

Documentation

Technical documentation support in 

Pavel Krivanek

Public secret:

Developers don't like to write documentation

- It's Time-Consuming
- It's Not "Fun"
- It Can Become Outdated Quickly
- Assumption of Obsolescence - code is "self-documenting"
- Belief in Tools (Javadoc or Doxygen)
- Lack of Incentives (ship features!)
- Not Everyone Feels Equipped
 - requires a different skill set
 - fear of incompleteness or inaccuracy

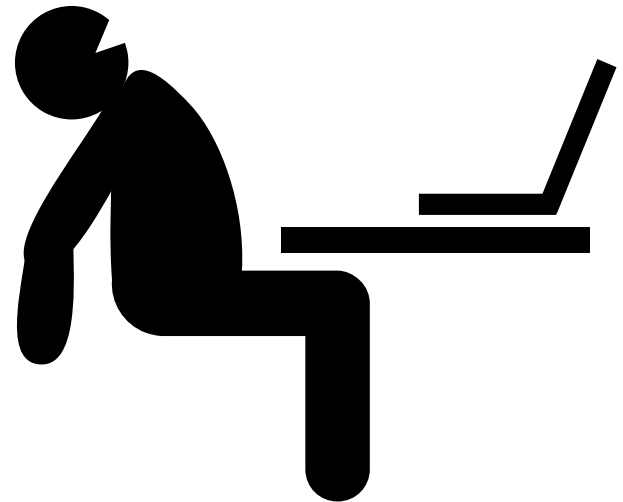


Good quality and up to date documentation is important

Lepiter, Documentation Browser...

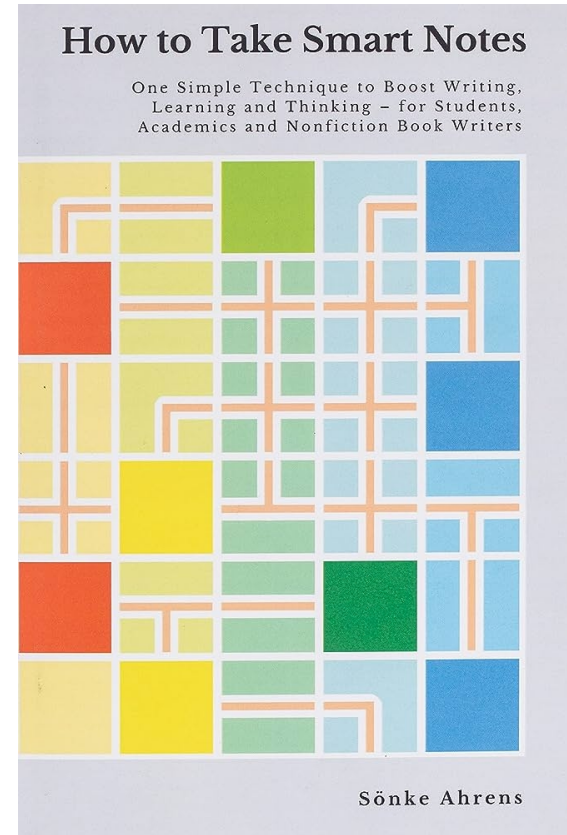
When?

- Sometime after...
- Sometime after
...but this time for real
- During the development
 - quickly outdated, rewrites
- Before
 - DDD - Documentation-driven development
- Literate programming
- Combination

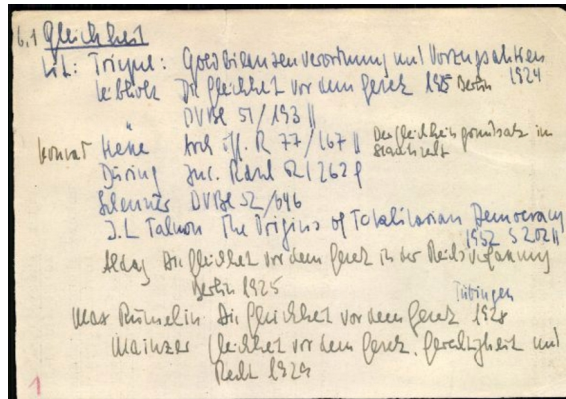


Smart notes

- Academic and nonfiction writing
- Based on a technique used by Niklas Luhmann
- Zettelkasten



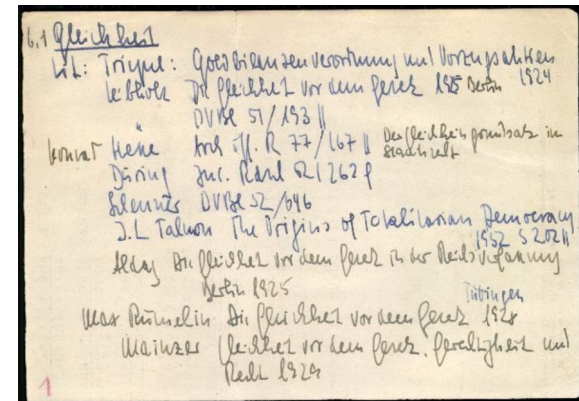
Zettelkasten



- Note-taking system
- Linked idea cards
- Individual idea per card with an ID
- Continuous knowledge growth

Zettelkasten

- When writing an output document, all the content is already there, it just needs to be filtered, cleaned and reviewed
- Can it be used for the code documentation?



Computer program

```
collectionAssert: aBlock
```

```
"Evaluate the assertion block for each of the FileAttribute cache modes"
```

```
self attributesCollectionDo:  
  [ :each | self assert: (aBlock value: each) ]
```

implementors



```
attributesCollectionDo: aBlock
```

```
"Evaluate the supplied block for each of the attribute cache modes"
```

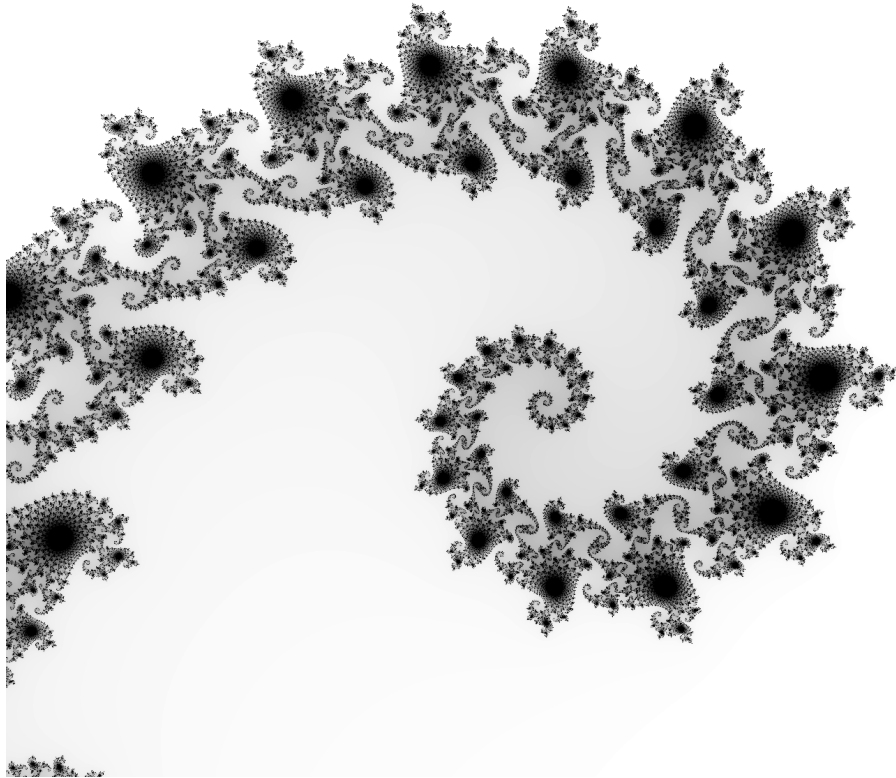
```
^self attributesCollection do: aBlock
```

```
actionsDo: aBlock
```

```
self actionMap do: aBlock
```

