Aspect-Oriented Web Development in PHP

Jorge Esparteiro Garcia

Faculdade de Engenharia da Universidade do Porto
jorge.garcia@fe.up.pt

Abstract. Aspect-Oriented Programming (AOP) provides another way of thinking about program structure that allows developers to separate and modularize concerns like crosscutting concerns. These concerns are maintained in aspects that allows to easily maintain both the core and crosscutting concerns. Much research on this area has been done focused on traditional software development. Although little has been done in the Web development context. In this paper is presented an overview of existing AOP PHP development tools identifying their strengths and weaknesses. Then we compare the existing AOP PHP development tools presented in this paper. We then discuss how these tools can be effectively used in the Web development.
Finally, is discussed how AOP can enhance the Web development and are presented some future work possibilities on this area.

Keywords: Aspect Oriented Programming, Web Development, AOP, PHP

1 Introduction

As web applications become more complex, it becomes harder to separate independent concerns. Aspect oriented programming (AOP) [9] paradigm offers various ways to separate concerns which can help us to reduce time and complexity of applications. AOP better separates concerns than previous methodologies (object oriented, procedure, etc.), thereby providing modularization of crosscutting concerns [10].

Despite AOP being a programming paradigm that can be used with the most common object oriented languages, much of the research has been done on developing standalone applications and little has been applied to Web development. Therefore, we believe Web development can be improved using aspect-oriented techniques.

AspectJ [8] is one of the most popular AOP proposal tools that offers the possibility to develop web applications using JSP (Java Server Pages). However, nowadays PHP is becoming the most widely used Web scripting. PHP has an edge over locked-in solutions such as JSP and ASP for most Web development work because it is a cross-platform technology.

PHP is, nowadays, one of the best and most popular script programming languages for innumerable web applications. Over 20 millions domains on web use PHP as the web programming language [12]. Specially suited for Web development, it´s recognized as one of the most used programming languages in the world.

Therefore in this work are presented the existing AOP PHP development tools and is made a comparison of these tools showing their strengths and weaknesses on the web development. It´s also discussed the impact of AOP Web development with a language with such a wide-spread use.

The rest of the this paper is as follows. Section 2 gives a brief overview of Aspect Oriented Programming. In Section 3 are presented the existing AOP PHP development tools. In Section 4 is made a comparison of these tools in the context of the web development. Section 5 concludes the paper and discusses the future work.

2 Aspect-Oriented Programming Web development

AOP is a new technology for separating crosscutting concerns into single units called aspects. An aspect is a modular unit of crosscutting implementation.

It encapsulates behaviors that affect multiple classes into reusable modules. With AOP, we start by implementing our project using our OO language (for example, Java), and then we deal separately with crosscutting concerns in our code by implementing aspects.

Finally, both the code and aspects are combined into a final executable form using an aspect weaver. As a result, a single aspect can contribute to the implementation of a number of methods, modules, or objects, increasing both reusability and maintainability of the code.

Figure 1 explains the weaving process. The original code doesn't need to know about any functionality the aspect has added; it only needs to be recompiled without the aspect to regain the original functionality.

2.1 Aspect-Oriented Programming Web Development Concerns

Although aspect oriented programming is a very young area of research, from the very beginning there have been concerns, such as synchronization or distribution, that have had the attention of researchers due to their clear crosscutting nature. However, there are other concerns that don´t crosscut so clearly, and haven´t had the focus of the aspect oriented community.

Following the proposal by Kilesev [8], Reina et al. [2] have addressed the following concerns on the development of web applications:

- **Security**. This concern is really authentication. Authentication is the process of determining whether someone is who is declared to be. This aspect tries to prevent that unauthenticated users have access to some web pages.

