

Documentation support in 



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- It Can Become Outdated Quickly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

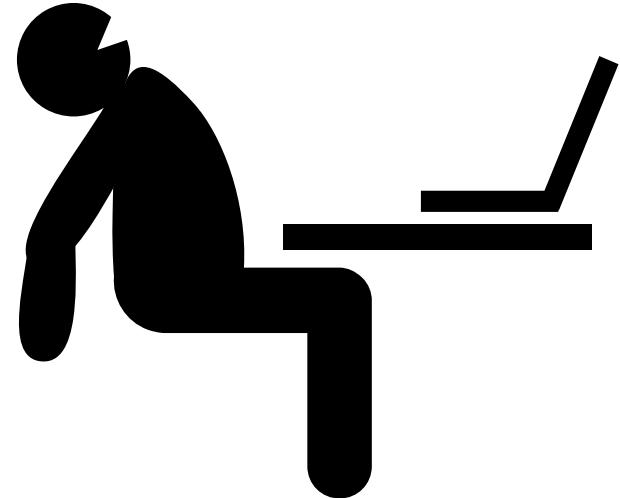


Good quality and up to date documentation is important

Lepiter, Documentation Browser...

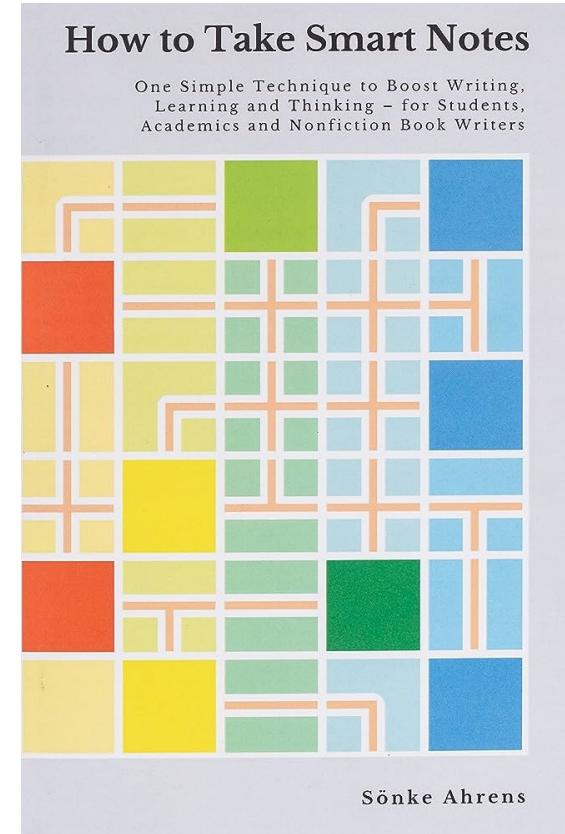
When?

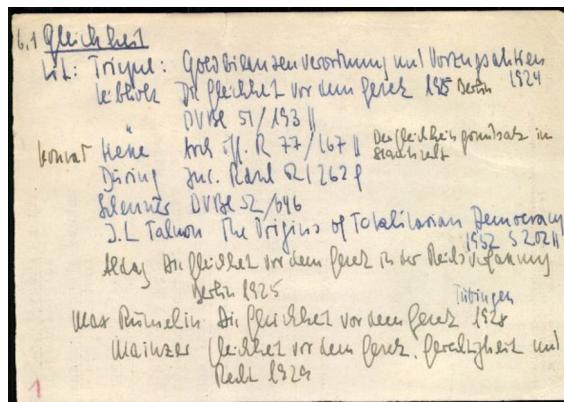
- Sometime after...
- Sometime after
...but this time for real
- During the development
 - quickly outdated, rewrites
- Before
 - DDD - Documentation-driven development
- Literate programming
- Combination



Smart notes

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



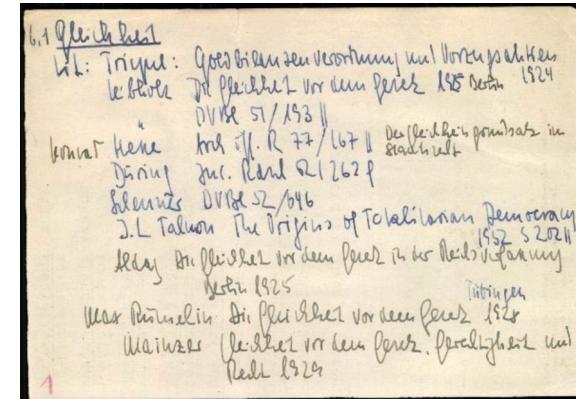


Zettelkasten

- 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) ]
```

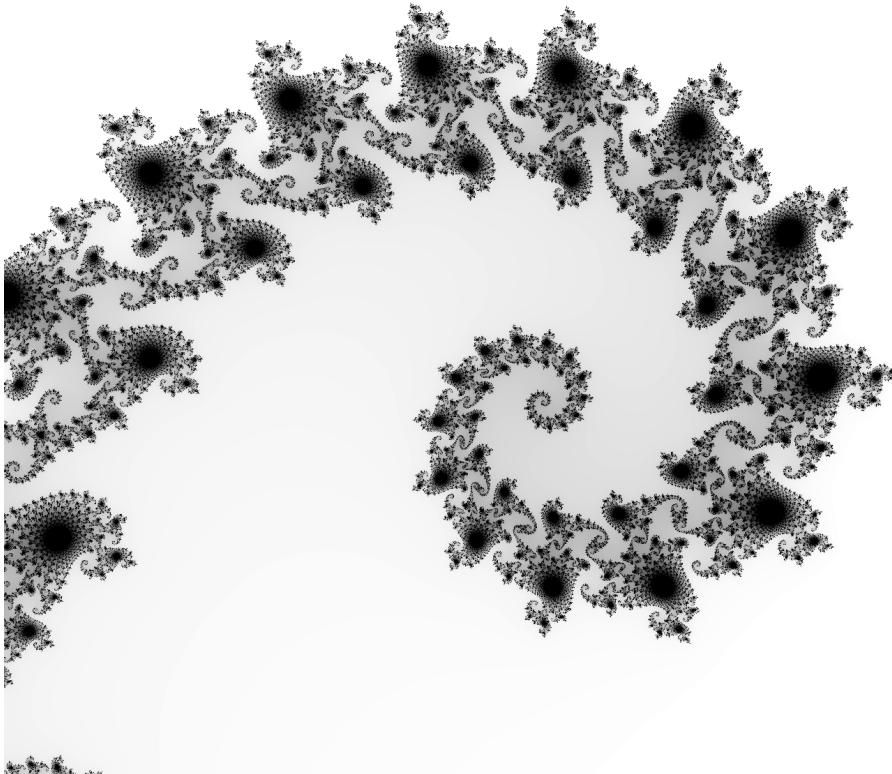
implementors

