar plugin for:
  - Emacs
  - Vim
  - TextMate
  - Atom

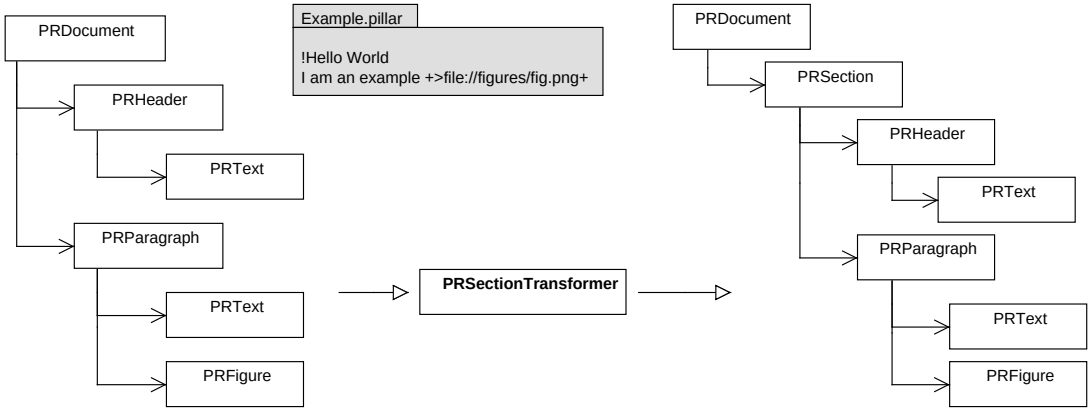
```
44
45 !!! Create a repository
46
47 In Voyage, all persistent objects are stored in a repository. The kind of repository
48
49 To use the in-memory layer for Voyage, an instance of ==VOMemoryRepository== needs to
50
51 [[[
52 repository := VOMemoryRepository new
53 ]]]
54
55 In this text, we shall however use the MongoDB backend. To start a new MongoDB reposi
56 ==VOMongoRepository==, giving as parameters the hostname and database name. For examp
57 ==mongo.db.url== execute the following code:
58
59 [[[
60 repository := VOMongoRepository
61   host: 'mongo.db.url'
62   database: 'databaseName'.
63 ]]]
```

How Pillar Works

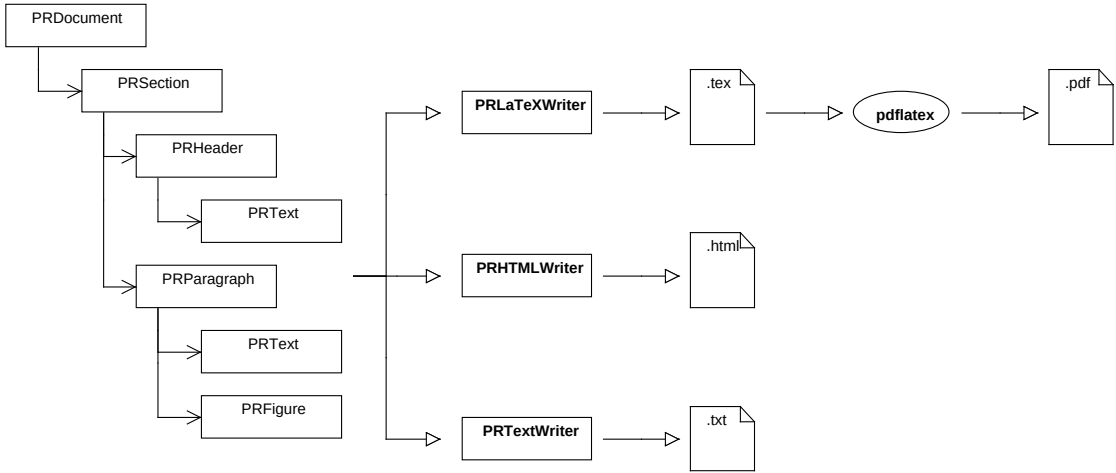
Parse


.Pillar
PRDocumentParser
PRDocument
PRHeader
PRText
PRParagraph
PRText
PRFigure
Example.pillar
!Hello World
I am an example +>file://figures/fig.png+