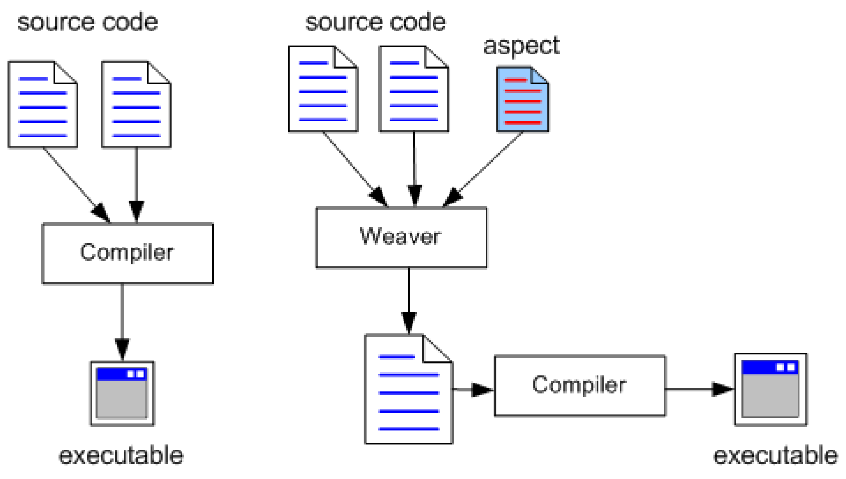


Fig. 1. Aspect Weaver

- **Design by Contract**. A contract is something that should be guaranteed before calling a method on a class, but, also, the class should guarantee certain properties after the call. This is a way to check if certain conditions are fulfilled before executing a method. Some programming languages have implemented this concern using the notion of assertion.

- **Exception Handling**. It is a simple way of applying an exception handling policy, in such a way that all exceptions should be handled by notifying the end user that something went wrong.

- **Logging**. This concern encapsulates the logger behavior. When certain points during the execution of a program are reached, a message is printed out. Tracing. It is a debugging tool very similar to a logger, but it only tracks one type of event, a method execution.

- **Profiling**. A debugging concern which measures the execution time consumed in some methods. This concern can be very helpful for detecting some bottlenecks. Pooling. Pooling is a strategy to obtain faster database connections. When a database and all its associated files are closed, the connection and server resources are released. If the same application needs the database services again, a new connection will have to be established and server resources will have to be asked for again, wasting resources, and, of course, slowing down the application. If we maintain a pool of connections and server resources, we will obtain faster database connections.

- **Caching**. Caching is the retention of data, usually in the application, to minimize network traffic flow and/or disk accesses. If database information is cached on the application server, the database server can be relieved of its repetitive work.

In the development of an web application there are some aspects that are crucial for the success of the final product. These key aspects are: pooling, caching and security.

On the one hand, pooling and caching are very important because they can have influence on response time, which is an important requirement, because a user can be bored waiting for a response, specially in a web application, where the response time can very exasperating to the user.

On the other hand, authentication is really important, because a web application can easily be altered by an intruder, and needs to be protected. But there are other key concepts, such as navigation [1], that should be addressed during the web development.

2.2 PHP Web development

There are several programming languages for the development of Web Interfaces. These programming languages are used primarily for developing server-side applications and dynamic content. Microsofts ASP.NET, PHP, Java, CGI, Perl are some of the technologies used on this area. PHP is currently one of the most popular server-side scripting systems on the Web.

One major part of PHP which has helped it become popular is that it is a very loose language; in particular, it is dynamically typed. The key technical contributor to PHP success is its simplicity, which translates into shorter development cycles, easier maintenance and lower training costs.

That is, the rules aren´t as strict with variables - they don´t have to be declared and they can hold any type of object. Further, unlike many other languages (like C++ and Java), arrays are able to hold objects of varying types, including other arrays.

PHP, like Perl or Javascript is a dynamically weakly typed interpreted language. However, like Java, classes (and in this case functions) are special entities within the language, they can't be directly referenced.

3 Aspect-Oriented PHP development tools

In this section are presented the existing tools for web development in PHP. There also presented some code examples of each implementation. We can consider two main methods of implementation of the extensions to support AOP in PHP. The Pre-Processing Implementation and the Runtime Weaving implementation.

In the Pre-Processing Implementation a preprocessor is used to perform source code transformations and carry the weaving process. Then the PHP source code produced can then be deployed in a standard PHP environment.

In the Runtime Weaving implementation, there is no previous source code generation, the weaving process is done dynamically and PHP code is executed normally as a "traditional" PHP web application.

3.1 Pre-Processing Weaving Implementations

There are already some different AOP implementations for PHP. The first implementations of all required pre-processing, this means it is necessary to run the PHP source through a processor first or patches against the engine. This pre-processing applies code transformations and integrates the aspects to generate final PHP code for the application.

