Generative and Model-Driven Approaches for PHP

Ghica van Emde Boas
Bronstee.com, the Netherlands
emdeboas@bronstee.com

ABSTRACT

The use of the PHP programming language is a fast-growing, and rapidly the language becomes mature enough to develop robust web-applications with. This paper investigates ways that model-driven principles can be exploited for PHP application development.

Keywords

Model Driven Software development, MDSD, PHP, Generative techniques, Java, MVC, XML.

1. Introduction

Let us start with describing a few IT-related cases:

1. My local rowing club has a professionally looking website where articles and photo's are posted, calendars of events are kept up-to-date etc[1]. It was made by a few volunteers in their evening hours using Typo3[2]. Typo3 is an open source Content Management Framework written in PHP.

2. Porsche Brazil has a very good website[3], made in three months with a small team, using Mambo[4], another Content Management System written in PHP.

3. Sept. 1, 2005. IBM Signs EURO 1.5 Billion Contract with ABN AMRO to Manage Infrastructure Services[5]. The Global Agreement includes a new innovation center to dvelop highly-advanced IT Services to support new financial products. ABN AMRO is the largest bank in the Netherlands, with world-wide operations.

The three cases described above cover a spectrum of software development efforts from very small to extremely large. The Model-Driven Software Development (MDSD) and related approaches such as Model-Driven Architecture (MDA) and Software Factories seem to target the large to very large projects, because expensive and extensive tools and processes are required.

The industrial revolution in the IT world has its consequences though. ABN AMRO is decreasing its IT staff, and IBM in the Netherlands does likewise. The bulk of the IT work will probably be done in cheaper countries. As a citizen of a western developed country, I have two choices: try to be one of the few left highly skilled staff, or try to work in the other end of the spectrum: the small shops, the community organizations, or the highly flexible web-sites of moderate to large companies.

The second option poses an interesting challenge of how we can exploit model-driven principles to this kind of application development. We would like to explore this further in this paper.

2. About PHP

Until today the PHP language has been largely ignored by the scientific and coporate communities. The same applies to methods for developing PHP applications. Many corporate developers view PHP as a simple web-scripting lanuage, not intended for serious use. Is this still justified?

In July 2004 the fifth version of PHP appeared. It supports advanced object-oriented facilities and this has brought methods and techniques developed earlier for the Java lanuage to the PHP programmer. A series of books, featuring PHP patterns, and object-oriented development methods start to appear. PHPUnit, ANT for PHP, object-relational mapping layers extended XML support and the like are available now to the PHP community.

Today PHP is the fifth most popular programming language overall, behind Java, C, C++, and Perl, closely followed by Visual Basic [11].

If we look at web-application programming, PHP may be the most popular language considering that Apache is installed on 72% of all servers[10] and 46% of those have the PHP module installed, followed by Frontpage 21%, Perl 12%, and Python 2%. Several modules could account for the usage of Java, my guess is that it adds up to around 5%.

2.1 Characteristics of PHP

Wikipedia[9] describes PHP as follows[7]: The PHP model can be seen as an alternative to Microsoft's ASP.NET/C#/VB.NET system, Macromedia's ColdFusion system, Sun Microsystems' JSP/Java system, and to the CGI/Perl system.

Famous examples of PHP applications include phpBB[6] as well as MediaWiki[8], the software behind Wikipedia.

Important to note about PHP is its suitablility for development in small teams. Because PHP is open source, it is easy and cheap to start using it. A considerable majority of people who contribute to open source efforts such as *phpClasses*[16] are very young, living in Argentina, Brazil, China, India, Indonesia, Poland, Russia, … and just a few are located in western Europe or the US.

2.2 A Prototypical PHP Script

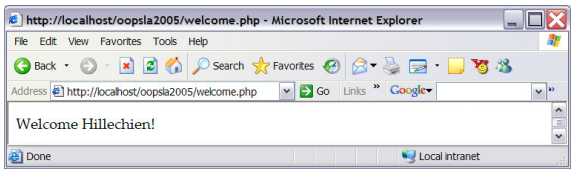
To set the scene for those of you who have never seen a PHP script before, here is a small example:

welcome.php

```
<?php $name='Hillechien'; ?>
<html>
<body>
Welcome <?php echo "$name"; ?>!
</body>
</html>
```

As you can see, a PHP script looks like a piece of HTML text, where anywhere a piece of PHP code can be placed, between <?php and ?> tags.

