introducing Zend FRAMEWORK

A First Look From a Main Contributor

by MIKE NABEREZNY

PHP: *5.0.4 or higher*

O/S: *Any supported by PHP*

SOFTWARE: *Zend Framework 0.1.3, a PDF viewer*

LINK: *http://framework.zend.com*

CODE DIRECTORY: *zendframework*

TO DISCUSS THIS ARTICLE VISIT:
http://forum.phparch.com/296

At the Zend/PHP Conference in October of 2005, Zend Co-founder and CEO Doron Gerstel gave a keynote address to a packed auditorium. It was here that he announced the PHP Collaboration Project.

The PHP Collaboration Project, as he described it, would be a remarkable effort that would create a collaborative playing field around PHP and bring together companies both small and large, individual contributors, and community members to enlarge the PHP ecosystem. The project's goal was very simply stated: further advance the use of the PHP programming language. Mr. Gerstel announced that a collaborative industry effort was necessary to continue PHP's penetration into business markets while at the same time maintaining PHP's already strong leadership position as the dominant programming language for the Web.

To accomplish these goals, Zend and its partners would initiate three parallel efforts under the umbrella of the PHP Collaboration Project. First, a proposal would be made to the Eclipse Foundation for the creation of a new, open-source development environment based on the Eclipse Rich Client Platform (RCP). Second, Zend would rebuild its Developer Zone from the ground up to be a better resource and collaboration hub for professional PHP developers. Third, Zend would spearhead the Zend Framework—an open-source software stack to power web applications written in PHP 5.

Delivering on the Promise

On March 3rd of this year, after months of hard work by dozens of people from many companies, the promise of the PHP Collaboration Project was realized when the first milestones were met on all three efforts. A formal proposal was submitted and approved by the Eclipse Foundation with support expressed by IBM, Intel, SAP, and Actuate. The new Zend Developer Zone also opened its doors to the public and posted its first article on professional PHP development.

On this day, the first preview release of the Zend Framework source code was also released, and a new project site opened to support it at *http://framework.zend.com*. The release, although still very much in its infancy, garnered so much fanfare among potential users in the PHP "blogosphere" that one blogger declared reviews of the framework to be a "blog worm," overwhelming the PHP news aggregators with Zend Framework reports. The release sparked a surge of downloads, and a *Slashdotting*. Hundreds of potential users subscribed to the mailing list that became active overnight.

> **"We believe this initiative will be a positive catalyst to help propel the open source community to new levels of innovation."**
>
> **- Jon Bork, Director of Intel's Open Source Program Office**

Goals of the Zend Framework

Zend Framework was immediately set apart from all other frameworks from its inception because it is backed with a promise of active development and continuous support by Zend, its many commercial partners, and prominent community members. This is perhaps the most important differentiator between the Zend Framework and prior efforts. However, some of the Zend Framework's goals are equally unique, as stated on its website.

Provide a repository of high quality components that are actively supported. These components must be written entirely in PHP 5 and be `E_STRICT` compliant. Full API and end-user documentation must always be provided and continuously maintained.

Provide a complete system for developing web applications powered by PHP 5. Zend Framework aims to provide an architecture for developing entire applications with no other library dependencies. This code will always be actively developed, tested, and supported by Zend and the PHP Collaboration Project. In time, this will also include automation and tooling developed by Zend and its partners to further enhance the development experience.

Don't change PHP—it's already a great platform (even without a framework!). This means keeping components as simple as possible for mainstream users by not trying to write a book about design patterns or trying to develop purposely large class hierarchies. Zend Framework's success should be driven by the same forces that drive PHP's success: simple and elegant solutions that work well and can be easily understood.

Embrace collaboration and community to further advance PHP 5 programming. The Zend Framework project welcomes and encourages other PHP frameworks. We celebrate diversity and choice as a very large part of PHP's success. Our framework will be as modular as possible so users may choose to use as much or as little of it as they like. Companies with existing code are encouraged to use components from the Zend Framework to further their own projects.

Positively contribute to the PHP 5 ecosystem and the PHP Collaboration Project. Zend Framework aims to further help PHP 5's adoption as an industry standard for businesses developing Web-based applications. This must be done through many channels: providing source code with a license and IP rights suitable for business purposes, promoting PHP best practices, and developing relationships with businesses using PHP 5.

