The Big Documentation Extravaganza

Stephan Schmidt <schst@php.net>
1&1 Internet AG

international PHP2004 conference

Agenda

- Why documentation is needed
- Types of documentation
- phpDocumentor
- DoxyGen
- DocBook / peardoc
- ReST

The Speaker

- PHP since 1999
- Working for 1&1 Internet AG
- Founder of PHP Application Tools (www.php-tools.net)
- Member of PEAR QA Core Team and active PEAR developer
- Regular contributor to various magazines
- Speaker at conferences around the globe

Why write documentation?

- You are not alone
  - Working in teams
  - Communication is important
- Your brain leaks
  - Do you remember what code you wrote four weeks ago actually does?
  - If yes, what about two years ago?
- You could be hit by a bus!

More reasons for documentation

- Helps you design new parts of your code
  - Write and document function prototypes

"PHP is as easy to read as English."

- Documentation is at a higher level
- There are people that don't "talk" PHP

Types of documentation

- Comments
  - Documentation in the source code
- API-documentation
  - May be generated from code (comments)
  - description of functions, classes, methods,…
- Tutorials
  - Helps you get started
  - What? Why? How?

Documentation must be useful

Comment must add value to your code, otherwise they are wasted bytes.

Bad:

```
// increment $i by one
$i++;
```

Good:

```
// create a unique cache key for the current request
$key = md5(serialize($this->_cacheData));
```

Creating API documentation

Document your code using DocBlocks

- Special comments that start with /**
- Classes, functions, variables, constants
- Contains a summary for these elements
  - Purpose
  - Function arguments / return value
  - Visibility (in PHP4)
  - Links for further reading

A typical DocBlock

```
/**
  * Multiply two integer values
  *
  * @access public
  * @param  int     value 1
  * @param  int     value 2
  * @return int     product of the
  *                 two values
  */
function multiply($a, $b) {}
```

History of DocBlocks in PHP

- Borrowed from JavaDoc
- phpDoc by Ulf Wendel
  - First public appearance on the PHP Kongress 2000 in Cologne
  - Only one release (1.0beta)
  - deprecated, not in CVS anymore
- ported to PEAR in 2002
  - Makes use of the tokenizer extension
  - Also only one release

phpDocumentor

- De-facto standard for DocBlocks in PHP
- Mainly developed by Joshua Eichhorn and Greg Beaver
- Tons of features
  - Creates HTML, PDF, CHM and DocBook
  - Source code highlighting
  - Creates tutorials, to-do lists, …
  - Includes README, CHANGELOG, etc.

phpDocumentor Tags

- Common tags
  - @var, @param, @return
  - @static, @abstract, @access (in PHP4)
  - @see, @uses, @link
  - @category, @package, @subpackage
  - @author, @copyright, @version
- Supports inline tags
- Supports DocBlock templates

phpDocumentor

- Easy to install using the PEAR installer

```
$ pear install phpDocumentor
```

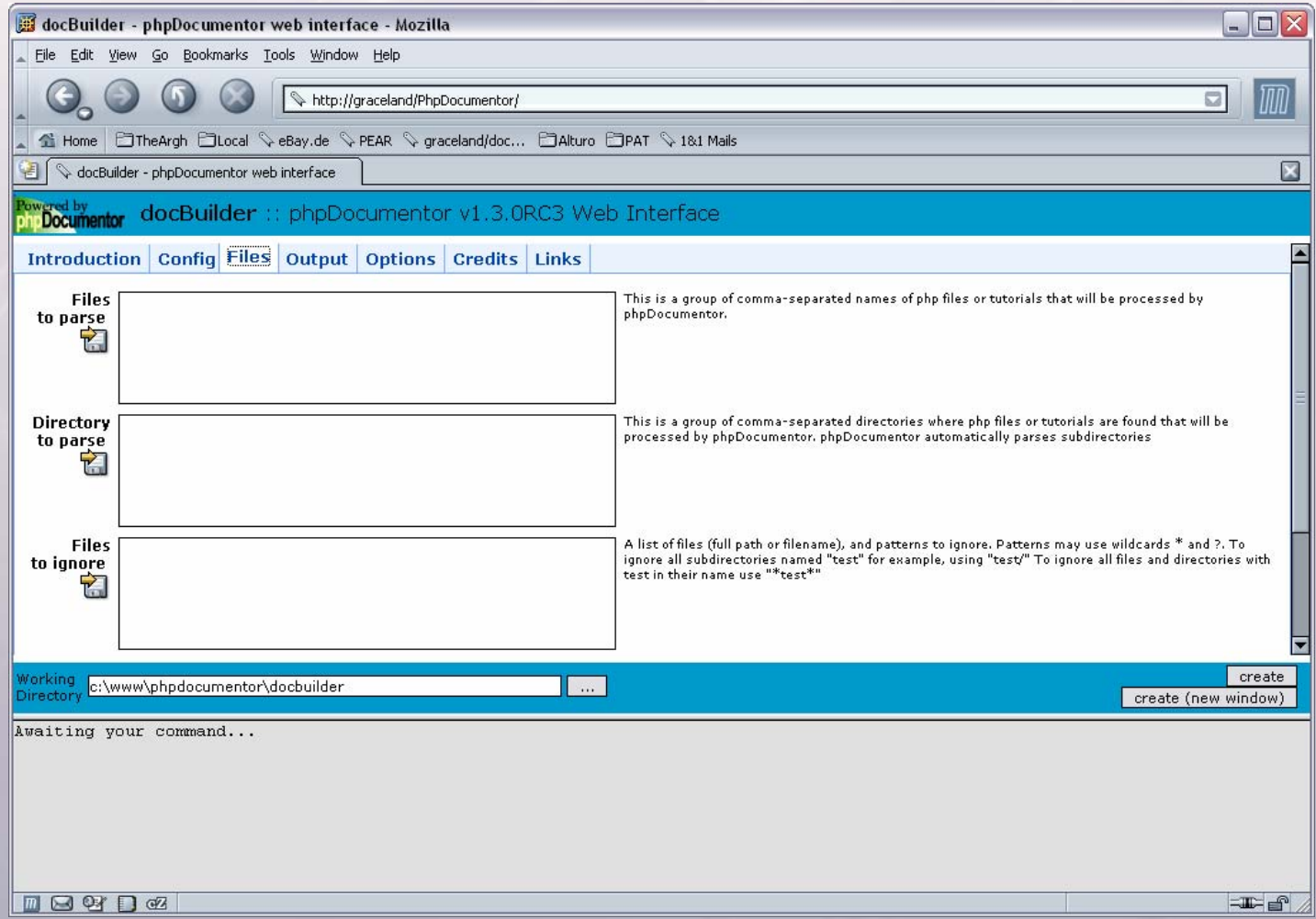
- Command-line interface
  - Use command line options
  - Create configuration files for common tasks
- Easy-to-Use web interface
- Creates more than one output format at once

