Dynamic
Languages World
Europe 2008

Shahar Evron | Zend Technologies

Zend Framework Components

...for non-framework development

The PHP Company

Agenda

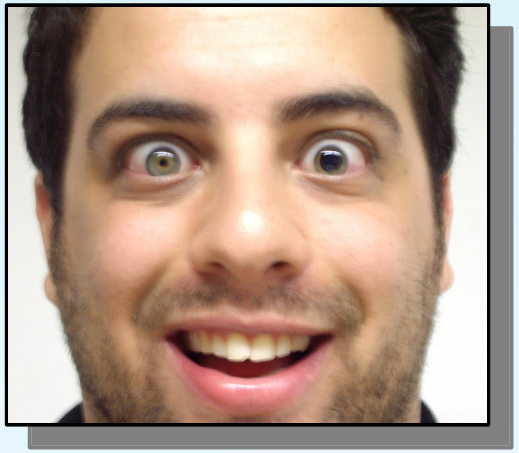
- Zend Framework's „Use At Will“ approach
- So why would I want to use Zend Framework?
- Show Us The Code! *(the interesting part)*
  - Zend_Mail
  - Zend_Db
  - Zend_Config
  - Zend_Http_Client
  - Zend_Service
  - Zend_Json
  - Zend_Log
- Some other things worth looking at

Prologue

- Who Are You?
  - PHP Programmers / Users?
  - Using Zend Framework?
  - Using other PHP Frameworks?
  - Using PEAR classes in your code?
  - Tend to reuse other people's code?

Prologue

- Who am I?
  - Programming in PHP for ~6 years
  - Working for Zend for the last 3 years
    - Services, Product Management
  - Part-time contributor to ZF
    - Not a part of my actual job ;-)
  - If you have a hard time with my name:
    - German: Shachar
    - Spanish: Shajar
    - Russian: Шахар
    - Arabic: شخر
    - Hebrew: שָׁחַר