For managers, Zend Framework represents a significant industry effort to supply quality PHP source code that's suitable for commercial projects and backed by real companies. For PHP 5 developers, Zend Framework offers something for almost everyone. It provides a library of useful components that can be used by any project, and it will also offer a complete, integrated software stack for building websites without depending on any other libraries.

Getting Started

Before beginning with the Zend Framework, it's important to recognize that it is still a preview release. While it is very exciting and useful in its present form, being a preview release means that it is not intended for use on production websites and that it will evolve from its present form before the final release. New features are constantly being added and some may even be removed as it is fine tuned. That said, some companies have already deployed Zend Framework in its present form on some high-traffic sites, including parts of *zend.com*. Also, many parts of the Zend Framework were extracted and then refactored from successful websites built by Zend and its PHP Collaboration Project partners.

The first hint that the Zend Framework is different from some traditional libraries is its distribution. The Zend Framework is always downloaded as one large body of code and not many separate pieces. This means that interdependent components are always tested together and problems caused by incompatible component versions can be minimized.

Zend and its partners initiated three parallel efforts under the umbrella of the PHP Collaboration Project.

The Zend Framework is very convenient to download and easy to install. The latest version can always be downloaded as a complete package from *http://framework.zend.com/download/zip*. Some users on UNIX-like operating systems prefer a tarball, which is always available from *http://framework.zend.com/download/tgz*.

After downloading, installation is very simple. Just extract the files to a directory on the development server where they will be used. First, after downloading a new version, be certain to review the `README.txt` file that ships with the distribution. This always contains the latest news and important instructions that are worth reading. The file will always be short and to the point. After reading it, the only other installation step is to add the `library/` directory from the archive to the system's `include_path`. If this is not possible or desirable for some reason, simply place a call to `set_include_path()` at the top of scripts that will use the framework.

Learning by Example

One of the key tenets of the Zend Framework is that it should be simple to use and very well documented. Zend believes that the framework should be simple enough for everyone, even new users, but powerful enough to be a staple of seasoned developers—just like PHP itself. Also in the PHP spirit, the Zend Framework has a great online manual, available at *http://framework.zend.com/manual*, that is full of examples showing how to use the code. The manual is currently available in several languages and more translations are underway..

While the Zend Framework ships with everything that one would expect in a framework, such as database adapters and a Model-View-Controller (MVC) system, these have been done well before and alone aren't compelling reasons to use the framework. It also ships with a number of very specialized components that fill common needs in web development that haven't been served well by prior efforts. The Zend Framework is all about solving the common problems that seem to be encountered by many PHP developers.

The best way to learn about new software is always to try and put it to practical use through some sort of real-life test. Let's look at four of these components in a realistic business scenario.

Sample Components

The following Zend Framework components will be demonstrated in this article:

`Zend_XmlRpc_Client` – With PHP 5 came the introduction of the excellent SOAP extension, which boiled down complicated SOAP-based web services to look and feel like native PHP objects. The XML-RPC protocol is still very popular on the web but PHP never delivered an extension like SOAP to make these services easy to use. PHP developers consuming XML-RPC web services often must manually convert data types or juggle cumbersome messaging objects. `Zend_XmlRpc_Client` fills the gap and provides a client that's very similar to SOAP in its fully automatic operation and ease of use.

`Zend_Pdf` – PDF generation has long been a point of distress for PHP developers. While there are existing

PDF packages available for PHP, there are some problems with them. Many packages either have a license which prevents them from being deployed for free in commercial settings, or require special PHP extensions to be loaded. Most are not feature-complete and it's not uncommon to find PHP programs scraping the output of PDF command line tools to fill gaps in these libraries. To solve these problems, the Zend Framework includes its own PDF generation toolkit that can both generate a new PDF and modify existing ones, without any of these problems.

Zend_Mail – Almost every PHP web application needs to send email. While there are many components available to do this, there are various levels of complexity and it's difficult to find a PHP 5 component to do it that's E_STRICT compliant. Zend_Mail provides a simple solution that solves the majority of email sending problems without unnecessarily forcing developers to learn about the complexity of email formats and MIME encoding.

The first hint that the Zend Framework is different from some traditional libraries is its distribution.

Zend_Search_Lucene – The Apache Lucene project is a fast and feature-rich search engine written in Java. PHP developers have long needed a search solution that is powerful but without requiring Java, web service calls, or complicated APIs. This component is a port of the Java Lucene engine, reimplemented entirely in PHP 5 source code (no extensions required). Since it is a direct port, Zend_Search_Lucene is fully binary compatible with the index file formats of its Java cousin. Zend_Search_Lucene brings unprecedented search capabilities to PHP applications and it doesn't even require a database.