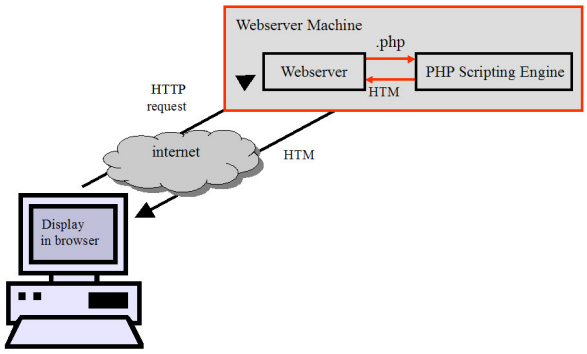
You probably guessed also that this script will output the following in a browser:



If you are familiar with JSP or Velocity, you will recognize that the way PHP is imbedded into HTML is very similar.

2.3 PHP and Server Architecture

The architecture of any PHP application is very simple: an HTTP request goes from a browser to a server. The server responds by sending an HTML stream back to the browser.



If the request is for a PHP script, the server will route the file through the PHP interpreter, which in turn can issue database requests etcetera.

To preserve state between interactions with a browser, PHP can keep *session* data. This allows for a conversation with a user, for example to fill a shopping cart, reserve an airline ticket, or similar.

2.4 Generative PHP

How can we exploit generative programming techniques in PHP?

2.4.1 Templates

In my long practice of developing tools for code generation, mainly targeting the Java environment, I found the use of template languages to be a powerful and easy to use means of rapidly developing enginges to generate code or text of various kinds.

In the Java world there seems to be only a small number of template languages. Except the language I made myself very long ago, for use with the predecessor of the FUUT-je tool, I only know of Velocity and JET.

The Velocity engine was originally developed as a simpler alternative to the rather complicated development in JSP (Java Server Pages).

EMF, the Eclipse Modeling Facility, heavily exploits templates to generate code for applications modeled in EMF. Their template language is called JET and it is very close to JSP in syntax and use.

We were pleasantly surprised when we found that the use of template languages is very natural to a PHP programmer, we will describe why in the next sections.

2.4.2 MVC Architecture

The good old Model-View Controller (MVC) architecture is very applicable to web-applications, including those developed in PHP.

The MVC pattern encourages developers to split the code into at least two parts (or more, except for very small applications), where the View is separated from the Model and the Control.

In the case of a web-program, the View is usually a piece of HTML text generated by the web-application. To make the views more flexible, it is common to use *templates* that specify a specific look and feel for a common back-end application. All popular Content Management Systems, including the ones mentioned above, use this method of customization for individual installations of their systems. There are even companies that sell templates for specific CMS'es.

2.4.3 PHP as Template language

From the *welcome.php* example it is clear that PHP is a template language itself, more or less a hybrid of both JET and Velocity, but with more functionality. Therefore it is easy to develop view templates directly in PHP.

Nevertheless, there is a debate about the suitability of PHP as a template language, and a large number of specialized template languages is in use. About every self-respecting PHP programmer seems to have developed his/her own template language. The most notable example of a template language that is widely used with PHP is Smarty[13]. Advocates of these template languages say that the full power of PHP should not be given to end users and that PHP is too complex as a template language.

A simple approach of using PHP itself as a view template is outlined by Brian Massasi[14]. This solution implements a *Template* class. The variables to be replaced in a template are passed in an array upon instantiation of an object of the Template class, and of course the template itself. The template is really a PHP script, but it should contain only HTML and simple view-logic. The template is executed a by including it as you would include any PHP script using the PHP *include()* function. Replacement of the variables in the template is automatic, because the data in the array is extracted to a set of variables with the PHP *extract()* function. It is worth mentioning that arrays in PHP are associative maps and that there are very powerful built-in functions to handle arrays.

