Inside the PHP Documentation

Dan Scott
dbs@php.net

Agenda

- Contributing to the PHP documentation:
  - PHP Manual
  - Editing user notes
  - Documentation bugs
  - Translations
  - Documentation HOWTO
- Becoming a contributor

PHP Manual

- Written in DocBook XML, an OASIS industry standard DTD for technical documentation
  - Also used for Linux Documentation Project, MySQL, PostgreSQL, Gentoo, Red Hat, GNOME, KDE, and countless other projects
  - Translated into dozens of languages
  - Updated and rebuilt regularly

DocBook basics: elements

- DTD and reference available online at http://docbook.org
- Verbose tagging hierarchy:

```
<para>
 <itemizedlist>
  <listitem><para>First bullet.</para></listitem>
 </itemizedlist>
</para>
```

DocBook basics: entities

- Entities are the variables of XML:
  - File entities:

```
<!ENTITY index SYSTEM 'index.xml'>
&index;
```

  - Text entities:

```
<!ENTITY notes '<title>Notes</title>'>
&reftitle.notes;
```

  - URLs:

```
<!ENTITY url.php 'http://www.php.net'>
&url.php;
```

Editing DocBook

- Standard suggestion: Emacs+PSGML
- My preference: (g|k)vim
- But any XML-aware editor with intelligent indenting and line wrapping will do
  - http://wiki.docbook.org/topic/DocBookAuthoringTools

Getting the phpdoc source

- Stored in the phpdoc CVS repository

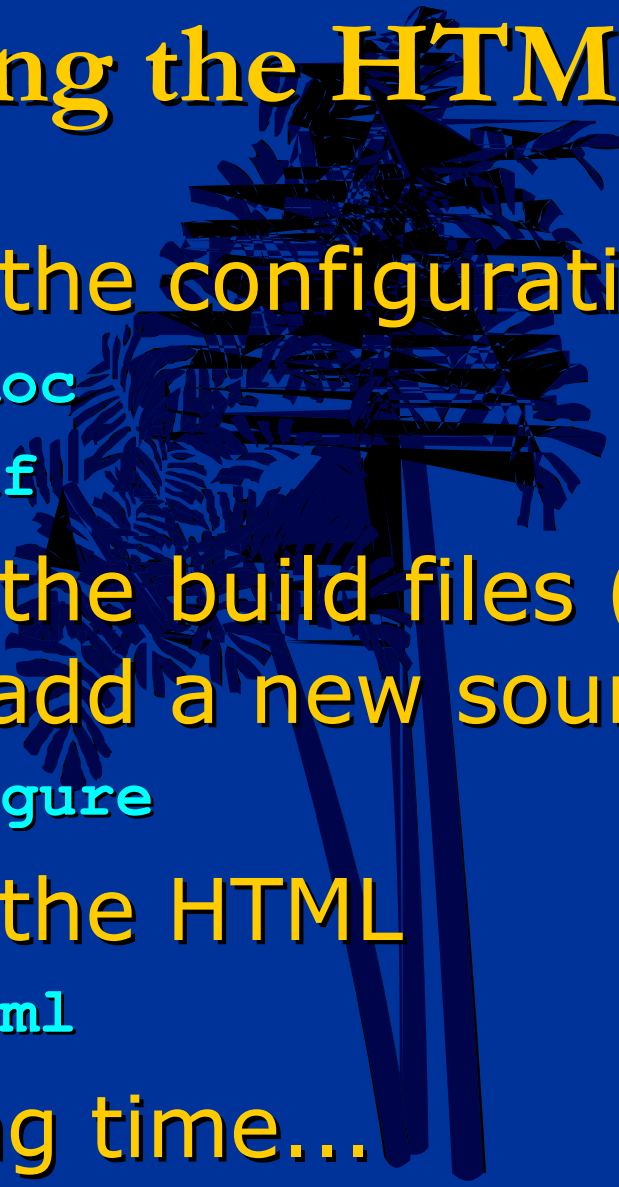
```
bash$ export
  REP=:pserver:cvsread@cvs.php.net/repository
bash$ cvs -d $REP login
password: phpfi
bash$ cvs -d $REP co phpdoc
```

- From now on you can update using:

```
bash$ cvs up
```

- Windows people -- use Cygwin tools

Building the HTML output



- Generate the configuration file

```
bash$ cd phpdoc
bash$ autoconf
```

- Generate the build files (repeat every time you add a new source file)

```
bash$ ./configure
```

- Generate the HTML

```
bash$ make html
```

- Wait a long time...