These components are only four out of a much larger body of useful components that build the foundation for the Zend Framework.

Solving a Real Problem

To demonstrate the use of these components, we'll build a simple solution to solve a real problem that might face mainstream PHP developers. In our simplified case, let's say that zend.com needs to create an invoice for a customer. This process will involve collecting data for the invoice, building the invoice itself, emailing the invoice to the customer, and later searching invoices that have been sent. All of these tasks can be done painlessly with the Zend Framework.

The first step in creating the invoice will be gathering the information for it. Like many modern e-commerce websites, zend.com has developed a network of internal web services based on XML-RPC to aggregate data throughout its network of servers. We'll simulate this system by connecting to an example server using Zend_XmlRpc_Client. This will collect the raw information required for the invoice.

Next, a customer invoice will be created. Text-based invoices are fine but one way to stand out, or simply to display information in a more readable fashion, is to provide an invoice in the Portable Document Format (PDF). We'll use a pre-made, rich PDF template that includes some niceties like the Zend logo. Zend_Pdf will be used to read the template and then populate it with the customer's data.

Once a nice PDF document has been generated for the invoice, Zend_Mail will email it off to the customer. Finally, the invoice will be stored in a search index to show how Zend_Search_Lucene could be used to find it again at some later time.

Retrieving the Invoice Data

For this exercise, an example web service has been created to simulate the internal service that would return the data for the invoice. Our web service server is hosted on framework.zend.com and is based on XML-RPC. The endpoint is http://framework.zend.com/xmlrpc. We'll connect to it with Zend_XmlRpc_Client with the code that is shown in Listing 1.

Most XML-RPC servers logically divide their methods into namespaces. For example, most contain a set of de facto "system." methods like system.listMethods(). For

VIRTUAL CLASSROOMS

Online Training Courses from php|architect

Zend PHP Essentials

Our introductory PHP course, *Zend PHP Essentials*, was developed for us and Zend Technologies by PHP expert Chris Shiflett, co-founder of the PHP Security Consortium. This **19-hour course** provides a thorough introduction to PHP development, with particular care to "doing things right" by covering **security**, **performance** and the best development techniques. Rather than cramming as much theory as possible, PHP Essentials provides a thoroughly practical approach to learning PHP—thus ensuring that each student will be able to write good PHP code in a **real-world setting** by the end of the course.

Zend PHP Certification Training

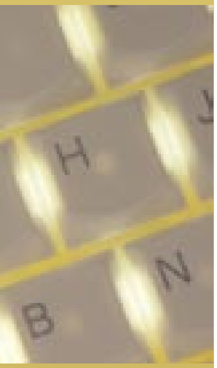
If you want to become a Zend Certified Engineer, this course is the best preparation tool that you'll ever find! Designed by some of the same Subject Matter Experts who also helped write the exam itself, this course covers every single topic that is part of the exam. The *Zend PHP Certification* Training (course) provides a complete overview of the exam, and doubles as an excellent refresher course in PHP for any developer.

Zend Professional PHP Development

This is our advanced course for the professional PHP developer. This course picks up from where *PHP Essentials* ends and provides a thorough, in-depth analysis of advanced features found in both PHP 4 and PHP 5, including object-oriented programming and design patterns, XML development, regular expressions, encryption, e-mail manipulation, performance management and advanced databases.

Course	Description	Start Dates	Duration	Tutoring	Prerequisites	Cost
Zend PHP Essentials	• Covers PHP 4 and PHP 5 • Provides a thorough practical introduction to PHP • Covers security and performance	**Anytime**	7 Sessions 19 Hours 3 Weeks	**YES**	**-**	**$769.99 US** ($999.99 CAD)
Zend PHP Certification Training	• Covers every topic in the exam • Provides an excellent refresher course for PHP at all levels	**Anytime**	7 Sessions 19 Hours 3 Weeks	**YES**	**Zend PHP Essentials**	**$644.99 US** ($838.99 CAD)
Zend Professional PHP Development	• Covers advanced PHP 4 and PHP 5 topics • Perfect for going "beyond the basics" and learning the true power of PHP	**Anytime**	7 Sessions 19 Hours 3 Weeks	**YES**	**Zend PHP Essentials**	**$769.99 US** ($999.99 CAD)