This method has some disadvantages to classic PHP developers because pre-processing adds an extra step to the development. This means developers can´t write the code and then test it, the source code has to be processed after written and then only after that can be tested. Another issue of this method is that PHP becomes no longer an interpreted language. The code that is produced won´t run natively on any interpreter, the PHP developer has to program on a Java fashion way to be able to develop AOP web applications.

Though, this implementations are very useful because they can really implement AOP in PHP and bring to the language other capabilities that were not present until now.

PHPAspect The phpAspect [11] compiler weaves aspects implementing crosscutting concerns as shown in the weaving chain in Figure 2. Inspired in AspectJ, the phpAspect is one of the most used solutions to develop PHP web applications using the proposals of AOP. As previous referred, the weaving process is static and based on Lex and Yacc analysis to generate XML parse trees. XSLT is used to perform the source code transformation on those trees. PHPAspect contains the usual jointpoints on AOP like method execution/call, attribute writing/reading, object construction/destruction, exception throwing, etc.

One important plugin on this tool is the PHPAspect Builder that provides check of aspect syntax and also offers the possibility to weave each php file contained in the project. Figure 3 shows a screenshot of this plugin.

Aspect-Oriented PHP Aspect-Oriented PHP(aoPHP) [3] uses a preprocessor for the PHP programming language written in Java. This preprocessor is responsible for the weaving process of the aspects and the base-code. aoPHP contains support to the methods: execution/call, field read and write, among others.

The aspects on this implementation are stored in files with the extension .aophp. aoPHP plans to evolve in direction of an Aspect-Oriented Language with a rich joint point model such as AspectJ.

AOP Library for PHP This library, developed by Dmitry Sheiko [13], can implement Aspect Oriented Programming (AOP) by executing the code of classes that enable orthogonal aspects at run-time.

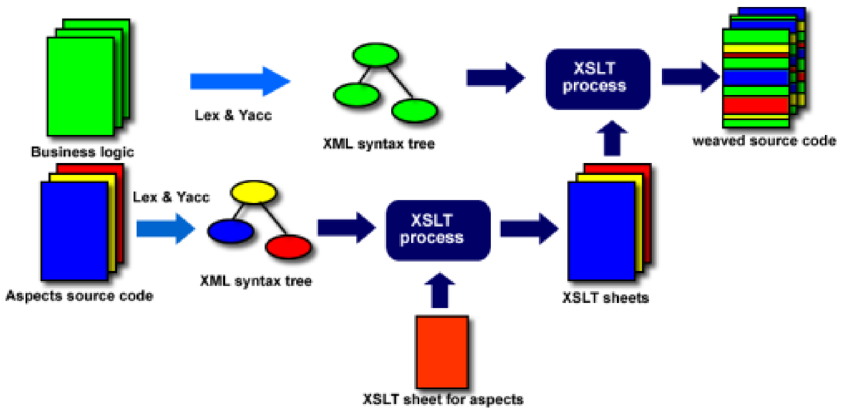
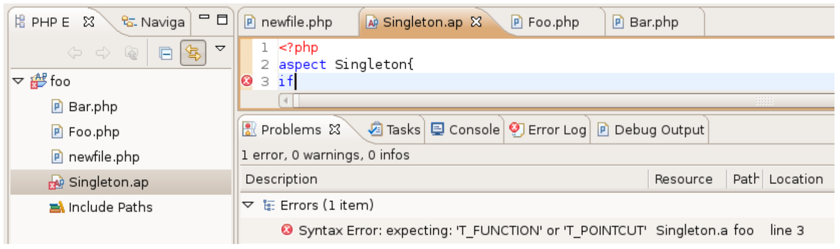


Fig. 2. PHPAspect´s weaving chain



```
<?php
aspect Singleton{
if
```

```
1 error, 0 warnings, 0 infos
Syntax Error: expecting: 'T_FUNCTION' or 'T_POINTCUT'  Singleton.a  foo  line 3
```

Fig. 3. PHPAspect Builder Screenshot

The intention is to provide and implement orthogonal aspects in separate classes that may be interesting to add, without affecting the main business logic to the application, like logging, caching, transaction control, etc.

The package provides base classes for implementing defining point cuts where the code of advice class is called to implement actions of the orthogonal aspects that an application may need to enable.