```
actionsDo: aBlock
self actionMap do: aBlock
```

senders

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

^self attributesCollection do: aBlock
```



Computer program
=
Fractal structure

...independently
on the
representation

```

;Read in the Joystick  $---FUDR
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 $ff01
    ; 543210-- TestVal
    lda #11000000 ;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 #000001100 ;Bit3=1 Y-axis
    sta $ff01
    ; 543210-- TestVal
    lda #11000000 ;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
    rola
    ;Up (True if Bit 7=0)

;Fix results
    eor b,#11010101 ;Flip Up and Left bits
    tfr b,a          ;A contains $---FUDR
    rts

EXPORT(int)
ve_main_with_parameters(MParameters *parameters)
{
    MImage image;
    int error;
    if (!parameters->isDefaultImage && !parameters->defaultImageFound)
    {
        MImage::error = ve_parameters_error_interactive_option_to_the_image;
        if (error)
            return 1;
    }
    if (parameters->isDefaultImage && !parameters->defaultImageFound)
    {
        MImage::error = ve_parameters_error_no_default_image_has_been_specified;
        if (error)
            return 1;
    }
    initCommandLine();
    setProcessArguments(parameters->processArgs, parameters->processArg);
    setEnvironmentVariables(parameters->environmentVariables);
    logInfo("Openning image: %s", parameters->imageFileName);
    //This initialization is required because it makes awful, awful, awful code to calculate
    //the location of the machine code.
    //Luckily, it can be cached.
    //setMachineCode();
    //Retrive the working directory.
    char *workingDirectoryBuffer = (char*)calloc(1, +FILENAME_MAX+1);
    if (!workingDirectoryBuffer)
    {
        logError("Out of memory.\n");
        return 1;
    }
    error = ve_path_get_current_working_dir_into(workingDirectoryBuffer, FILENAME_MAX);
    if (error)
    {
        logError("Failed to obtain the current working directory: %s\n", ve_error_code_to_string(error));
        return 1;
    }
    logDebug("Working Directory %s", workingDirectoryBuffer);

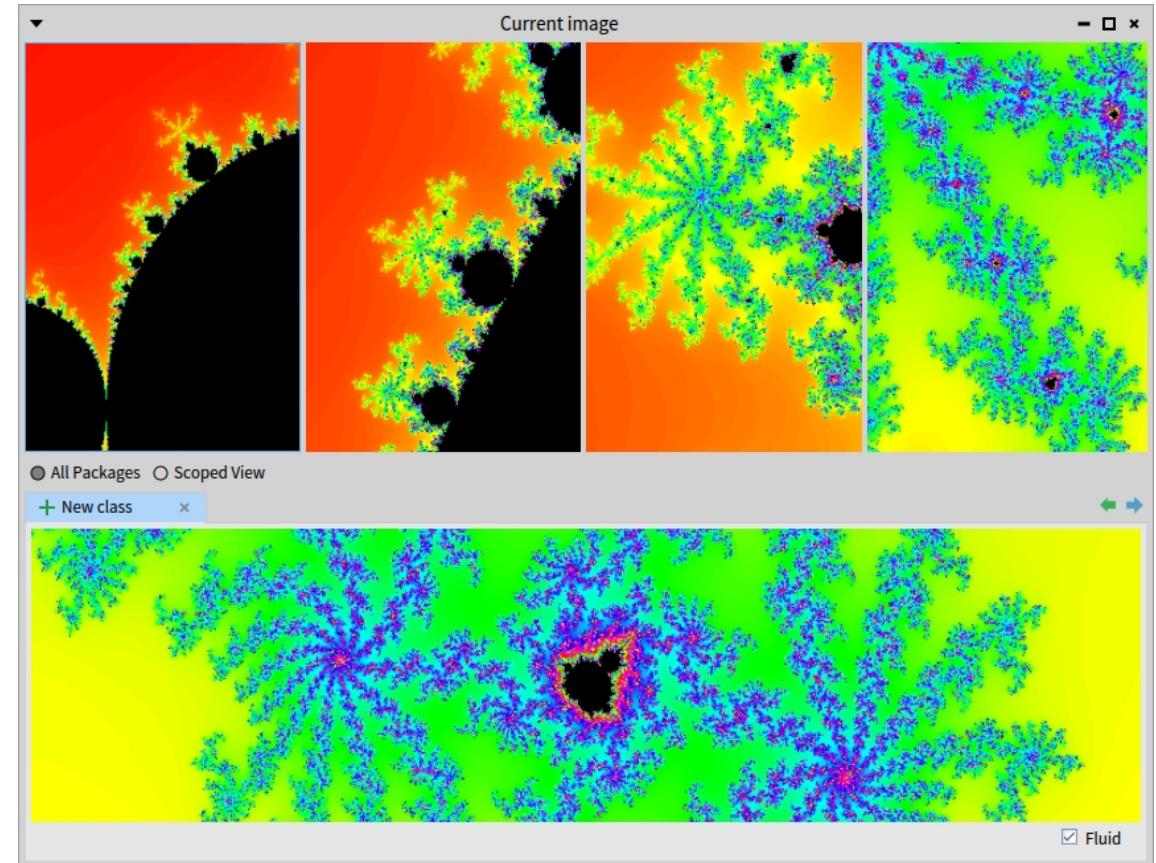
    LOG_SIZEOF(int);
    LOG_SIZEOF(long);
    LOG_SIZEOF(void);
    LOG_SIZEOF(char);
    LOG_SIZEOF(double);
    LOG_SIZEOF(double);

    if (ve_main_in_worker_thread)
    {
        vmbWorkerThread = parameters->isworker;
        return vmbWorkerThread(parameters);
    }
    else
    {
        return runMainWithRead(parameters);
    }
}

EXPORT(int)
ve_main(int argc, const char** argv, const char** env)
{
    MParameters parameters;
    parameters.init(&parameters);
    parameters.environmentVector = env;
    parameters.processArgs = argv;
    parameters.processArg = argv[0];
    if (!parameters.success_on_parsing_the_parameters)
    {
        MImage::error = ve_parameters_parse(argv, argv, &parameters);
        if (error)
            return 1;
        if (error == VM_ERROR_EXIT_WITH_SUCCESS)
            return 0;
    }
    //Do we need to select an image file interactively?
    if (parameters.isInteractive && parameters.isDefaultImage && !parameters.defaultImageFound && !parameters.imageFileName)
    {
        uiFileDialog();
        filiodialog.title = "Select Pharo image to Open";
        filiodialog.filter = "Pharo images (*.image)";
        filiodialog.filterDescription = "Pharo images (*.image)";
        filiodialog.filterExtension = ".image";
        filiodialog.defaultExtension = ".image";
        filiodialog.defaultFile = DEFAULT_IMAGE_NAME;
        error = ve_file_dialog_cw_modal_open(&fileDialog);
        if (fileDialog.success)
        {
            ve_file_dialog_destroy(&fileDialog);
        }
    }
    parameters.imageFileName = strdup(fileDialog.selectedFileName);
    parameters.isDefaultImage = false;
    ve_file_dialog_destroy(&fileDialog);
    int exitCode = ve_main_with_parameters(&parameters);
    ve_main_with_parameters(&parameters);
    return exitCode;
}

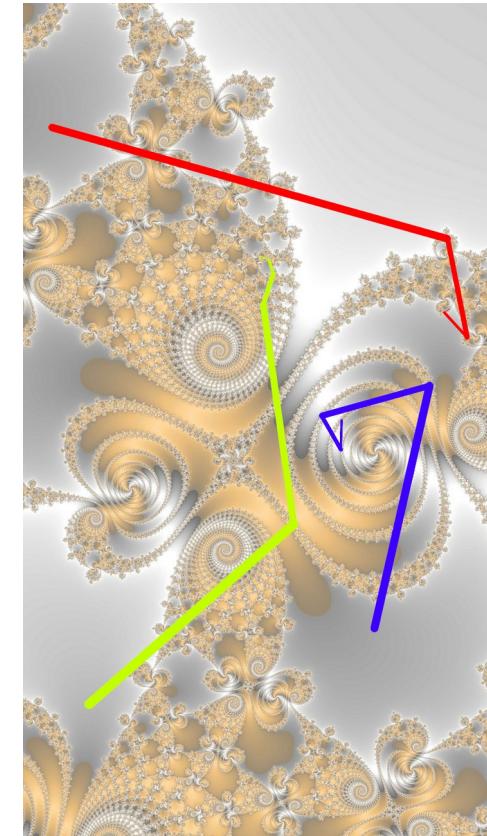
```

