Version 1.0

Tobias Schlitt (toby@php.net)
Stefan Neufeind (neufeind@php.net)

PEAR (PHP Extension and Application Repository) – An introduction. This article gives an overview of the usage of PEAR and the benefits you can gain by using PEAR and it's packages.

Agenda

About us

Stefan Neufeind is a student from Neuss (Germany) and will shortly finish as a Bachelor of Computer Science. Beside his studies he works as a freelancer for SpeedPartner.de. Stefan is very engaged for PEAR development, the QA team and the documentation team. He currently maintains the packages Image_Graph, Net_Traceroute, HTTP_Session and Validate.

Tobias Schlitt is currently employed at Deutsche Bank AG and additionally works as a software architect for the eBranch project. Since 4 years he's addicted to PHP and contributes actively to PEAR. Tobias maintains the packages Net_FTP, Image_Text, Image_Tools and Log_Parser and works beside that actively on the PEAR website. He is also engaged in the PEAR QA team.

WTFIP

> *What the fuck is PEAR*

This question has been asked many times and we are trying to answer it inside this article.

PEAR means in written words „PHP Extension and Application Repository“. And the name is program. PEAR provides a huge variety of high quality open source components to make your life with PHP more comfortable and secure.

PEAR was founded by Stig S. Bakken in 1999 and since then a huge community has been established around it. PEAR currently contains over 220 packages, which are maintained by more than 500 developers. In 2003 the „PEAR Group“ has been established, as an institution of regulation and contact point for external interests.

PEAR's goal is not to provide the highest amount of PHP classes and packages, but to offer high quality and reliable code packages for use in any kind of PHP project. The PEAR community actively tries to avoid redundant code inside the clearly structured repository. Where possible redundant proposed packages are merged into the existing packages to improve their flexibility, functionality and quality.

The PEAR community constantly improves PEAR and PHP development in general by defining standards and creating a common PHP package infrastructure. For example, the community has defined common coding standards, which have been taken over by many other projects and companies to improve the readability of code and to enable foreign developers to easily dig into their work. Unittests and inline documentation using phpdoc are highly recommended to maintain quality and a cleanly documented API.

PEAR institutions

The website

One of the most important institutions of PEAR is the PEAR website (http://pear.php.net). Beside the listing of all available packages sorted in categories the website provides a large online documentation in several languages about the PEAR packages and PEAR in general. The online presence also covers the PEAR bug tracking system, where everyone is free to file bugs and enhancement-suggestions concerning packages, the website, documentation and the bug system itself.

Another tool available on the website is PEPr (pronounced 'pepper' – PEar Proposal system). PEPr's main goal is to allow active and potential developers to propose new packages to PEAR. The system handles the complete workflow of a proposal, which includes the formal proposal of a package (including source, examples etc.), discussion of the package idea and the voting of the community on if the package should added to PEAR.

The PEAR website additionally provides information and direct contact options to the developers, as well as detailed information on their packages, release history and notes.

For developers, the website offers functionality like registering new packages and uploading new releases of a package. The releases are verified directly after uploading, to raise their quality.

Last but not least, the website integrates with the backend of the PEAR installer. The XML_RPC services are hosted there and make use of the common PEAR infrastructure, as the website does.

Support

PEAR provides several ways of support for users and interested developers. Besides mailinglists you can reach the developers on IRC, contact the PEAR QA team or get in touch with the PEAR group. The PEAR website even offers you to get in direct contact with every maintainer by email.

Mailinglists are a very essential institution of PEAR. Several mailinglists exist for different purposes. A general support and user mailinglists can be reached at pear-general@lists.php.net. It's main purpose is help with installation and usage of the PEAR installer(s) and the packages. The developers, and everyone interested in the technical part of PEAR, meet at pear-dev@lists.php.net. Beside those two major discussion points there are additonal special purpose mailinglists e.g. for QA handling (pear-qa@lists.php.net) and documentation (pear-doc@lists.php.net).

The public mailinglists are listed on the website inside the support section (available at http://pear.php.net/support.php). Everyone interested in the specific topic of a list is invited to subscribe and post to it. Contributors are welcome at any time.

The official IRC channel of PEAR named #pear can be accessed on the efnet IRC servers. The partner program PECL provides the channel #php.pecl. In both channels several developers are available around the clock and can help you with most of your problems with PEAR in “realtime”.

Beside the official support institutions, PEAR lists external resources on it's support website (http://pear.php.net/support.php) where you can gain help from. There are several tutorials, talk slides and 3rd party websites noted.