Documenting a new extension

- HOWTO contains DocBook templates for overview, configuration, constants, and functions
- You *could* copy and paste each template, rename the files, and rename the internal IDs in each file
- Or you could save yourself a lot of work by running the xml_proto.php script

Running xml_proto.php

- Accepts two arguments
  - Name of extension
  - C source file pattern for function prototypes and constants
- For example, to generate the documentation framework for PDO:

```
bash$ php scripts/xml_proto.php pdo \
~/php-src/ext/pdo/*.c
```

- Output quality is directly related to adherence to PECL coding standards

Prototypes in code

- xml_proto.php converts prototype comments to DocBook XML templates

```
/* {{{ proto resource db2_procedures
   (resource connection[, string qualifier])
Returns a result set listing the stored
   procedures registered in a database */
```

xml_proto.php result

```
<?xml version="1.0" encoding="iso-8859-1"?>
<refentry id="function.db2-procedures">
 <refnamediv>
  <refname>db2_procedures</refname>
  <refpurpose>
   Returns a result set listing the stored procedures
   registered in a database
  </refpurpose>
 </refnamediv>
 <refsect1 role="description">
  &reftitle.description;
  <methodsynopsis>
   <type>resource</type><methodname>db2_procedures</methodname>
   <methodparam>
      <type>resource</type><parameter>connection</parameter>
   </methodparam>
   <methodparam choice="opt">
      <type>string</type><parameter>qualifier</parameter>
   </methodparam>
   ...
```

Function structure

- Standard structure for functions:
  - Function name & short description
  - Function signature
  - Description
  - Exceptions
  - Change log
  - Parameters
  - Return values
  - Examples
  - See also

Structure: Exceptions

```
<refsect1 role="exceptions">
 &reftitle.exceptions;
 <para>
  When does this function throw exceptions?
 </para>
</refsect1>
```

Structure: Change log

```
<refsect1 role="changelog">&reftitle.changelog;
  <para>   <informaltable>
    <tgroup cols="2">
     <thead>      <row>
       <entry>&Version;</entry>
       <entry>&Description</entry>
      </row>
     </thead>
     <tbody>      <row>
       <entry>Enter the PHP version of change here</entry>
       <entry>Description of change</entry>
      </row>
     </tbody>
    </tgroup>
   </informaltable>  </para>
 </refsect1>
```

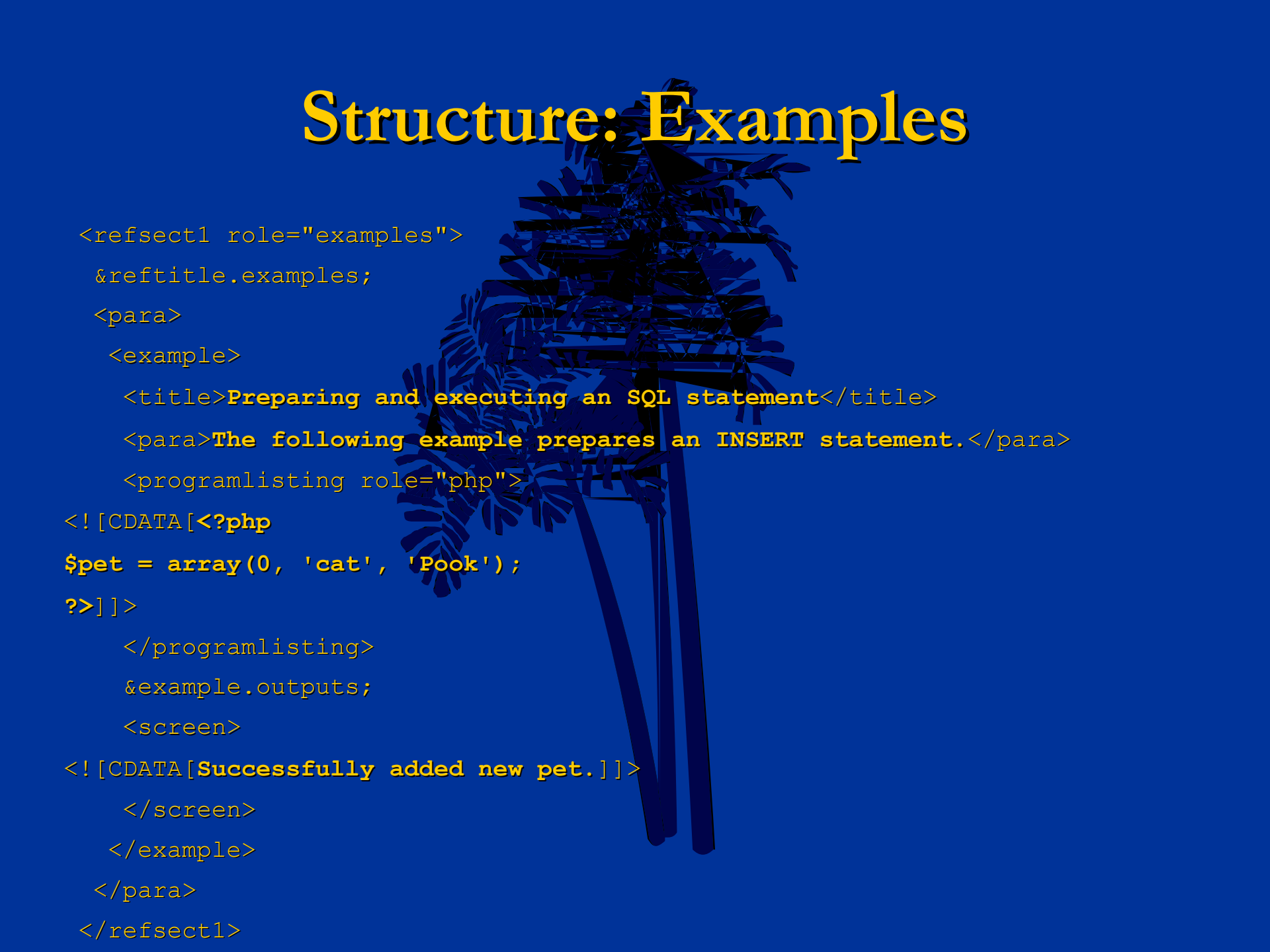
Structure: Parameters

```
<refsect1 role="parameters">
  &reftitle.parameters;
  <para>
   <variablelist>
    <varlistentry>
     <term><parameter>database</parameter></term>
     <listitem>
      <para>
       For a cataloged connection to a database,
       <parameter>database</parameter> represents
       the database alias in the DB2 client catalog.
      </para>
    </varlistentry>
   </variablelist>
  </para>
 </refsect1>
```