senders





Computer program
=
Fractal structure

...independently on the representation

```

;Read in the Joystick $---FUCLR
ReadJoystick:
    clrb                ;We'll build up the result in B

    lda #%11111111     ;Disable Keyboard
    sta $FF02

;Fire button
    lda $FF00          ;Bit 0=Fire
    rora               ;Get Fire (Bit 0)
    rolb

;X axis Tests
    lda #%00000100    ;Bit3=0 X-axis
    sta $if01         ;PIA0-A Control
                    ; 543210-- TestVal

    lda #%11100000    ;Test DAC>56 (Right)
    sta $FF20         ;Store test value into DAC
    lda $FF00         ;Bit 7=Test Result
    rola              ;Right (True if Bit 7=1)
    rolb
                    ; 543210-- TestVal

    lda #%00011100    ;Test DAC>7 (Left)
    sta $FF20         ;Store test value into DAC
    lda $FF00         ;Bit 7=Test Result
    rola              ;Left (True if Bit 7=0)
    rolb

;Y axis Tests
    lda #%00001100    ;Bit3=1 Y-axis
    sta $if01         ;PIA0-A Control
                    ; 543210-- TestVal

    lda #%11100000    ;Test DAC>56 (Down)
    sta $FF20         ;Store test value into DAC
    lda $FF00         ;Bit 7=Test Result
    rola              ;Down (True if Bit 7=1)
    rolb
                    ; 543210-- TestVal

    lda #%00011100    ;Test DAC>7 (Up)
    sta $FF20         ;Store test value into DAC
    lda $FF00         ;Bit 7=Test Result
    rola              ;Up (True if Bit 7=0)
    rolb

;Fix results
    eorb #%11101010  ;;Flip Up and Left bits
    tfr b,a          ;A contains $---FUCLR
    rts
    
```

```

EXPORT(int)
ve_main_with_parameters(VParameters *parameters)
{
    // HACK: In some cases we need to add an explicit --interactive option to the image;
    // otherwise error = ve_parameters_image_interactive_image_parameter(parameters);
    if (error) {
        return 1;
    }

    if(parameters->setDefaultImage && !parameters->defaultImageFound){
        // If error != 0, image has been specified, and no default image has been found.
        ve_printImage(stdout);
        return 1;
    }

    installErrorHandlers();
    setProcessArguments(parameters->processArgv, parameters->processArgv);
    setProcessEnvironment(parameters->env, parameters->env);
    logInfo("Opening image: %s", parameters->imageFilename);

    // This initialization is required because it makes awful, awful, awful code to calculate
    // the location of the machine code.
    // Usually, it can be cached.
    setStackPageAddress();

    // Retrieve the working directory.
    char *workingDirectory = (char*)calloc(1, FILENAME_MAX);
    if(!workingDirectory){
        logError("Failed to obtain the current working directory: %s", ve_error_code_to_string(error));
        return 1;
    }
    error = ve_path_get_current_working_dir_into(workingDirectory, FILENAME_MAX);
    if(error){
        logError("Failed to obtain the current working directory: %s", ve_error_code_to_string(error));
        return 1;
    }
    logInfo("Working Directory %s", workingDirectory);

    LOG_SIZE(int);
    LOG_SIZE(long);
    LOG_SIZE(long long);
    LOG_SIZE(void*);
    LOG_SIZE(void**);
    LOG_SIZE(void***);
    LOG_SIZE(void****);
    LOG_SIZE(double);

    #if defined(PHARO_VE_16_BITS)
        veMainThread = parameters->isWorker;
        return veMainThread(parameters);
    #else
        return veMainThread(parameters);
    #endif
}

EXPORT(int)
ve_main(int argc, const char** argv, const char** env)
{
    VParameters parameters;
    ve_parameters_init(&parameters);
    parameters.environmentVector = env;
    parameters.processArgv = argv;
    parameters.processArgv = argv;

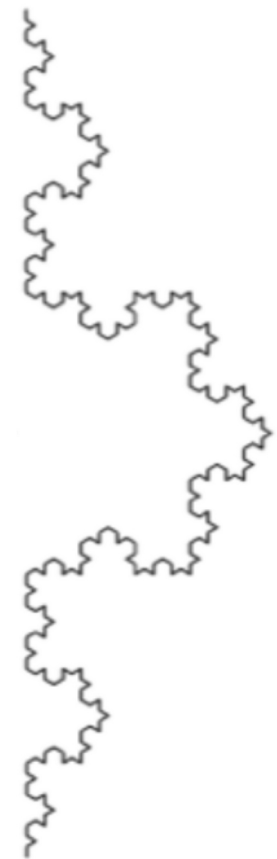
    // Did we succeed on parsing the parameters?
    ve_errorCode error = ve_parameters_parse(&parameters);
    if(error){
        if(error == VE_ERROR_EXIT_SUCCESS) return 0;
        return 1;
    }

    // Do we need to select an image file interactively?
    if(parameters->interactiveSelectImage && parameters->setDefaultImage && !parameters->defaultImageFound && !ve_file_dialog_is_none()){
        veFileDialog fileDialog;
        fileDialog.title = "Select Pharo Image to Open";
        fileDialog.message = "Choose an image file to execute";
        fileDialog.filterExtension = "Pharo Image (*.image)";
        fileDialog.filterExtension = ".image";
        fileDialog.defaultFilename = "image";
        fileDialog.defaultFilename = "image";

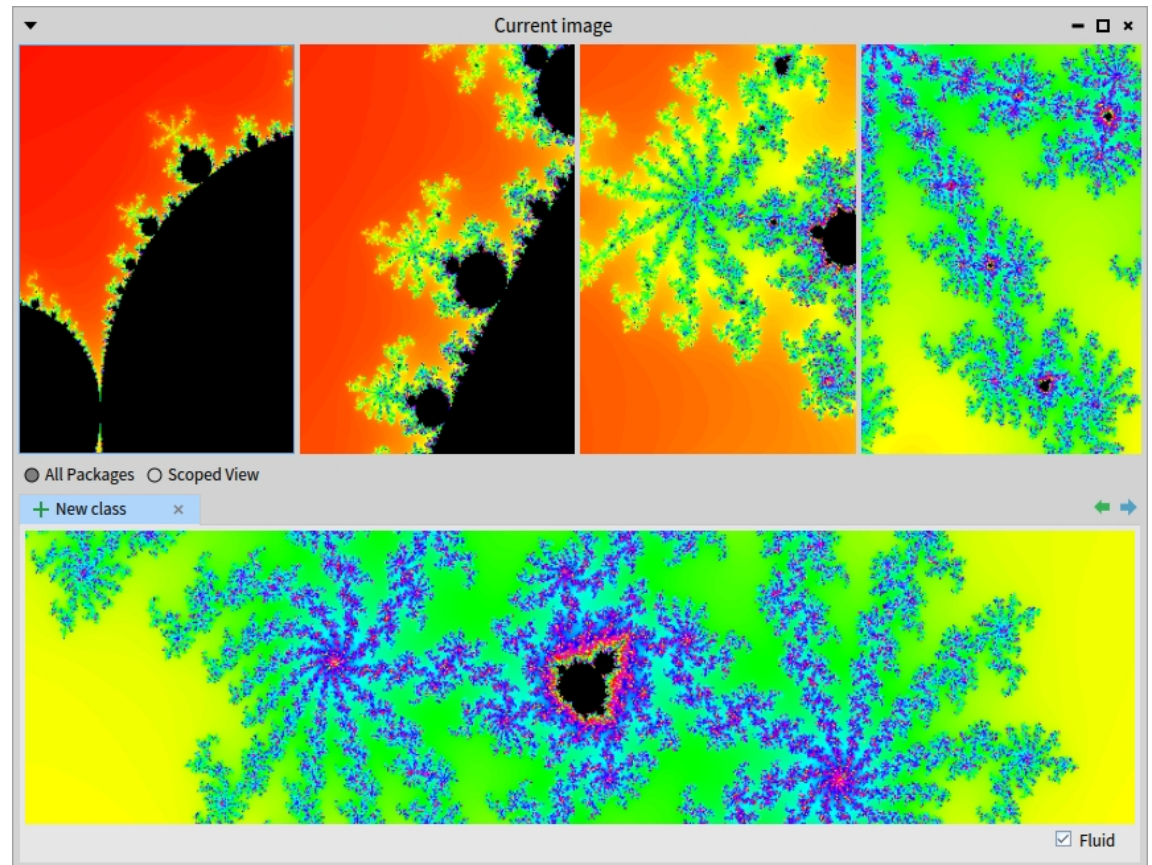
        error = ve_file_dialog_run_modal_open(&fileDialog);
        if(!fileDialog.succeeded){
            ve_file_dialog_destroy(&fileDialog);
            return 0;
        }

        parameters.imageFilename = strdup(fileDialog.selectedFilename);
        parameters.setDefaultImage = fileDialog.selectedFilename;
        ve_file_dialog_destroy(&fileDialog);

        int exitCode = ve_main_with_parameters(&parameters);
        parameters.destroy(&parameters);
        return exitCode;
    }
}
    
```