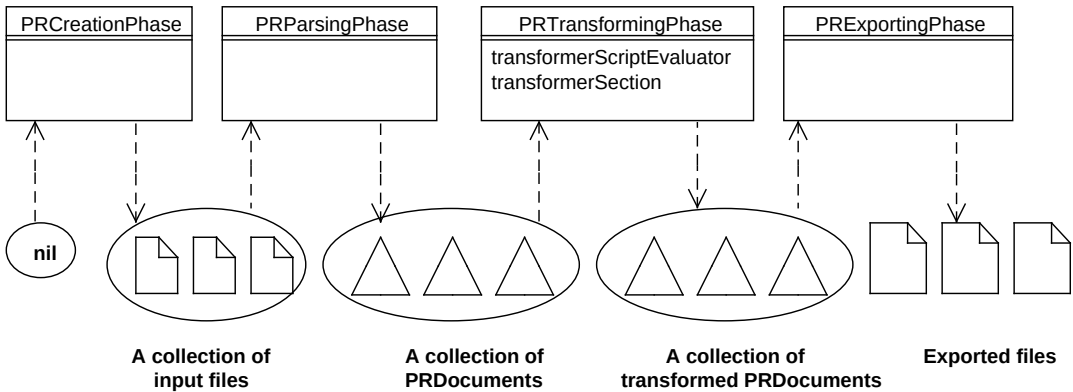
Transformation



Export


PRDocument
PRSection
PRHeader
PRText
PRParagraph
PRText
PRFigure
PRLaTeXWriter
PRHTMLWriter
PRTextWriter
.tex
pdflatex
.pdf
.html
.txt

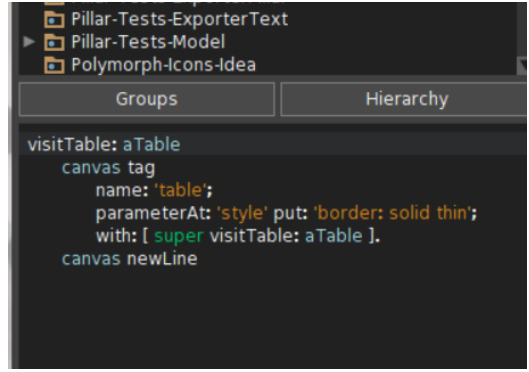
Phases System



Easy Extension

Adding an Exporter

- Write a format from a document model
- Just write a visitor for the document
- Check your exporter with tests



Adding an Annotation

- Syntax extension point
- Can take parameters
  - Describe your parameters with `Magritte`
- Work well with transformers

Examples:

- Include a Pillar file

```
${inputFile:folder/file.pillar}$
```

- Define boundaries of a Slide

```
${slide:title=title of the slide}$
```

- Divide into columns

```
${column:width=50}$
```

Adding an Annotation

```
PRAbstractAnnotation subclass: #MyNewAnnotation
  instanceVariableNames: ''
  classVariableNames: ''
  category: 'Pillar-Model-Document'
```

Adding an Annotation

```
PRAbstractAnnotation subclass: #MyNewAnnotation
  instanceVariableNames: ''
  classVariableNames: ''
  category: 'Pillar−Model−Document'
```

```
MyNewAnnotation class>>#tag
  ^ #myTag
```

Adding an Annotation

```
PRAbstractAnnotation subclass: #MyNewAnnotation
  instanceVariableNames: ''
  classVariableNames: ''
  category: 'Pillar−Model−Document'
```

```
MyNewAnnotation class>>#tag
  ^ #myTag
```

```
MyNewAnnotation>>#descriptionWidth
  <magritteDescription>
  ^ MANumberDescription new
    accessor: #width;
    required: true;
    min: 0 max: 100;
    yourself
```

Adding a Transformer

- Transform a document
  - Visit the document to transform it
- Basic transformers structure exist

Adding a Transformer

```
PRNodeTransformer subclass: #PRFileInclusion
  instanceVariableNames: ''
  classVariableNames: ''
  category: 'Pillar−Model−Transformer'
```

Adding a Transformer

```
PRNodeTransformer subclass: #PRFileInclusion
  instanceVariableNames: ''
  classVariableNames: ''
  category: 'Pillar-Model-Transformer'
```

```
PRFileInclusion>>#visitInputFileAnnotation: anInputFileAnnotation
  "I load the file and if the file exist I replace the node of the annotation by the content of the file."

  | file |
  file := anInputFileAnnotation fileWithConfiguration: self configuration.
  file exists
    ifTrue: [ self replace: (anInputFileAnnotation parseFile: file withConfiguration: self configuration) children ]
    ifFalse: [ anInputFileAnnotation errorFileNotFound: file ]
```

Adding a Transformer

- You can:
  - Define a priority
  - Define a keyword to disable a transformer
  - Generate documentation of the transformer

```
PRTransformingPhase>>#transformerInputFileOn: aCollection
  <pillarTransformer: 1 key: 'fileInclusion' documentation:
    'I visit a document and transform an ==inputFile== annotation into the content of the file.'>
  aCollection
    do: [ :each |
      PRFileInclusion new
        configuration: self configuration;
        start: each ]
```

