

Zend_Service_Recaptcha - Christer Edvartsen & Pádraic Brady

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Service_Recaptcha Component Proposal

Proposed Component Name	Zend_Service_Recaptcha
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Service_Recaptcha
Proposers	Christer Edvartsen Pádraic Brady Matthew Weier O'Phinney (Zend Liaison)
Revision	1.1 - Ready For Zend Review (wiki revision: 12)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

Let's get the definition out of the way 😊. A CAPTCHA is a test to determine whether the entity being tested is a real breathing human being or a computer driven facsimile. They are used predominantly to prevent automated actions, such as making comments or posting forum messages, by programs such as bots which are generally designed to distribute spam (or worse).

The effort expended by the real humans is immense. It's estimated upwards of 60-70 million CAPTCHAs are solved everyday. Not about to let that time go down in history as a waste of electrons, the reCAPTCHA team have developed their own implementation where the solutions are utilised to maximise the efficiency of digitising books where OCR errors are common.

The benefit of reCAPTCHA is not just goodwill. The program itself has a high rate of success, offers built in alternatives for individuals without the benefit of perfect eyesight, and uses a public/private key security to prevent chained attacks (i.e. using the answers of real humans to drive those of bots elsewhere).

2. References

- [What is reCAPTCHA?](#)

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** allow insertion of a reCAPTCHA into forms.
- This component **will** allow verification of reCAPTCHA answers.
- This component **will** optionally output XHTML valid markup (this is unsupported by reCAPTCHA itself).
- This component **will not** assume the role of generating a custom themed reCAPTCHA.
- This component **will** allow setting of SSL mode, and a custom error message.
- This component **will** allow setting of options for a RecaptchaOptions JS container (see reCAPTCHA API documentation).
- This component **may** integrate with Zend_Form once CAPTCHA integration is finalised.

Please note that XHTML output will, and must, be disabled by default. reCAPTCHA does not officially support an XHTML option as yet, so this will remain entirely for use at the user's risk.

4. Dependencies on Other Framework Components

- Zend_Http_Client
- Zend_Json
- Zend_Exception

Zend_Json is utilised to create the RecaptchaOptions Javascript container. It is almost a negligible dependency which can be removed with very little impact.

5. Theory of Operation

The detailed workflow of reCAPTCHA begins with outputting the actual CAPTCHA into a form. This requires the inclusion of Javascript into the current View, sourced from an offsite script URL and a noscript element for non-javascript enabled clients. These elements must be inserted within <form> tags which may require a backend integration point to Zend_Form (one is in development) or at least detailed documentation of a method which allows for this.

More dynamic pages can make use of the AJAX API. The AJAX API will not be addressed by this proposal since it is managed on the client side using Javascript alone. Validation may of course use this component.

Once a reCAPTCHA has been appended to a relevant form, the response may then be validated using the Service API. The process, in terms of userland implementation, is quite simple. Refer to the default use cases a typical user can expect below.

Internally, the source code for the component is expected to make use of other components to whatever extent is possible to reduce duplication.

6. Milestones / Tasks

- Milestone 1: Assemble use cases and design comments based on draft source code
- Milestone 2: Assemble a unit test suite for common functionality and implement
- Milestone 3: Pending review comments, complete initial development
- Milestone 4: Complete acceptance testing, verify unit test coverage
- Milestone 5: Let's assume documentation is written during development

7. Class Index

- Zend_Service_Recaptcha
- Zend_Service_Recaptcha_Exception
- Zend_Service_Recaptcha_Response
- Zend_Service_Recaptcha_Mailhide
- Zend_Service_Recaptcha_Mailhide_Exception

This is a preliminary class listing. Pending public and Zend review, the list may expand as necessary.

8. Use Cases

UC-01

At its simplest, output of a reCAPTCHA can be generated utilising default options.

```
<?php

$captcha = new Zend_Service_Recaptcha('PUBLIC_KEY', 'PRIVATE_KEY');
echo $captcha;
```

UC-02

Validation then follows, again using default options.

```
<?php

$captcha = new Zend_Service_Recaptcha('PUBLIC_KEY', 'PRIVATE_KEY');
$result = $captcha->verify(
    $_POST['recaptcha_challenge_field'],
    $_POST['recaptcha_response_field']
);
if($result->isValid()) {
    // CAPTCHA correctly answered
} else {
    // give user a second chance, or give them the boot
}
```

UC-03

The component allows for two customisation means. Three parameter options for enabling SSL compatible requests, setting a custom error message on invalid responses, and to enable XHTML output. There is also support for reCAPTCHA specific options which are prepended to any Javascript to tailor the reCAPTCHA as documented in the reCAPTCHA client API.

```
<?php

require_once 'Zend/Service/ReCaptcha.php';
$captcha = new Zend_Service_Recaptcha('PUBLIC_KEY', 'PRIVATE_KEY');
$captcha->setParams(array(
    'ssl'=>true,
    'xhtml'=>true
));
$captcha->setOptions(array(
    'theme'=>'white',
    'lang'=>'fr',
    'tabindex'=>2
));
echo $captcha;
```

9. Class Skeletons

Source code for the above component has been in progress and may be made available upon request. Once a more finalised version is complete, it will be attached to this proposal (if not already reviewed).

```
]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>
```