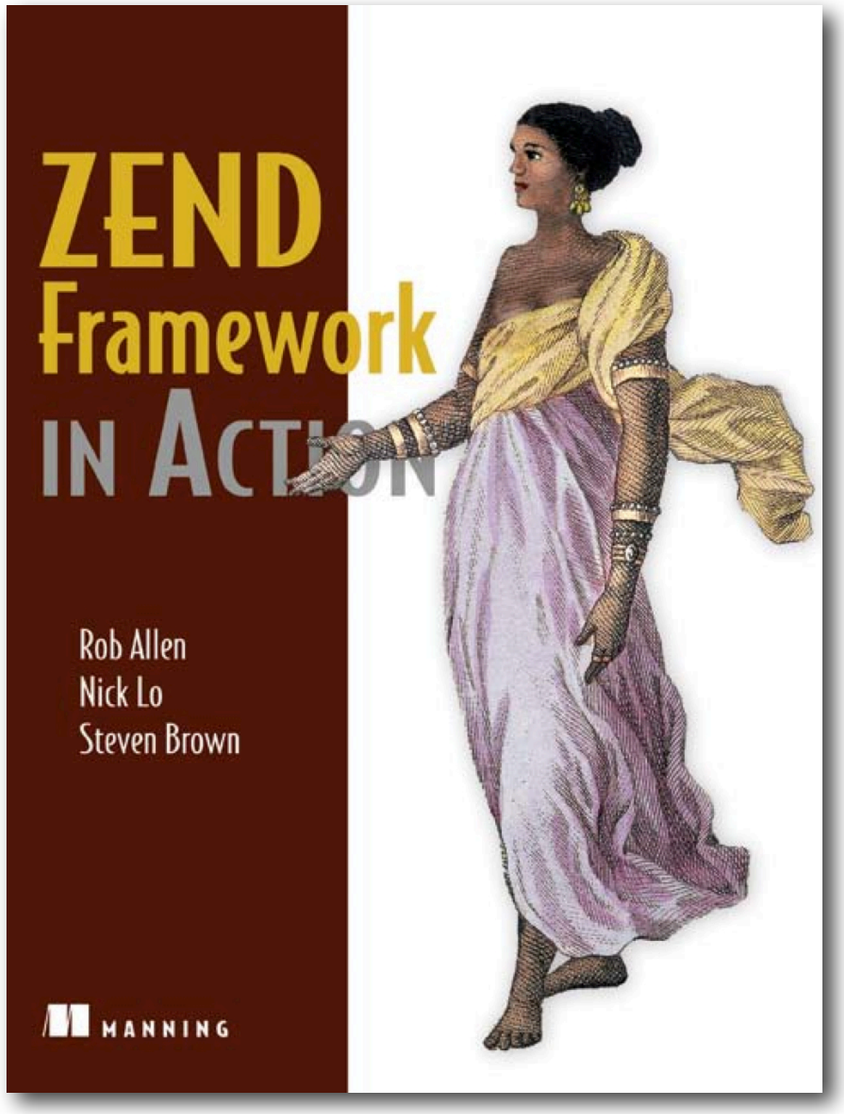
Zend Framework Certification

(a refresher)

Rob Allen

Rob Allen?


php


ZEND
Framework
IN ACTION
Rob Allen
Nick Lo
Steven Brown
MANNING

Zend Framework Certification?

The exam

- 75 questions
- 1.5 hours
- Multiple choice & fill in the blank

What you need to know

- Zend Framework 1.5
- http://framework.zend.com/releases/ZendFramework-1.5.3/ZendFramework-1.5.3.zip
- Manual: http://framework.zend.com/releases/ZendFramework-1.5.3/ZendFramework-1.5.3-manual-en.zip

What you need to know

- Know your configuration options
- Know the available constants
- If there are interfaces involved, learn them!
- Know method names
- If there is more than one way to do something, know them all!

What's not in it?