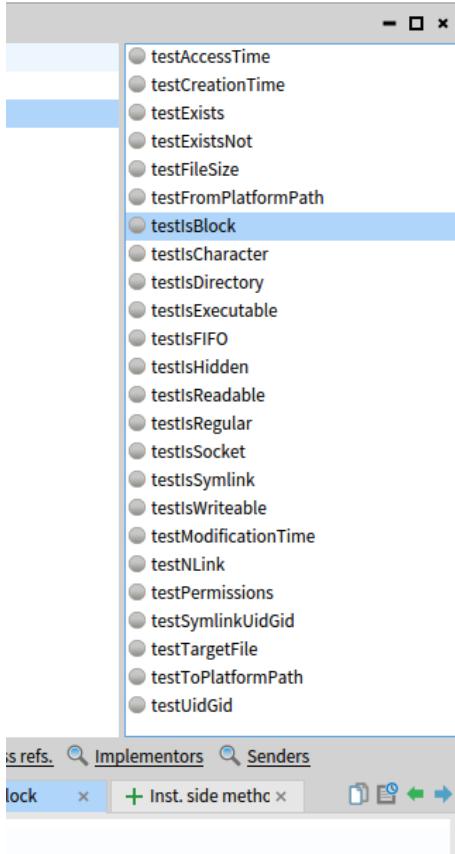
Some tools do it better!



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
 -





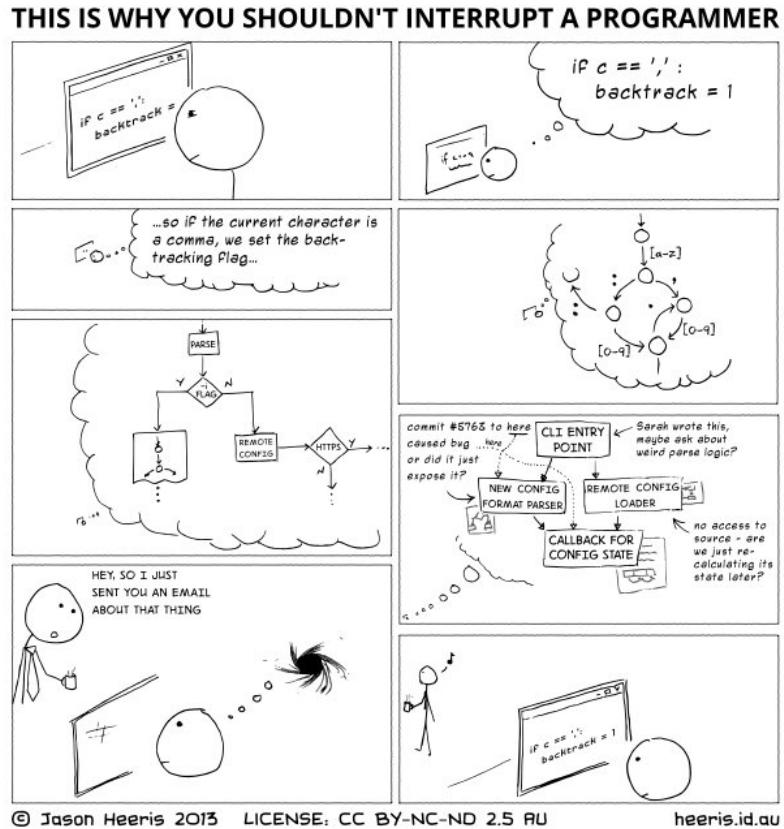
Pharo – code indexing

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



Quick
notes





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...



