

# 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

```
(E:\Programme\MikTeX\tex\latex\beamer\base\translator\dicts\translator-basic-dictionary\translator-basic-dictionary-English.dict)
(E:\Programme\MikTeX\tex\latex\beamer\base\translator\dicts\translator-bibliography-dictionary\translator-bibliography-dictionary-English.dict)
(E:\Programme\MikTeX\tex\latex\beamer\base\translator\dicts\translator-environment-dictionary\translator-environment-dictionary-English.dict)
(E:\Programme\MikTeX\tex\latex\beamer\base\translator\dicts\translator-months-dictionary\translator-months-dictionary-English.dict)
(E:\Programme\MikTeX\tex\latex\beamer\base\translator\dicts\translator-numbers-dictionary\translator-numbers-dictionary-English.dict)
(E:\Programme\MikTeX\tex\latex\beamer\base\translator\dicts\translator-theorem-dictionary\translator-theorem-dictionary-English.dict)
No file Pillar.nav.
(E:\Programme\MikTeX\tex\latex\psnfs\ts1phv.fd) [1(C:/ProgramData/MiKTeX/2.9/pdftex/config/pdftex.map)] [2] (Pillar.urb)
(E:\Programme\MikTeX\tex\latex\amsfonts\umsa.fd)
(E:\Programme\MikTeX\tex\latex\amsfonts\umsb.fd) [3] (Pillar.urb) [4]
(Pillar.urb) [5] [6] (Pillar.urb) [7] [8] (Pillar.urb) [9] (Pillar.urb)
[10] (Pillar.urb) [11] (Pillar.urb) [12] (Pillar.urb) [13] [14] (Pillar.urb)
[15] (Pillar.urb) [16] (Pillar.urb) [17] (Pillar.urb) [18] [19] (Pillar.urb)
[20] (Pillar.urb) [21] [22] (Pillar.urb) [23] [24] (Pillar.urb) [25]
(Pillar.urb) [26] [27] (Pillar.urb)
Runaway argument?
(You can help!
! File ended while scanning use of \@xdblarg.
<inserted text>
\par
1.477 \end{frame}
?
```

# 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



The logo for 'pier' is written in a bold, lowercase, rounded font. The letter 'i' is replaced by a vertical gradient bar that transitions from red at the top to orange at the bottom. The rest of the letters are in a dark grey or black color.

!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



Save

Meta

Attachments ▾

Link

```
1 !!Example
2
3 *You can add a link>http://www.smalltalkhub.com/#!/
  ~/Pier/Pillar*
4
5 [[[language=smalltalk
6 Weather isRaining
7     ifTrue: [self takeMyUmbrella]
8     ifFalse: [self takeMySunglasses]
9 ]]]
10
11 |!Language |!Coolness
12 |Smalltalk | Hypra cool
13 |Java      | baaad
14
15 + Example of figure !>file:///figures/pharo.png|width
  =20|label=Foo+
16
```

## Preview

# 1. Example

You can add a link

```
Weather isRaining
    ifTrue: [self takeMyUmbrella]
    ifFalse: [self takeMySunglasses]
```

Language	Coolness
Smalltalk	Hypra cool
Java	baaad



1.1. Example of figure !

<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

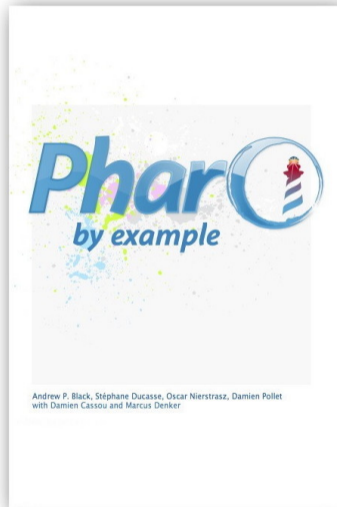
67

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. ''