- To be clear: ZF 1.5, not 1.9!
- No Zend_Application, Zend_Dojo, etc
- No writing reams of code
- Questions are one component each

The curriculum

The curriculum

- Authentication & authorisation
- Coding conventions
- Database
- Diagnostics & maintenance
- Filtering & validation
- Infrastructure
- Internationalisation
- Web services
- Search
- Forms
- Mail
- MVC
- Performance
- Security

Authentication & Authorisation

Sample question

The result of a Zend_Auth adapter's `authenticate` method is:

a. an instance of `Zend_Auth_Result`
b. an instance of `Zend_Auth_Authentication`
c. a boolean
d. a string

Zend_Auth

- Singleton: Zend_Auth::getInstance()
- Adapters
  - implement Zend_Auth_Adapter_Interface
  - authenticate() returns a Zend_Auth_Result
  - operations on a result:
    - getCode(), getIndentity(), getMessages(), isValid()

Zend_Auth

- Persistence
  - Zend_Auth_Storage_Interface
  - Default: Zend_Auth_Storage_Session
- Authenticate:
  - Zend_Auth::authenticate() or use adapter directly

Zend_Acl

- Jargon:
  - role: grouping of users
  - resource: something to be protected
  - privilege: type of access required
- Resource:
  - implement Zend_Acl_Resource_Interface
  - Single inheritance

Zend_Acl

- Role:
  - implement Zend_Acl_Role_Interface
  - multiple inheritance
- Privilege:
  - just a string

Zend_Acl

```
$acl = new Zend_Acl();

$acl->add(new Zend_Acl_Resource('rsrc'));

$roleGuest = new Zend_Acl_Role('guest');
$acl->addRole($roleGuest);
$acl->addRole(new Zend_Acl_Role('staff'), $roleGuest);
$acl->allow($roleGuest, null, 'view');
$acl->allow('staff', 'rsrc', array('edit', 'create'));
$acl->deny('staff', 'rsrc', 'delete');

echo $acl->isAllowed('guest', null, 'view') ?
        "allowed" : "denied";
```

Coding conventions

Sample question

The recommended length of a code line is:

a. 120 characters
b. 100 characters
c. 80 characters
d. 60 characters

Code style

- don't use ?> for only PHP code
- 4 spaces, not tabs
- 80 characters
- line termination is \n (0x0A)
- no short tags
- control statements:

```
if ($x == 1) {
    $x ++;
} else if ($x == 2) {
} else {
}
```

Class naming

- Class name map to directory
- App_Test_456 => App/Test/456.php
- Multiple words: Zend_XmlRpc
- Filenames: alphanumeric, underscore, hyphen
- Interfaces: ends with the word _Interface
- Abstract classes: end with _Abstract

Method/variable names

- Name: alphanumeric, underscore, hyphen
- starts with lowercase letter
- multi-word: camelCase
- prefix for accessors: `get` or `set`
- private or protected: start with underscore

Constants

- Name: alphanumeric, underscore
- ALL_CAPS
- Only in classes using `const`

Databases

Sample question

Zend_Db::quote() is used to quote a table name so that it is safe to use in an SQL statement:

a. True
b. False

Zend_Db

- Adapters connect to the database.
- Construct directly or `Zend_Db::factory()`
- Zend_Db::factory('Pdo_Mysql', $params);
- lazy connection: force with `getConnection()`

Zend_Db

```
$sql = 'SELECT * FROM news WHERE id = ?';
$result = $db->fetchAll($sql, 2);
```

Quotes:

```
$tableName = $db->quoteIdentifier('news');
    `news`
```

Zend_Db

```
$string = $db->quote("This'll work");
```

```
'This''ll work'
```

```
$sql = $db->quoteInto("SELECT * FROM news
    WHERE title = ?", "This'll work");
```

```
SELECT * FROM bugs
    WHERE title = 'This''ll work'
```

Zend_Db

```
$data = array(
    'title' => 'Test',
    'body' => 'Something interesting'
);
$db->insert('news', $data);

$n = $db->update('news', $data, 'id = 2');

$n = $db->delete('news', 'id = 3');
```

Zend_Db_Statement

```
$sql = 'SELECT * FROM news WHERE
    category = ? AND published = ?';
$stmt = new Zend_Db_Statement_Mysqli($db, $sql);
$stmt->execute(array('current', 1));
```

```
$stmt = $db->query('SELECT * FROM news');
while ($row = $stmt->fetch()) {
    echo $row['title'];
}
// or
$rows = $stmt->fetchAll();
```

