SDN Community Contribution

(This is not an official SAP document.)

Disclaimer & Liability Notice

CAPTCHA for BSP Applications

Applies To:

SAP Web Application Server, ABAP/Business Server Pages (BSP)

Summary

This PHP class code enables the inclusion of CAPTCHA technology within SAP Web AS BSP applications. One needs to make a calling script for this class. This calling script is triggered as an external command inside the BSP application. Detailed information on implementation is covered in the SDN blog, "Die Mensch-Machine"

By: Eddy De Clercq

Company: K.U. Leuven

Date: 17 July 2005

The Code

```
<?php
/**
  * PHP-Class bsp_captcha Version 1.0
  *
  * Original code hn_captcha Version 1.2
  * Original code author: Horst Nogajski
  * Original License: GNU GPL (http://www.opensource.org/licenses/gpl-license.html)
  *
  * Modifications and port for SAP Web AS BSP application by: Eddy De Clercq
  * For SAP Developers Network http://sdn.sap.com
  * Detailed info on implementation is covered in the web log: https://www.sdn.sap.com/sdn/weblogs.sdn?blog=/pub/wlg/1952
  **/


class bsp_captcha
{


    ////////////////////////////////
    //
    //  PUBLIC PARAMS
    //
```

```
        /**
          * @shortdesc Absolute path to a Tempfolder (with trailing slash!).
This must be writeable for PHP and also accessible via HTTP, because the
image will be stored there.
          * @type string
          * @public
          *
          **/
        var $tempfolder;


        /**
          * @shortdesc Absolute path to folder with TrueTypeFonts (with
trailing slash!). This must be readable by PHP.
          * @type string
          * @public
          *
          **/
        var $TTF_folder;


        /**
          * @shortdesc A List with available TrueTypeFonts for random char-
creation.
          * @type mixed[array|string]
          * @public
          *
          **/
        var $TTF_RANGE  =
array('COMIC.TTF','JACOBITE.TTF','LYDIAN.TTF','MREARL.TTF','RUBBERSTAMP.TTF',
'ZINJARON.TTF');


        /**
          * @shortdesc How many chars the generated text should have
          * @type integer
          * @public
          *
          **/
        var $chars      = 6;
```

```
        /**
          * @shortdesc The minimum size a Char should have
          * @type integer
          * @public
          *
          **/
        var $minsize    = 20;

        /**
          * @shortdesc The maximum size a Char can have
          * @type integer
          * @public
          *
          **/
        var $maxsize    = 40;

        /**
          * @shortdesc The maximum degrees a Char should be rotated. Set it
to 30 means a random rotation between -30 and 30.
          * @type integer
          * @public
          *
          **/
        var $maxrotation = 30;

        /**
          * @shortdesc Background noise On/Off (if is Off, a grid will be
created)
          * @type boolean
          * @public
          *
          **/
        var $noise      = TRUE;
```

```
        /**
          * @shortdesc This will only use the 216 websafe color pallette for
the image.
          * @type boolean
          * @public
          *
          **/
        var $websafecolors = FALSE;


        /**
          * @shortdesc Outputs configuration values for testing
          * @type boolean
          * @public
          *
          **/
        var $debug = FALSE;

        /**
          * @shortdesc defines public key
          * @type string
          * @public
          *
          **/
        var $public_key = '';

        /**
          * @shortdesc defines private key
          * @type string
          * @public
          *
          **/
        var $private_key = '';
```

```
////////////////////////////////
//
//  PRIVATE PARAMS
//

    /** @private **/
    var $lx;                // width of picture
    /** @private **/
    var $ly;                // height of picture
    /** @private **/
    var $jpegquality = 80;  // image quality
    /** @private **/
    var $noisefactor = 9;   // this will multiplyed with number of chars
    /** @private **/
    var $nb_noise;          // number of background-noise-characters
    /** @private **/
    var $TTF_file;          // holds the current selected TrueTypeFont
    /** @private **/
    var $filename;          // filename of captcha picture