- All our courses are delivered **entirely online** using an innovative system that combines the convenience of the Internet with the unique experience of being in a real classroom.
- All sessions take place in real time, and the students can interact directly with the instructor as if they were in a real classroom either via voice or text messaging.
- In most cases, our system requires no software installation and works with the majority of operating systems and browsers, including Windows, Mac OS and Linux, as well as Internet Explorer, Firefox and Safari.
- All courses include a generous amount of homework and in-class exercises to ensure that the students assimilate each topics thoroughly.
- Tutoring is available (via e-mail) throughout the duration of the entire course.
- Each class includes a complete set of recordings that the students can peruse at their leisure.



For more information, visit our website at ***http://www.phparch.com/phptraining***
or **call us toll-free** at (877) 630-6202 (416-630-6202 outside Canada and the U.S.)

LISTING 1

```
<?php

/**
 * Example using of Zend_XmlRpc_Client to query an
 * XML-RPC web service.  This example retrieves sample
 * invoice data from the Zend Framework website.
 */

require_once ‘Zend/XmlRpc/Client.php’;


// Instantiate a new XML-RPC client using the Zend Framework
// website as the XML-RPC endpoint.
$server = new Zend_XmlRpc_Client(‘http://framework.zend.com/xmlrpc’);


// Call phparch.getInvoice() on the server to get
// the dummy invoice data and dump the resulting array.
// All of the conversions to native PHP types are automatic.
Zend::dump( $server->phparch->getInvoice() );

?>
```

LISTING 2

```
array(2) {
  [“details”]=>
  array(7) {
    [“id”]=>
```

LISTING 2 (CONT’D)

```
    int(1)
    [“customer_name”]=>
    string(12) “Andi Gutmans”
    [“date”]=>
    string(11) “15-Mar-2006”
    [“subtotal”]=>
    string(5) “$1.50”
    [“ca_sales_tax”]=>
    string(5) “0.00%”
    [“shipping”]=>
    string(5) “$0.00”
    [“grand_total”]=>
    string(5) “$1.50”
  }
  [“items”]=>
  array(1) {
    [0]=>
    array(6) {
      [“product_name”]=>
      string(18) “First product name”
      [“quantity”]=>
      string(1) “1”
      [“list_price”]=>
      string(5) “$1.50”
      [“discount”]=>
      string(5) “$0.00”
      [“unit_price”]=>
      string(5) “$1.50”
      [“total_price”]=>
      string(5) “$1.50”
    }
```

The Zend Framework includes its own PDF generation toolkit that can both generate new PDFs and modify existing ones.

the purposes of this text, our namespace is phparch. The Zend_XmlRpc_Client attempts to make remote methods look and feel as similar to native PHP methods as possible. It uses the object overloading capabilities introduced in PHP 5 to map the XML-RPC namespaces into familiar PHP object syntax. For example, a call to system.listMethods() would be written in PHP as $server->system->listMethods().

The example server contains only one method required for this exercise: phparch.getInvoice(). The return payload from the server will be an XML response but this will automatically be converted to a PHP array by Zend_XmlRpc_Client. The array of invoice data is shown in Listing 2. Most of the invoice fields have been returned as strings to simplify the examples.

Building the Invoice

Zend_Pdf can be used to create PDF documents entirely from scratch and includes primitives for tasks like drawing lines, shapes, and color fills.

Instead of drawing the entire invoice programmatically, Zend_Pdf allows an existing PDF document to be imported and then modified. We’ve pre-made a PDF template for the invoice and stored it at *http://framework.zend.com/demos/phparch/invoice-template.pdf*. The template contains all of the graphical elements and nice styling but lacks any actual invoice data.

Listing 3 shows how Zend_Pdf is used to take an array of data for the invoice, draw text onto the PDF invoice template, and then save the invoice to disk. The resultant PDF document can be opened with Adobe Acrobat or any suitable PDF viewer.

Sending to the Customer

After putting polishing on the invoice document with Zend_Pdf, it is ready to be delivered to the customer. Zend_Mail is a simple component that takes the grunt work out of constructing and sending email.