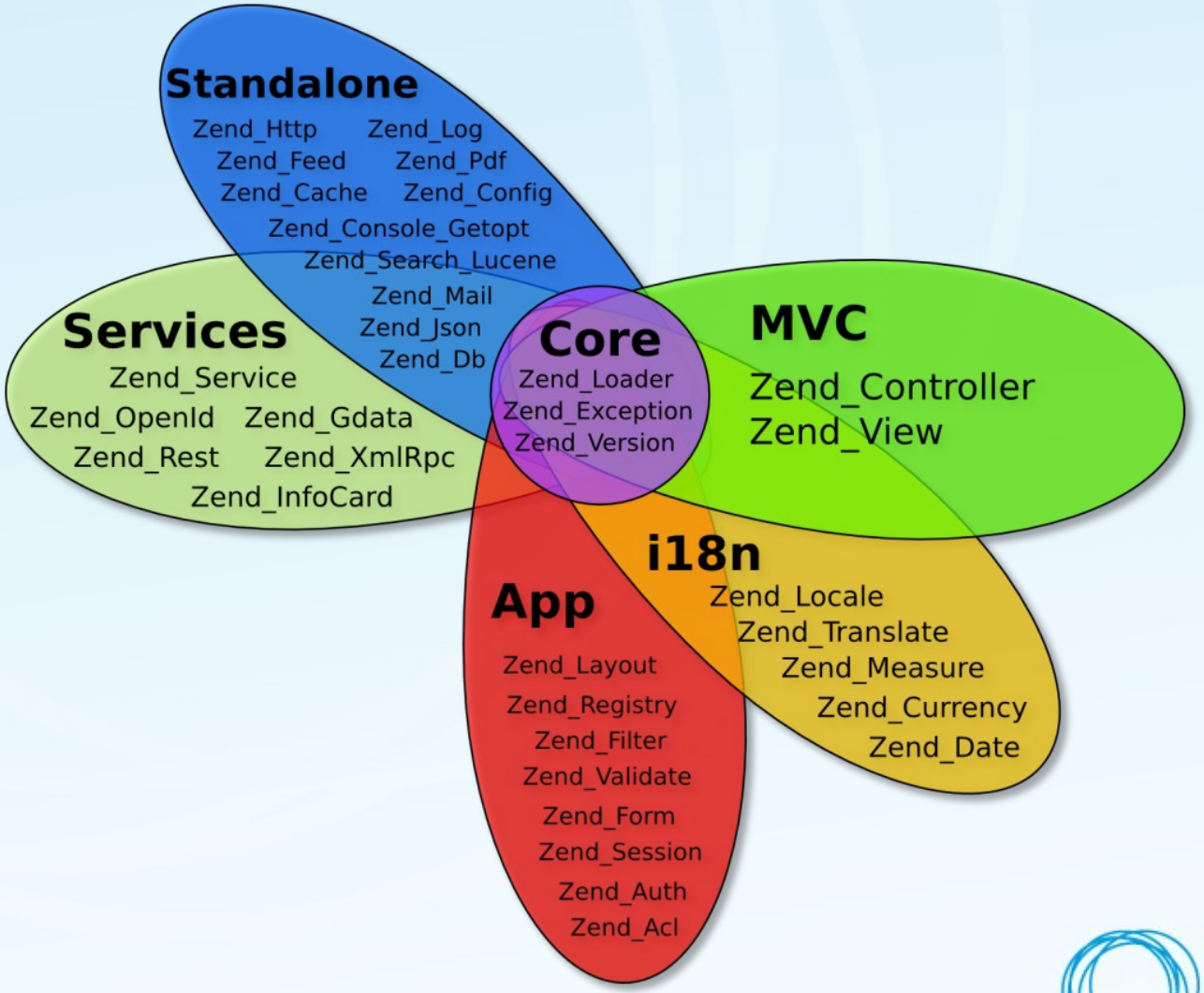
Dynamic
Languages World
Europe 2008

Why Zend Framework?

Fire At Will!

- Zend Framework embraces a **use-at-will** philosophy
- In practice, this means we try to have little or no dependencies between components
  - You don't have to use the entire framework
  - You don't have to build your application in a particular way
  - ...But if you want to, you can!
- There are some components (mostly the MVC stuff) that have loose bindings to other components
- Many components have very little dependencies, and can simply be *includ*ed and used in any application

Ooohh... Pretty Colors!



If No Framework, Why Zend Framework?

- Because it's high quality
  - Strict standards from day one
  - Unit Tested (80%+ code coverage)
  - Used and reviewed by a large community of users
- Because it's clean IP and friendly licensed
- Because it's there!
  - Well, it's one good option – you have others of course ;)

Dynamic
Languages World
Europe 2008

Ok, we get it,
Now show us the Code!

Zend_Mail

Zend_Mail

- Allows you to easily compose and send e-mail messages from your PHP applications
- Simplified control over recipients, headers, etc.
- Easy creation of multipart/alternative HTML messages
- Easy attachment handling
- Supports different delivery transports
- Allows you to read e-mail messages from POP3, IMAP, Mbox and Maildir (not discussed here)

Zend_Mail

Sending a simple message through Zend_Mail:

```php
// Load the Zend_Mail class
require_once 'Zend/Mail.php';

// Instantiate a new message object
$message = new Zend_Mail('utf-8');

// Use the fluent interface to set message properties
$message->setFrom('shahar.e@zend.com', 'Shahar E')
        ->setSubject('Hello, world!')
        ->setBodyText("Hi, what's up?")
        ->addHeader('Importance', 'high')
        ->addTo('someone@example.com', 'Some One')
        ->addCc('other.guy@example.com', 'Other Guy')
        ->addBcc('t.person@example.com', 'The Third Person');

// Send the message!
$message->send();
```

Zend_Mail

Creating `multipart/alternative` messages

```php
// ...Load the Zend_Mail and prepare the message object as before...

$text = <<<WELCOME_MSG
Hello Shahar,

*Thank you* for registering to my new site.
You are welcome to log in at http://example.com/log-in

The example.com team!
WELCOME_MSG;

// Set the plain text and HTML body parts
$message->setBodyText($text);
$message->setBodyHtml(nl2br(preg_replace(
          array('/\*(.+?)\*/', '|https?://\S+|'),
          array('<strong>\1</strong>', '<a href="\0">\0</a>'),
          $text)));

// Send the message!
$message->send();
```