Zend_Db_Select

```
$select = $db->select();
$select = $db->select()
    ->from('news', 'news.title, news.id')
->join(array('i' => 'images'),
    'news.id' => 'i.parent_id')
->where('id > 100')
->order('date_published DESC')
->limit(20);
```

Zend_Db_Table

```
class News extends Zend_Db_Table_Abstract
{
    protected $_name = 'news';
    protected $_primary = 'news_id';

    protected function init()
    {
    }
}


Zend_Db_Table_Abstract::setDefaultAdapter($db);
$newsTable = new News(array('db'=>'db_one'));
```

Zend_Db_Table

```
$table = new News();

$data = array('title'=> 'item one',
    'publish_date' => new Zend_Db_Expr('NOW()'));
$table->insert($data);
```

```
$data = array(
    'title' => 'Fixed item one'
);
$where = $table->getAdapter()->quoteInto('id = ?', 1);
$table->update($data, $where);


$table->delete($where);
```

Zend_Db_Table

```
$rows = $table->find(1);  // returns a rowset
$theRow = $rows->current();

$select = $table->select();
$select->where('id < ?', 100);
```

```
$rows = $table->fetchAll($select);  // returns a rowset

$row = $table->fetchRow($select);   // returns a row

$initialData = array('author'=>1);
$newRow = $table->createRow($initialData);
$newRow->title = 'new';
$newRow->save();
```

Relationships

```
class Authors extends Zend_Db_Table_Abstract
{
    protected $_dependentTables = array('Posts');
}
```

```
class Posts extends Zend_Db_Table_Abstract
{
    protected $_referenceMap    = array(
        'Author' => array(
            'columns'           => 'created_by',
            'refTableClass'     => 'Authors',
            'refColumns'        => 'id'
        ));
}
```

One to many

```
$authorsTable = new Authors();
$rob = $authorsTable->find(1)->current();

$postsByRob = $rob->findDependentRowset('Posts');
// or
$postsByRob = $rob->findDependentRowset(new Posts());
// or
$postsByRob = $rob->findPosts();

// Other way
$postsTable = new Posts();
$post = $postsTable->find(3)->current();
$author = $post->findParentRow('Authors');
$author = $post->findParentAuthors();
```

Many to many

```
class PostsTags extends Zend_Db_Table_Abstract
{
    protected $_referenceMap    = array(
        'Posts' => array(
            'columns'           => 'post_id',
            'refTableClass'     => 'Posts',
            'refColumns'        => 'id'
        )
        'Tags' => array(
            'columns'           => 'tag_id',
            'refTableClass'     => 'Tags',
            'refColumns'        => 'id'
        ));
}
```

Many to many

```
$postTable = new Posts();
$post = $postTable->find(1)->current();

$tags = $post->findManyToManyRowset('Tags', 'PostsTags');
// or
$tags = $post->findTagsViaTagsPosts();
```

Diagnostics & Maintenance

Sample question

Which of the following is NOT a Zend_Log writer:

a. `Zend_Log_Writer_Db`

b. `Zend_Log_Writer_Xml`

c. `Zend_Log_Writer_Stream`

d. `Zend_Log_Writer_Mock`

Zend_Log

- Writers
  - extend `Zend_Log_Writer_Abstract`
- Filters
  - implement `Zend_Log_Filter_Interface`
- Formatters
  - implement `Zend_Log_Formatter_Interface`

Zend_Log

```
$writer = new Zend_Log_Writer_Stream('php://output');
$logger = new Zend_Log($writer);

$logger->log('Informational message', Zend_Log::INFO);
$logger->info('Informational message');
```

```
EMERG   = 0;  // Emergency: system is unusable
ALERT   = 1;  // Alert: action must be taken immediately
CRIT    = 2;  // Critical: critical conditions
ERR     = 3;  // Error: error conditions
WARN    = 4;  // Warning: warning conditions
NOTICE  = 5;  // Notice: normal but significant condition
INFO    = 6;  // Informational: informational messages
DEBUG   = 7;  // Debug: debug messages
```