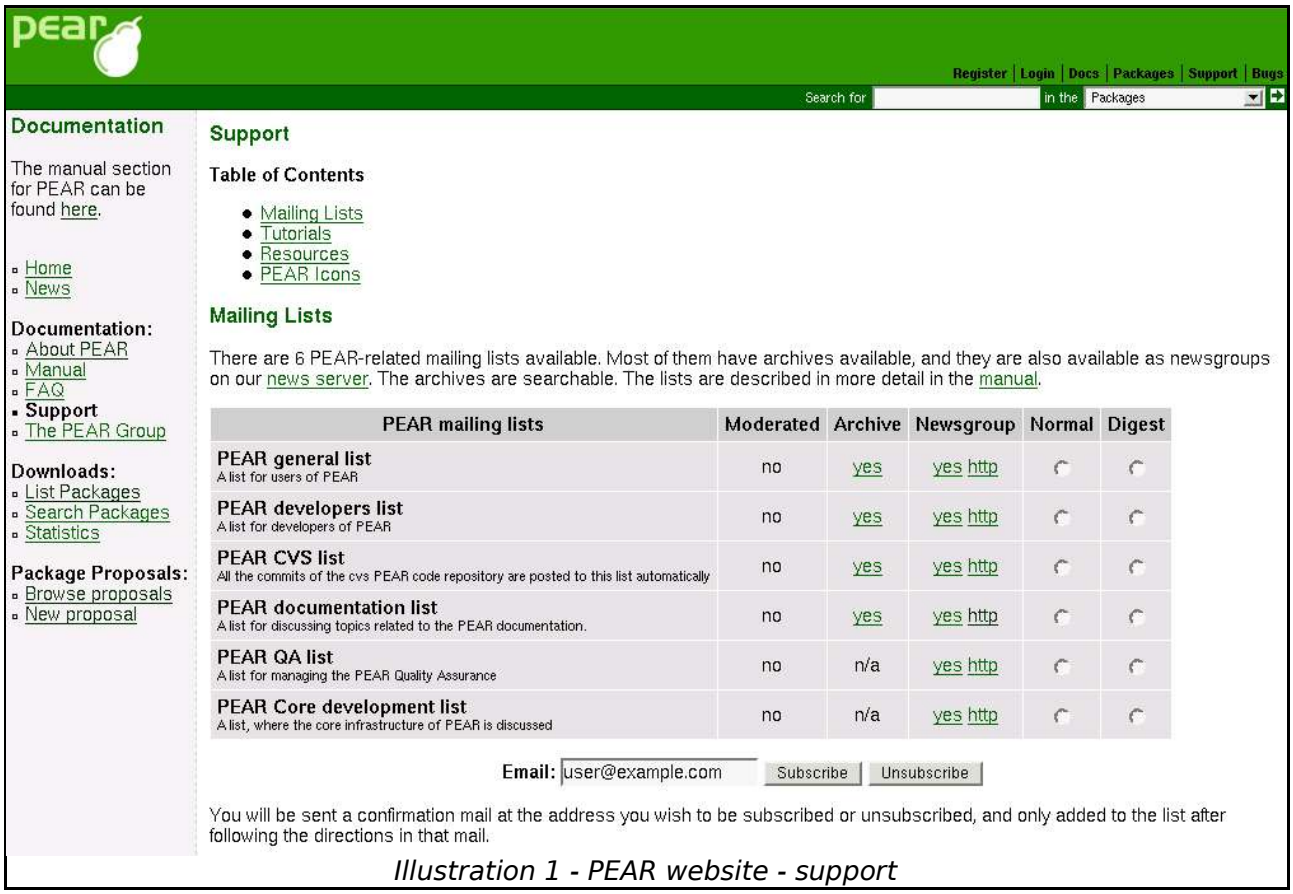

Illustration 1 - PEAR website - support

Hierarchy

PEAR does not implement a strict hierarchy as other projects do. Everyone is free to propose a new solution to any issue at any time and everyone is free to criticize everything. Although there exist special groups for different purposes everybody is free to work actively on those topics.

The highest institution in PEAR is the PEAR group. It has been announce by the founder of PEAR in 2003, to implement a central regulation institute. The PEAR group's purpose is not to lead or rule the community, but to step in on unresolvable issues inside the community and to provide a single point of contact to the world outside PEAR. It consists of a clearly defined group of people.