2.4.4 Code Generation with PHP

Many PHP applications use a relational database to store data about web-site users, orders, products, digital images and everything else that applications may store in a database. The most popular of the database management system in use is MySQL, available on almost every webserver where PHP is installed.

Writing an access layer for a specific set of tables is a tedious task, therefore several attempts have been made to automate this task and generate code for it. As we have seen, it is not difficult to write templates to do this in PHP.

There is a good article by Jack Herrington about generating PHP database access layers[18]. The most interesting product mentioned in this article is phpCodeGenie[19]. If you have a database schema, phpCodeGenie is able to generate PHP code to access this database and to produce forms for entering data into the tables.

A distinct disadvantage of generating code out of a database schema is that a schema may not be expressive enough to generate good code. For example, if the schema says that a field is a BLOB, does this field contain a large piece of text, an image or binary data? Likewise, Unix timestamps cannot be distinguished from integers without having additional knowledge.

Another real problem with this approach is the difficulty of customization. As soon as something changes in the database schema, the code must be regenerated and all customizations must be re-coded.

3. Model-Driven PHP

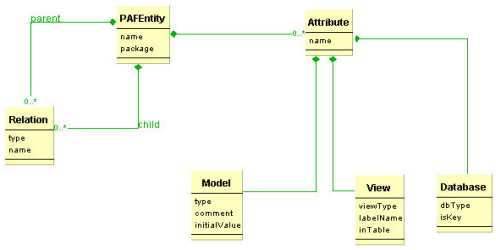
In the previous section we have tried to indicate that there is a need for models that are more expressive than a database schema if you would like to generate good code.

The MDSD (or MDA, DSL, Software Factory) world intends to provide these more expressive models and the means to create code or executables from them, by transformation or code generation.

I tried to make an example of a model-driven framework for PHP using the FUUT-je tool. My PHP Application Framework supports facilities to use multiple languages, user authorization and authentication, a flexible menu system and a database access layer that recognizes images, timestamps etc. FUUT-je easily generates tons of working PHP code for this. The framework can be found at Sourceforge[20]. I found however that this approach is not suitable to the PHP environment, because it requires non-PHP tooling that needs to be installed on the user client in addition to Java. In addition a steep learning curve is required to understand how to fit code into this framework and how to customize it to your website's needs.

3.1.1 Model-View-Persistence Unit

A fresh start led to defining a simplified meta-model.



In this model, a *model* is defined as you would in any statical analysis model. In addition, for every attribute you can specify how it will be represented in a browser window and how it will be stored in a database. The model-entity together with its view and database aspects is called a *Model-View-Persistence Unit*. Relationships between multiple MVP-Units are not implemented yet.

No PHP code is generated for an MVP-Unit, only an XML configuration file. Generic features in PHP5 allow to read the XML definition and instantiate an object of class *PAFEntity* that behaves like a class with attributes, getters/setters, database access methods, and data-display forms as specified in the MVP-Unit. PAFEntity can also output the DDL needed to create the database table for the MVP-Unit. The PAFEntity class is barely 400 lines of code, very small and lightweight therefore.

Customization can be done by defining a subclass of PAFEntity and overriding or adding methods. A factory script will automatically instantiate an object of the subclass instead of an object of PAFEntity when found. This provides some resilience against changes of the MVP-Unit. Regeneration of the XML file will usually not invalidate a customized subclass.

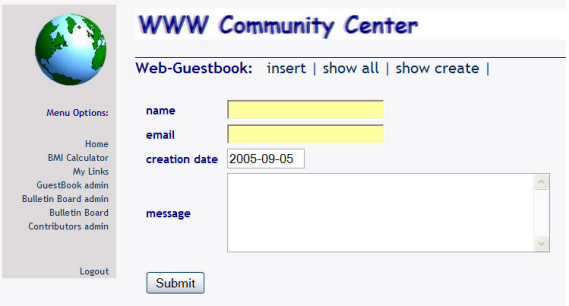
In the near future we will implement a PHP application that allows you to specify MVP-Units and generate XML from it, for now we have made an Excel spreadsheet skeleton and a VisualBasic Macro that should allow most developers to specify MVP-Units easily.