Zend_Mail

Creating `multipart/alternative` messages - result:

```
Subject: Hello, world!
From: "Shahar E" <shahar.e@zend.com>
Importance: high
Content-Type: multipart/alternative; charset="utf8";
 boundary="=_96d67bd80a67ddc3fa0d70097a48504e"
MIME-Version: 1.0

--=_96d67bd80a67ddc3fa0d70097a48504e
Content-Type: text/plain; charset="utf8"
Content-Transfer-Encoding: quoted-printable

Hello Shahar,=0A=0A*Thank you* for registering to my new site. =0AYou are=
 welcome to log in at http://example.com/log-in=0A=0AThe example.com team!

--=_96d67bd80a67ddc3fa0d70097a48504e
Content-Type: text/html; charset="utf8"
Content-Transfer-Encoding: quoted-printable

Hello Shahar,<br />=0A<br />=0A<strong>Thank you</strong> for registering=
 to my new site. <br />=0AYou are welcome to log in at <a href=3D"http://e=
xample.com/log-in">http://example.com/log-in</a><br />=0A<br />=0AThe exam=
ple.com team!

--=_96d67bd80a67ddc3fa0d70097a48504e--
```

Zend_Mail

Attaching files

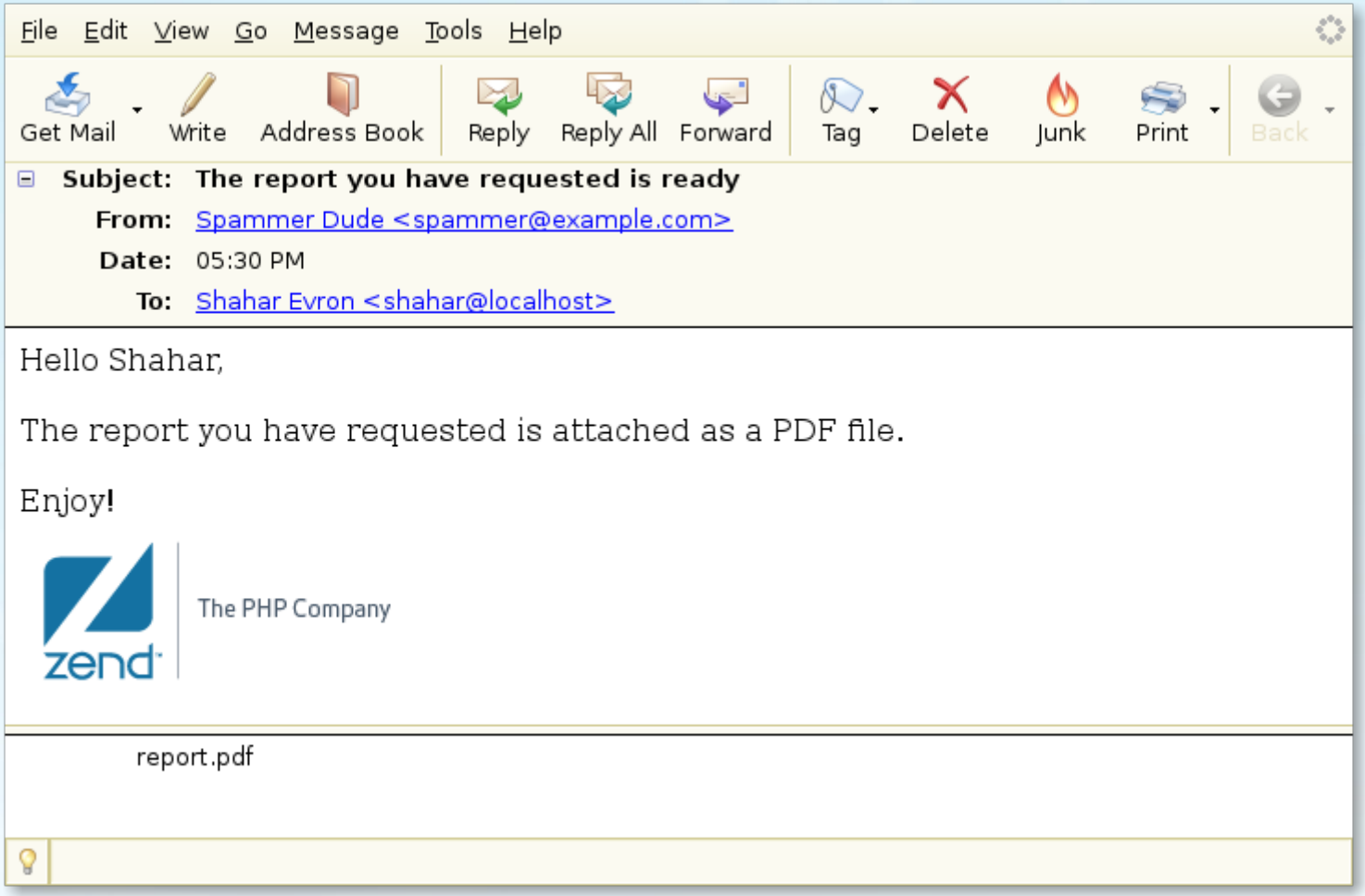
```php
// Load the Zend_Mail class
require_once 'Zend/Mail.php';

// Instantiate a new message object
$message = new Zend_Mail('utf-8');

// Use the fluent interface to set message properties
$message->setFrom('spammer@example.com', 'Spammer Dude')
        ->setSubject('The report you have requested is ready')
        ->addTo('shahar@localhost', 'Shahar Evron');

// Add a PDF attachment (Will be base64 encoded)
$pdf = $message->createAttachment(file_get_contents('report.pdf'));
$pdf->type     = 'application/pdf';
$pdf->filename = 'report.pdf';

// ... Continued on next slide ...
```