Documentation support in Pharo

Documentation support

Enter you search Search Previous Next Previous chapter Next chapter

Show header Edit mode Show preview/edit mode Sensitive Case Regex mode

Title: Introduction
 Key: 91bclk2xxrcv7dyhcvj7ay9tk ref:// include://

Next: Documentation entities

Header hide/show Edit/Perview Open

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 entitites

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.

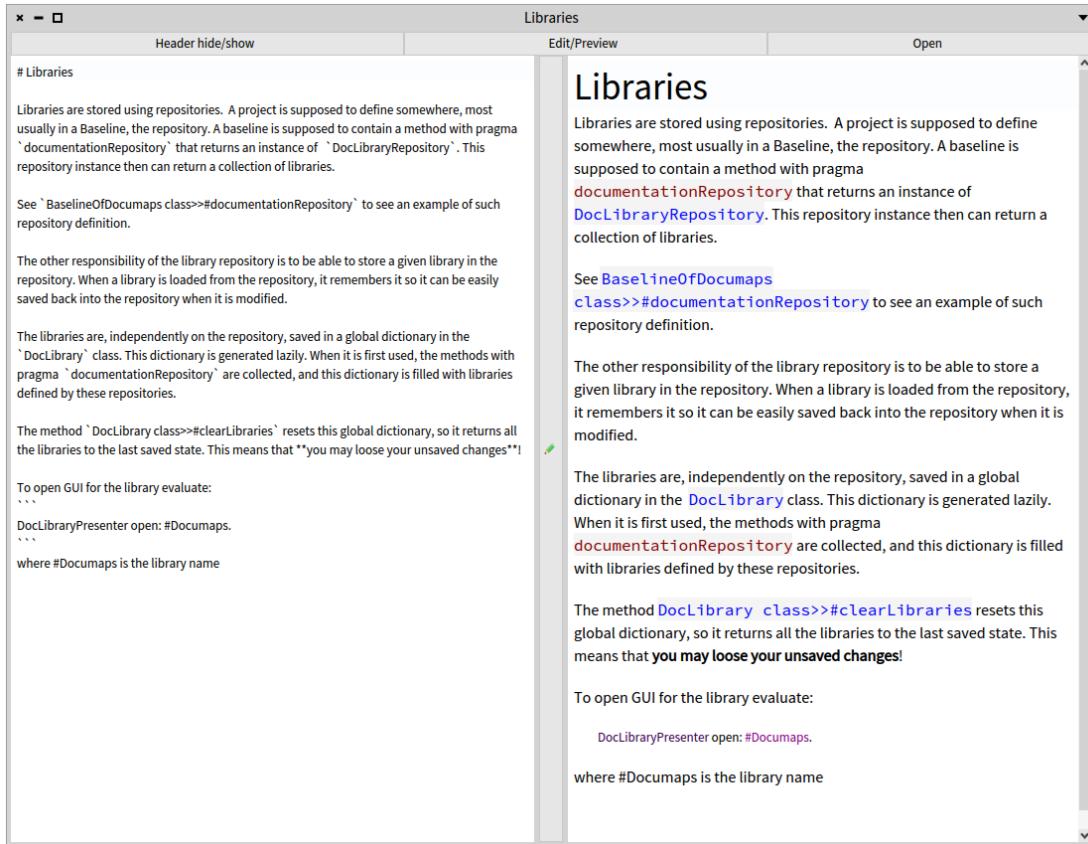
Every chapter has its **unique key**. The title can be changed but not the key.

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.

Documentation support in



Basics

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

Model

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

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

Documentation support in **Pharo**

CreateProcessW function (processsthreadsapi.h)

[View](#) [Edit](#) [Feedback](#)

In this article

System
Windows
Process
Threads
Handle

Create a new process and its primary thread. The new process runs in the security context of the calling process.
If the calling process is impersonating another user, the new process uses the token for the calling process and the impersonation token. To do this the token must be the security context of the user represented by the impersonation token or the `CREATE_UNICODE_NAME` handle.

Syntax

```
NTSTATUS CreateProcessW(
    _In_opt_ _In_opt_ _Inout_opt_ _Handle hThread,
    _In_opt_ _In_opt_ _Inout_opt_ _Handle hProcess,
    _In_opt_ _In_opt_ _Inout_opt_ _SECURITY_ATTRIBUTES *lpThreadAttributes,
    _In_opt_ _In_opt_ _Inout_opt_ _SECURITY_ATTRIBUTES *lpProcessAttributes,
    _In_opt_ _In_ _BOOL bInheritHandles,
    _In_opt_ _In_ _DWORD dwCreationFlags,
    _In_opt_ _In_ _LPVOID lpEnvironment,
    _In_opt_ _In_ _LPVOID lpCurrentDirectory,
    _In_opt_ _In_ _LPCSTR lpImageName,
    _In_opt_ _In_ _LPCVOID lpCommandLine,
    _In_opt_ _In_ _LPVOID lpProcessParameters
);
```

The calling process can inherit handles from the new process. If the `bInheritHandles` parameter is set to `TRUE`, all handles created by the new process will be inheritable by the calling process. This parameter is optional.

The `lpThreadAttributes` parameter can be `NULL` if the application does not intend to create threads in the new process. In this case, the `lpThreadAttributes` parameter must be `NULL` and the `dwCreationFlags` parameter must not contain the `CREATE_SUSPENDED` flag.

If the `lpThreadAttributes` parameter is not `NULL`, it must contain the command string for the new thread. The `lpThreadAttributes` parameter must be a pointer to a `CREATE_THREAD_ATTRIBUTE_LIST` structure.