LISTING 3

```
<?php

/**
 * Build on the previous example by querying the web service
 * for sample data and then populating a PDF template to
 * generate an invoice.
 */

require_once ‘Zend/XmlRpc/Client.php’;
require_once ‘Zend/Pdf.php’;


// Get the dummy data for an invoice.  In an actual application,
// be sure to sanitize any data received from foreign sources.
$s = new Zend_XmlRpc_Client(‘http://framework.zend.com/xmlrpc’);
$invoice = $s->phparch->getInvoice();


// Load the PDF template, go to the first page, and
// set the font for drawing text.
$pdf = Zend_Pdf::load(‘/path/to/your/invoice-template.pdf’);
$page = $pdf->pages[0];
$page->setFont(new Zend_Pdf_Font_Standard(
                    Zend_Pdf_Const::FONT_HELVETICA), 12);


// Write the customer details on the invoice.  Simply
// call drawText() with the coordinates on the page.
$details = $invoice[‘details’];
$page->drawText($details[‘id’],            165, 560);
$page->drawText($details[‘customer_name’], 165, 540);
$page->drawText($details[‘date’],          165, 520);


// Write a line item.  This would normally be done in
// a loop for all items but is simplified for the example.
$item = $invoice[‘items’][0];
$page->drawText($item[‘product_name’],       72, 455);
$page->drawText($item[‘quantity’],          290, 455);
$page->drawText($item[‘list_price’],        320, 455);
$page->drawText($item[‘discount’],          380, 455);
$page->drawText($item[‘unit_price’],        440, 455);
$page->drawText($item[‘total_price’],       500, 455);


// Write the remaining fields of the invoice.
$page->drawText($details[‘subtotal’],      500, 205);
$page->drawText($details[‘ca_sales_tax’],  500, 185);
$page->drawText($details[‘shipping’],      500, 165);
$page->drawText($details[‘grand_total’],   500, 145);


// Save the finished PDF document.
$pdf->save(‘/path/to/your/output.pdf’);

?>
```

LISTING 4

```
<?php

/**
 * Demonstrate the Zend_Mail component by emailing the
 * completed PDF document from the previous example.
 */

require_once ‘Zend/Mail.php’;


// Create a new Zend_Mail object
$mail = new Zend_Mail();
$mail->setFrom(‘sender@example.com’,  ‘Sender Name’);
$mail->addTo(‘recipient@example.com’, ‘Recipient Name’);
$mail->setSubject(‘Invoice’);
$mail->setBodyText(‘Please see the invoice attached.’);


// Add the PDF file as an attachment to this mail.  This
// example is simplified; always check that the file
// was read successfully by file_get_contents().
$file = file_get_contents(‘/path/to/your/output.pdf’);
$a    = $mail->addAttachment($file);
$a->filename =’output.pdf’;


// Send the finished email to the customer.
$mail->send();

?>
```

LISTING 5

```
<?php

/**
 * Demonstrates creating a simple search index
 * using Zend_Search_Lucene.
 */

require_once ‘Zend/Search/Lucene.php’;


// The first argument is the path to an empty directory where
// the index will be created.  The seecond argument tells
// Zend_Search_Lucene to create a new index instead of reading
// an existing one.
$index = new Zend_Search_Lucene(‘/path/to/your/index’, true);


// Add a contact to the index.  This could have come from
// the web service as demostracted earlier or anywhere else.
$doc = new Zend_Search_Lucene_Document();
$doc->addField(
        Zend_Search_Lucene_Field::Text(‘customer’,
                                       ‘phparchitect’) );
$doc->addField(
        Zend_Search_Lucene_Field::Text(‘contact’,
                                       ‘Sean Coates’) );
$index->addDocument($doc);


// Add another contact to the index.
$doc = new Zend_Search_Lucene_Document();
$doc->addField(
        Zend_Search_Lucene_Field::Text(‘customer’,
                                       ‘phparchitect’) );
$doc->addField(
        Zend_Search_Lucene_Field::Text(‘contact’,
                                       ‘Marco Tabini’) );
$index->addDocument($doc);


// Commit these changes to the index.
$index->commit();

?>
```