Using the CLI Interface

- Specify options in command line (total of 23 options), three are needed
  - Source file(s)/dirs
  - Output format
  - Target directory

```
phpdoc
   -d /home/schst/pear/pear/XML_Serializer/
   -o HTML:frames:earthli
   -t ./XML_Serializer_Docs
```

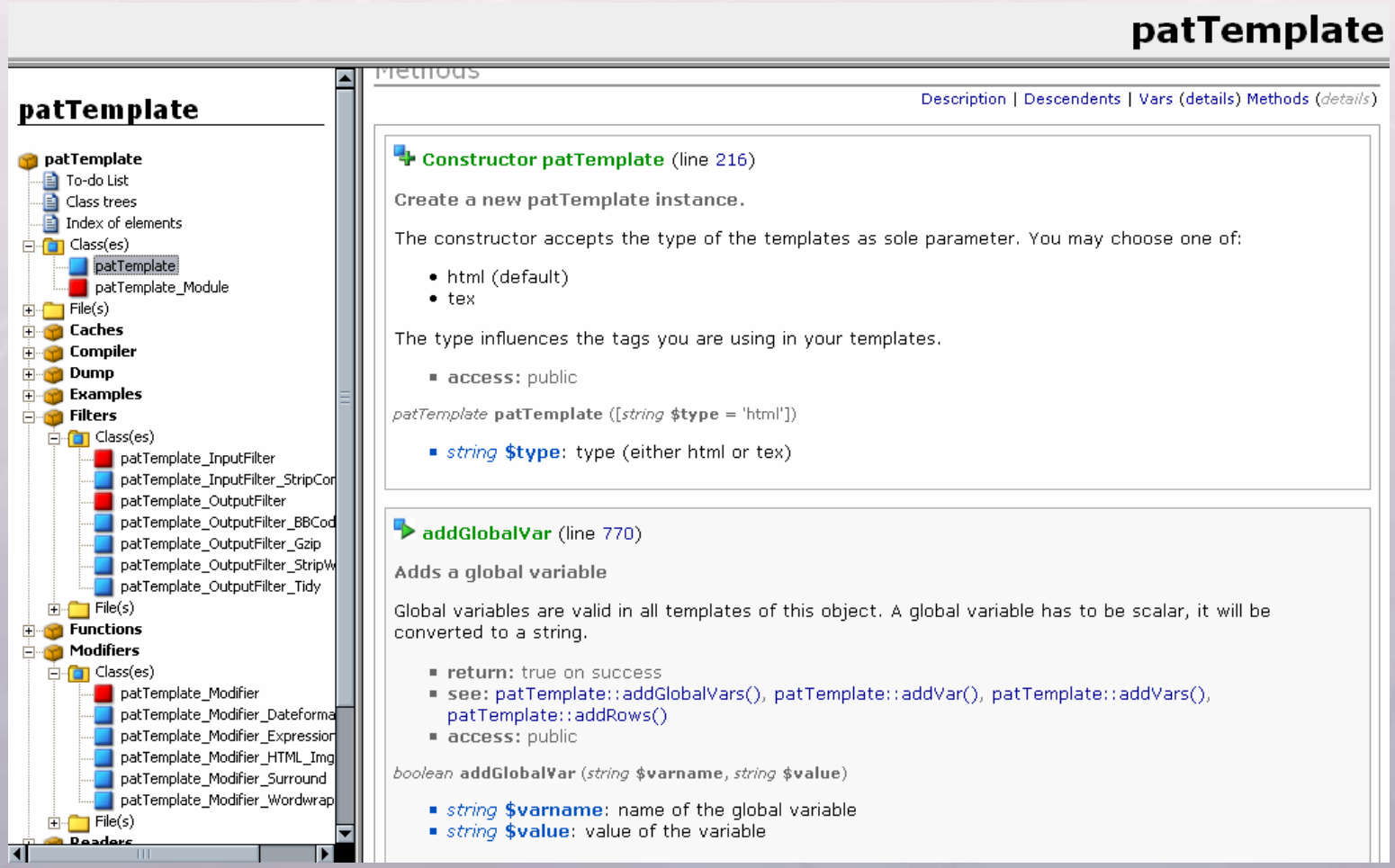
Web interface



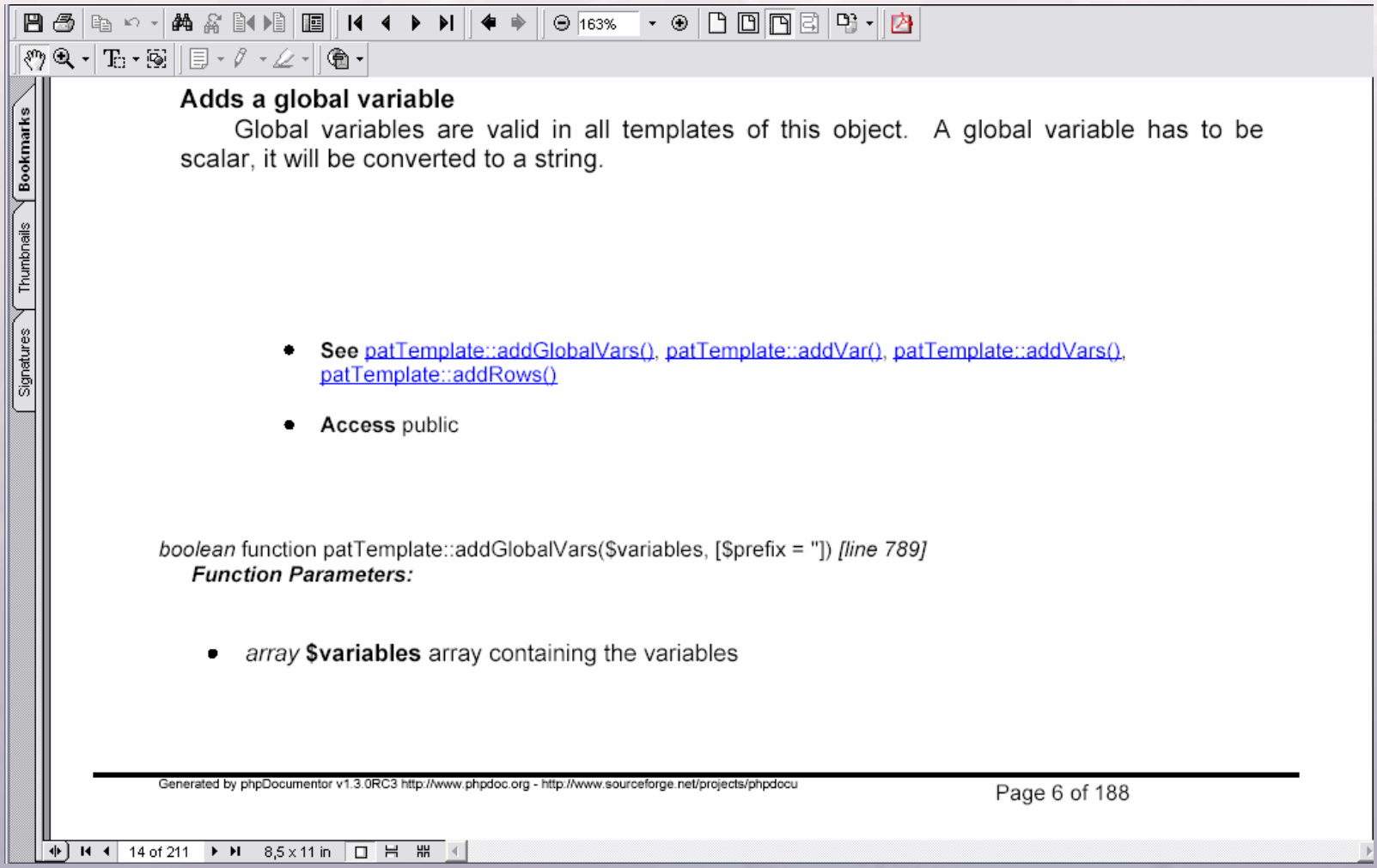
Advanced Features

- Ability to include README files
- Ability to create tutorials that contain more information than just API documentation
  - Based on SGML Syntax
  - Possible for packages, classes and functions
- Include Links to source code of Examples

HTML Output


patTemplate
patTemplate
To-do List
Class trees
Index of elements
Class(es)
patTemplate
patTemplate_Module
File(s)
Caches
Compiler
Dump
Examples
Filters
Class(es)
patTemplate_InputFilter
patTemplate_InputFilter_StripCor
patTemplate_OutputFilter
patTemplate_OutputFilter_BBCod
patTemplate_OutputFilter_Gzip
patTemplate_OutputFilter_StripW
patTemplate_OutputFilter_Tidy
File(s)
Functions
Modifiers
Class(es)
patTemplate_Modifier
patTemplate_Modifier_Dateforma
patTemplate_Modifier_Expressior
patTemplate_Modifier_HTML_Img
patTemplate_Modifier_Surround
patTemplate_Modifier_Wordwrap
File(s)
Readers
Methods
Description | Descendents | Vars (details) Methods (details)
Constructor patTemplate (line 216)
Create a new patTemplate instance.
The constructor accepts the type of the templates as sole parameter. You may choose one of:
html (default)
tex
The type influences the tags you are using in your templates.
access: public
patTemplate patTemplate ([string $type = 'html'])
string $type: type (either html or tex)
addGlobalVar (line 770)
Adds a global variable
Global variables are valid in all templates of this object. A global variable has to be scalar, it will be converted to a string.
return: true on success
see: patTemplate::addGlobalVars(), patTemplate::addVar(), patTemplate::addVars(), patTemplate::addRows()
access: public
boolean addGlobalVar (string $varname, string $value)
string $varname: name of the global variable
string $value: value of the variable

PDF Output



Adds a global variable

Global variables are valid in all templates of this object. A global variable has to be scalar, it will be converted to a string.