The `lpProcessAttributes` parameter can be `NULL`. In that case, the function uses the value pointed to by `lpThreadAttributes` as the handle to the new process. If the `lpProcessAttributes` and `lpThreadAttributes` parameters are `NULL`, the new process will be created using pointers to the `lpThreadAttributes` and `lpProcessAttributes` parameters of the `lpThreadAttributes` parameter.

If the `lpThreadAttributes` parameter is not `NULL`, the first whitespace-delimited token of the command line specifies the handle name. The `lpThreadAttributes` parameter must include the `dwThreadAttributes` argument, which contains the expression for the `lpThreadAttributes` parameter. If the name does not contain an expression, it is appended. If there is no handle name, the parameter must include the `dwThreadAttributes` argument, which contains the handle path. The system ignores the handle path if the name does not contain a directory path. The system searches for the handle in the following sequence:

1. The directory where the application loads.
2. The current working directory.
3. The 32-bit Windows system directory for the `GetModuleFileName` function to get the path of this directory.
4. The 32-bit Windows system directory. There is one handle that contains the path of this directory, but it is closed.
5. The Windows directory. The `GetModuleFileName` function begins its search for the handle in the Windows directory.
6. The system directory. The `GetModuleFileName` function begins its search for the handle in the system directory.

The system adds a terminating null character to the command line string to separate the file name from the arguments.

This allows the original string to be reused for external processing.



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.

Note that the function returns before the process has finished initialization. If a required DLL cannot be located or fails to initialize, the process is terminated. To get the termination status of a process, call GetExitCodeProcess.

By default, passing `TOKEN_ALL_ACCESS` as the value of the `dwDesiredAccess` parameter causes all inheritable handles to be inherited by the new process. This can be problematic for applications which create processes having multiple threads simultaneously yet desire each process to inherit different handles. Applications can use the `UpdateThread` or `SetThreadDescriptorByName` functions with the `PROC_THREAD_ATTRIBUTE_INHERIT` parameter to provide a list of handles to be inherited by a particular process.

To avoid this problem, do not pass `NILLL` for `lpThreadAttributes`. If you do pass `NILLL` for `lpThreadAttributes`, use question marks around the handleable paths in `lpThreadAttributes`, as shown in the example below:

```
NTSTATUS CreateProcessW(
    _In_opt_ _In_opt_ _Inout_opt_ _Handle hThread,
    _In_opt_ _In_opt_ _Inout_opt_ _Handle hProcess,
    _In_opt_ _In_opt_ _Inout_opt_ _SECURITY_ATTRIBUTES *lpThreadAttributes,
    _In_opt_ _In_opt_ _Inout_opt_ _SECURITY_ATTRIBUTES *lpProcessAttributes,
    _In_opt_ _In_ _BOOL bInheritHandles,
    _In_opt_ _In_ _DWORD dwCreationFlags,
    _In_opt_ _In_ _LPVOID lpEnvironment,
    _In_opt_ _In_ _LPVOID lpCurrentDirectory,
    _In_opt_ _In_ _LPCSTR lpImageName,
    _In_opt_ _In_ _LPCVOID lpCommandLine,
    _In_opt_ _In_ _LPVOID lpProcessParameters
);
```

Remarks

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

Security Remarks

The first parameter, `lpApplicationName`, can be `NILLL`, in which case the executable name must be in the white-space-delimited string pointed to by `lpCommandLine`. If the executable path name has a space in it, there is a risk that a different executable could be run because of the way the function parses spaces. The following example is dangerous, because the function will attempt to run "Program.exe", if it ends instead of "MyProgram".

```
NTSTATUS CreateProcessW(
    _In_opt_ _In_opt_ _Inout_opt_ _Handle hThread,
    _In_opt_ _In_opt_ _Inout_opt_ _Handle hProcess,
    _In_opt_ _In_opt_ _Inout_opt_ _SECURITY_ATTRIBUTES *lpThreadAttributes,
    _In_opt_ _In_opt_ _Inout_opt_ _SECURITY_ATTRIBUTES *lpProcessAttributes,
    _In_opt_ _In_ _BOOL bInheritHandles,
    _In_opt_ _In_ _DWORD dwCreationFlags,
    _In_opt_ _In_ _LPVOID lpEnvironment,
    _In_opt_ _In_ _LPVOID lpCurrentDirectory,
    _In_opt_ _In_ _LPCSTR lpImageName,
    _In_opt_ _In_ _LPCVOID lpCommandLine,
    _In_opt_ _In_ _LPVOID lpProcessParameters
);
```

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

To avoid this problem, do not pass `NILLL` for `lpThreadAttributes`. If you do pass `NILLL` for `lpThreadAttributes`, use question marks around the handleable paths in `lpThreadAttributes`, as shown in the example below:

```
NTSTATUS CreateProcessW(
    _In_opt_ _In_opt_ _Inout_opt_ _Handle hThread,
    _In_opt_ _In_opt_ _Inout_opt_ _Handle hProcess,
    _In_opt_ _In_opt_ _Inout_opt_ _SECURITY_ATTRIBUTES *lpThreadAttributes,
    _In_opt_ _In_opt_ _Inout_opt_ _SECURITY_ATTRIBUTES *lpProcessAttributes,
    _In_opt_ _In_ _BOOL bInheritHandles,
    _In_opt_ _In_ _DWORD dwCreationFlags,
    _In_opt_ _In_ _LPVOID lpEnvironment,
    _In_opt_ _In_ _LPVOID lpCurrentDirectory,
    _In_opt_ _In_ _LPCSTR lpImageName,
    _In_opt_ _In_ _LPCVOID lpCommandLine,
    _In_opt_ _In_ _LPVOID lpProcessParameters
);
```

Examples

For an example, see [Creating Processes](#).