    /** @private **/
    var $gd_version;        // holds the Version Number of GD-Library
    /** @private **/
    var $r;
    /** @private **/
    var $g;
    /** @private **/
    var $b;


////////////////////////////////
//
//  CONSTRUCTOR
//
```

```
        /**
          * @shortdesc Extracts the config array and generate needed params.
          * @private
          * @type void
          * @return nothing
          *
          **/
        function bsp_captcha($config,$secure=TRUE)
        {

            // Test for GD-Library(-Version)
            $this->gd_version = $this->get_gd_version();
            if($this->gd_version == 0) die("There is no GD-Library-Support
enabled. The Captcha-Class cannot be used!");
            if($this->debug) echo "\n<br>-Captcha-Debug: The available GD-
Library has major version ".$this->gd_version;



            // extracts config array
            if(is_array($config))
            {
                if($secure && strcmp('4.2.0', phpversion()) < 0)
                {
                    if($this->debug) echo "\n<br>-Captcha-Debug: Extracts
Config-Array in secure-mode!";
                    $valid = get_class_vars(get_class($this));
                    foreach($config as $k=>$v)
                    {
                        if(array_key_exists($k,$valid)) $this->$k = $v;
                    }
                }
                else
                {
                    if($this->debug) echo "\n<br>-Captcha-Debug: Extracts
Config-Array in unsecure-mode!";
                    foreach($config as $k=>$v) $this->$k = $v;
```

```
            }
        }


        // check TrueTypeFonts
        if(is_array($this->TTF_RANGE))
        {
            if($this->debug) echo "\n<br>-Captcha-Debug: Check given
TrueType-Array! (".count($this->TTF_RANGE).")";
            $temp = array();
            foreach($this->TTF_RANGE as $k=>$v)
            {
                if(is_readable($this->TTF_folder.$v)) $temp[] = $v;
            }
            $this->TTF_RANGE = $temp;
            if($this->debug) echo "\n<br>-Captcha-Debug: Valid TrueType-
files: (".count($this->TTF_RANGE).")";
            if(count($this->TTF_RANGE) < 1) die('No Truetypefont
available for the CaptchaClass.');
        }
        else
        {
            if($this->debug) echo "\n<br>-Captcha-Debug: Check given
TrueType-File! (".$this->TTF_RANGE.")";
            if(!is_readable($this->TTF_folder.$this->TTF_RANGE)) die('No
Truetypefont available for the CaptchaClass.');
        }


        // select first TrueTypeFont
        $this->change_TTF();
        if($this->debug) echo "\n<br>-Captcha-Debug: Set current
TrueType-File: (".$this->TTF_file.")";



        // get number of noise-chars for background if is enabled
        $this->nb_noise = $this->noise ? ($this->chars * $this-
>noisefactor) : 0;
```

```
            if($this->debug) echo "\n<br>-Captcha-Debug: Set number of noise
characters to: (".$this->nb_noise.")";



            // set dimension of image
            $this->lx = ($this->chars + 1) * (int)(($this->maxsize + $this-
>minsize) / 1.5);
            $this->ly = (int)(2.4 * $this->maxsize);
            if($this->debug) echo "\n<br>-Captcha-Debug: Set image dimension
to: (".$this->lx." x ".$this->ly.")";


        }




        function make_captcha()
        {
            // create Image and set the apropriate function depending on GD-
Version & websafecolor-value
            if($this->gd_version >= 2 && !$this->websafecolors)
            {
                $func1 = 'imagecreatetruecolor';
                $func2 = 'imagecolorallocate';
            }
            else
            {
                $func1 = 'imageCreate';
                $func2 = 'imagecolorclosest';
            }
            $image = $func1($this->lx,$this->ly);
            if($this->debug) echo "\n<br>-Captcha-Debug: Generate ImageStream
with: ($func1())";
            if($this->debug) echo "\n<br>-Captcha-Debug: For colordefinitions
we use: ($func2())";



            // Set Backgroundcolor
```

```
            $this->random_color(224, 255);
            $back =  @imagecolorallocate($image, $this->r, $this->g, $this-
>b);
            @ImageFilledRectangle($image,0,0,$this->lx,$this->ly,$back);
            if($this->debug) echo "\n<br>-Captcha-Debug: We allocate one
color for Background: (".$this->r."-".$this->g."-".$this->b.")";