${endColumns}$
```

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

The image shows two slide thumbnails from a presentation. The top thumbnail is a title slide with the text "What is Pillar ?" centered on a white background. The bottom thumbnail is a content slide titled "Problem" in red. It contains a bulleted list of four items: "We want more **documentation**", "We want many formats" (with sub-bullets for "pdf", "html", and "epub"), "We do not want to rewrite everything", and "We want it simple". To the right of the list, there is a note: "If you want them to RTFM, make a better FM." Both thumbnails have a small footer that reads "What is Pillar ?" and a small number "2" in the bottom right corner.

# 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

The screenshot shows the homepage of the Ecstatic static website generator. At the top, there is a dark navigation bar with the word "Ecstatic" on the left, and "Home" and "Blog" links on the right. The main content area is light gray and features a large white box on the left containing the title "Ecstatic", a descriptive paragraph, a bulleted list of principles, and a "Getting started" link. To the right of this box is a vertical menu with three items: "Install", "Your first Site", and "Fast Development". At the bottom, a dark footer bar contains contact information, social media links, and copyright details.

Ecstatic

Home Blog

## Ecstatic

Ecstatic is a static web-site generator engine written entirely in **Pharo**. 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](#)

[Install](#)

[Your first Site](#)

[Fast Development](#)

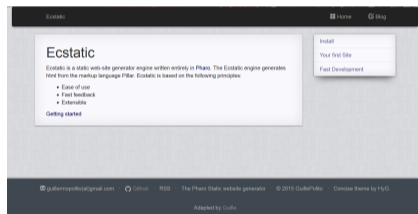
✉ guillermopolito(at)gmail.com · [Github](#) · [RSS](#) · The Pharo Static website generator · © 2015 GuillePolito · Concise theme by HyG.

Adapted by Guille



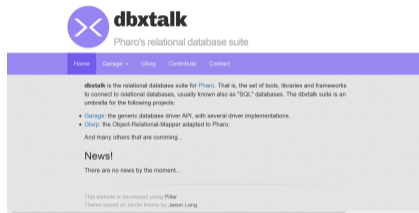
# Ecstatic: Static Websites

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



# Ecstatic: Static Websites

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



<http://guillep.github.io/ecstatic>

# Documentation renderer

Prototype by **Kasper Osterbye**

The screenshot shows the Cocoon documentation renderer interface. The top pane displays a class hierarchy with 'CcDocumentationGeneration' selected. The middle pane shows the class side view with tabs for 'Groups', 'Hierarchy', 'Class side', and 'Comments'. The 'Class side' tab is active, showing the class definition and its slots. The bottom pane shows the rendered HTML output for the class side view.

```
Object subclass: #CcDocumentationGeneration
instanceVariableNames: 'stream'
classVariableNames: ''
category: 'Cocoon-Core'
```

```
!Header 1
!!Header 2
!!!Header 3
!!!!Header 4

- Unordered List item 1
- Unordered List item 2
# Ordered list item 1
# Ordered list item 2
; head 1
; item 1
; head 2
; item 2

- ""bold""
- "italic"
- --strikethrough--
- __underscore__
```

The screenshot shows the Cocoon documentation renderer interface. The top pane displays a class hierarchy with 'CcDocumentationGeneration' selected. The middle pane shows the class side view with tabs for 'Groups', 'Hierarchy', 'Class side', and 'Comments'. The 'Comments' tab is active, showing the class definition and its slots. The bottom pane shows the rendered HTML output for the comments view.

```
Object subclass: #CcDocumentationGeneration
slots: { #stream }
classVariables: { }
category: 'Cocoon-Core'
```

```
Header 1
Header 2
Header 3
Header 4

* Unordered List item 1
* Unordered List item 2
1 Ordered list item 1
2 Ordered list item 2
head 1
item 1
head 2
item 2

* bold
* italic
* strikethrough
* underscore
```

# 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

```
Emacs: Voyage.pier

!!! Create a repository

In Voyage, all persistent objects are stored in a repository. The kind of repository that
To use the in-memory layer for Voyage, an instance of ==VOMemoryRepository== needs to be

[[[
repository := VOMemoryRepository new
]]]