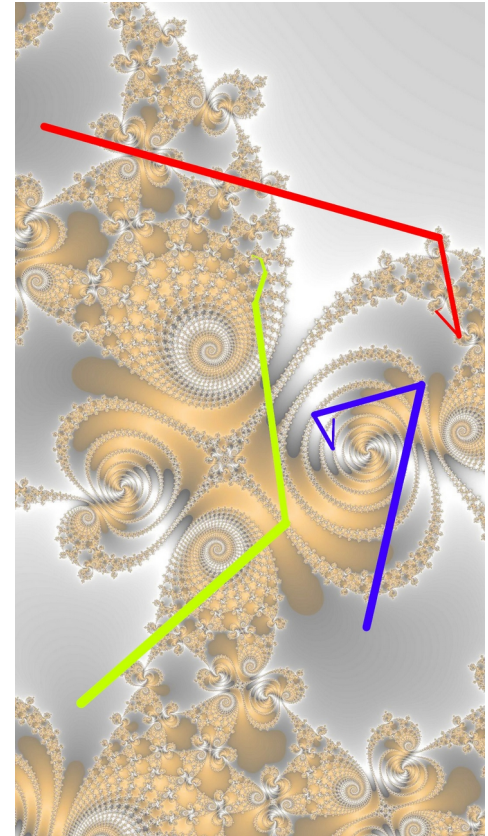


Some tools
do it better!



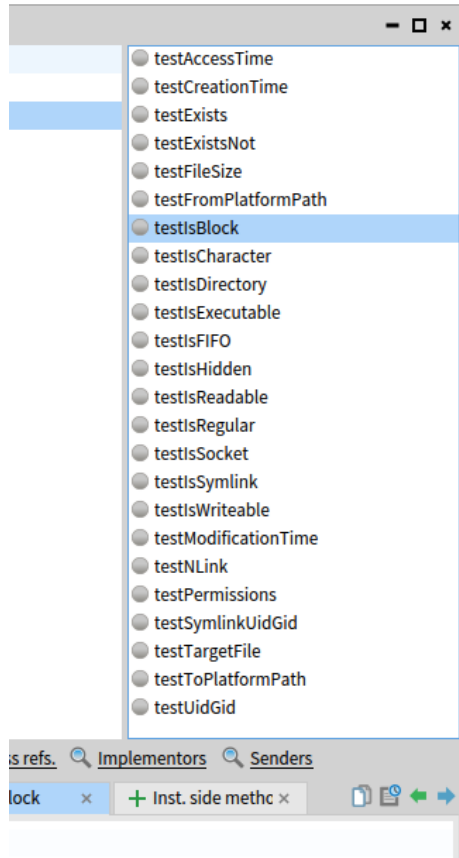
How to find a path in a code fractal?

- Various ways of indexing
 - Alphabetical
 - Order of creation
 - Various control flows
 - TDD logic
 - Dependency order
 - Loading order
 -