Zend_Mail

Attaching files (contd.)

```
// Add a logo to the message - referenced from the message HTML body
$img = $message->createAttachment(file_get_contents('logo.png'));
$img->type        = 'image/png';
$img->id          = 'logo-image-png@example.com';
$img->filename    = 'logo.png';
$img->disposition = Zend_Mime::DISPOSITION_INLINE;

// Set the message body
$message->setBodyHtml(
          "Hello Shahar,<br /><br />" .
          "The report you have requested is attached as a PDF file.<br /><br />" .
          "Enjoy!<br />" .
          '<img src="cid:logo-image-png@example.com" alt="Example.com Logo" />'
);

// Set the message MIME-type to multipart/related
$message->setType(Zend_Mime::MULTIPART_RELATED);

// Send the message!
$message->send();
```

Zend_Mail


File Edit View Go Message Tools Help
Get Mail
Write
Address Book
Reply
Reply All
Forward
Tag
Delete
Junk
Print
Back
Subject: The report you have requested is ready
From: Spammer Dude <spammer@example.com>
Date: 05:30 PM
To: Shahar Evron <shahar@localhost>
Hello Shahar,
The report you have requested is attached as a PDF file.
Enjoy!
zend
The PHP Company
report.pdf

Zend_Mail

Sending multiple e-mails efficiently

- Using the PHP *mail()* function is bad for multiple e-mails
- On each mail you send, you:
  - Fork a *sendmail* process
  - *sendmail* will open a connection to the SMTP server, send the message and close the connection
  - The *sendmail* process is then closed
- This is why Zend_Mail provides the SMTP transport
  - A full implementation of an SMTP client in PHP
  - Allows you to open a connection once and reuse it to send several messages
  - No forking is needed, does not depend on *sendmail*

Zend_Mail

Sending multiple e-mails efficiently

```php
// Load the Zend_Mail and Zend_Mail_Transport_Smtp classes
require_once 'Zend/Mail.php';
require_once 'Zend/Mail/Transport/Smtp.php';

// Set up a transport using smtp.example.com as SMTP server
$smtpConn = new Zend_Mail_Transport_Smtp('smtp.example.com');

// ... $msg1, $msg2 and $msg3 are Zend_Mail message objects

// Send a bunch of e-mail messages
$msg1->send($smtpConn);
$msg2->send($smtpConn);
$msg3->send($smtpConn);
```

Zend_Db_Adapter
Zend_Db_Select

Zend_Db_Adapter and Zend_Db_Select

- Zend_Db_Adapter provides an abstract, vendor-agnostic RDBMS connection layer
  - Mostly similar in concept and in interface to PDO
  - Actually, it will usually use PDO under the hood
  - But can also use other PHP RDBMS client extensions
  - It also provides some convenience methods for escaping and running queries
- Zend_Db_Select provides even better vendor-independence by abstracting SELECT SQL queries
  - This means you (almost) don't have to write any SQL code!