Loose team-structures exist for several other purposes. The youngest one (founded early in 2004) is the PEAR QA team, which is dedicated to review packages and point out and clear QA issues. A common task for the QA team is

to find common solution to PHP4/5 migration problems, carefully track package changes with regards to BC changes (backward compatibility), provide suggestions for improvement of code readability/reliability and announce them to the community.

The documentation team assists the package developers regarding package documentation, handle documentation translations and care for the common PEAR documentation on the website. In contrast to the PEAR group, everyone is free to join these teams and help. Even users and external developers are welcome to contribute, if they want to give something back to PEAR and do not want to maintain a whole package or are not able to. For example, it is often very helpful to improve package documentation from a users point of view. And (almost) everybody can easily help translating documentation.

Packages

To structure the huge variety of packages PEAR provides, categories were introduced at very early stage of the PEAR project. They group packages into logical categories and allow for an easier overview. Examples for category names are „Authentication“, „File_System“, „HTML“, „Networking“ or „XML“.

Through the website one can browse those categories, take a detailed look at the contained packages and download the latest releases. The installer (described further below) is a helpful tool for package handling.

PEAR's intention is to avoid duplication of packages where possible. For that, redundancy is checked during the package proposal by the community. The usual way to resolve package duplication is to get in touch with the current maintainer of the existing implementation and check whether it's possible to incorporate the new suggestions and improve the existing functionality on a collaborative basis.

Illustration 2 - PEAR website - package overview

Development of packages

PEAR generally allows it's maintainers to be free in the development of their packages. However certain rules exist to structure development, to provide quality and to make life for users easier. One rule is that maintainers are strongly encouraged to design the API of their packages as flexible and generic as possible and to not write code again which already exist in a different part of the PEAR repository. Code duplication can often be resolved by adding dependencies. PEAR packages can depend on other PEAR packages, PHP extensions and PHP versions, but may not depend on external PHP code. External PHP dependencies have to be replaced by internal ones. Dependencies are shown on the package websites and are also handled by the PEAR installer.

Standards

Beside those „flexible“ rules PEAR also defines fixed rules every developer has to stick to. The coding standards for example clearly define details like indention, function and variable naming and certain coding constructs. Another example is the definition of reliable package states and a common versioning system.

A version of a package, which is still under heavy development and maybe not yet ready to use is called „dev“. The next state is „alpha“, which indicates that the package should work, but maybe has some missing features and/or still has several potential bugs. Beside that, during „alpha“ stage the package maintainer is allowed to break BC (backwards compatibility) by changing the API of his package.

API breaks are definitely forbidden in beta stage. If a package is declared „beta“ you can be sure that the API will not change in any next version unless a new major version (e.g. 2.0) is developed. Beta packages might still have some bugs but should run fairly stable.

Before a package is really called „stable“ (which is the highest release state), it should run through a number of RC stages (release candidates) which indicate that the package is almost ready for production use and that heavy testing is now required from the user's side (especially to guarantee BC to earlier versions). RC releases are still in state “beta”.

Several other rules exist in PEAR to unify the development of packages. Beside the mentioned common names for methods are defined, as well as a standard for package testing (PHPT and PHPUnit tests). Testing can also be done automatically through the PEAR installer.

PEAR installer

The PEAR installer provides an easy way for installation, upgrade and removal of packages. Together with the PEAR website it's one of PEAR's core components. By accessing the XML_RPC-interface provided by the PEAR website it is able to negotiate version numbers and states of packages.

Installing the installer

The PEAR installer is shipped (in a stable form) together with PHP since 4.3.0. If your distribution is shipped without the PEAR installer or contains an outdated/broken release, it's usually easy to install the PEAR base system with a little magic. The website http://go-pear.org provides an easy “installer for the installer”. If you're running Linux and have access to the CLI-version of PHP simply run

lynx -source http://pear.php.net/go-pear | php -q

```
Below is a suggested file layout for your new PEAR installation.
To change individual locations, type the number in front of the
directory.  Type 'all' to change all of them or simply press Enter
to accept these locations.

 1. Installation prefix           : /usr
 2. Binaries directory            : $prefix/bin
 3. PHP code directory ($php_dir) : $prefix/share/php
 4. Documentation base directory  : $php_dir/docs
 5. Data base directory           : $php_dir/data
 6. Tests base directory          : $php_dir/tests

1-6, 'all' or Enter to continue:

The following PEAR packages are bundled with PHP: DB, Net_Socket,
Net_SMTP, Mail, XML_Parser, PHPUnit.
Would you like to install these as well? [Y/n] :

Loading zlib: ok
Downloading package: PEAR.............ok
Downloading package: Archive_Tar......ok
Downloading package: Console_Getopt....ok
Downloading package: XML_RPC..........ok
Bootstrapping: PEAR...................(remote) ok
Bootstrapping: Archive_Tar............(remote) ok
Bootstrapping: Console_Getopt.........(remote) ok
Downloading package: DB...............ok
Downloading package: Net_Socket.......ok
Downloading package: Net_SMTP.........ok
Downloading package: Mail.............ok
Downloading package: XML_Parser.......ok
Downloading package: PHPUnit..........ok
```

```
Extracting installer..................ok
```