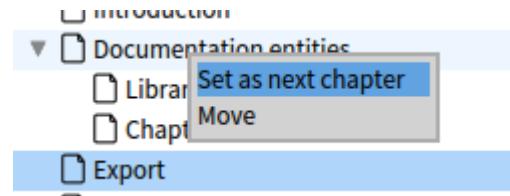
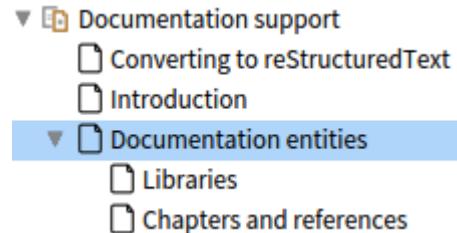
Note

The `lpThreadAttributes` handle defines `ThreadAttributes`, an alias which automatically refers to the `THREAD_ATTRIBUTES` structure of this handle based on the definition of the `UNICODE_STRING` preprocessor constant. Misuse of the preceding handle alias with code that is not encoding valid can lead to anomalies that result in compilation or runtime errors. For more information, see [Commons for Pointer Preprocessors](#).



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:

3z9zm765drcfzotsq9dvc48d4

ref://

include://

Serialization

- External tools friendly
- Multiple projects documentation

- Library folder
 - evxt7d77f32hb18qgh8kt082w (book folder)
 - description.md
 - title: 'Documaps'
 - key: 'evxt7d77f32hb18qgh8kt082w'
 - 3ckid6ubue60xjwpy8dxv2qls.md
 - key: '3ckid6ubue60xjwpy8dxv2qls'
 - parent: 'evxt7d77f32hb18qgh8kt082w'
 - nextChapter:
 - # Replay of events

The subject of this chapter is...

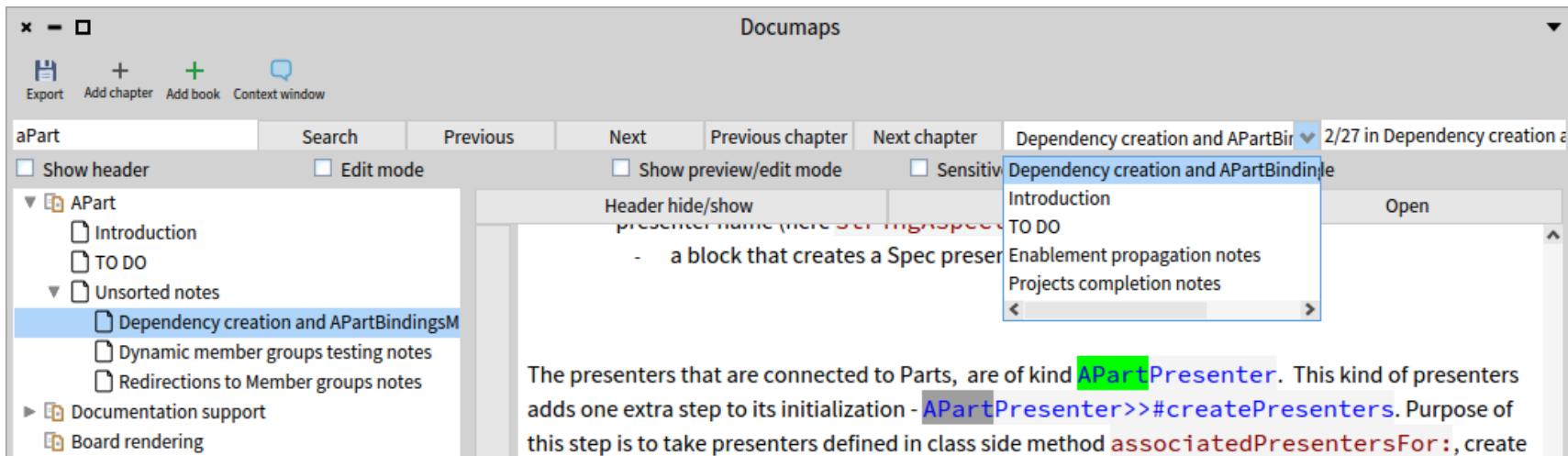
Serialization

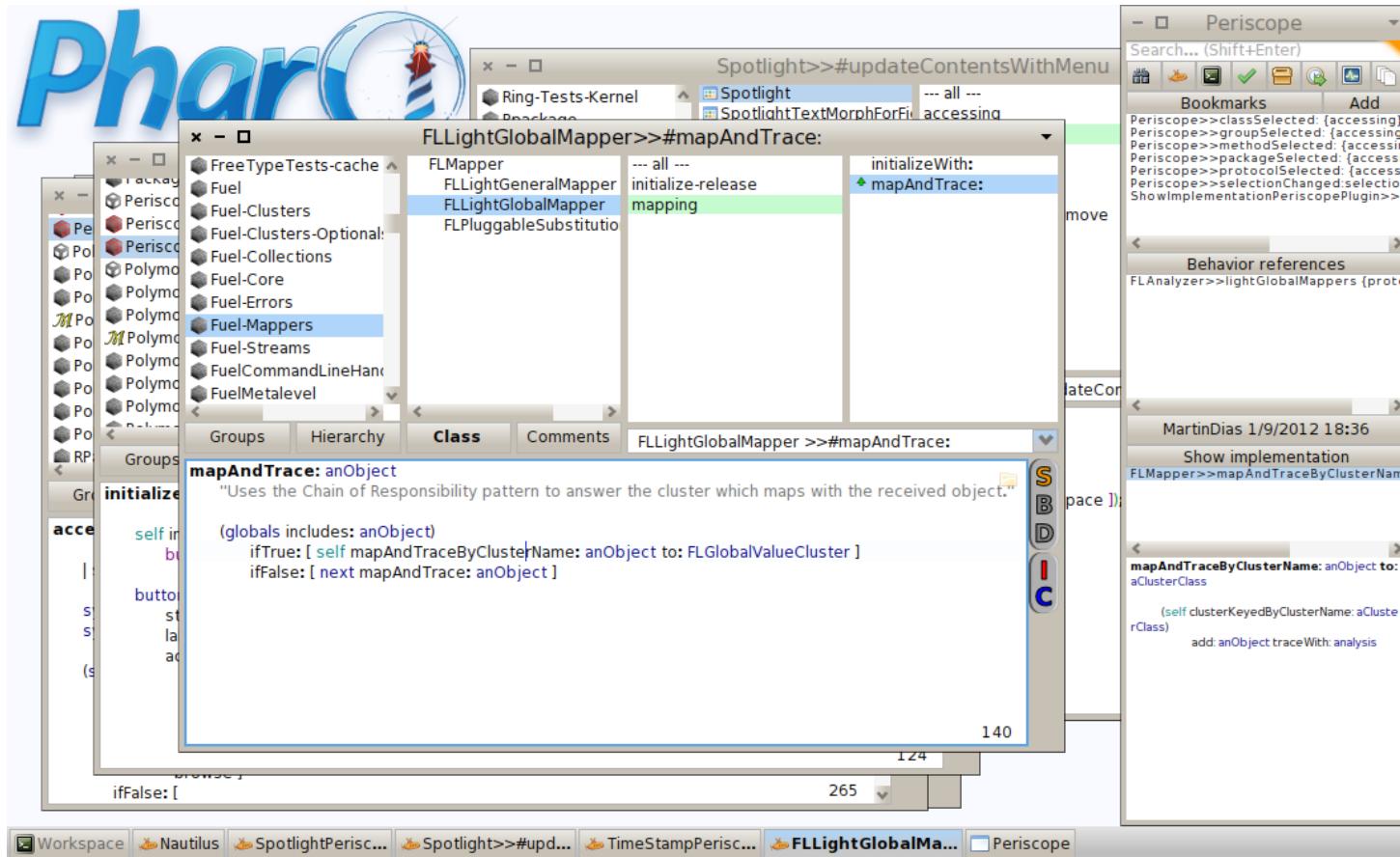
- Git & other tools friendly
- Stored in the Git repository next to the source code
- SPHINX support
 - Python Documentation Generator
 - By Massimo Nocentini



