

PhpObjects Overview

Hermann Röscheisen

June 26, 2004

Abstract

PHPOBJECTS is a framework implementing a powerful component object model which pushes PHP beyond the limits of a simple scripting environment. This paper describes the key features of PHPOBJECTS including installation and quick start.

1 Preliminary Note

PHPOBJECTS is a kind of feasibility study or experimental software not yet suited for high traffic sites. Its purpose is to testify that an object-oriented web component framework in PHP is practicable at least for sites with little to medium traffic.

PHPOBJECTS has some limitations caused by the underlying Zend engine, especially concerning shared resources and the time spent in the parser on each request. Some of this problems can be addressed by customizing PHP itself, for example with cache software.

Independent from performance problems inherent to PHP as a scripting environment, PHPOBJECTS in particular has a potential performance flaw as it is a complex framework located on top of this. In fact creating web pages using PHPOBJECTS is slower than generating pages with regular PHP. — This is the price paid for the benefit of working with reusable components and writing less code with less errors.

The common way other people do PHP, either procedural or object-oriented, is called *regular* PHP in this paper.

2 Quick Setup

This quick setup ignores any security considerations important for real-life environments. Normally frameworks shouldn't be placed under the document

root of the webserver as users may access private files containing confidential data, such as the `bundle.xml` which reveals the internal structure of a project or one of the configuration files which can contain database passwords.

1. Copy the applications with extension `.poa` and the frameworks with extension `.pof` to the document root of your webserver.
2. Make the document root directory itself writable to the PHP user or create a suitable `.tmp` directory for every application , e. g. `Test.poa.tmp` for `Test.poa` and make it writable to the PHP user.
3. On some unix flavours make the adaptor `index.php` of every application executable.
4. Test the application, its url should be `http://localhost/Test.poa`, if this doesn't work try addressing the adaptor explicitly with this url: `http://localhost/Test.poa/index.php`. If this works you also may try the page request handler serving pages which are marked as externally accessible: `http://localhost/Test.poa/index.php/p/Main`.

3 Runtime Architecture

PHPOBJECTS consists of frameworks with code and resources reusable across different applications. Frameworks are directories with the extension `.pof` and applications are directories with the extension `.poa`. PHPOBJECTS itself consists of several frameworks notably the core foundation with `CFCore` and `CFFoundation` as well as the web page services with `POEngine`, `POParser`, `PODynamicElements` and `POValidatedElements`. Applications are gateways which are externally visible to users. An application bundle primarily contains a single procedural `index.php` file which transfers control to the fully object-oriented PHPOBJECTS framework. This file is called *adaptor*. All frameworks used need to be on the application's search path, classes and resources required can be loaded automatically.

3.1 Bundles

The central structure of PHPOBJECTS is the *Bundle*. A bundle is a collection of classes, resources and webserver assets. There are two flavours of bundles, *Applications* and *Frameworks*. For security reasons bundles shouldn't be installed under the document root of the webserver. Instead you install them elsewhere where the PHP runtime has access and provide a symbolic link to its `docroot`.

3.1.1 Application or Main Bundle

The application or main bundle is a bundle which contains the adaptor script `index.php` that is used to serve *all* requests to a specific application. The main bundle is a regular framework except that it cannot be used in other bundles.

3.1.2 Framework

Frameworks are bundles which are reusable. Different applications may use the same single installed framework. Frameworks may use other frameworks. Unlike libraries which use the paradigm *call my functions* a framework uses the paradigm *don't call us, we call you*. This means that several methods of your objects are called by the framework automatically with no additional effort. A framework is also a library but should be more. For example if you use `$this->pageWithName("Main")` in your logic code the PHPOBJECTS environment knows in which framework this page resides. Its template is getting parsed and its logic class instantiated and returned. All happens automatically, because *we call you*.

3.2 Urls

PHPOBJECTS generates urls looking strange at first glance. After the addressing of the adaptor (`index.php`) there are more url segments which aren't query parameters.

3.2.1 Page Request Handler

For example `http://localhost/Test.poa/index.php/p/Main` is a legal url which instructs the framework to use the page request handler to serve this request by returning the page named `Main`. As it doesn't use query parameters this url is search engine friendly. For security reasons not all pages are externally addressable, for example page 2 of a registration workflow shouldn't ever be bookmarkable.

3.2.2 Direct Action Request Handler

`http://localhost/Test.poa/index.php/a/Stats/get` is another search engine friendly pattern which uses the direct action request handler to serve the request. Primarily this loads a class named `Stats` and triggers a method in this class named `getAction` which in turn is responsible for generating the response. For security reasons the class needs to be a subclass of

`PODirectAction` and the action method must end with the postfix `Action`. This prevents hackers from executing arbitrary methods in the class which are not intended as direct action methods.

3.2.3 Component Request Handler

`http://localhost/testapp.poa/index.php/c/23.0.0.3.2.1` is a pattern which utilizes the component action request handler. This triggers an action within a previously rendered page of an existing session. Such an url is therefore intentionally not bookmarkable. This searches the session's page store for page 23, the page with context id 23. The tree of subcomponents and elements is walked until its element id matches the sender id 0.0.3.2.1. The action connected to the corresponding element is executed and returns the response.

4 Terms and concepts

PHPOBJECTS establishes another way of building web pages. Instead of designing projects around urls and query parameters design in objects, methods and properties is encouraged. Web pages are composed of reusable components similar to JavaBeans in desktop applications. Reusable components have properties that can be read and set. Components can have a logic class which implements specific business logic. Components bundled in frameworks are reusable across projects.

4.1 Terms

Projects in terms of PHPOBJECTS are different from projects in regular PHP. PHPOBJECTS has some key differences:

- every unit of html is a component
- a page is a special sort of component
- components are constructed of static html and/or other components and/or controls called dynamic elements
- components consist of at least one and up to three files: a mandatory template (`.html`), an optional description (`.pod`) and an optional class (`.class.php`)

- logic code and template code is completely separated, actually embedding PHP code in a template is impossible
- pages are addressed by name and not by full path in the document root, therefore they must have unique names in the scope of all frameworks bound to the application. Typically this is achieved by two-letter-prefixing
- there is only one `echo` statement in the whole project and it is executed automatically by the framework once at the end of a request
- all resources including components and classes are organized in reusable frameworks, also called bundles
- one web site or one microsite is normally one PHPOBJECTS application with an unique base url
- per application there is only one external visible `.php` file, the adaptor which handles all requests, e. g. `/backapp/index.php`
- an application uses frameworks, frameworks may use other frameworks
- all internal urls are constructed by the framework
- there are hooks that can be used to customize framework behaviour

5 Tuning

Serious performance improvements are achieved with Turck MMCache. A tuned version of PhpObjects has only been tested with PHP 4.3.4 on Mac OS X 10.3.2 and Apache 2. This is not the build in apache version, rather it's provided by the people at <http://www.serverlogistics.com>. The PHP 4 module for Apache 2 contains Turck MMCache which is deactivated by default. You need to install two packages, *Complete PHP 4* and *Complete Apache 2*. Activate it and set it also to handle session save and restore. You will encounter a significant performance improvement.