In this text, we shall however use the MongoDB backend. To start a new MongoDB repository
==VOMongoRepository==, giving as parameters the hostname and database name. For example,
==mongo.db.url== execute the following code:

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

# Text Editor Plugins

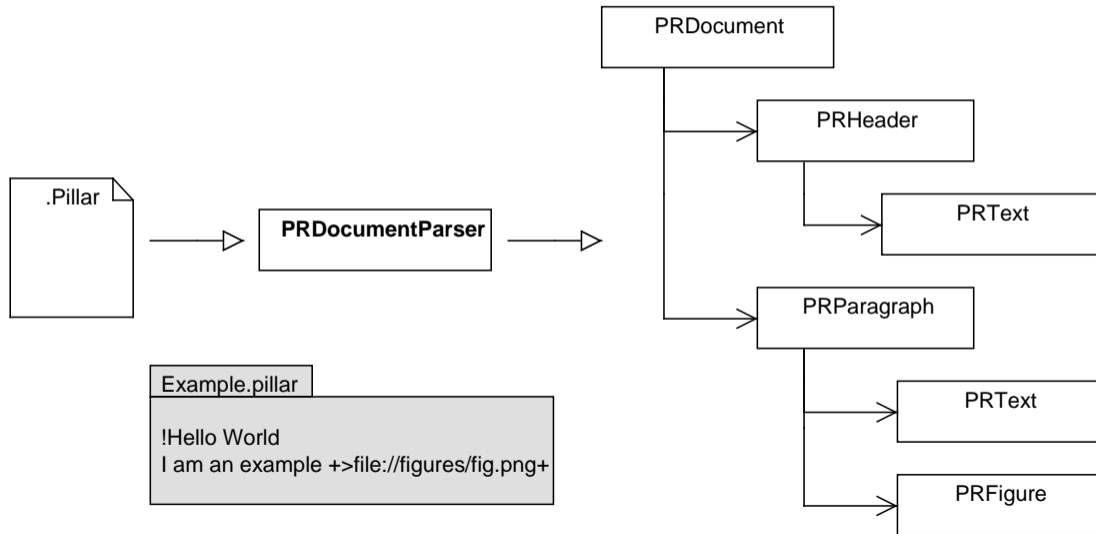
## ■ You can find Pillar plugin for:

- ▶ Emacs
- ▶ Vim
- ▶ TextMate
- ▶ Atom

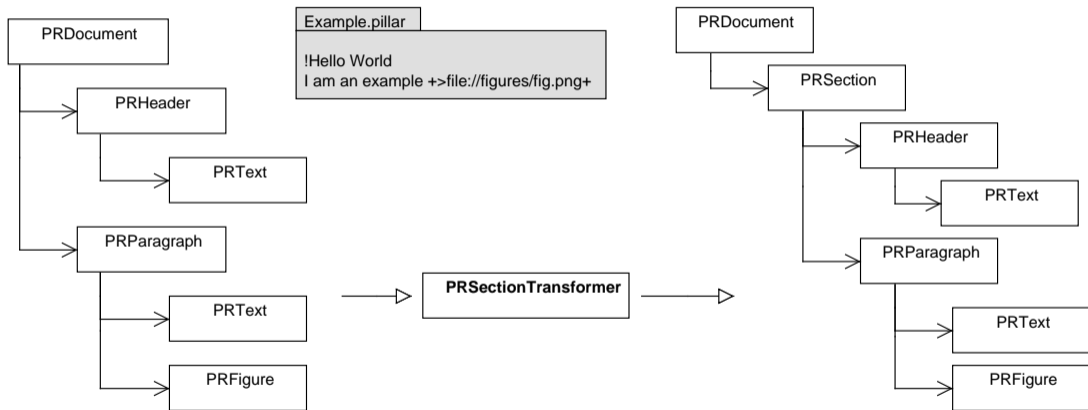
```
44
45 !!! Create a repository
46
47 In Voyage, all persistent objects are stored in a repository. The kind of repository
48
49 To use the in-memory layer for Voyage, an instance of ==VOMemoryRepository== needs to
50
51 [[[
52 repository := VOMemoryRepository new
53 ]]]
54
55 In this text, we shall however use the MongoDB backend. To start a new MongoDB repositi
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

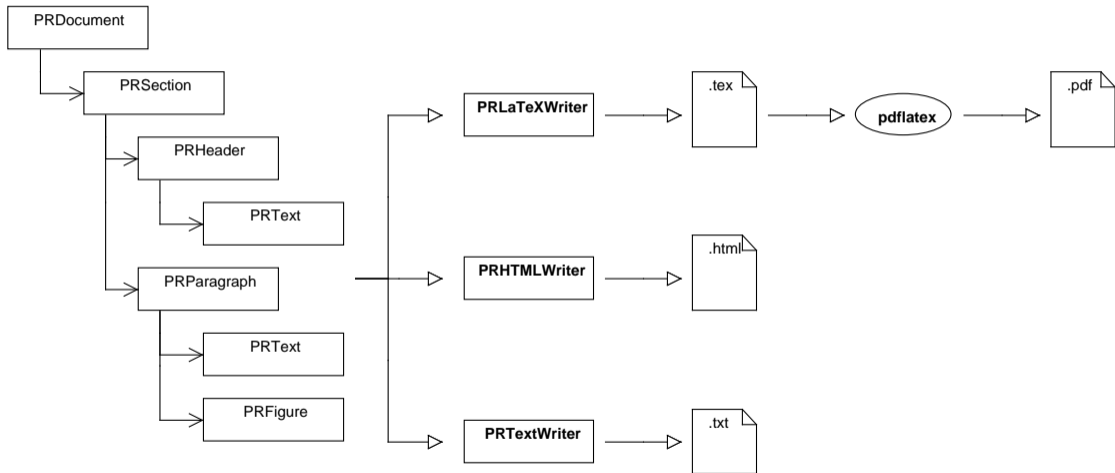
# Parse



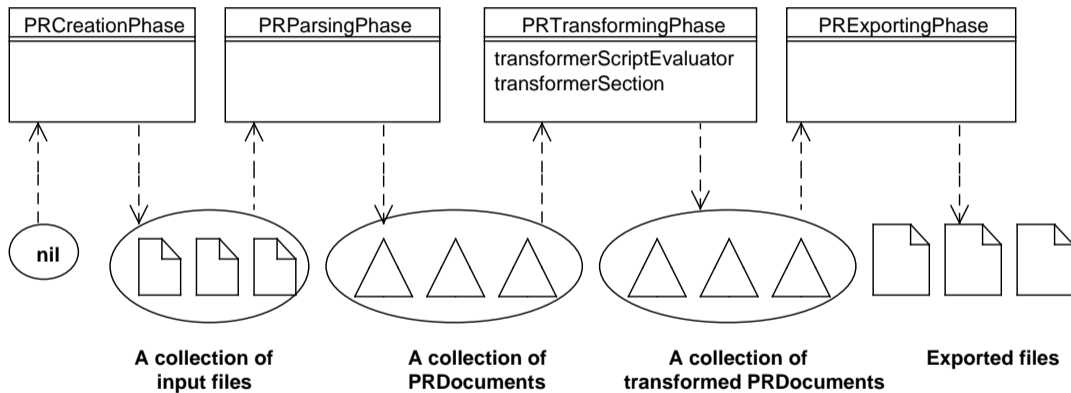
# Transformation



# Export



# Phases System

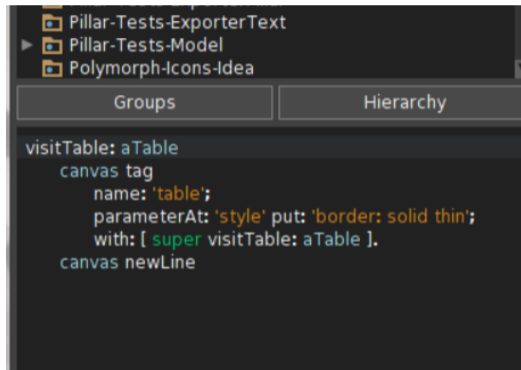




# Easy Extension

# Adding an Exporter

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



The screenshot shows an IDE interface. At the top, a file explorer displays a tree structure with folders: Pillar-Tests-ExporterText, Pillar-Tests-Model, and Polymorph-Icons-Idea. Below the explorer are two tabs: 'Groups' and 'Hierarchy'. The main editor area shows the following code snippet:

```
visitTable: aTable  
  canvas tag  
    name: 'table';  
    parameterAt: 'style' put: 'border: solid thin';  
    with: [ super visitTable: aTable ].  
  canvas newLine
```

# 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.'>
aCollection
do: [ :each |
  PRFileInclusion new
    configuration: self configuration;
    start: each ]
```

# Conclusion

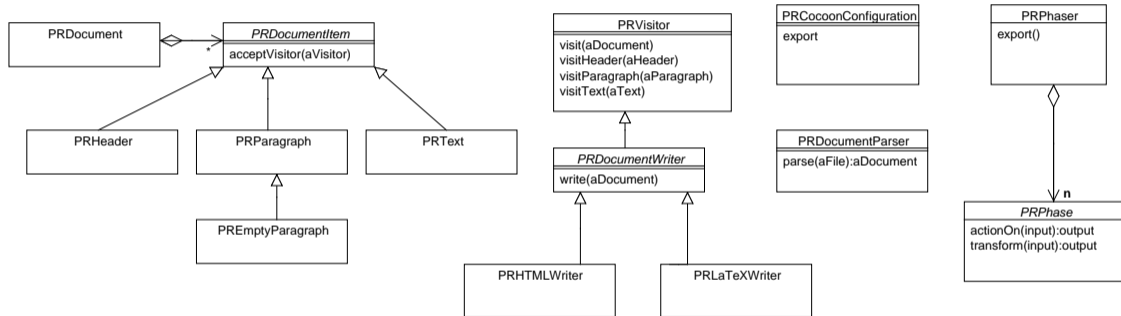
# Conclusion

- 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

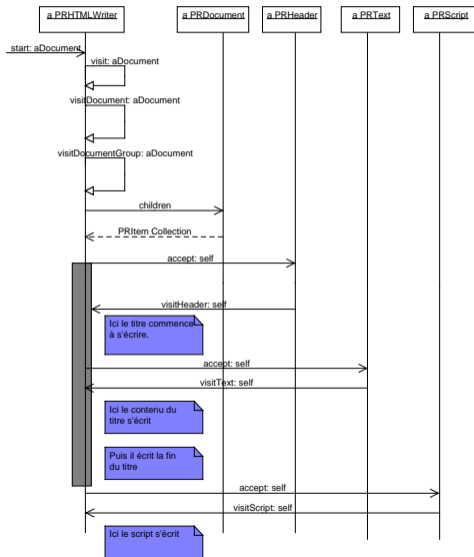


# Annexes

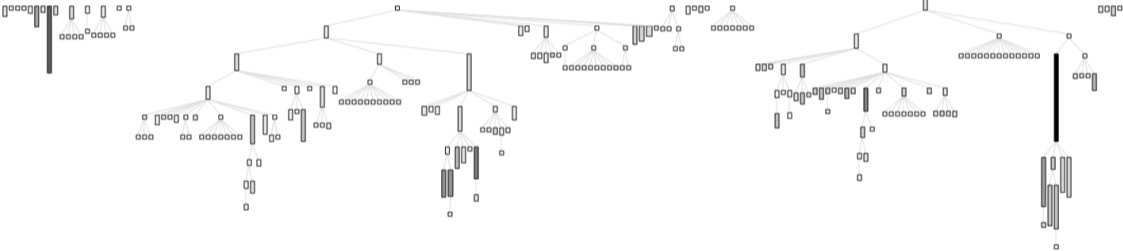
# Partial Diagram of Pillar



# 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      ! Right      ! Centered

Header      Header      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      \*anchor\*

Link

External      \*Google>www.google.fr\*

Link

Image      +Caption>file://image.png|w-  
idth=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