Visual and Audio PHP CAPTCHA Generation Class

PhpCaptcha is a library for generating visual and audio CAPTCHAs (completely automated public Turing test to tell computers and humans apart). You can read more about CAPTCHAs at Wikipedia.

It can help you to prevent/reduce:

- Automated sign-ups in registration forms.
- Comment spam in blogs and guestbooks.
- Brute force attacks on login systems.

A Word of Warning

OK, so this might sounds strange given what follows but please consider whether you really need to use a CAPTCHA before implementing this on your site. Although it's legitimate to use CAPTCHAs in some situations you really need to be aware of the inherent accessibility pitfalls before implementing. I'd also encourage you to fully investigate the alternatives such as the Akismet spam filtering WordPress plug-in and API before resorting to a CAPTCHA to solve your spam problems.

Right, I've said my piece - on with the script.

Origins

The library is loosely based on an article I wrote for SitePoint which was published on 9th November 2005 - Toughen Forms' Security with an Image.

Supported Features

- Multiple random TrueType fonts
- Character rotation
- Optional chararacter shadow support
- Optional site owner display text
- Random custom background images
- Font size selection
- Greyscale or colour lines and characters
- Character set selection
- Integration of validation function for checking the user entered code with the generated code

Requirements

The library requires PHP 4 compiled with GD 1 or 2 (Image Generation) and FreeType text support. It should work fine with Linux, Mac OS X or Windows based systems although you will need to change the default temporary file storage directory specified when using with Windows. PHP session support is also required. The audio CAPTCHA requires the Flite text to speech synthesis engine - more details on set up below.

Download

You can download a zip file of the source code here.

License

PhpCaptcha is licensed under the Free BSD license.

Hosting

If you need suitable hosting checkout DreamHost. They offer really good value accounts starting from $7.95 per month which should be suitable for running this CAPTCHA class. If you enter the promo code "**ejeliot**"

Subscribe to RSS Feeds

Posts feed
Comments feed
Quick Links feed

728 readers
BY FEEDBURNER

Connect/Save

bookmark on del.icio.us

tags: accessibility Security validation captcha forms audio php Programming class webdev

saved by 296 people

I am ejeliot on del.icio.us.
Add me to your network.

Sponsors

tutvus
Web Templates
Web Hosting
powerpoint templates
Programming Tutorials
Web design and Seo specialists

Recent Posts

Problems building MySQLdb on Mac OS X 10.4 (Power PC) (0)
Automatic versioning of CSS, JavaScript and Images (7)
27 million microformats on Kelkoo (3)
CSS Sprite Generator Source Code Released (7)
Major update to PhpDelicious (0)
Simple caching proxy for Google Charts API (1)
Accessible, SEO friendly text chopping method (5)
Opacity bug in Opera 9.5 Beta (2)
First steps with Django (2)
User generated translations (0)

Last 12 Months

April 2008 (2)
March 2008 (1)

when signing up you'll save **$49.95**. It is also possible to download and compile the Flite text to speech synthesis engine required for the audio captcha. See instructions below.

Implementation Steps

Step 1

Unzip and copy the php-captcha.inc.php to a directory within your site.

Step 2

To create a basic visual CAPTCHA with the minimal options create a new file (visual-captcha.php) containing the code shown below. You can download some suitable TrueType fonts here.

```
01. <?php
02.    require('php-captcha.inc.php');
03.    $aFonts = array('fonts/VeraBd.ttf', 'fonts/VeraIt.ttf', 'fonts/Vera.ttf');
04.    $oVisualCaptcha = new PhpCaptcha($aFonts, 200, 60);
05.    $oVisualCaptcha->Create();
06. ?>
```

Step 3

To create an audio CAPTCHA create a new file (audio-captcha.php) containing the code shown below. For this to work you'll need to ensure that your have a working installation of Flite.

```
01. <?php
02.    require('php-captcha.inc.php');
03.    $oAudioCaptcha = new AudioPhpCaptcha('/usr/bin/flite', '/tmp/');
04.    $oAudioCaptcha->Create();
05. ?>
```

You need to pass the path to the Flite binary and the temporary directory you want to use for storing generated audio CAPTCHAs to the class constructor. Alternatively you can modify the corresponding constants in the library file and then omit the parameters in the constructor.

Step 4

Include the visual and audio CAPTCHAs in your application/form with the following code. The audio CAPTCHA should ideally follow the visual CAPTCHA in the source code. This will ensure the visual CAPTCHA has generated a random code before the audio CAPTCHA is called.

```
01. <p><img src="visual-captcha.php" width="200" height="60" alt="Visual CAPTCHA"
    /></p>
02. <p><a href="audio-captcha.php">Can't see the image? Click for audible
    version</a></p>
```

Step 5

On form submission you need to check the code the user enters with the one generated by the CAPTCHA. You can do this with the following code assuming that the user entered code was submitted in an HTML POST form field "user_code".

```
01. <?php
02.    require('php-captcha.inc.php');
03.    if (PhpCaptcha::Validate($_POST['user_code'])) {
04.       echo 'Valid code entered';
05.    } else {
```

```
06.      echo 'Invalid code entered';
07.   }
08. ?>
```

Please note that the Validate method needs to be called statically, i.e you don't create an instance of the class before calling it.