- **See** patTemplate::addGlobalVars(), patTemplate::addVar(), patTemplate::addVars(), patTemplate::addRows()
- **Access** public

boolean function patTemplate::addGlobalVars($variables, [$prefix = '']) *[line 789]*

Function Parameters:

- *array* **$variables** array containing the variables

Generated by phpDocumentor v1.3.0RC3 http://www.phpdoc.org - http://www.sourceforge.net/projects/phpdocu

Page 6 of 188

DoxyGen

- Documentation System for C++, C, Java, Objective-C, IDL
- But also works for PHP
- Extracts DocBlocks but also creates documentation from undocumented source files
- Creates HTML, LaTeX, RTF, PDF, CHM, XML and Unix man pages

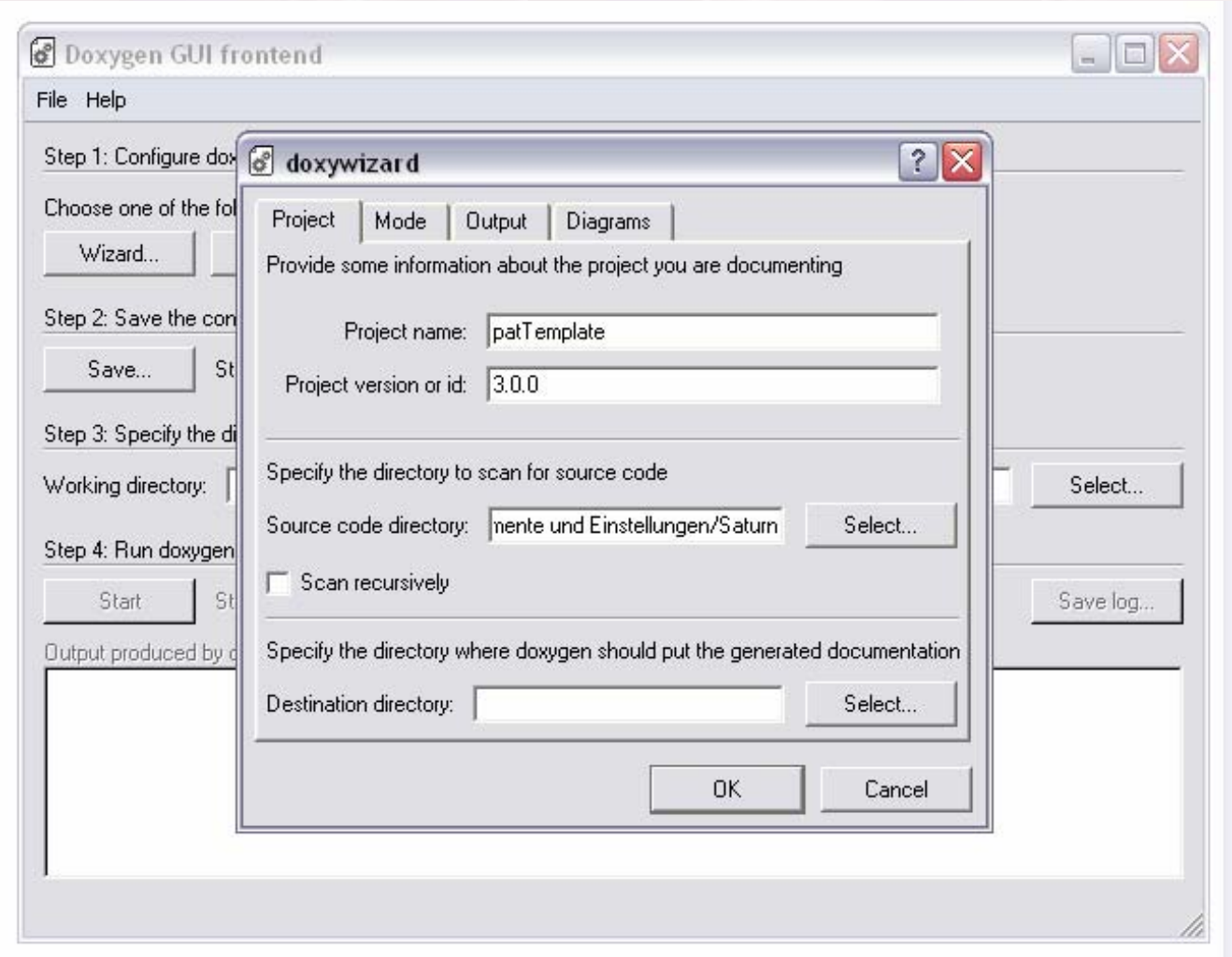
DoxyGen Features

- Supports documentation of files, namespaces, classes, variables, functions and defines (plus a lot entities not available in PHP).
- Creates class diagrams from your code as EPS or PNG with image maps
- Includes references to source code and examples