Pharo – code indexing

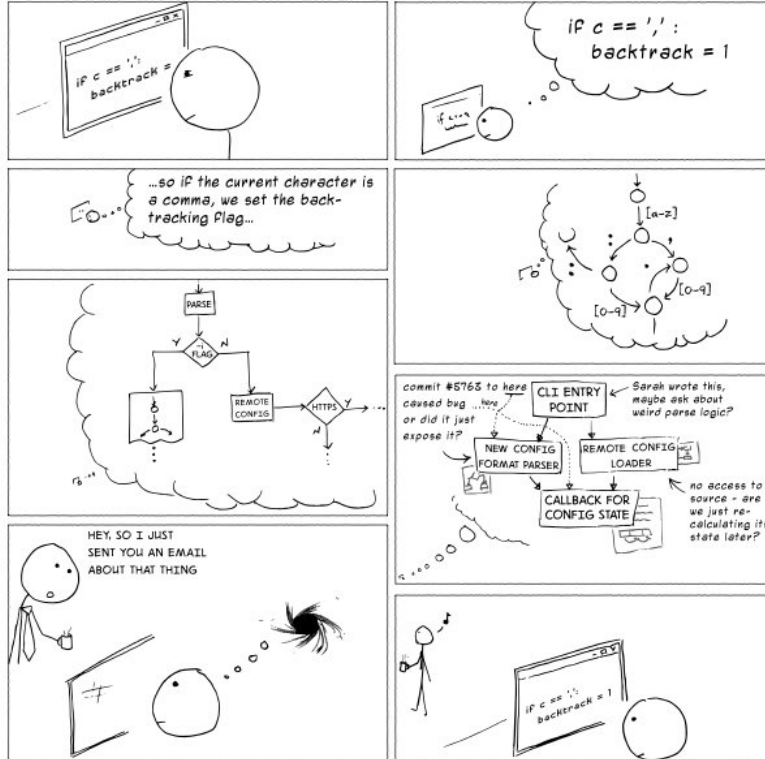
- alphabetical
- by protocols
- by variables
- mostly one class only...



Quick
notes



THIS IS WHY YOU SHOULDN'T INTERRUPT A PROGRAMMER



© Jason Heeris 2015 LICENSE: CC BY-NC-ND 2.5 AU heeris.id.au

Tracking the thought process

Writing is thinking

Documentation snippets
+
Code indexing
+
Quick notes
+
Logged rubber duck
+
Thought process tracking

ONE TOOL
???

A cool name required!

Metacello new

baseline: 'DocumentationSupport';

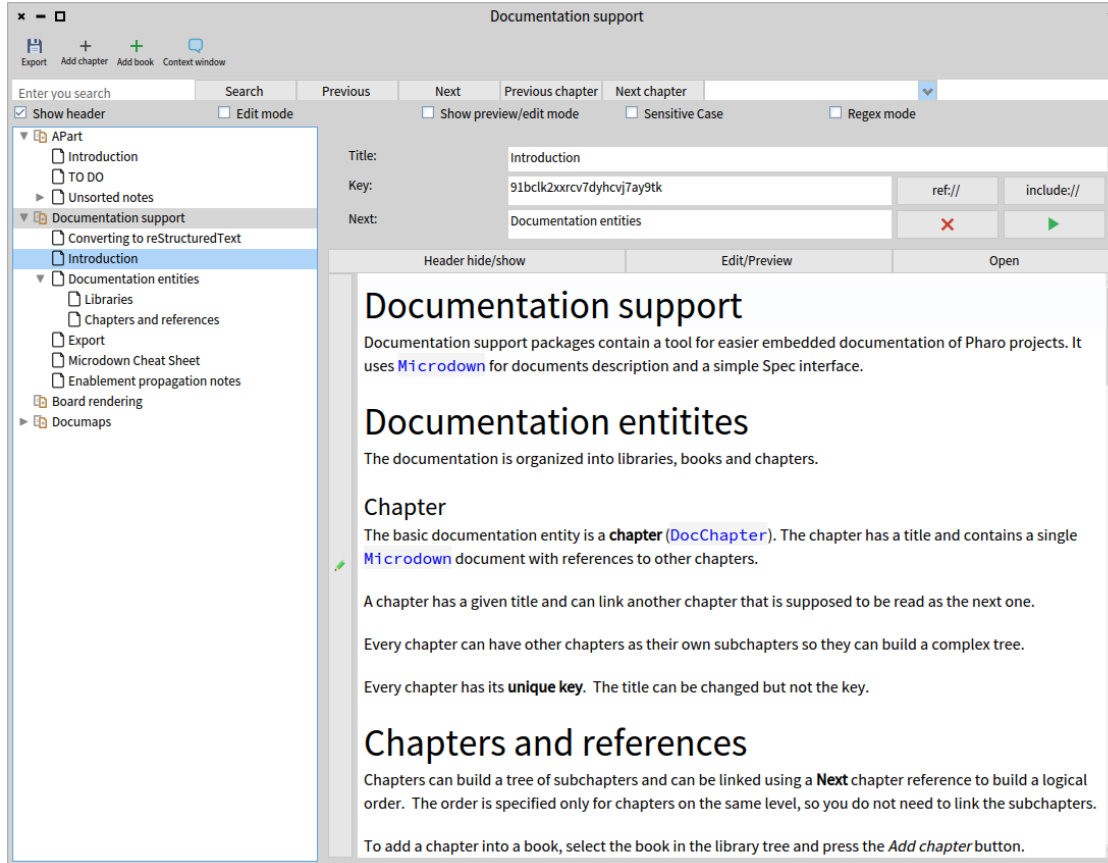
repository: 'github://bauing-schmidt/DocumentationSupport:main';

load.

Documentation Support

- Prototype
- Vision and requested features specification
- Asked RMoD Team at Inria to improve it
 - Intern **Leo Frere**
 - improved Search
 - UI enhancements...





The screenshot shows the 'Documentation support' application window. It features a sidebar with a tree view of the documentation structure, including sections like 'APart', 'Documentation support', and 'Documentation entities'. The main content area displays the text of the selected 'Introduction' chapter, which describes the tool's purpose and usage. The interface includes search and navigation controls at the top, and a metadata table for the current document.

Header hide/show	Edit/Preview	Open
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Documentation support

Documentation support packages contain a tool for easier embedded documentation of Pharo projects. It uses [Microdown](#) for documents description and a simple Spec interface.

Documentation entities

The documentation is organized into libraries, books and chapters.

Chapter

The basic documentation entity is a **chapter** ([DocChapter](#)). The chapter has a title and contains a single [Microdown](#) document with references to other chapters.

A chapter has a given title and can link another chapter that is supposed to be read as the next one.