Structure: Return values

```
<refsect1 role="returnvalues">
 &reftitle.returnvalues;
 <para>
  What the function returns on success.
 </para>
 <para>
  What the function returns on failure.
  See also the &return.success; entity.
 </para>
</refsect1>
```

Structure: Examples



```
 <refsect1 role="examples">
  &reftitle.examples;
  <para>
   <example>
    <title>Preparing and executing an SQL statement</title>
    <para>The following example prepares an INSERT statement.</para>
    <programlisting role="php">
<![CDATA[<?php
$pet = array(0, 'cat', 'Pook');
?>]]>
    </programlisting>
    &example.outputs;
    <screen>
<![CDATA[Successfully added new pet.]]>
    </screen>
   </example>
  </para>
 </refsect1>
```

Structure: See also

```
<refsect1 role="seealso">
  &reftitle.seealso;
  <para>
   <simplelist>
    <member><function>db2_exec</function></member>
    <member><function>db2_fetch_assoc</function></member>
    <member><function>db2_fetch_both</function></member>
    <member><function>db2_fetch_into</function></member>
    <member><function>db2_fetch_row</function></member>
    <member><function>db2_prepare</function></member>
    <member><function>db2_result</function></member>
   </simplelist>
  </para>
 </refsect1>
```

Generating docs for one extension

- Generating the HTML for every extension takes a long time
- You can hack one of the generated files to build just one extension:
  - Open `phpdoc/entities/builtin-extensions.xml`
  - Comment out every other extension (`<!-- -->`)
  - Watch `make html` fly through your build

Updating documentation

- APIs change, examples get added or extended, and (rarely) docs might need to be corrected
- Updating documentation:
  - `cvs up` to get the latest version
  - Make your changes
  - TEST! `make test`, `make html`...
  - `cvs commit` *`filename [filename...,]`*
- Commit white space changes separately to ease translation

Cool documentation tools

- Periodic analysis of documentation by extension at http://phpdoc.info/meta/
  - Number of open documentation bugs
  - Function aliases
  - Functions missing examples
  - Undocumented functions
    - Stub documentation
    - Compared with PHP extension source
- Great spot to start tackling problems

LiveDocs

- LiveDocs server: automatically synchronizes source files, builds full-text search index, & generates XHTML output from XML source
- Check out LiveDocs source from http://cvs.php.net/livedocs/
- http://docs.php.net/ is a LiveDocs server

Translations

- Check out a specific language:

```
bash$ cvs -d $REP co phpdoc-nl
```

- Build a specific language:

```
bash$ ./configure --with-lang=nl
bash$ make html
```

- (First time): Copy the English source file into the language tree
- Track revisions in comments or side file

Editing user notes

- User annotation is a major feature of the PHP manual; dozens of user notes are submitted each day
- Problem: Average quality of submissions is low
  - Questions, bug reports, spam, redundant information, bad suggestions...
- Therefore, PHP documentation needs user note editors

Accessing user notes

- Prerequisite: CVS account with commit access to phpdoc repository
- Three access points to user notes:
  - Subscribe to php-notes mailing list
  - Read the php-notes newsgroup
  - Use the handy "Manage user notes" interface at http://www.phpdoc.info/notes/manage.php

Managing user notes

- Rejecting a user note:
  - Sends an email to the submitter with links to PHP Buzilla, mailing lists, etc
- Deleting a user note:
  - Silently deletes the user note -- good for spam, or if you incorporate the note into the documentation
- Editing a user note:
  - Enables you to add an inline [Editor's note] at the top of the entry

Documentation bugs

- PHP bugs (http://bugs.php.net) often turn into documentation bugs
- If you have a CVS account, and you understand the problem, you can commit changes and return as "Fixed in CVS"

Becoming a contributor

- Subscribe to the phpdoc mailing list
- Hang out at #phpdoc on IRC
- Start by submitting patches to the phpdoc mailing list to earn your CVS account stripes
- Or offer to document a highly visible extension

References

- Documentation HOWTO:
  - http://doc.php.net/dochowto/
- phpdoc mailing list:
  - phpdoc-subscribe@lists.php.net
- #phpdoc IRC channel on irc.freenode.net
- DocBook reference:
  - http://docbook.org/