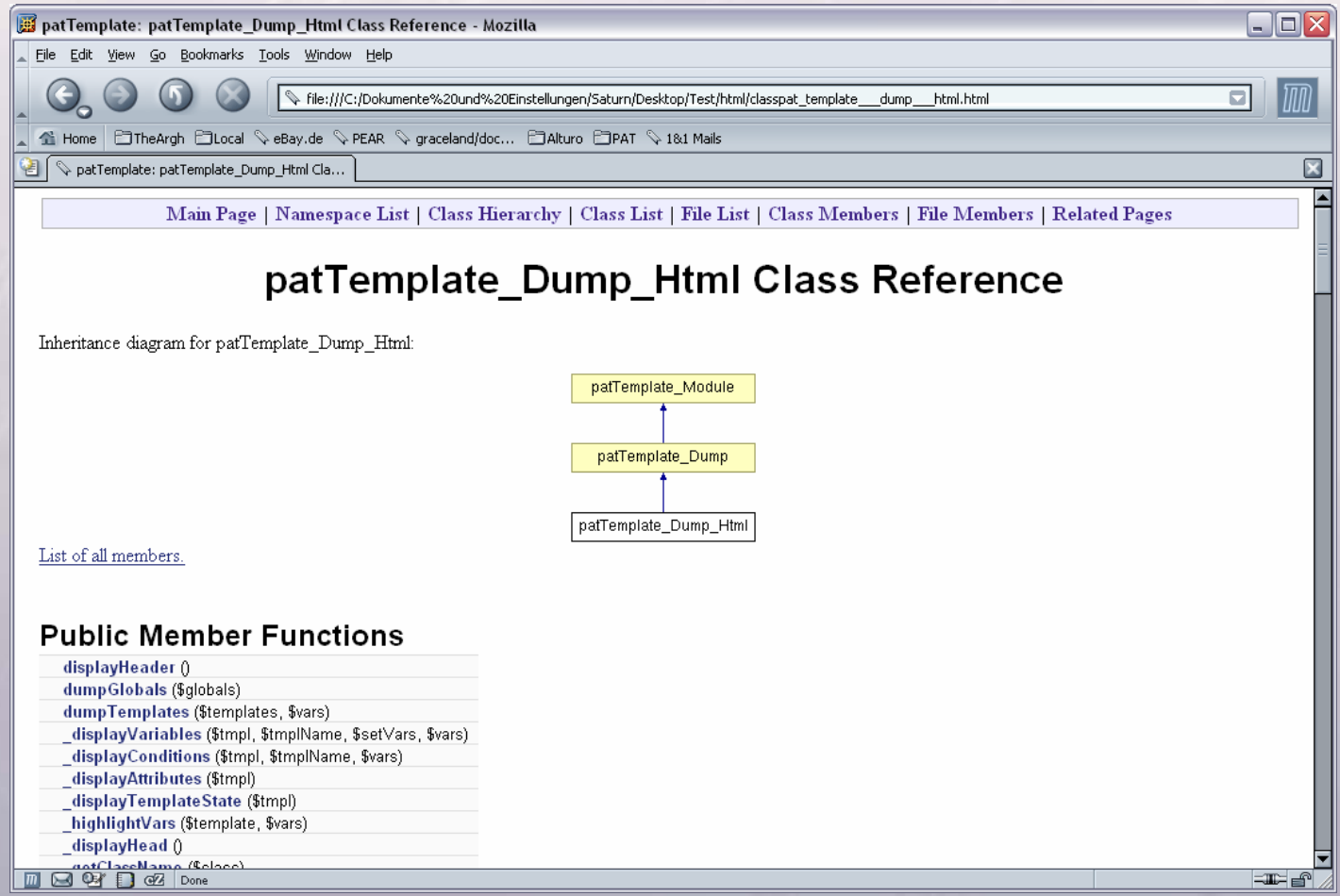
DoxyGen Features (cont.)

- some kind of Wiki markup inside DocBlocks
- HTML tags in documentation are allowed
- allows you to mark visibility for methods and properties
- more than 150 configuration options
- reads from configuration files
- Provides a wizard