Some Examples

Some examples of CAPTCHAs with various options set are shown below.

CAPTCHA with site owner display text


```
01. // include captcha class
02. require('php-captcha.inc.php');
03. // define fonts
04. $aFonts = array('fonts/VeraBd.ttf', 'fonts/VeraIt.ttf', 'fonts/Vera.ttf');
05. // create new image
06. $oPhpCaptcha = new PhpCaptcha($aFonts, 200, 60);
07. $oPhpCaptcha->SetOwnerText('Source: www.ejeliot.com');
08. $oPhpCaptcha->Create();
```

CAPTCHA with site owner display text and character shadows


```
01. // include captcha class
02. require('php-captcha.inc.php');
03. // define fonts
04. $aFonts = array('fonts/VeraBd.ttf', 'fonts/VeraIt.ttf', 'fonts/Vera.ttf');
05. // create new image
06. $oPhpCaptcha = new PhpCaptcha($aFonts, 200, 60);
07. $oPhpCaptcha->DisplayShadow(true);
08. $oPhpCaptcha->SetOwnerText('Source: www.ejeliot.com');
09. $oPhpCaptcha->Create();
```

CAPTCHA with default settings

```
01. // include captcha class
02. require('php-captcha.inc.php');
03. // define fonts
04. $aFonts = array('fonts/VeraBd.ttf', 'fonts/VeraIt.ttf', 'fonts/Vera.ttf');
05. // create new image
06. $oPhpCaptcha = new PhpCaptcha($aFonts, 200, 50);
```

```
07. $oPhpCaptcha->Create();
```

CAPTCHA with custom background image

```
01. // include captcha class
02. require('php-captcha.inc.php');
03. // define fonts
04. $aFonts = array('fonts/VeraBd.ttf', 'fonts/VeraIt.ttf', 'fonts/Vera.ttf');
05. // create new image
06. $oPhpCaptcha = new PhpCaptcha($aFonts, 200, 50);
07. $oPhpCaptcha->SetBackgroundImages('images/captcha.jpg');
08. $oPhpCaptcha->Create();
```

CAPTCHA with colour

```
01. // include captcha class
02. require('php-captcha.inc.php');
03. // define fonts
04. $aFonts = array('fonts/VeraBd.ttf', 'fonts/VeraIt.ttf', 'fonts/Vera.ttf');
05. // create new image
06. $oPhpCaptcha = new PhpCaptcha($aFonts, 200, 50);
07. $oPhpCaptcha->UseColour(true);
08. $oPhpCaptcha->Create();
```

Audio CAPTCHA

You can listen to an example of the audio CAPTCHA alternative on any of my blog post pages.

Configuration Options

The methods listed below allow you to refine the look and feel as well as the behaviour of the generated CAPTCHA. They should all be called before the "Create" method.

01. **SetWidth(int iWidth)** - set the width of the CAPTCHA image. Defaults to 200px.
02. **SetHeight(int iHeight)** - set the height of the CAPTCHA image. Defaults to 50px.
03. **SetNumChars(int iNumChars)** - set the number of characters to display. Defaults to 5.
04. **SetNumLines(int iNumLines)** - set the number of interference lines to draw. Defaults to 70.
05. **DisplayShadow(bool bShadow)** - specify whether or not to display character shadows.
06. **SetOwnerText(sting sOwnerText)** - owner text to display at bottom of CAPTCHA image, discourages attempts to break your CAPTCHA through display on porn and other unsavoury sites.
07. **SetCharSet(variant vCharSet)** - specify the character set to select characters from. If left blank defaults to A-Z. Can be specified as an array of chracters e.g. array('1', 'G', '3') or as a string of characters and character ranges e.g. 'a-z,A-Z,0,3,7'.
08. **CaseInsensitive(bool bCaseInsensitive)** - specify whether or not to save user code preserving case. If setting to "false" you need to pass "false" as the second parameter to the "Validate" function when checking the user entered code.
09. **SetBackgroundImages(variant vBackgroundImages)** - specify one (a string) or more (an array) images to display instead of noise lines. If more than one image is specified the library selects one at random.

10. **SetMinFontSize(int iMinFontSize)** - specify the minimum font size to display. Defaults to 16.
11. **SetMaxFontSize(int iMaxFontSize)** - specify the maximum font size to display. Defaults to 25.
12. **UseColour(bool bUseColour)** - if true displays noise lines and characters in randomly selected colours.
13. **SetFileType(string sFileType)** - specify the output format jpeg, gif or png. Defaults to jpeg.

TrueType Fonts

If you don't have any TrueType fonts you can use with this class you can find some here.

Flite text to speech synthesis engine

You can download the source distribution of Flite from here. It is also available as a Debian package as well as RPMs which are suitable for use on RedHat and Mandrake Linux. If you're compiling from source ensure that your server has plenty of memory. Most compile time errors are caused by lack of memory. To download and compile Flite from source run the following commands on the command line:

```
01. wget http://www.speech.cs.cmu.edu/flite/packed/flite-1.3/ flite-1.3-release.tar.gz
02. tar zxvf flite-1.3-release.tar.gz
03. cd flite-1.3-release
04. ./configure
05. make
```

You can test if Flite installed successfully by running the following command to generate a sample wav file:

```
01. flite -t 'Hello World' -o hello.wav
```