Zend_Db_Adapter

Connecting to a MySQL database using the factory method

```php
require_once 'Zend/Db.php';

// Connecting to MySQL through MySQLi
$config = array(
    'adapter'  => 'MYSQLI',
    'hostname' => 'localhost',
    'dbname'   => 'wiki_db',
    'username' => 'wiki_db_user',
    'password' => 'xxxxxx'
);

// Use the factory method to connect to the DB
$db = Zend_Db::factory($config['adapter'], $config);
```

Zend_Db_Adapter

Connecting to an SQLite DB and inserting some data

```php
// Same connect method – but this time to SQLite 3.x
$config = array(
    'adapter'  => 'PDO_SQLITE',
    'dbname'   => 'var/data/dbfile.sq3'
);

$db = Zend_Db::factory($config['adapter'], $config);

// Insert a new row to the articles table
$db->insert('articles', array(
    'created_by' => (int) $user_id,
    'created_at' => $_SERVER['REQUEST_TIME'],
    'title'      => $_POST['title'],
    'content'    => $_POST['content']
));
```

Zend_Db_Select

Fetching data from the DB using Zend_Db_Select

```php
// Create the SELECT query
$select = $db->select()
              ->from('articles', array('id', 'title'))
              ->join('users', 'articles.author = users.id',
                  array('name', 'email'))
              ->where('links_to', 5)
              ->order('articles.title')
              ->limit(16);

// Execute the query, return a result set object
$results = $select->query();

--- Produced SQL query: ---
SELECT "articles"."id", "articles"."title", "users"."name",
"users"."email" FROM "articles" INNER JOIN "users" ON articles.author =
users.id WHERE (links_to = 5) ORDER BY "articles"."title" ASC LIMIT 16
```

Zend_Config

Zend_Config

- The purpose of Zend_Config is to allow easy management of and access to configuration data
- Several configuration storage formats are supported
  - Native PHP arrays
  - INI or XML files
  - Adding support for other formats is easy!
- Supports some cool features as well
  - Nesting configuration
  - Inheritance between configuration sections
  - Default values

Zend_Config

Sample INI file used in next slide (config.ini)

```
[ production ]
debug = Off
path.web.root = /
path.web.host = example.com

db.adapter  = MYSQLI
db.hostname = 10.1.2.3
db.username = produser
db.password = xxxxxxxx
db.dbname   = wikidb

[ development : production ]
debug = On
path.web.host = dev.example.com
db.adapter  = PDO_SQLITE
db.dbname   = wikidb.sq3
```

Zend_Config

Using Zend_Config_Ini to manage configuration data

```php
$env  = getenv('MYAPP_ENVIRONMENT');
if (! $env) $env = 'production';

// Load configuration data from file
$cfg = new Zend_Config_Ini('config.ini', $env);

// Connect to DB using our configuration settings
$db = Zend_Db::factory($cfg->db->adapter, $cfg->db->toArray());

// Will always print '/' regardless of environment
echo $cfg->path->web->root;

// Will be true or false - depending on environment
var_dump($cfg->debug);

// Fetching with default value
$hostname = $cfg->path->web->get('host', $_SERVER['HOST_NAME']);
```

Zend_Http_Client

Zend_Http_Client

- A PHP-implemented HTTP client
  - Can be used to pass data around using HTTP
  - Serves as the basis for web-service client components in ZF
- Does not require any extensions (CURL, pecl_http)
- Does not require allow_url_fopen
- Supports some advanced HTTP features
  - Cookie persistence
  - File uploads
  - Proxy, authentication, HTTPS over proxy

Zend_Http_Client

Reusing cookies over consecutive requests

```php
// Set up the client and enable cookie persistance
$client = new Zend_Http_Client('https://www.example.com/user/login');
$client->setCookieJar();

// Set the login parameters
$client->setParameterPost(array(
    'username' => 'shahar.e',
    'password' => 'xxxxxxxxx'
));

// Send a POST request
$response = $client->request(Zend_Http_Client::POST);

// Send a GET request to fetch the restricted content
$client->resetParameters();
$client->setUri('http://www.example.com/restricted');
$response = $client->request(Zend_Http_Client::GET);
```