3.2 Runtime Weaving

The extensions that implement this method use aspect weavers that work in application runtime, taking advantage of the interpreted nature of the PHP language, and weaving the aspects on demand.

aspectPHP aspectPHP [4] is another implementation that works in application runtime, taking advantage of the interpreted nature of the PHP language.

A first version of this tool was adapted from aoPHP implementation described on Section 3.1. On this version they assume all the aspects are already located in the same directory, as separate files "*.aspect". The aspects were then loaded in sequence by the weaver, to weave them with the original code.

Another version based on the Zend compiler was released. This version was released due to in the previous implementation is unlikely that the PHP call-site will be captured, especially when the functions are not inside a script file, but included from other program files. Zend compiler was then adopted changing the compiler to support aspects.

GAP: Generic Aspects for PHP GAP [6] is the first implementation in AOP PHP that supports dynamic weaving, genericity and an extensible pointcut language.

This is a project under development by Sebastian Bergmann, the author of PHPUnit [5] (software testing framework for PHP based on JUnit of Java), for the creation of an AOP extension that takes full advantage of the new functionalities made available by PHP 5, such as the Reflection API or the overloading methods call(), get() and set(). This project, which is not available for public use yet, uses the PHP Runkit [7] extension, a new and powerful reimplementation of the Classkit extension mentioned previously. Figure 5 shows the aspect declaration with the AOP Library for PHP in GAP.

Figure 4 illustrates the GAP weaving chain. This chain uses a streams filter written in PHP, the GAP Weaver hooks into the loading of the source code of classes (represented in green color) and aspects (represented in blue color). The first weaving stage performs source code transformations then, it passes the generated source code to the compiler, integrated in the PHP Interpreter. The second weaving stage operates on the bytecode generated by the compiler. It uses the Runkit extension to complete the insertion of generic hooks into the classes.

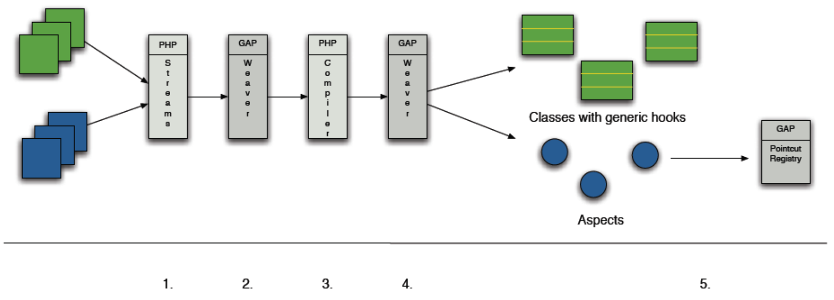


Fig. 4. The weaving chain of GAP

In this implementation, the definition of aspects is obtained through the creation of classes whose methods correspond to the kind of advice to apply (before, after or around). The creation of pointcuts and association of advice to the joint points is carried through with the aid of annotations that precede the class definition. An example of GAP aspect that logs all method calls can be seen in Figure 6.

```
<?php

require_once 'Class.php'
require_once 'aop.lib.php'
$aspect = new Aspect;
$pointCut = $aspect->pointcut('all AClass::aMethod';
$pointCut->_before('... before advice code ...';
$pointCut->_after('... after advice code ...';
$pointCut->destroy();
$object = new AClass($aspect);
?>
<?php class AClass {
private $aspect;
public function __construct($aspect) {
$this->aspect = $aspect;
}
public function aMethod() {
Advice::_before($this->aspect);
// ... base code ...
Advice::_after($this->aspect);
}
} }?>
```

Fig. 5. Aspect declaration with the AOP Library for PHP in GAP.

An example of GAP aspect that logs all method calls can be seen in Figure 6.

4 Comparison of the Tools

All tools presented in this paper are in a very early stage of development. Therefore we cannot yet make some deep considerations about this AOP PHP development tools.

Although, we can make some comparison based on the weaving process method used by each tool and the possibilities of improvement of each tools features and capabilities.

4.1 Weaving Process Method

In the Pre-Processing Weaving Implementation method there are some disadvantages when compared to the Runtime Weaving process method, due to the need to use a preprocessor to perform source code transformations and carry the weaving process.