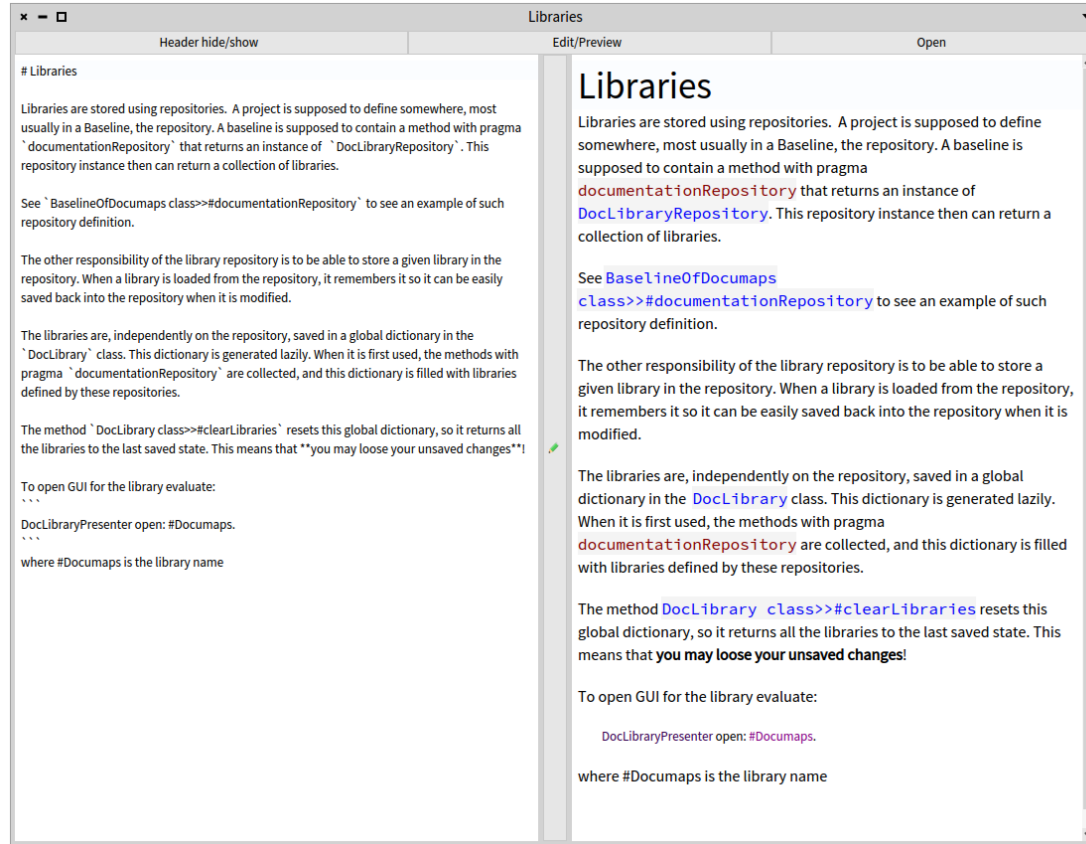
Every chapter can have other chapters as their own subchapters so they can build a complex tree.

Every chapter has its **unique key**. The title can be changed but not the key.

Chapters and references

Chapters can build a tree of subchapters and can be linked using a **Next** chapter reference to build a logical order. The order is specified only for chapters on the same level, so you do not need to link the subchapters.

To add a chapter into a book, select the book in the library tree and press the *Add chapter* button.



Basics

- Microdown
- Spec
- Immediate preview
- Changes propagation to all windows

Model

Title:	Introduction		
Key:	91bclk2xxrcv7dyhcvj7ay9tk	ref://	include://
Next:	Documentation entities	✘	▶

- Basic unit: Chapter
- The best name?
- “free flowing”
- Key (for references)
- Title independent
- Organized to Books, Libraries

CreateProcessW function (processthreadsapi.h)

In this article

- Notes
- Parameters
- Return value
- Errors
- See also

Creates a process and its primary thread. The new process can be the newly created or the calling process.

Syntax

```
BOOL WINAPI CreateProcessW(
    [in] LPCTSTR lpApplicationName,
    [in, optional] LPCTSTR lpCommandLine,
    [in] LPSECURITY_ATTRIBUTES lpProcessAttributes,
    [in] LPSECURITY_ATTRIBUTES lpThreadAttributes,
    [in] BOOL bInheritHandles,
    [in] DWORD dwCreationFlags,
    [in, optional] LPCTSTR lpCurrentDirectory,
    [in] LPCTSTR lpFileName,
    [in, optional] LPCTSTR lpCommandLine,
    [in] LPSECURITY_ATTRIBUTES lpProcessAttributes,
    [in] LPSECURITY_ATTRIBUTES lpThreadAttributes,
    [in] LPCTSTR lpCurrentDirectory);
```



Return value

If the function succeeds, the return value is nonzero.
If the function fails, the return value is zero. To get extended error information, call GetLastError.

By default, passing TRUE as the value of the inheritHandles parameter causes all inheritable handles to be inherited by the new process. This can be problematic for applications which create processes from multiple threads simultaneously yet desire each process to inherit different handles. Applications can use the UpdateProcThreadAttribute function with the PROC_THREAD_ATTRIBUTE_HANDLE_LIST parameter to provide a list of handles to be inherited by a particular process.

```
LPCTSTR lpApplicationName = _T("cmd.exe");
LPCTSTR lpCommandLine = _T("cmd.exe /c dir");
LPCTSTR lpCurrentDirectory = _T(".");
```

Remarks

The process is assigned a process identifier. The identifier is valid until the process terminates. It can be used to identify the process, as specified in the OpenProcess function to open a handle to the process. The valid thread in the process is also assigned a thread identifier. It can be specified as the OpenThread function to open a handle to the thread. The identifier is valid until the thread terminates and can be used to uniquely identify the thread after the process. These identifiers are returned in the PROCESS_INFORMATION structure.

Security Remarks

The first parameter, lpApplicationName, can be NULL, in which case the executable name must be in the white space defined using pointers to by lpCommandLine or path name has a space in it, there is a risk that a different executable could be run in case of the way the function parses queries. The following example is dangerous because the function will attempt to run "Programmer", if it fails, instead of "MyApp.exe".

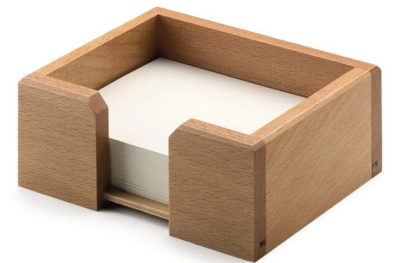
```
LPCTSTR lpApplicationName = _T("cmd.exe");
LPCTSTR lpCommandLine = _T("Program Files\\MyApp - /C");
```

If a malicious user were to create an application called "Program.exe" in a system, any program that invokes the CreateProcess function using the Program Files directory will run this application instead of the intended application.

```
LPCTSTR lpApplicationName = _T("cmd.exe");
LPCTSTR lpCommandLine = _T("Program Files\\MyApp - /C");
```

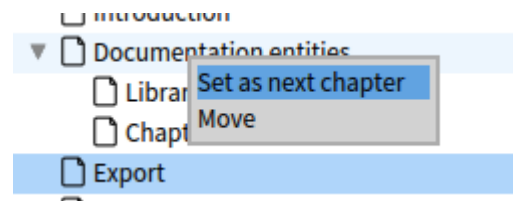
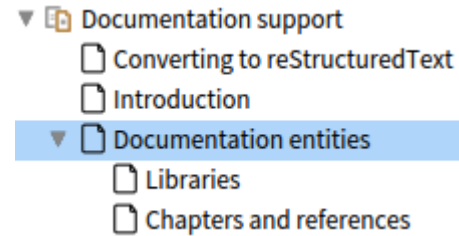
Examples

For an example, see Creating Processes.
Note
The process identifier handle defines CreateProcess as an alias, which automatically selects the USER or USER32 version of this handle based on the definition of the UNICODE preprocessor constant. Mixing usage of the resulting symbol with code that not resulting symbol can lead to compilation, but that is compilation or runtime errors. For more information, see Constants for Function Prototypes.