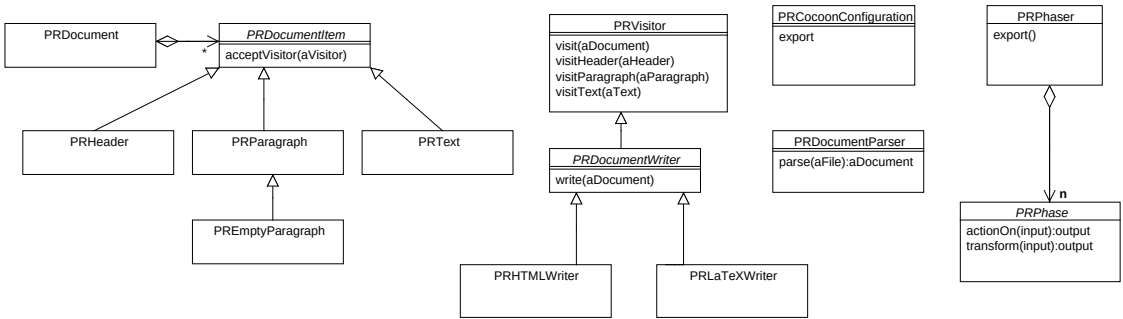
Conclusion

Conclusion

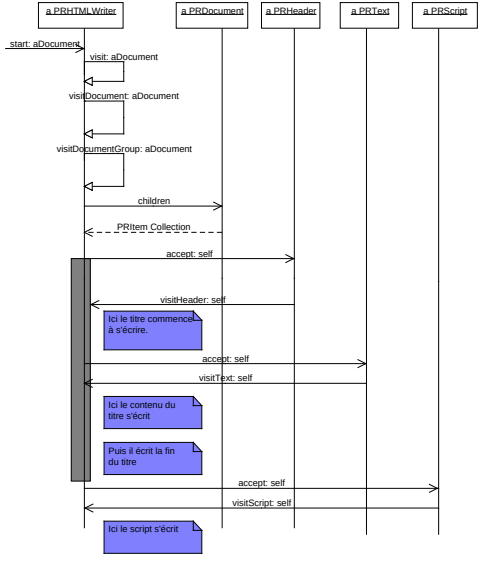
- One Format to Rule Them All
- Easy to extends
- Future Work
  - Improve the Pharo renderer
  - An `Epub` and a `DocBook` exporter would be great.
  - We have a large TODO list

Annexes

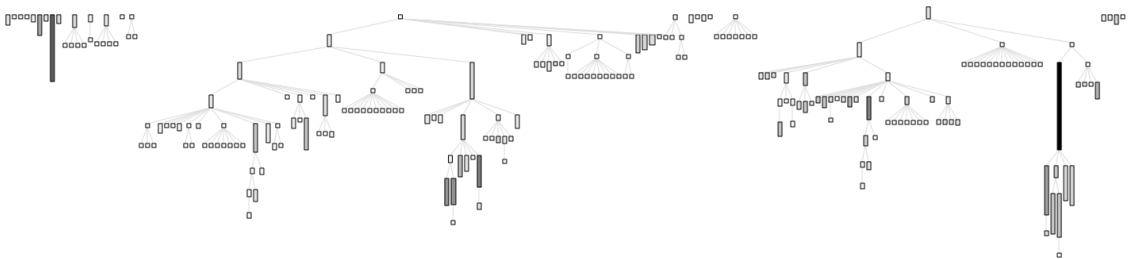
Partial Diagram of Pillar



Sequence Diagram



Moose analysis of Pillar



Non Exhaustive Syntax of Pillar

Headers

```
!Header 1
!!Header 2
!!!Header 3
!!!!Header 4
!!!!!Header 5
!!!!!!Header 6
```

Lists

```
- Unordered List
# Ordered List
```

Description (on a new line)

```
;head
:item
```

Table

```
|! Left Header |! Right Header |! Centered Header
|{ Left Item |} Right Item | Centered Item
```

Emphasis

```
""bold""
''italic''
--strikethrough--
__underscore__
==inline code==
@@subscript@@
^^sup-script^^
```

Code blocks

```
[[[label=helloScript|caption=How to print Hello World|language=Smalltalk
Transcript show: 'Hello World'.
]]]
```

Raw

```
{{{latex:
this is how you inject raw \LaTeX in your output file
}}}
```

Links

Anchor	`@anchor`
Internal Link	`*anchor*`
External Link	`*Google>www.google.fr*`
Image	`+Caption>file://image.png|width=50|label=label+`

Annotations (on a new line)

Annotation	`@@note this is a note`
Todo	`@@todo this is to do`

Comment

```
% each line starting with % is commented
```