There are two main issues. Preprocessing adds an extra step to the development: it's no longer code then test as in traditional php development; but has become code, preprocess then test. The benefit of php being an interpreted

```
<?php
/* @pointcut allInvocations : method(* *->*(..));
* @after allInvocations : Logging->log();
*/
class Logging {
public function log($joinPoint) {
printf(
"%s->%s() called %s->%s()\n",
$joinPoint->getSource()
->getDeclaringClass()
->getName(),
$joinPoint->getSource()
->getName(),
$joinPoint->getTarget()
->getDeclaringClass()
->getName(),
$joinPoint->getTarget()
->getName()
);
}
} ?>
```

Fig. 6. GAP aspect that logs all method calls

language is lost. The second issue is that you have stopped writing PHP, but have started writing a dialect of PHP; your new dialect won't run natively on any interpreter.

Some preprocessors like Aspect-Oriented PHP move the step of preprocessing out of the programmers hands to the hands of apache, so that apache will preprocess the PHP before handing it off to the Zend engine, but it´s still a Pre-processing that is executed in the PHP Hypertext PreProcessor language. This is highly redundant and very slow.

Developing Web applications using a tool with the Runtime Weaving process method can take some advantages. The main advantage is, of course, not having to deal with the step of preprocessing that allows to apply the aspects in runtime. It also can be very annoying to a php developer to deal with the world of class libraries, frameworks or applications that are need to the Pre-Processing Weaving Method.

5 Conclusions

This paper presented various solutions and approaches to support the AOP paradigm in the PHP Web development. Depending the method used for the weaving process the implementations were classified as Pre-Processing implementations or Runtime Weaving implementations.

Despite AOP being a programming paradigm that can be used with the most common object oriented languages, much of the research has been done on developing standalone applications and little has been done applied to Web development. Therefore, we believe Web development can be improved using aspect-oriented techniques and this paper shows some good web development tools to help the improvement of Web development.

On Section 4 we shown that the Runtime Weaving method can be very interesting applied on the Web Development, in particularly the fact of offering the developer the possibility to apply aspects in runtime without the need of the step of preprocessing.

Despite the possibilities of applying PHP to AOP Web development, the solutions that were presented and studied in this paper, are all on a very early stage of development. This means, that doesn´t exists a PHP tool that can explore and implement the AOP paradigm like an OO language as Java. Although, these tools seem to very promising specially GAP, that can change the PHP web development in the next few years.

References

1. J. Torres A. M. Reina. Separating the navigational aspect. In *Proceedings of the Workshop of Aspect-Oriented Programming for Distributed Computing Systems*, Viena, Austria, 2002.
2. M.Bonilla A. M. Reina, J. Torres. Aspect-oriented web development vs. non aspect-oriented web development. In *Workshop AAOS2003: Analysis of Aspect Oriented Software*, Darmstadt, Alemania, 2003.
3. aoPHP. Website visited on January 3th 2008 at http://www.aophp.net/.
4. aspectPHP. Website visited on January 5th 2008 at http://www.cs.toronto.edu/ yijun/aspectPHP/.
5. S. Bergmann. PHPUnit website Visited on January 6th 2008 at http://www.phpunit.de.
6. Sebastian Bergmann and Günter Kniesel. Generic aspects for php. In *Proceedings of EWAS 2006*, Netherlands, 2006.
7. S. Golemon. Runkit extension for PHP website, Visited on January 6th 2008 at http://pecl.php.net/package/runkit.
8. Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and William G. Griswold. An overview of AspectJ. *Lecture Notes in Computer Science*, 2072:327–355, 2001.
9. Gregor Kiczales, John Lamping, Anurag Menhdhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-oriented programming. In Mehmet Akşit and Satoshi Matsuoka, editors, *Proceedings European Conference on Object-Oriented Programming*, volume 1241, pages 220–242. Springer-Verlag, Berlin, Heidelberg, and New York, 1997.

10. C. Lopes and W. Hursch. Separation of concerns, 1995.
11. phpAspect. Website visited on January 5th 2008 at http://phpaspect.org/.
12. PHP.net. PHP.net website visited on January 24th 2008 at http://www.php.net/usage.php.
13. D. Sheiko. *Aspect Oriented Software Development and PHP*, volume 5. In php — architect, 2005. issue 4, pages 17-25.