Chapter

- Subchapters
- Optional explicit order
- Both set by drag&drop



References

- Direct reference

[Displayed reference text](ref://3z9zm765drcfzotsq9dvc48d4)

- navigates to the given chapter

- References with inclusion

[Displayed reference text](include://3z9zm765drcfzotsq9dvc48d4)

- includes given chapter
- for longer documentation documents

Key:

Serialization

- External tools friendly
- Multiple projects documentation

- Library folder
 - evxt7d77f32hb18qgh8kt082w (book folder)
 - description.md

```
title: 'Documaps'
key: 'evxt7d77f32hb18qgh8kt082w'
```
 - 3ckid6ubue60xjwpy8dxv2q1s.md

```
key: '3ckid6ubue60xjwpy8dxv2q1s'
parent: 'evxt7d77f32hb18qgh8kt082w'
nextChapter:

# Replay of events

The subject of this chapter is...
```

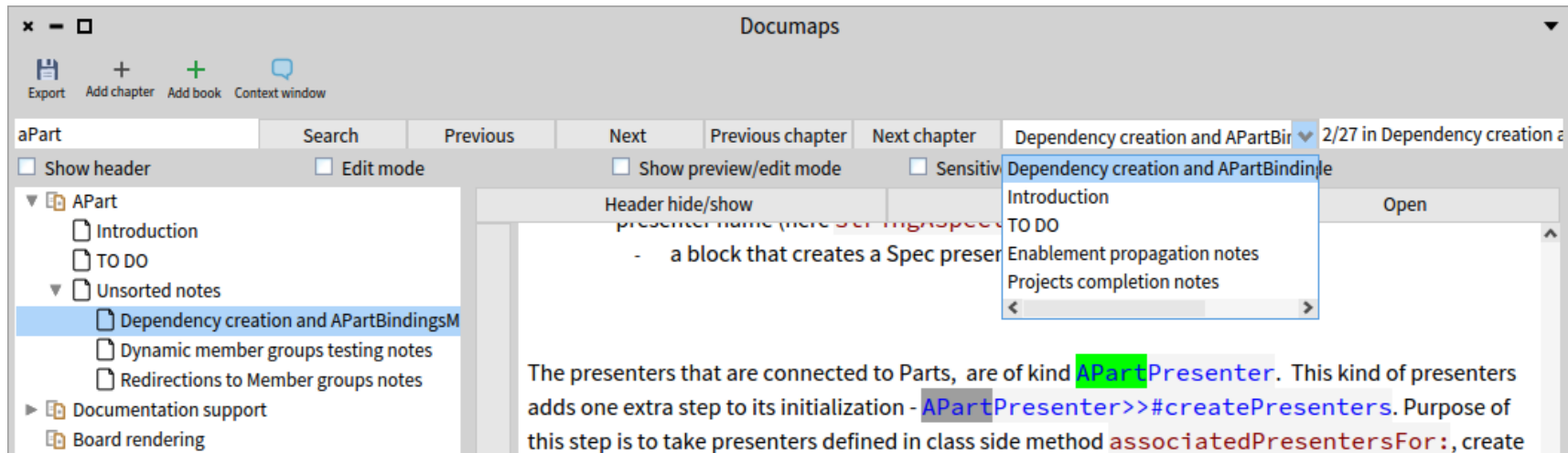
Serialization

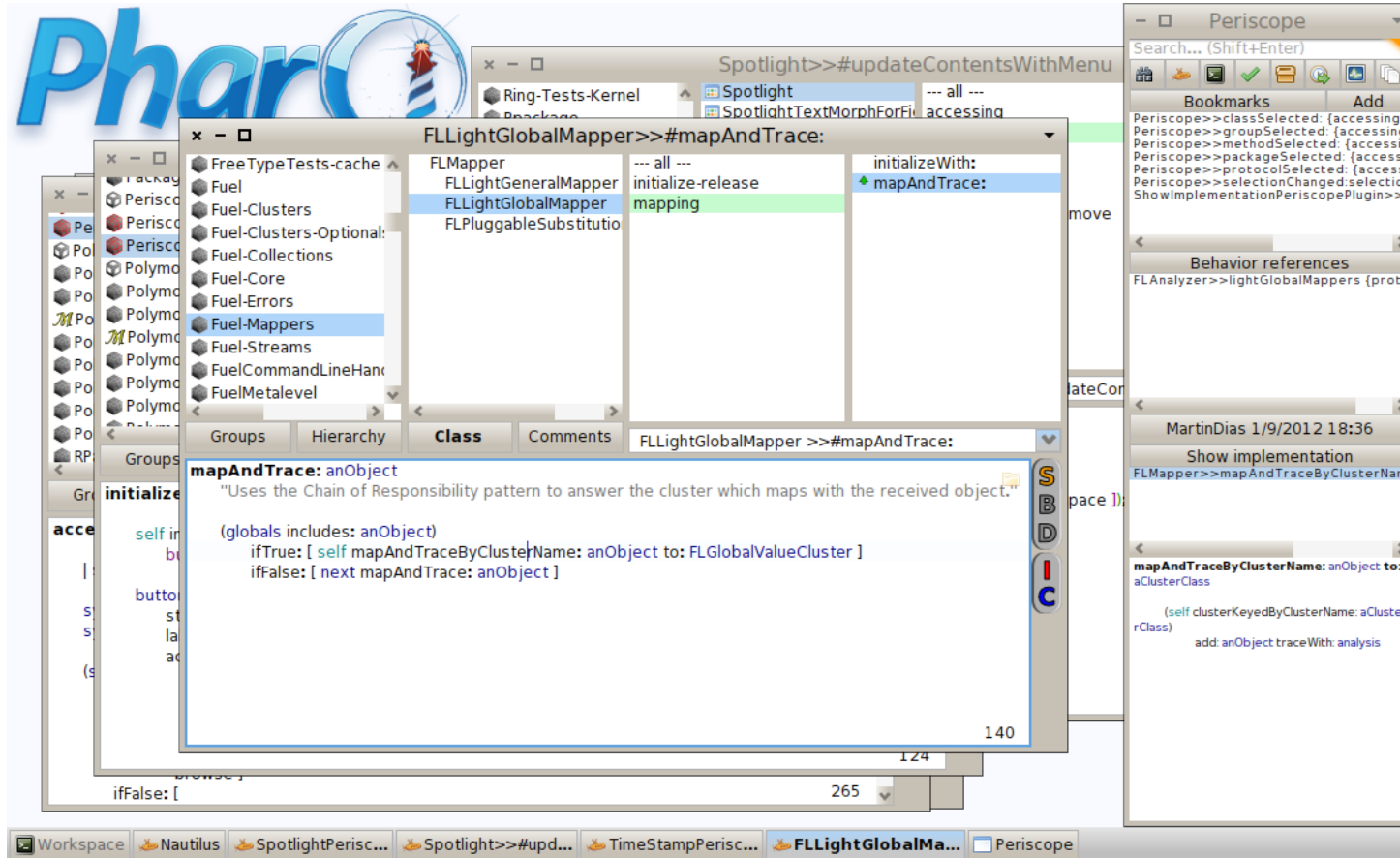
- Git & other tools friendly
- Stored in the Git repository next to the source code
- SPHINX support
 - Python Documentation Generator
 - By Massimo Nocentini