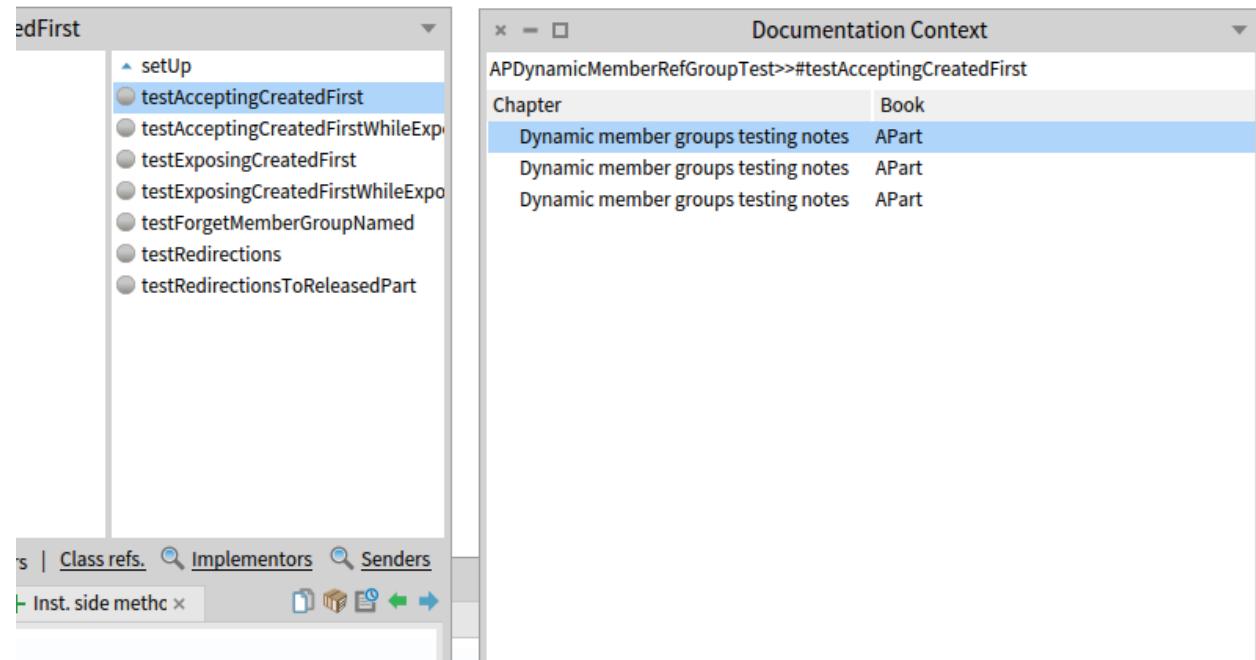
Search

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

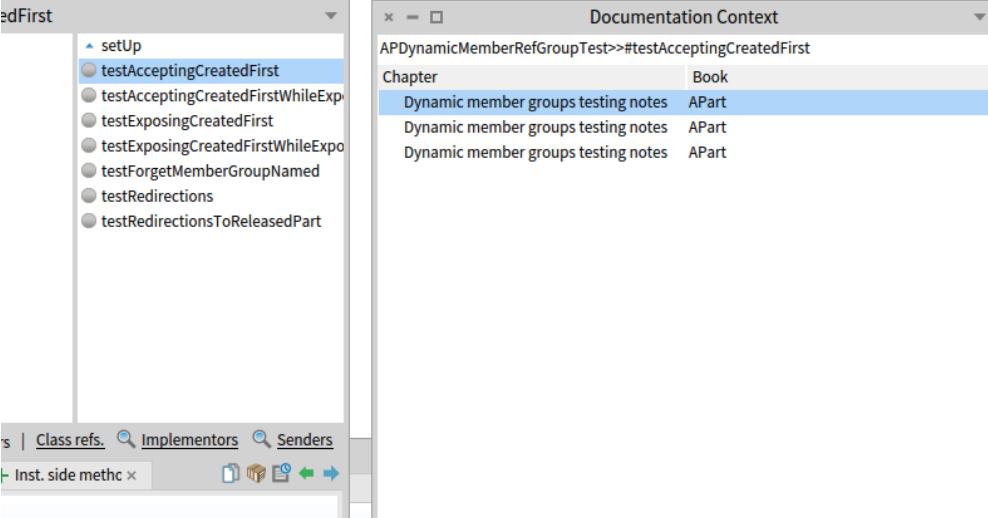


Documentation support in **Pharo**


Documentation Context



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



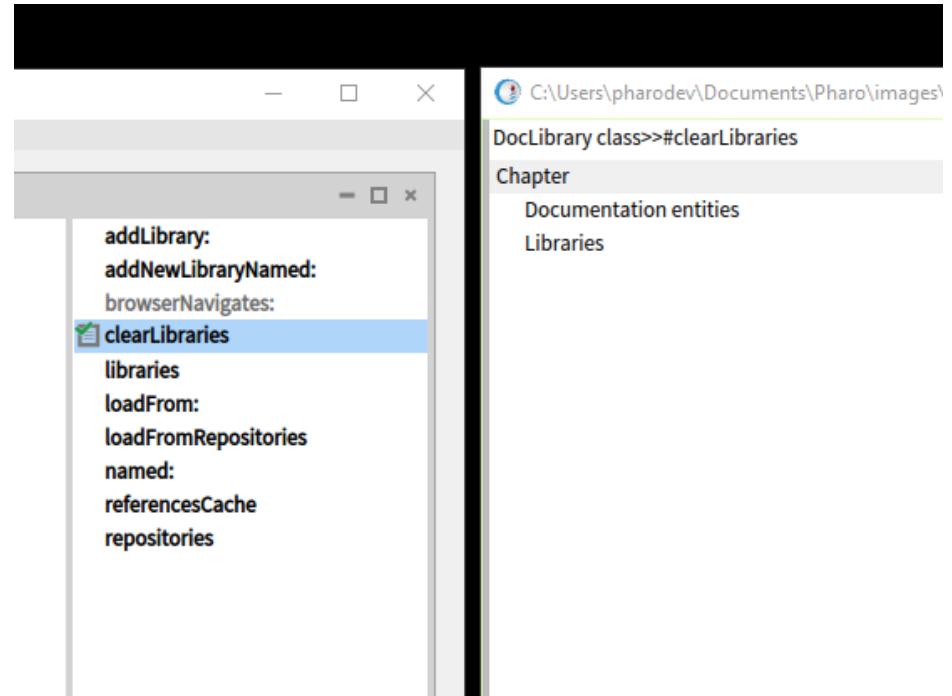
Just the beginning

Future ideas

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