LISTING 6

```
<?php

/**
 * Demonstrates searching the index that was created
 * by the previous example.
 */

require_once ‘Zend/Search/Lucene.php’;


// Open the index for reading.
$index = new Zend_Search_Lucene(‘/path/to/your/index’);


/**
 * Find a document within the index.  Some interesting
 * permutations to try:
 *   ‘customer:phparchitect’
 *   ‘contact:tabini’
 *   ‘customer:phparchitect -contact:tabini’
 */
$hits = $index->find(‘customer:phparchitect’);


// Display the search results.  It is not recommend to mix
// presentation in the program body, this is only an example.
foreach ($hits as $hit) {
    echo ‘Score: ‘         . $hit->score        . ‘<br/>’;
    echo ‘Customer Name: ‘ . $hit->customer     . ‘<br/>’;
    echo ‘Contact: ‘       . $hit->contact      . ‘<br/>’;
    echo ‘<br />’;
}

?>
```

Zend_Mail was contributed by one of the PHP Collaboration partners, 100days.de, a German software consultancy. It was extracted from code that they've used on many websites. As such, Zend_Mail users benefit from features implemented out of their experience sending high volumes of email, such as protection against header injections.

The PHP community has also proven itself to be invaluable and many members of the community have made significant contributions.

In Listing 4, a simple email is built and then the PDF file is added as an attachment. It is then sent off to the customer in one step. The billing process is now complete.

Indexing and Searching

Data such as this invoice would normally be stored in a database. However, this presents an opportunity to demonstrate Zend_Search_Lucene, which is also suited for the task.

Documents in Zend_Search_Lucene are divided into fields. A document may have any number of arbitrary fields, and many different documents can be stored in the search index. Our invoice can also be logically divided into fields and then stored in the index as Zend_Search_Lucene documents. Listing 5 shows everything that is needed to build a simple index with a couple of sample documents.

Searching documents is simple and queries can be built with Google-like strings or by constructing query objects. Listing 6 demonstrates the former and pulls documents from the index based on various criteria. This is all the code that is required to perform some fairly sophisticated searches.

These are very simplistic examples that are not the best demonstration of Zend_Search_Lucene's capabilities. However, they do serve to show the simplicity of the component and how quickly developers can index and then search documents. For more information on Zend_Search_Lucene, readers are invited to visit the online manual to explore the search capabilities in greater depth.

Wrapping Up

Through these examples, we've seen that the Zend Framework offers powerful capabilities coupled with ease-of-use. These small snippets can easily be expanded to enhance the core services of any PHP 5 application.

These are only some of the components that are presently offered. Components for reading RSS and Atom feeds, as well as extensive web services support for Amazon, Flickr, and Yahoo! are other examples. The components in this article, as well as the others, will be explored in greater detail in future articles.