Search

- Jumping between results (bidirectional)
- Quick jumping between found chapters



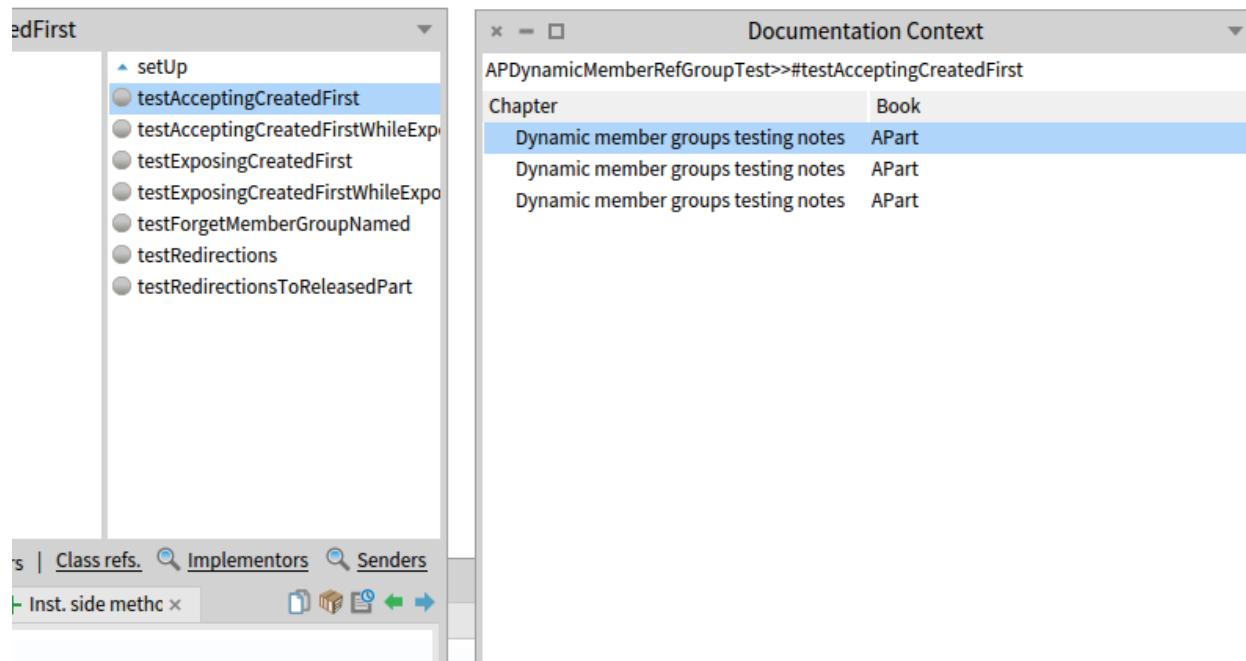


The screenshot shows the Pharo IDE interface. The Spotlight tool is active, displaying the class hierarchy for `FLLightGlobalMapper`. The `mapAndTrace` method is highlighted. The Periscope tool is also active, showing the implementation of the `mapAndTrace` method. The implementation uses the Chain of Responsibility pattern to answer the cluster which maps with the received object.

```

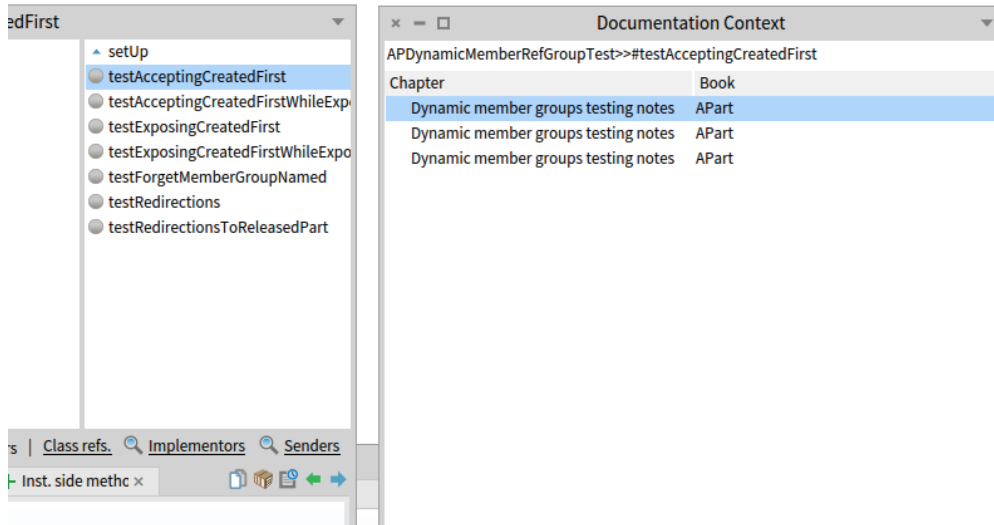
mapAndTrace: anObject
    "Uses the Chain of Responsibility pattern to answer the cluster which maps with the received object."
    (globals includes: anObject)
    ifTrue: [ self mapAndTraceByClusterName: anObject to: FLGlobalValueCluster ]
    ifFalse: [ next mapAndTrace: anObject ]
    
```

Documentation Context



- Standalone window
- Calypso aware
- Shows references to the currently browsed method, class, package
- Bidirectional references

Just the beginning

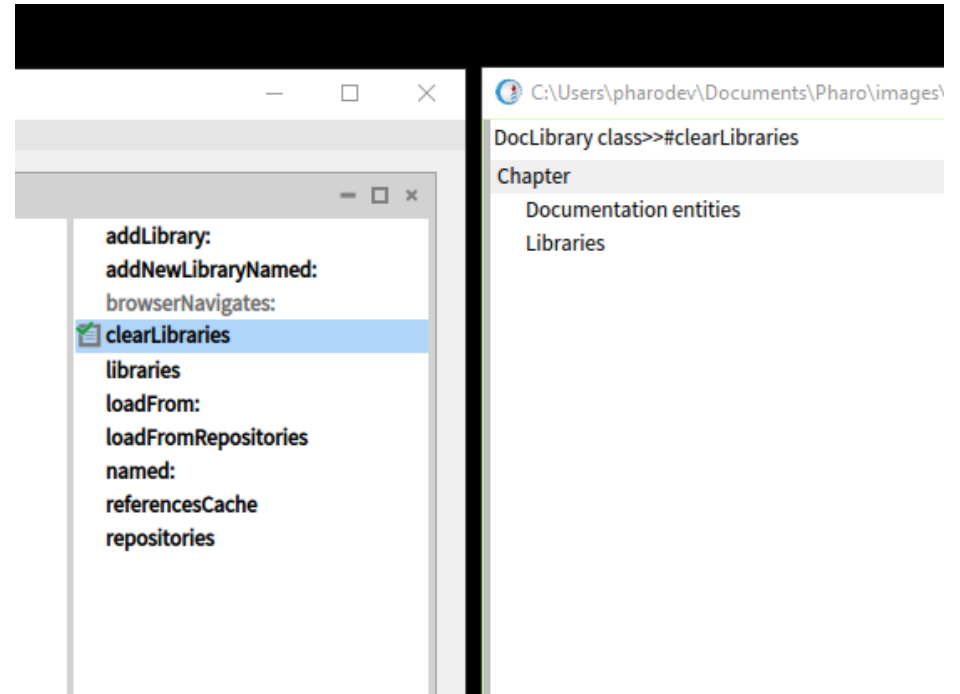


Future ideas

- Immediate preview
- Smarter context display
- Cooperation with the source editor
- Automatic browser history logging
- Bookmarks
-