Zend_Log writers

`Zend_Log_Writer_Stream`: write to a file or PHP output buffer
`Zend_Log_Writer_Db`: write to a database table
`Zend_Log_Writer_Null`: disable logging (e,g, during tests)
`Zend_Log_Writer_Mock`: write to a public var for unit testing

```
$writer1 = new Zend_Log_Writer_Stream('/path/to/logfile1');
$writer2 = new Zend_Log_Writer_Stream('/path/to/logfile2');

$logger = new Zend_Log();
$logger->addWriter($writer1);
$logger->addWriter($writer2);
```

Zend_Log formatters

Zend_Log_Formatter_Simple

```
'%timestamp% %priorityName% (%priority%): %message%' . PHP_EOL;
```

Zend_Log_Formatter_Xml

```
<logEntry>
  <timestamp>2007-04-06T07:24:37-07:00</timestamp>
  <message>informational message</message>
  <priority>6</priority>
  <priorityName>INFO</priorityName>
</logEntry>
```

Attach the formatter to the writer:

```
$writer = new Zend_Log_Writer_Stream('php://output');
$formatter = new Zend_Log_Formatter_Simple('%message%' . PHP_EOL);
$writer->setFormatter($formatter);
```

Zend_Log filters

```
$filter = new Zend_Log_Filter_Priority(Zend_Log::CRIT);


$logger->addFilter($filter);
// or
$writer->addFilter($writer1);


$logger->info('Informational message');  // blocked
$logger->emerg('Emergency message');  // logged
```

Zend_Debug

```
Zend_Debug::dump($var, 'label', $echo=true);
```

`$echo` controls if the output is echoed.

To set SAPI:

```
Zend_Debug::setSapi('cli');
```

Filtering & Validation

Sample question

What is the output of:

```
$filterChain = new Zend_Filter();
$filterChain->addFilter(new Zend_Filter_StripTags())
            ->addFilter(new Zend_Filter_StringTrim());
```

```
echo $filterChain->filter(' a <b>test </b>');
```

a. `' a test '`

b. `'a test '`

c. `'a test'`

d. `' a test'`

Zend_Filter

- Filters implement `Zend_Filter_Interface`
  - single method: `filter()`

```
$htmlEntities = new Zend_Filter_HtmlEntities();
echo $htmlEntities->filter('Tom & Jerry');
  // outputs: Tom &amp;amp; Jerry

// or

echo Zend_Filter::get('Tom & Jerry', 'HtmlEntities');
```

Zend_Filter

- Alnum
- Alpha
- BaseName
- Digits
- Dir
- HtmlEntities
- Int
- RealPath
- StringToLower
- StringToUpper
- StringTrim
- StripTags

Chaining filters

```
$filterChain = new Zend_Filter();
$filterChain->addFilter(new Zend_Filter_Alpha())
            ->addFilter(new Zend_Filter_StringToLower());

$username = $filterChain->filter($_POST['username']);
```

Filters are applied in this order:

1. `Zend_Filter_Alpha`
2. `Zend_Filter_StringToLower`

(`$username` is lowercase alphabetic characters only)

Zend_Filter_Input

```
$filters = array(
    '*'         => 'StringTrim',
    'account'   => 'StringToUpper',
);
$validators = array(
    'account' => array('Alpha', 'allowEmpty'=>true);
);

$data = array('account'=>' Abc 1234');
$input = new Zend_Filter_Input($filters, $validators, $data);
```

Zend_Filter_Input

```
if ($input->hasInvalid() || $input->hasMissing()) {
  $messages = $input->getMessages();
}
if ($input->hasUnknown()) {
  $unknownFields = $input->getUnknown();
}

// retrieve data:
if ($input->isValid()) {
    $account = $input->account; // escaped data
    $account = $input->getEscaped('account');
    $account = $input->getUnescaped('account');
}
```