Zend_Http_Client

Sending files over HTTP using POST file upload

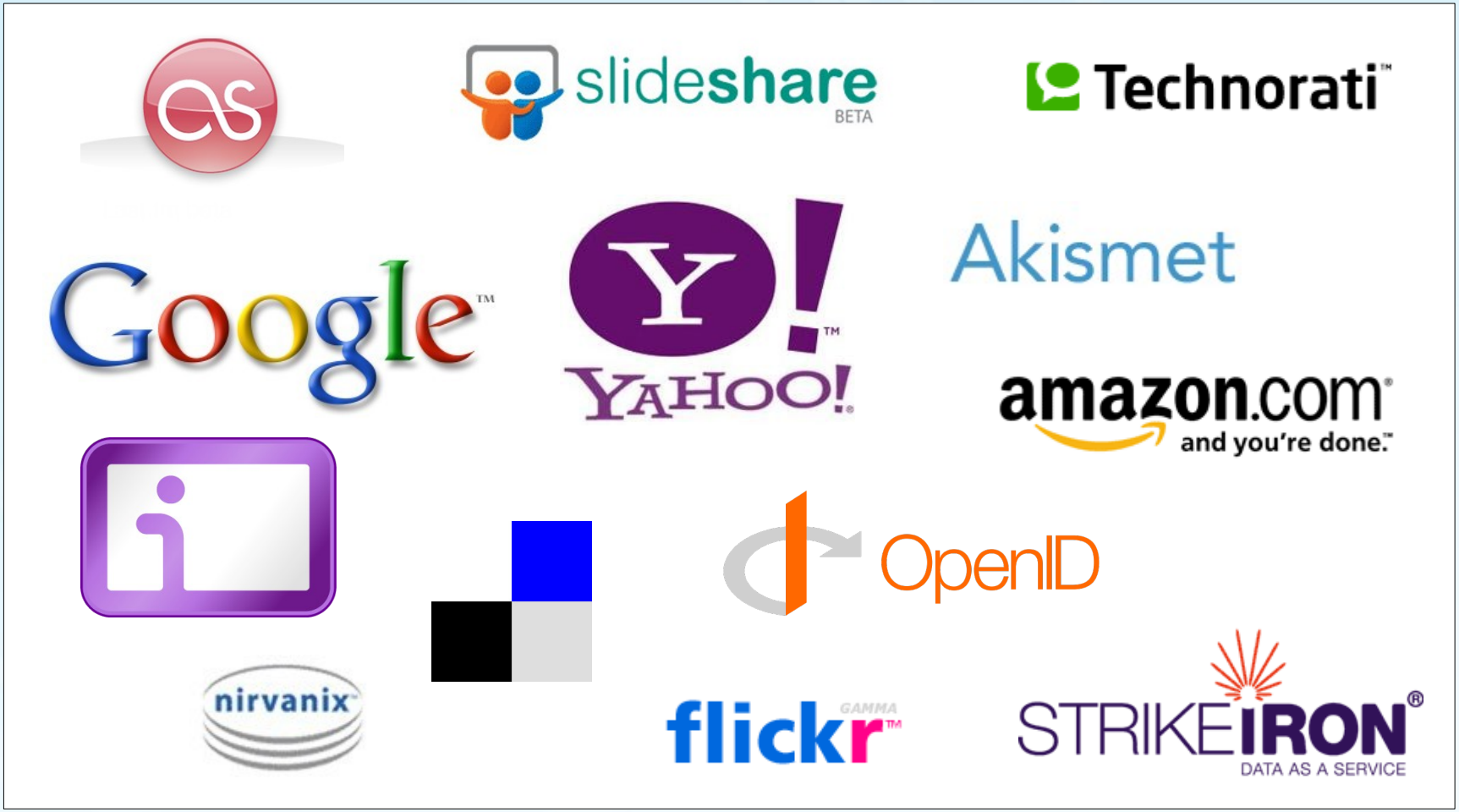
```
// Set up the client and enable cookie persistance
$client = new Zend_Http_Client('http://www.example.com/upload-photo');

// Add a file from disk
$client->setFileUpload('/home/shahar/photos/image1.jpg', 'photo1');

// Add a file from buffer
$client->setFileUpload('mock-file.jpg', 'photo2', $image_data,
    'image/jpg');

// Send files
$response = $client->request();

// Files will be available on server-side PHP as
// $_FILES['photo1'] and $_FILES['photo2']
```

Zend_Service

Zend_Service

- Zend Framework aims to provide easy-to-use web service clients for many web service providers
  - Yahoo!
  - Flickr
  - AudioScrobbler
  - Amazon
  - ...and many others
- New service clients are added on every release
- There are also the Gdata and Infocard clients
- There are also generic REST and XMLRPC clients