Native OS windows

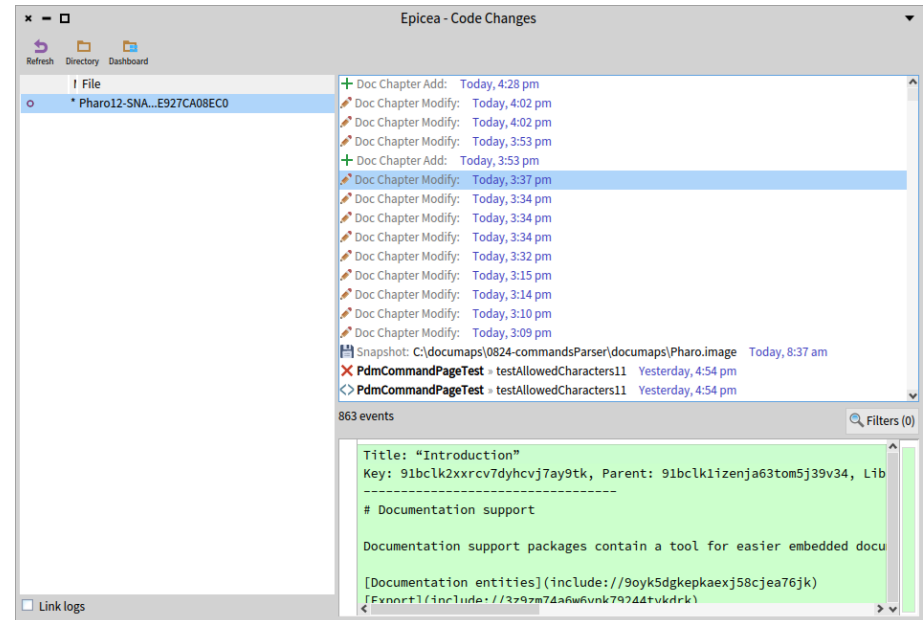
Multiple monitors



(ClyFullBrowserMorph openOn: ClyNavigationEnvironment currentImage) openInExternalWindow

Epicea integration

- Prevents loss of contents
- Not every single character
- Currently an experimental feature



Integration issues

An external tool will never
have close, solid integration
with the IDE

- Tonel format?
- Iceberg integration?

```
commitDocumentationWith: script
```

```
| scriptFile repo win p |  
scriptFile := 'commitScript.sh' asFileReference ensureDelete.  
scriptFile writeStreamDo: [ :s |  
    s nextPutAll: script withUnixLineEndings ].
```

```
repo := IceRepository registry detect: [ :each |  
    each name = 'documaps' ].
```

```
LibC system: (self gitBashCommand format: {  
    repo location asFileReference fullName.  
    scriptFile pathString }).
```

```
repo workingCopy isDetached iffFalse: [ ^ self ].  
win := (IceTipCheckoutPreviewBrowser onBranch:  
    (IceTipRepositoryModel on: repo) branchModel) open.  
p := win presenter.  
p checkoutStrategyList selectItem:  
    (p checkoutStrategyList items detect: [ :e |  
        e class = IceCheckoutDoNotLoadPackages ]).  
(p instVarNamed: #button) click
```

Integration issues

- Refactorings?
- References to the documentation directly from the method code?
- Integrity checks and validation of up-to-dateness of the documentation?
- Support of future tools

Humble proposal

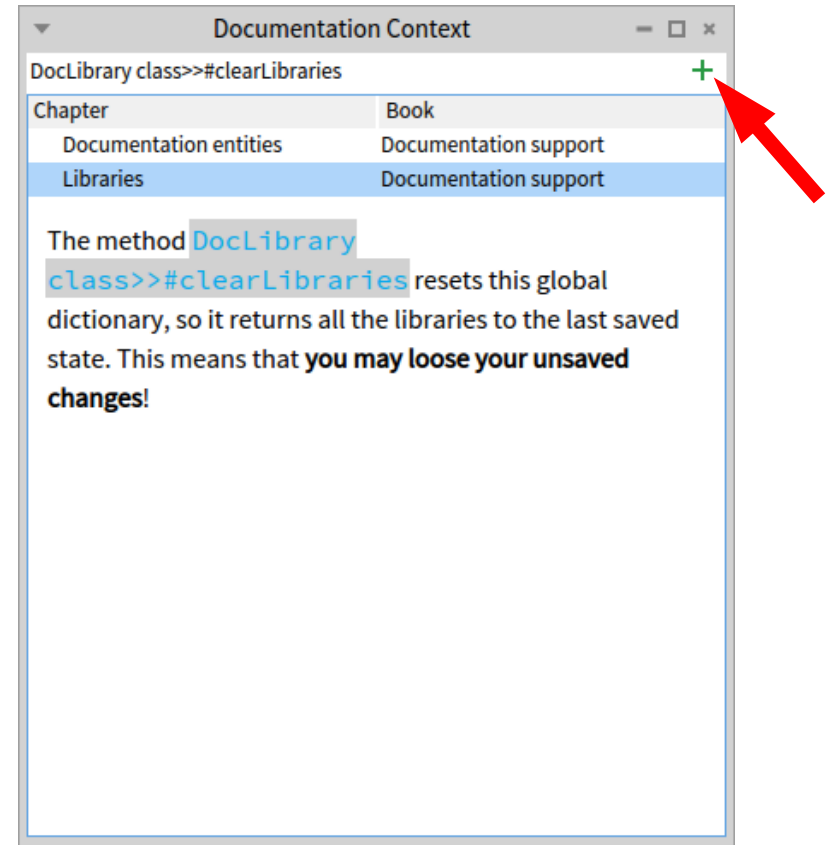
*Let's make documentation entities
the first-class citizens*



Humble proposal

- Part of the Pharo language metamodel
- Built-in support in current tools
- Minimal API allowing future extensions
 - just know about mutual references to other metamodel entities
 - enable future evolving

- No grammar changes required
- Relatively small effort with immediate gain
- Almost everything else is file-based
- Description of packages, instance variables etc.
- Usage for automatic logging
 - manually executed tests
 - visited methods...



Thank you for your attention!

Questions?