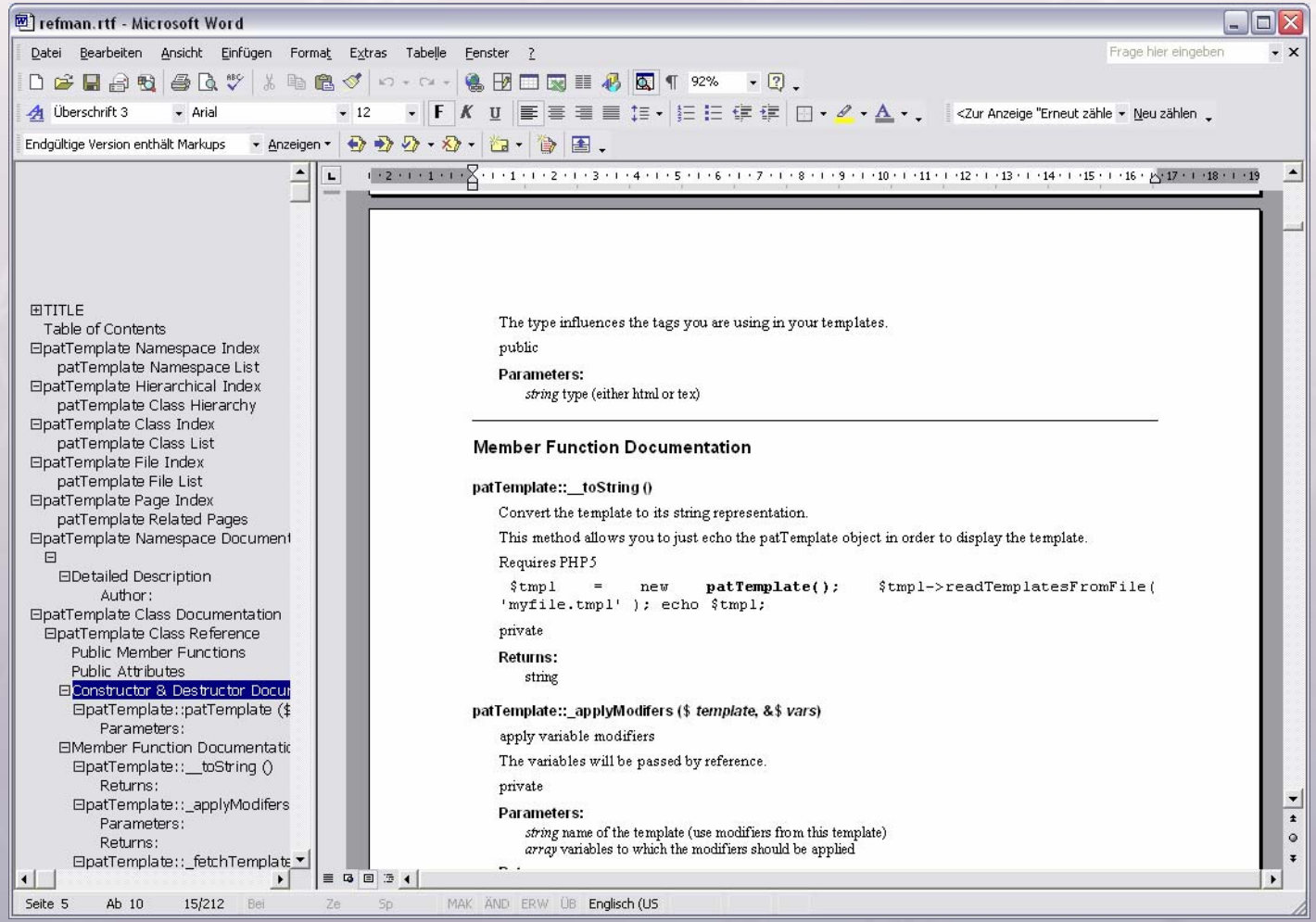
DoxyWizard


Doxygen GUI frontend
File Help
Step 1: Configure dox
Choose one of the fol
Wizard...
Step 2: Save the con
Save...
Step 3: Specify the di
Working directory:
Step 4: Run doxygen
Start
Output produced by d
Select...
Save log...
doxywizard
Project
Mode
Output
Diagrams
Provide some information about the project you are documenting
Project name:
patTemplate
Project version or id:
3.0.0
Specify the directory to scan for source code
Source code directory:
nente und Einstellungen/Saturn
Select...
Scan recursively
Specify the directory where doxygen should put the generated documentation
Destination directory:
Select...
OK
Cancel