> *Through an interactive setup this will download all needed core-components and install them in the appropriate locations. If you don't have a lynx (or similar) at hand you can also simply download the script from http://go-pear.org/ and execute it manually using PHP at the command line.*

If you prefer to use your browser to perform the installation or don't have command line access, simply place the script on your webserver and open it's URL in your browser.

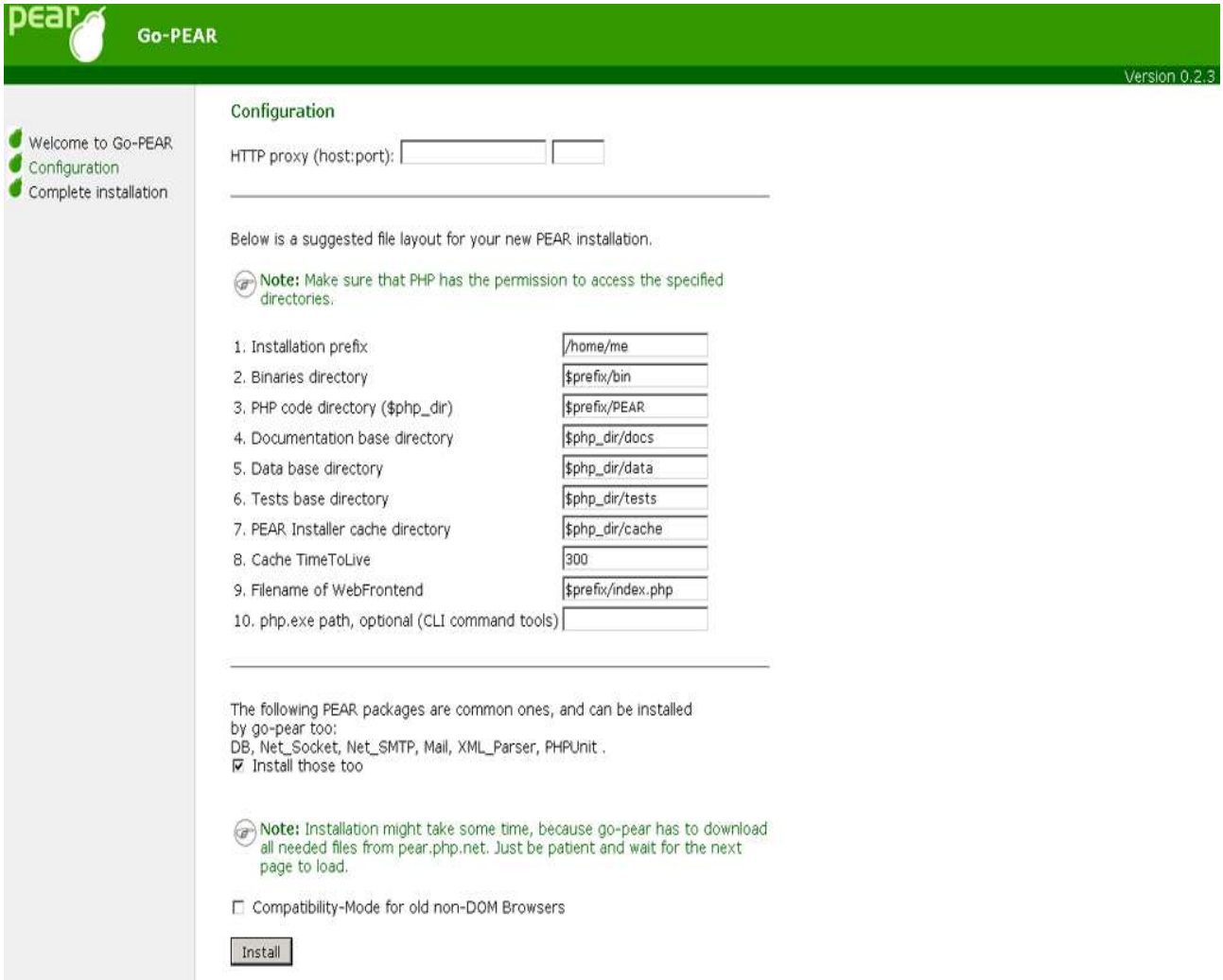

Illustration 3 - go-pear via webbrowser

Using the installer

Once PEAR is installed the command line installer “pear” offers you commands that make package handling quite simple. For most common tasks you'll use “pear <command>” where command is “install”, “uninstall”, “list”, “list-upgrades” or “upgrade-all”. Entering “pear help” will give you a detailed overview of the commands and their options. If packages have (optional)

dependencies the PEAR installer will tell you about this when you're doing an install/upgrade. As you can see in the illustration PEAR will even update the main component (itself) when needed. Using the upgrade/upgrade-all comments allows you to easily track package updates.

```
pear upgrade-all
Will upgrade date
Will upgrade http
Will upgrade mail
Will upgrade net_smtp
Will upgrade pear
downloading Date-1.4.2.tgz ...
Starting to download Date-1.4.2.tgz (42,318 bytes)
............done: 42,318 bytes
downloading HTTP-1.2.3.tgz ...
Starting to download HTTP-1.2.3.tgz (3,515 bytes)
...done: 3,515 bytes
downloading Mail-1.1.3.tgz ...
Starting to download Mail-1.1.3.tgz (13,415 bytes)
...done: 13,415 bytes
downloading Net_SMTP-1.2.6.tgz ...
Starting to download Net_SMTP-1.2.6.tgz (9,106 bytes)
...done: 9,106 bytes
downloading PEAR-1.3.1.tgz ...
Starting to download PEAR-1.3.1.tgz (95,968 bytes)
...done: 95,968 bytes
upgrade-all ok: Date 1.4.2
upgrade-all ok: HTTP 1.2.3
upgrade-all ok: Net_SMTP 1.2.6
upgrade-all ok: Mail 1.1.3
upgrade-all ok: PEAR 1.3.1
```

There are also two alternatives available if you prefer to install your PEAR packages by using a webbrowser (use PEAR-package “PEAR_Frontend_Web”) or using a desktop application (use PEAR-package “PEAR_Frontend_Gtk”).

You can install packages from the PEAR repository automatically (“pear install <packagename>”). But the PEAR package format also allows distribution of packages as files. This way you install your own packages from (downloaded) files via “pear install mypackage.tgz” or even use direct URLs for installation.