Native OS windows

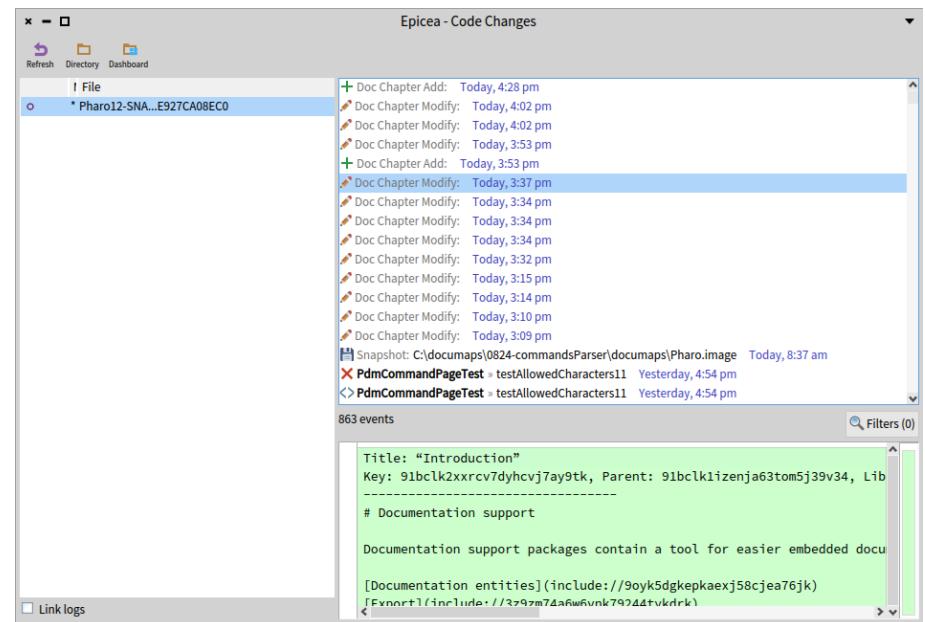
Multiple monitors



(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 ifFalse: [ ^ 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

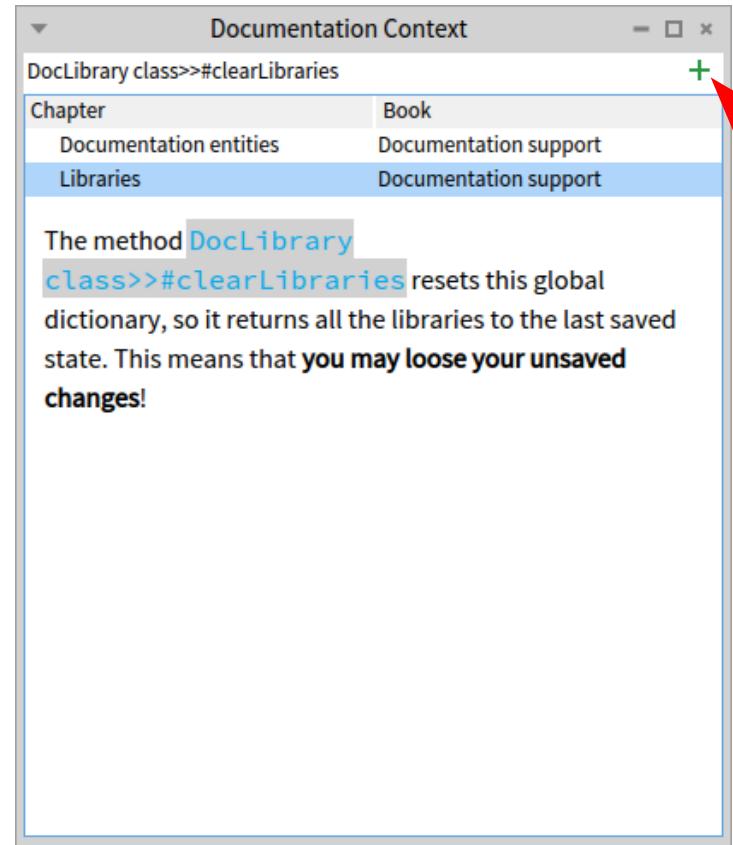
*Let's make documentation entities
the first-class citizens*



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?