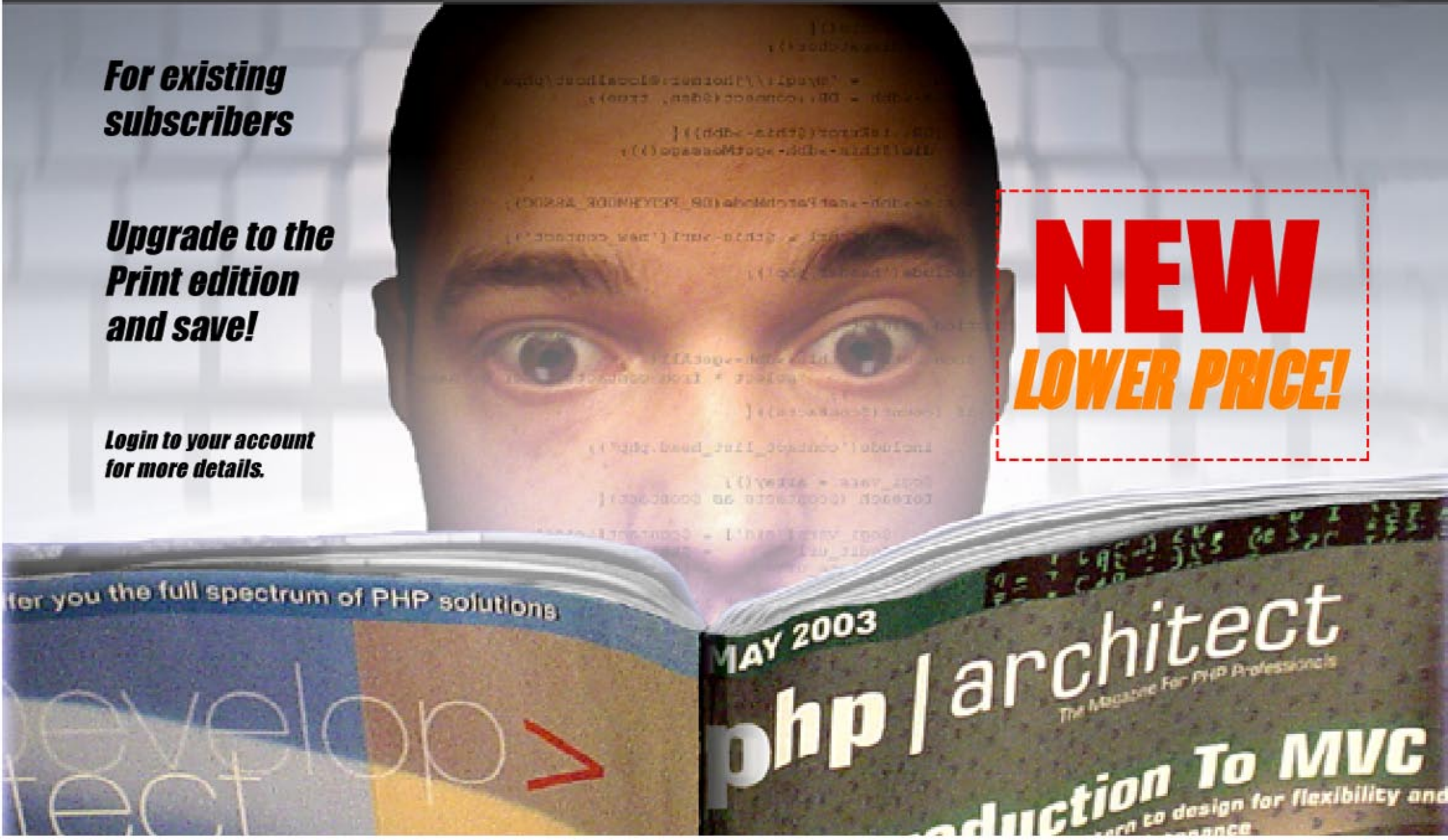
Next Steps

As exciting as these components are, the Zend Framework aims to be much more than just a component library. Quite a few developers are working hard (and being paid) to integrate these components into a full software stack that will include an MVC architecture with many tools for developing and deploying applications. While it's still early, a tremendous amount of progress has been made and the number of industry partners involved is rapidly turning the Zend Framework's vision into reality.

The PHP community has also proven itself to be invaluable and many members of the community have made significant contributions. Readers are very strongly encouraged to join the mailing list, contact Zend, and get involved with this industry-wide effort that has already begun to shape the future landscape of PHP development. ■

MIKE NABEREZNY is a senior software engineer for Zend Technologies in Cupertino, CA, where he develops the Zend Framework and other products. He blogs about PHP and related technologies on his website, *http://mikenaberezny.com*.

You'll never know what we'll come up with next


For existing subscribers
Upgrade to the Print edition and save!
Login to your account for more details.
NEW
LOWER PRICE!
fer you the full spectrum of PHP solutions
MAY 2003
php|architect
The Magazine For PHP Professionals
duction To MVC

php|architect
The Magazine For PHP Professionals

Visit: http://www.phparch.com/print for more information or to subscribe online.

php|architect Subscription Dept.
P.O. Box 54526
1771 Avenue Road
Toronto, ON M5M 4N5
Canada

Your charge will appear under the name "Marco Tabini & Associates, Inc." Please allow up to 4 to 6 weeks for your subscription to be established and your first issue to be mailed to you.

*US Pricing is approximate and for illustration purposes only.

NEW LOWER PRICE!

Name: ______________________

Address: ______________________

City: ______________________

State/Province: ______________________

ZIP/Postal Code: ______________________

Country: ______________________

Payment type:
VISA □ Mastercard □ American Express □

Credit Card Number: ______________________

Expiration Date: ______________________

E-mail address: ______________________

Phone Number: ______________________

Choose a Subscription type:

□ Canada/USA	$ 77.99 CAD	($59.99 US*)
□ International Air	$105.19 CAD	($80.89 US*)
□ Combo edition add-on (print + PDF edition)	$ 14.00 CAD	($10.00 US)

Signature: **Date:**

*By signing this order form, you agree that we will charge your account in Canadian dollars for the "CAD" amounts indicated above. Because of fluctuations in the exchange rates, the actual amount charged in your currency on your credit card statement may vary slightly.

To subscribe via snail mail - please detach/copy this form, fill it out and mail to the address above or fax to +1-416-630-5057