Zend_Service



Zend_Service_Akismet

Checking a blog post comment for spam using Akismet

```php
require_once  'Zend/Service/Akismet.php';

// Create the client object using the API key and blog URL
// Get your API key from http://akismet.com/ (free for personal use)
$akismet = new Zend_Service_Akismet($apiKey, 'http://example.com/blog');

// Check a blog post comment for spam
$commentData = array(
    'user_ip'         => $_SERVER['REMOTE_ADDR'],
    'user_agent'      => $_SERVER['HTTP_USER_AGENT'],
    'comment_type'    => 'trackback',
    'comment_author'  => 'Honest Bender',
    'comment_email'   => 'fake123@hotmail.com',
    'comment_content' => "Free pr0n and cheap Rolex!"
);
if ($akismet->isSpam($commentData)) {
    echo "Die, Evil Spammer!!!";
} else {
    echo "Thanks for your comment!";
}
```

Zend_Service_Akismet

Reporting new spam or false positives to Akismet

```php
$commentData = array(
    'user_ip'          => $_SERVER['REMOTE_ADDR'],
    'user_agent'       => $_SERVER['HTTP_USER_AGENT'],
    'comment_type'     => 'trackback',
    'comment_author'   => 'Honest Bender',
    'comment_email'    => 'fake123@hotmail.com',
    'comment_content' => "Free pr0n and cheap Rolex!"
);

// Report new spam to Akismet
$akismet->submitSpam($commentData);

// Report false positive (ham) to Akismet
$akismet->submitHam($commentData);
```

Zend_Json

Zend_Json

- Allows to easily encode PHP data as JSON* and vice versa
- Can encode / decode PHP scalars, arrays and objects
- Will utilize ext/json if available – if not, will use the written-from-scratch JSON encoder and decoder

* JavaScript Object Notation, http://www.json.org/

Zend_Json

Encoding PHP to JSON

```php
require_once 'Zend/Json.php';

$article = array(
    'title'     => 'How to make a good script great',
    'author'    => 'Some Guy',
    'published' => time(),
    'tags'      => array('howto', 'scripting', 'script', 'guide')
);

// Encode the PHP array
echo Zend_Json::encode($article);


--- Output ---
{"title":"How to make a good script great","author":"Some Guy",
"published":1211837898,"tags":["howto","scripting","script","guide"]}
```

Zend_Json

Decoding JSON to PHP

```
// Decode JSON. Will decode to an associative array by default
$article = Zend_Json::decode($json);

--- Output ---
stdClass Object
(
    [title] => How to make a good script great
    [author] => Some Guy
    [published] => 1211837898
    [tags] => Array
        (
            [0] => howto
            [1] => scripting
            [2] => script
            [3] => guide
        )
)
```

Zend_Json

Decoding JSON to PHP

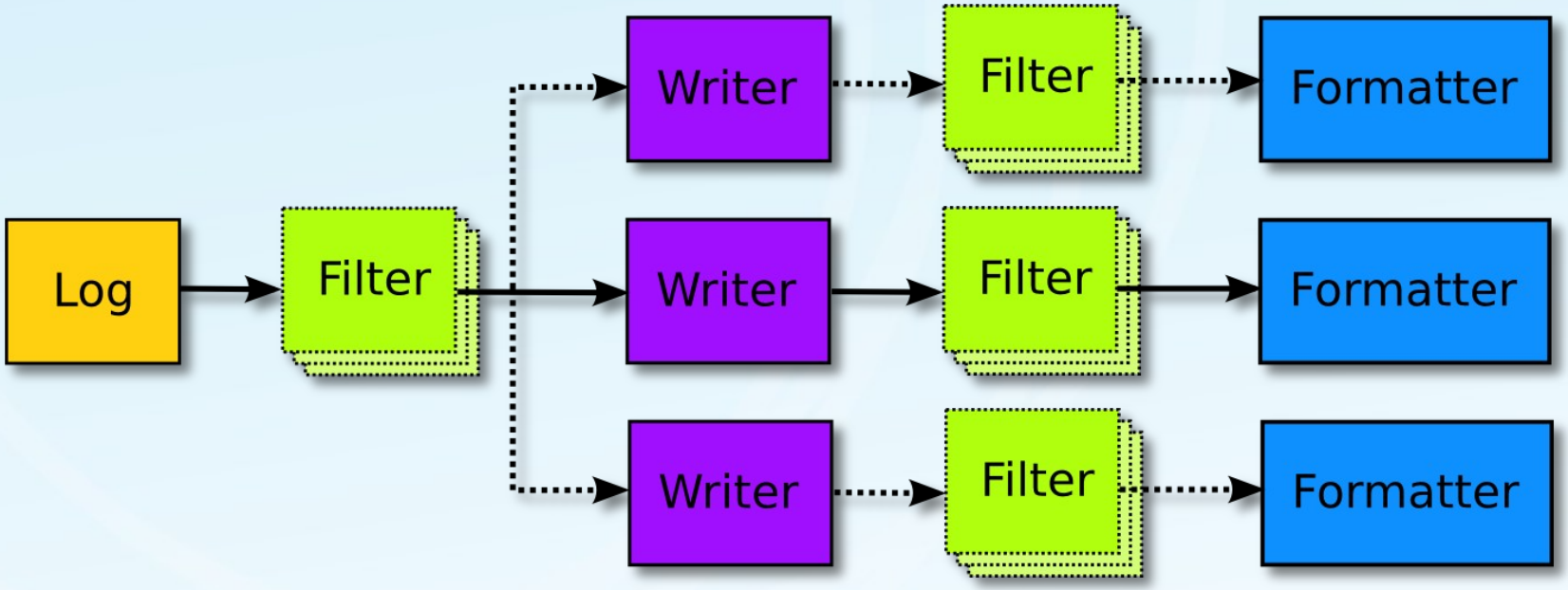
```
// Decode JSON to stdClass object
$article = Zend_Json::decode($json, Zend_Json::TYPE_OBJECT);

--- Output ---
Array
(
    [title] => How to make a good script great
    [author] => Some Guy
    [published] => 1211837898
    [tags] => Array
        (
            [0] => howto
            [1] => scripting
            [2] => script
            [3] => guide
        )
)
```