Zend_Validate

- Implement `Zend_Validate_Interface`
  - two methods: `isValid()` & `getMessages()`

```
$validator = new Zend_Validate_EmailAddress();
$validEmail = $validator->isValid($email);
// or $validEmail = Zend_Validate::is($email, 'EmailAddress');

if ($validEmail) {
    // email is valid
} else {
  foreach($validator->getMessages()as $messageId => $message) {
    echo "Validation failure '$messageId': $message\n";
  }
}
```

Zend_Validate

- Alnum
- Alpha
- Barcode
- Between
- Ccnum
- Date
- Digits
- EmailAddress
- Float
- GreaterThan
- Hex
- Hostname
- InArray
- Int
- Ip
- LessThan
- NotEmpty
- Regex
- StringLength

Chaining validators

```
$chain = new Zend_Validate();
$chain->addValidator(new Zend_Validate_StringLength(6, 12), true)
                ->addValidator(new Zend_Validate_Alnum());
```

```
if ($chain->isValid($username)) {
  // valid username
}
```

Custom messages

```
$validator = new Zend_Validate_StringLength(8);
$validator->setMessage(
    'The string \'%value%\' must be at least %min% characters',
    Zend_Validate_StringLength::TOO_SHORT);
```

```
$validator->setMessages(array(
    Zend_Validate_StringLength::TOO_SHORT =>
        'The string \'%value%\' is too short',
    Zend_Validate_StringLength::TOO_LONG  =>
        'The string \'%value%\' is too long'
));
```

Infrastructure

Sample question

Which configuration file format is not supported by `Zend_Config`? (Choose 2)

a. XML
b. INI
c. JSON
d. YAML

Zend_Version

- Class constant: `Zend_Version::VERSION`
- Static method:

```
Zend_Version::compareVersion($version)
```

- Comparison

```
$cmp = Zend_Version::compareVersion('1.5.0');
  //  -1 => 1.5.0 is less than this version
  //   0 => same version
  //  +1 => 1.5.0 is greater than this version
```

Zend_Registry

- Static usage:

```
Zend_Registry::set('db' => $db);
$db = Zend_Registry::get('db');
```

- Registry object:

```
$registry = new Zend_Registry(
        array('db' => $db));

Zend_Registry::setInstance($registry);
```

Zend_Config

- Object oriented access to configuration
- Adapters load specific file types
- Adapters support single inheritence
- Merge two config objects with merge():

```
$config->merge($config2);
```

Zend_Config_Ini

```
[production]
database.adapter = pdo_mysql
database.params.host = db.example.com

[staging : production]
database.params.host = dev.example.com
```

```
$config = new Zend_Config_Ini('config.ini',
                'staging');
```

Zend_Config_Xml

```
<?xml version="1.0"?>
<configdata>
  <production>
    <database>
      <adapter>pdo_mysql</adapter>
      <params><host>db.example.com</host></params>
    </database>
  </production>
  <staging extends="production">
    <database>
      <params><host>dev.example.com</host></params>
    </database>
  </staging>
</configdata>
```

Zend_Loader

- `Zend_Loader::loadFile($file, $dirs, $once)`
  - security check on format of filename
  - searches $dirs in order, then include_path
- `Zend_Loader::loadClass($file, $dirs)`
  - substitutes _ for /
  - searches $dirs in order, then include_path

Plugin loader

- Specify class prefix and path not on include_path
- Paths are search in LIFO order.

```
$loader = new Zend_Loader_PluginLoader();
$loader->addPrefixPath(
 'Zend_View_Helper', 'Zend/View/Helper/');
```

Zend_Session

- Zend_Session_Namespace
  - accessor objects to Zend_Session
- Zend_Session
  - manages $_SESSION
  - standard PHP config options apply
  - start: `Zend_Session::start();`

Zend_Session

```
$session = new Zend_Session_Namespace('rob');
if (isset($session->count)) {
  $session->count++;
} else {
  $session->count = 1;
}

foreach ($session as $key => $value) {
    echo "$key = '$value'\n";
}
```

Expiration

```
$s = new Zend_Session_Namespace('expireAll');
$s->a = 'apple';
$s->p = 'pear';
```

```
$s->setExpirationHops(5);
$s->setExpirationSeconds(60);

// expire only the key "a" in 5 seconds
$s->setExpirationSeconds(5, 'a');
```

Internationalisation

&

Localisation

Sample question

Which Zend_View_Helper_Translate method is used to change language?