PAFEntity name	label		timestamp	date
BBItem	Bulletin Board Item	Create PAF-XML	20050801105027	August 1, 2005

	Model	View		Database
name	**type**	**viewType**	**labelName**	**dbType**
id	int	textfield(5) readonly	id	int auto_increment
author	string	textfield(25)	sender	varchar(50)
date	date	textfield(10)	date	int
message	string	textarea(60, 3)	message	text
status	string	select(published, unpublished)	status	varchar(20)
creationDate	date	textfield(10) readonly	creation date	int
modificationDate	date	textfield(10) readonly	last modification	int

The screenshot above shows an excel sheet for an MVP-Unit. It defines an item for a bulletin-board application. I hope it convinces you that it is easy to define these units. For readability, we did not show all columns.

MVP-Units provide for a complete and customizable database access layer, with no programming required. In a real application, additional classes and scripts will be needed. Just to give an impression what a simple application could look like, here a partial screenshot:



MVP-Units are just a start of what can be done with simple means in a PHP environment. More elaborate model transformation tools may be feasible in the future when browsers become capable of running interactive graphical applications.

4. Conclusion

A silent revolution is happening in the IT world. While the Internet matures, a new kind of applications is emerging: small services, distributed information, online commerce, to name just a few. A new generation of software developers to produce these applications is programming differently. They work as individuals and not within corporations.

The new languages that help developing these applications are not very different from what we had before. What makes them different is their capability to blend with HTML.

The premier example of such a language is PHP. PHP has an abundance of built-in database connectivity. PHP has many features adapted to developing web based applications. PHP is easy to learn and the newest version 5 supports object-oriented features that makes it look like an un-typed version of Java. PHP is free and it is available for all popular platforms.

Model-Driven Software Development faces the challenge of finding methods that can help this fascinating new area of IT in a way that is suitable for the PHP programming environment. We tried to make a start with designing tools that can be used by almost everyone.

We acknowledge that the large and complex Java tools will be instrumental in improving the large-scale software developments, and we hope that enough knowledge will result that can be applied to small-scale development projects done with PHP.

REFERENCES

[1] Koninklijke Roei en Zeil Vereniging "Het Spaarne", http://www.hetspaarne.nl

[2] Typo3, http://www.typo3.com

[3] Porsche Brazil, http://www.porsche.com.br

[4] Mambo, http://www.mamboserver.com/

[5] IBM press release, http://www-1.ibm.com/press/PressServletForm.wss?MenuChoice=pressreleases&TemplateName=ShowPressReleaseTemplate&SelectString=t1.docunid=7866&TableName=DataheadApplicationClass&SESSIONKEY=any&WindowTitle=Press+Release&STATUS=publish

[6] phpBB, http://en.wikipedia.org/wiki/PhpBB

[7] WikiPedia about PHP, http://en.wikipedia.org/wiki/PHP

[8] MediaWiki, http://en.wikipedia.org/wiki/MediaWiki

[9] WikiPedia, http://en.wikipedia.org/wiki/Wikipedia

[10] Netcraft, http://www.netcraft.com/Survey/

[11] Security Space, http://www.securityspace.com/s_survey/data/man.200508/apachemods.html

[12] The TIOBE Programming Community index, http://www.tiobe.com/tpci.htm.

[13] Smarty, http://smarty.php.net/

[14] http://www.massassi.com/php/articles/template_engines/ Template Engines, by Brian Massassi

[15] FUUT-je, http://dev.eclipse.org/viewcvs/indextech.cgi/~checkout~/gmt-home/download/index.html

[16] PhpClasses, http://www.phpClasses.org

[17] MySQL, http://www.mysql.com

[18] Jack Herrington, database access layers, http://freshmeat.net/articles/view/843/.

[19] PhpCodeGenie, http://phpcodegenie.sourceforge.net/.

[20] PHP Application Framework, http://www.sf.net/projects/paf

[21] We acknowledge trademarks or registered trademarks of Sun Microsystems Inc., International Business Machines Corporation, and others.