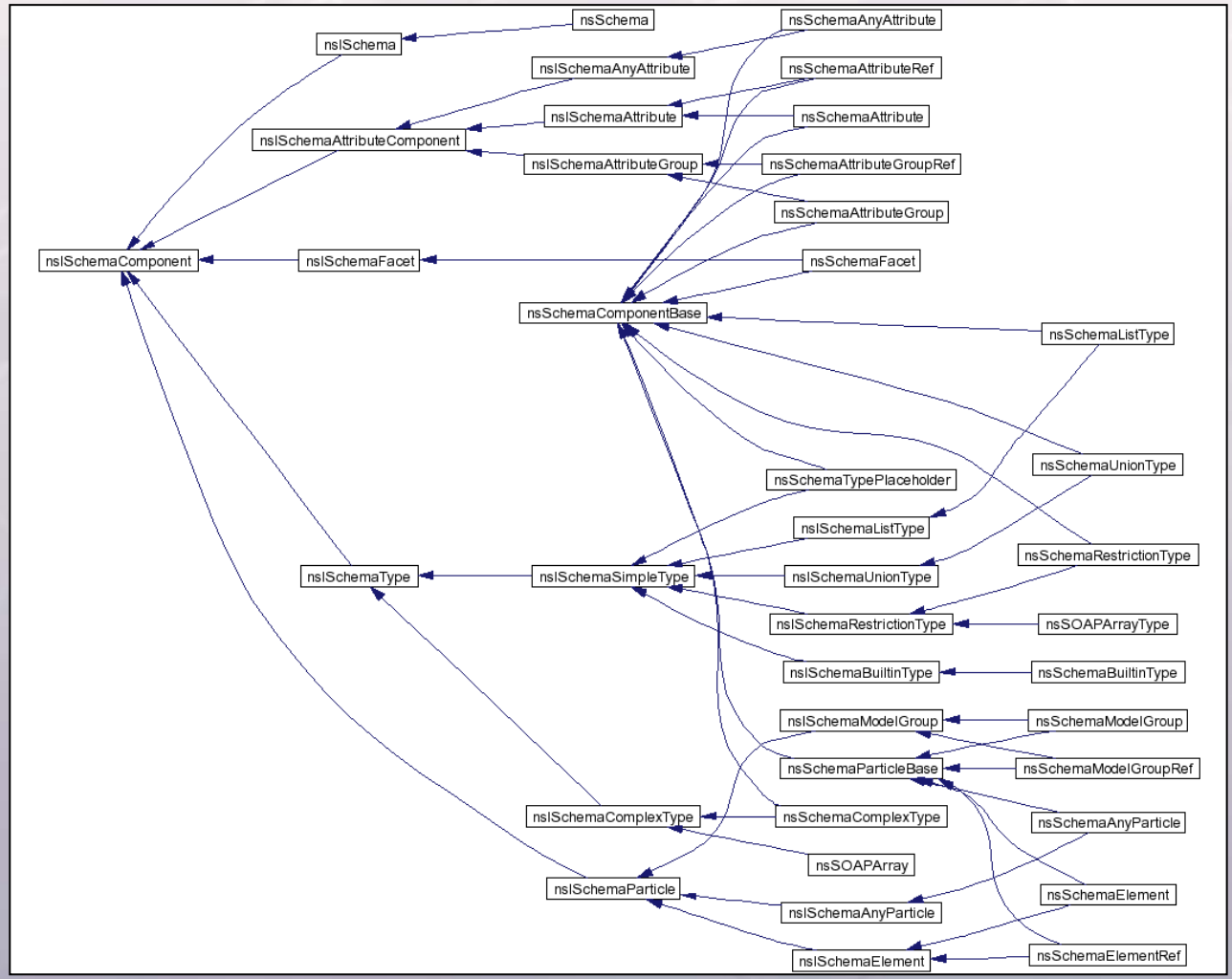
DoxyGen HTML Output



DoxyGen RTF Output



Graphical class hierarchies



Who uses DoxyGen

- Mozilla
- Xerxes
- KDevelop
- phpOpentracker
  http://www.phpopentracker.de/apidoc/
- SPL
  http://www.php.net/~helly/php/ext/spl/

phpDocumentor vs. DoxyGen

phpDocumentor

- easy to use
- made for PHP
- written in PHP
- several layouts
- created peardoc files

DoxyGen

- Windows GUI
- not limited to PHP
- RTF output
- creates class diagrams

DocBook

- Set of SGML tags for describing articles, books and other prose documents
- Designed for writing documentation
- Created 1991 by HaL Computer Systems and O'Reilly
- Now managed by OASIS
- Making heavy use of external entities
- extremely complex

DocBook elements

- Sets
- Books
- Divisions
- Components (chapters)
- Sections
- Meta information
- Block level elements (lists, paragraphs,…)
- Inline elements (Emphasis, Quote, etc.)

DocBook sample document

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V3.1//EN" [
 <!ENTITY sect2 SYSTEM "section2.sgm">
]>
<article>
 <artheader>
  <title>My Article</title>
  <author>
   <honorific>Dr</honorific><firstname>Ed</firstname>
   <surname>Wood</surname>
  </author>
 </artheader>
 <para> ... </para>
 <sect1>
  <title>On the Possibility of Going Home</title>
  <para> ... </para>
 </sect1>
 &sect2;
 <bibliography> ... </bibliography>
</article>
```

Publishing DocBook

Make use of stylesheets

- FOSIs
- DSSSL
- CSS
- XSL

You'll probably use DSSSL (as php.net uses it)

peardoc

- PEAR uses DocBook (like the PHP documentation does)
- configure file will create external entities from file system: **`&package.xml.xml-util.intro;`** refers to /package/xml/xml-util/intro.xml
- all <refsection/> tags have id attributes: **`&package.xml.xml-util.intro.example;`** creates a URL with a fragment #example

Contributing documentation

- Requires openjade and DSSSL stylesheets
- The rest is done by configure/make

```
$ cvs -d :pserver:cvsread:phpfi@cvs.php.net:/repository
   login
$ cvs -d :pserver:cvsread:phpfi@cvs.php.net:/repository
   checkout peardoc
$ cd peardoc
$ autoconf
$ ./configure [--with-lang=en]
$ make
```

Directory structure

```
en/
  package/                          (chapter)
    xml.xml                         (category overview)
    xml/                            (category files)
      xml-util.xml                  (package overview)
      xml-wddx.xml                  (package overview)
      xml-util/                     (package files)
        intro.xml                   (intro to package)
        example.xml                 (example for package)
        xml-util/                   (class)
          createtag.xml             (method of class)
          isvalidname.xml           (method of class)
      xml-wddx/                     (package files)
```

Contributing documentation

- Create directory for the new package (replace _ with -)
- Create overview file in category directory:

```
<sect1 id="package.xml.xml-util">
 <title>XML_Util</title>
 <para>
  Collection of often needed methods that help you
   creating XML documents.
 </para>
  &package.xml.xml-util.intro;
  &package.xml.xml-util.example;
</sect1>
```

Contributing documentation

- Add directories for classes in your package
- Write DocBook files (or let phpDocumentor do this for you)
- Add the new package to the overview page
- Rebuild the documentation
- commit

reStructuredText

- Easy-to-read plain-text markup, like Wiki
- Has been built to create Python documentation
- Great to write short documentation
- DocUtils convert it to
  - HTML
  - LaTeX (PDF)
  - XML

Using DocUtils

- Python with XML-support is needed (xml.dom.minidom)
- Get DocUtils from http://docutils.sourceforge.net
- Install using
  python setup.py install
- Write your documents
- Transform it to HTML/LaTeX/XML

An example document

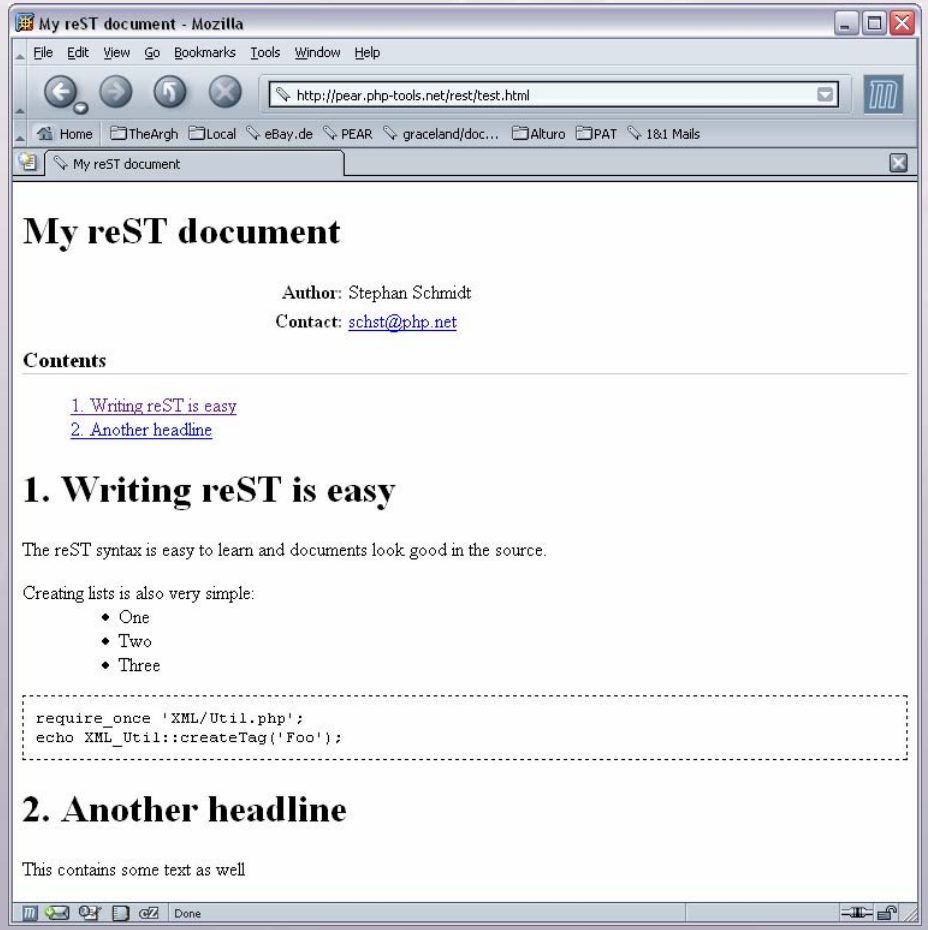
```
=================
My reST document
=================

:Author:        Stephan Schmidt
:Contact:       schst@php.net

.. contents:: Contents
.. section-numbering::

Writing reST is easy
====================
The reST syntax is easy to learn and documents look good in the source.

Creating lists is also very simple:
        - One
        - Two
        - Three

::

    require_once 'XML/Util.php';
    echo XML_Util::createTag('Foo');
```

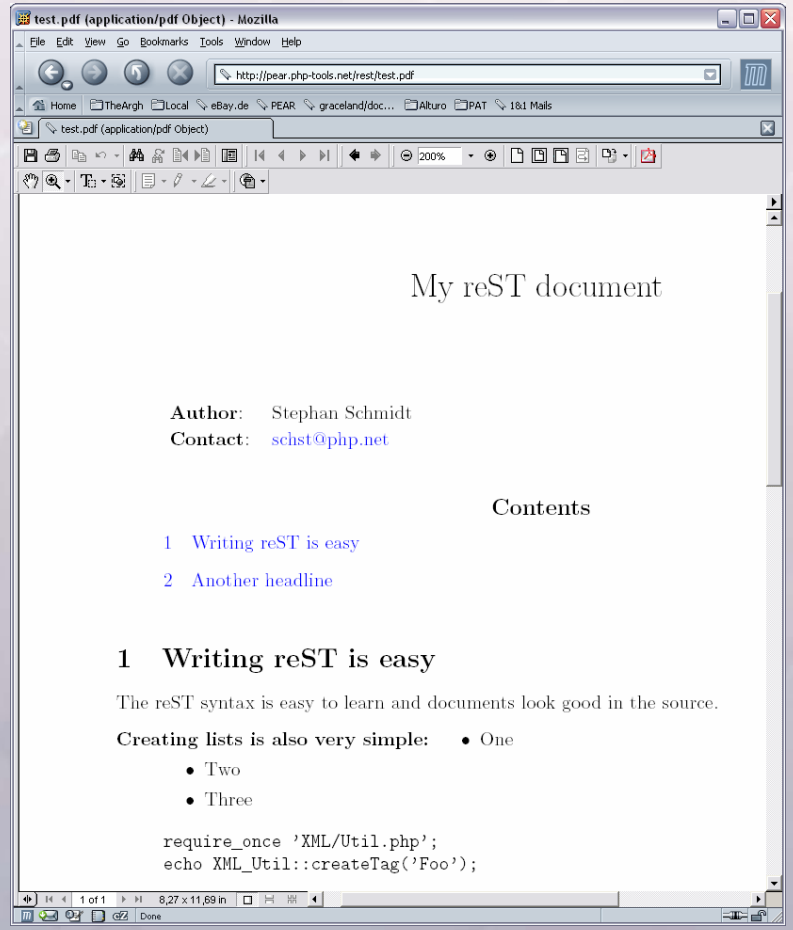
Publishing reST

- DocUtils package provides Python scripts:
  - rst2html.py
  - rst2latex.py
  - rst2xml.py
  - rst2pseudoxml.py
- Usage:

```
$ rst2html sourcefile targetfile
```

Publishing reST





Further reading and downloads

- phpDocumentor
  http://www.phpdoc.org
- DoxyGen
  http://www.doxygen.org
- DocBook
  http://www.docbook
- reST
  http://docutils.sourceforge.net/

The End

Thanks for your attention.

schst@php.net

http://www.php-tools.net