Zend_Locale

- Automatic Locales:
  - `$locale = Zend_Locale('browser');`
  - `$locale = Zend_Locale('environment');`
  - `$locale = Zend_Locale('framework');`
  - `$locale = Zend_Locale('auto');`

- More common:
  - `$locale = new Zend_Locale('en_US');`

Zend_Locale_Format

```
$locale = new Zend_Locale ('de_DE')
$number = Zend_Locale_Format::getNumber(2435.837,
    array('precision' => 2, 'locale' => $locale));
// prints: 2.435,84
```

Also:

```
getDate(), getDateFormat(), getFloat(),
getInteger(), getTime(), getTimeFormat()
```

Zend_Translate

- Adapters:
  - Array, Tbx, Xliff, Csv, Tmx, XmlTm, Gettext, Qt
- Source file layout:

Single-structure:	all files in one directory
Language-structured:	one language per directory
Application-structured:	multiple files per language
Gettext-structured:	legacy gettext organisation
File-structured:	files related to source code

Zend_Translate

```
$translate = new Zend_Translate('gettext',
 '/path/de.mo', 'de', $options);
$translate->addTranslation('/path/fr.mo', 'fr');
```

```
Zend_Translate::setCache($cache);
```

Options in $options array:

```
clear: Remove previous translations (boolean)
scan: Where to find the locale info
      (Zend_Translate::LOCALE_DIRECTORY or
       Zend_Translate::LOCALE_FILENAME)
```

Zend_Translate

```
// as before
$translate = new Zend_Translate('gettext',
 '/path/de.mo', 'de');
$translate->addTranslation('/path/fr.mo', 'fr');
```

```
// usage
$translate->_("welcome %1$s", $name);
$translate->setLocale('fr_CH');
$translate->_("welcome %1$s", $name);
```

Translate view helper

- Register Zend_Translate instance:

```
Zend_Registry::set('Zend_Translate',$adapter);
```

- Can set within view if required:

```
$this->translate()->setTranslator($adapter);
$this->translate()->setLocale('en');
```

- Use within view:

```
<?php echo $this->translate('simple'); ?>
<?php echo $this->translate('%1$s in %2$s',
        $time, $countryName); ?>
<?php echo $this->translate('%1$s in %2$s',
        array($time, $countryName)); ?>
```

Zend_Date

```
$date = new Zend_Date($unixtimestamp,
          Zend_Date::TIMESTAMP);
$date = new Zend_Date($row->date_created,
          Zend_Date::ISO_8601);
$array = array('year'=>2009,'month'=>8,
    'day'=>7,'hour'=6,'minute'=>5,'second'=>4);
$date = new Zend_Date($array);

if(Zend_Date::isDate('30 Feb 09') == false) {
  // not a date
}
```

Zend_Date operations

```
$date->add($date, $part, $locale);
$date->sub($date, $part, $locale);
$date->compare($date, $part, $locale);
$date->get($date, $part, $locale);
$date->set($date, $part, $locale);
```

Parts:

```
Zend_Date::TIMESTAMP, Zend_Date::TIMEZONE
Zend_Date::DAY, Zend_Date::MONTH, Zend_Date::YEAR
Zend_Date::MINUTE, Zend_Date::SECOND,
(lots more!)
```

Zend_Currency

```
$currency = new Zend_Currency('en_GB');

echo $currency->toCurrency(1234.56);
// output: £1,234.56
```

```
$currency->setLocale('fr_FR');
echo $currency->toCurrency(1234.56);
// output: 1 234,56 €
```

Web services

Sample question

The result of a call to a `Zend_Rest_Client` method is an instance of:

XmlRpc client

- Calling a server method:
  - $client->call('test.sayHello', array($arg1, $arg2));
  - Proxy:
    - $server = $client->getProxy();
    - $server->test->sayHello($arg1, $arg2);
- Use Zend_XmlRpc_Value to convert type

XmlRpc client

- Error handling: Exceptions
  - Zend_XmlRpc_Client_HttpException
  - Zend_XmlRpc_Client_FaultException
- but, doRequest() returns a Zend_XmlRpc_Response with isFault() true

XmlRpc server

```
$server = new Zend_XmlRpc_Server();
$server->addFunction('addTwoNumbers');
$server->addFunction('divideTwoNumbers');

$server->setClass('Services_Math', 'ns1');
$server->setClass('Services_Finance', 'ns2');

echo $server->handle();
```

XmlRpc server

Catches exceptions from your method and converts to a Zend_XmlRpc_Server_Fault.