Packages can not just be “classes” but also applications. Distributing applications in package format allows for easy installation/removal and automatic dependency-handling. (e.g. PhpOpenTracker by Sebastian Bergmann uses the PEAR installer: http://www.phpopentracker.de)

Soon to come: Channel support

Since the installer is one of PEAR's core components there are already several plans for further improvment. One of the most anticipated new features will be “channel support”. A “channel” is an additional repository beside the PEAR

repository, that you will be able to add to your configuration. This will allow you and your clients to install classes and applications from various sources, making distribution of updates a lot easier.

User projects

Now you have a vague idea what PEAR has to offer and how it may simplify your daily work. Not yet convinced? Here is a list of projects that are already proud to be using PEAR packages in their applications:

- Horde (Framework)
- TikiWiki (Wiki application)
- S9Y (Weblog application)
- Savant (Template engine)
- Seagull PHP Framework (Framework)
- patUser (Usermanagement package)
- YAWP, Yawiki (Wiki application)

In addition to that, PEAR is used on a large variety of even high traffic websites and in commercial projects. More and more ISPs also offer ready-to-go PEAR installations in their webhosting packages.

Summary

So what has PEAR got to offer for you?

- enterprise-ready components
- high-quality packages
- reusable, flexible classes
- improved security by 100 eyes watching the code
- help via community and direct contact to maintainers
- easy and fast installation/upgrading of packages
- automatic installation of needed packages via dependencies
- free licenses (PHP license, LGPL, ...):
  can even be used "for free" in commercial applications

Questions, ideas, feedback?

We hope to have piqued your interest in PEAR. Maybe consider taking a closer look at the available components and see how your projects can benefit from using PEAR packages.

Your feedback on this introduction is very much appreciated. Please feel free to contact us by email, come around on IRC, post your thoughts on the pear-general mailinglist ... or meet us at the next conf :-))