Zend_Log

Zend_Log

- Provides powerful yet easy to use logging capabilities
- Supports different “writers” - can write to several backends (at the same time even!)
- Uses a default set of error levels, you can add more
- You can apply filters when logging – this allows, for example, to dynamically change the logging level
- You can apply formatters on writers – this allows advanced message formatting - for example XML

Zend_Log


Log
Filter
Writer
Filter
Formatter
Writer
Filter
Formatter
Writer
Filter
Formatter

Zend_Log

Quick Start: Logging to standard output

```php
require_once 'Zend/Log.php';
require_once 'Zend/Log/Writer/Stream.php';

// Create a simple log that will write to STDOUT
$log = new Zend_Log(new Zend_Log_Writer_Stream('php://stdout'));

// Log a warning message
$log->log('Something might be wrong', Zend_Log::WARN);

// A simpler way to do the same thing
$log->warn('Something might be wrong');

Output:
2008-05-27T01:15:56+03:00 WARN (4): Something might be wrong
```

Zend_Log

- You can apply one or more Filter objects to both
  - The Log object – this will affect all writers
  - A particular Writer object – this will affect only one write
- Several filters exist, and you can easily create your own filter classes
- The most common method of filtering messages is by priority
  - This lets you control the logging verbosity level at run time, without changing your code

Zend_Log

- Default priorities correlate to the *syslog* standards:
  - EMERG (0)
  - ALERT (1)
  - CRIT (2)
  - ERR (3)
  - WARN (4)
  - NOTICE (5)
  - INFO (6)
  - DEBUG (7)
- You can add your own if needed

Zend_Log

Filtering messages according to priority

```php
$log = new Zend_Log();

// Log all messages to a debugging log
$log->addWriter(new Zend_Log_Writer_Stream('debug.log'));

// Log only important messages to STDERR
$stdErr = new Zend_Log_Writer_Stream('php://stderr');
$stdErr->addFilter(new Zend_Log_Filter_Priority(Zend_Log::WARN));
$log->addWriter($stdErr);

// Log an informational message - will only appear in debug.log
$log->info('The application is starting up');

// Log an error message - will appear in both logs
$log->err('Something very bad has happened');
```

Dynamic Languages World Europe 2008

Epilogue

Some more components worth looking at

- Zend_Pdf
- Zend_Search_Lucene
- Zend_Memory
- Zend_Auth
- Zend_Acl
- Zend_Console_Getopt
- I18N and L10N components
- ... and the MVC parts as well ;)

Got Questions?

- Questions?
- This presentation will be available at:
  - http://prematureoptimization.org/
  - Slideshare: http://www.slideshare.net/shahar
- Feedback and questions: shahar.e@zend.com

Dynamic
Languages World
Europe 2008

Thank You!