Whitelist your exceptions using:

```
Zend_XmlRpc_Server_Fault::attachFaultException('My_Project_Exception');
```

XmlRpc server

Cache server definitions between requests:

```
$cacheFile = dirname(__FILE__) . '/xmlrpc.cache';
if (!Zend_XmlRpc_Server_Cache::get($cacheFile, $server)) {
    $server->setClass(...);

    Zend_XmlRpc_Server_Cache::save($cacheFile, $server));
}

echo $server->handle();
```

Rest client

```
$client = new Zend_Rest_Client('http://path/to/service');
$result $client->methodName($arg1)->get();
```

Must follow method with:

`->get()`, `->put()`, `->post()` or `->delete()`.

`$result` is a `Zend_Rest_Client_Response`

Rest server

```
$server = new Zend_Rest_Server();
$server->addFunction('addTwoNumbers');

$server->setClass('Services_Math');
echo $server->handle();
```

Zend_Service

- Zend_Service_Abstract
  - Uses an underlying Zend_HttpClient
  - setHttpClient() / getHttpClient() allow for customisation of the client object.
- Be aware of the services supported:
  - `Akismet`, `Amazon`, `Audioscrobbler`
  - `Delicious`, `Flickr`, `Nirvanix`, `Simpy`,
  - `Slideshare`, `StrikeIron`, `Technorati`,
  - `Yahoo`

Search

Sample question

Which of the following field types is tokenised by `Zend_Search_Lucene`?

a. Keyword
b. UnIndexed
c. UnStored
d. Binary

Documents

- Documents are atomic objects
- Documents are divided into named fields
- UTF-8 internally (auto converted on input)
- Stored on filesytem

Indexing

```
$index = Zend_Search_Lucene::create('/data/myindex');

$doc = new Zend_Search_Lucene_Document();
$doc->addField(Zend_Search_Lucene_Field::UnIndexed(
    'url', $url));
$doc->addField(Zend_Search_Lucene_Field::UnStored(
    'contents', $contents));
$doc->addField(Zend_Search_Lucene_Field::Text(
    'description', $description));

$index->addDocument($doc);
```

Index operations

```
$numDocuments = $index->count();

$numberOfIndexedFields = $index->count
```

```
// 'id' is a reserved field. If you use:
$id = $hit->getDocument()->id;
```

Field types

Name	Indexed	Stored	Tokenised
Keyword	Yes	Yes	No
UnIndexed	No	Yes	No
Text	Yes	Yes	Yes
UnStored	Yes	No	Yes
Binary	No	Yes	No

Updating

```
//To update: remove and re-add
//Therefore: store a unique reference for each doc

$docRef = $document->docRef;
$term = new Zend_Search_Lucene_Index_Term(
                        $docRef, 'docRef');
$query = new Zend_Search_Lucene_Search_Query_Term($term);
$results = $this->find($query);
foreach ($results as $doc) {
    $index->delete($doc->id);
}
// add updated document
```

Querying

- Query parser (from a string)

```
$hits = $index->find($query);
```

- Programmatic querying

```
$term = new Zend_Search_Lucene_Index_Term('php',
    'category');
$query = new Zend_Search_Lucene_Search_Query_Term($term);
$hits = $index->find($query);
```

- Use results

```
foreach ($hits as $hit) {
    $document = $hit->getDocument();
    $document->getField('title'); // or $hit->title
}
```

Search performance

- Indexes consists of many segments
- Memory usage
  - Terms dictionary is always in memory
  - Each segment has it's own terms dictionary.
- Optimisation restores index quality (merges)
  - `$index->optimize();`

Good luck!

Provide feedback on this talk: http://joind.in/882