            // allocates the 216 websafe color palette to the image
            if($this->gd_version < 2 || $this->websafecolors) $this-
>makeWebsafeColors($image);



            // fill with noise or grid
            if($this->nb_noise > 0)
            {
                // random characters in background with random position,
angle, color
                if($this->debug) echo "\n<br>-Captcha-Debug: Fill background
with noise: (".$this->nb_noise.")";
                for($i=0; $i < $this->nb_noise; $i++)
                {
                    srand((double)microtime()*1000000);
                    $size   = intval(rand((int)($this->minsize / 2.3),
(int)($this->maxsize / 1.7)));
                    srand((double)microtime()*1000000);
                    $angle  = intval(rand(0, 360));
                    srand((double)microtime()*1000000);
                    $x      = intval(rand(0, $this->lx));
                    srand((double)microtime()*1000000);
                    $y      = intval(rand(0, (int)($this->ly - ($size /
5))));
                    $this->random_color(160, 224);
                    $color  = $func2($image, $this->r, $this->g, $this->b);
                    srand((double)microtime()*1000000);
                    $text   = chr(intval(rand(45,250)));
                    @ImageTTFText($image, $size, $angle, $x, $y, $color,
$this->change_TTF(), $text);
                }
```

```
            }
            else
            {
                // generate grid
                if($this->debug) echo "\n<br>-Captcha-Debug: Fill background
with x-gridlines: (".(int)($this->lx / (int)($this->minsize / 1.5))."
)";
                for($i=0; $i < $this->lx; $i += (int)($this->minsize / 1.5))
                {
                    $this->random_color(160, 224);
                    $color  = $func2($image, $this->r, $this->g, $this->b);
                    @imageline($image, $i, 0, $i, $this->ly, $color);
                }
                if($this->debug) echo "\n<br>-Captcha-Debug: Fill background
with y-gridlines: (".(int)($this->ly / (int)(($this->minsize / 1.8))).")";
                for($i=0 ; $i < $this->ly; $i += (int)($this->minsize / 1.8))
                {
                    $this->random_color(160, 224);
                    $color  = $func2($image, $this->r, $this->g, $this->b);
                    @imageline($image, 0, $i, $this->lx, $i, $color);
                }
            }


            // generate Text
            if($this->debug) echo "\n<br>-Captcha-Debug: Fill forground with
chars and shadows: (".$this->chars.")";
            for($i=0, $x = intval(rand($this->minsize,$this->maxsize)); $i <
$this->chars; $i++)
            {
                $text   = strtoupper(substr($this->private_key, $i, 1));
                srand((double)microtime()*1000000);
                $angle  = intval(rand(($this->maxrotation * -1), $this-
>maxrotation));
                srand((double)microtime()*1000000);
                $size   = intval(rand($this->minsize, $this->maxsize));
                srand((double)microtime()*1000000);
                $y      = intval(rand((int)($size * 1.5), (int)($this->ly -
($size / 7))));
```

```
                $this->random_color(0, 127);
                $color  =  $func2($image, $this->r, $this->g, $this->b);
                $this->random_color(0, 127);
                $shadow = $func2($image, $this->r + 127, $this->g + 127,
$this->b + 127);
                @ImageTTFText($image, $size, $angle, $x + (int)($size / 15),
$y, $shadow, $this->change_TTF(), $text);
                @ImageTTFText($image, $size, $angle, $x, $y - (int)($size /
15), $color, $this->TTF_file, $text);
                $x += (int)($size + ($this->minsize / 5));
            }
            @ImageJPEG($image, $this->get_filename(), $this->jpegquality);
            $res = file_exists($this->get_filename());
            if($this->debug) echo "\n<br>-Captcha-Debug: Safe Image with
quality [".$this->jpegquality."] as (".$this->get_filename().") returns:
(".($res ? 'TRUE' : 'FALSE').")";
            @ImageDestroy($image);
            if($this->debug) echo "\n<br>-Captcha-Debug: Destroy
Imagestream.";
            if(!$res) die('Unable to safe captcha-image.');
        }

        /** @private **/
        function makeWebsafeColors(&$image)
        {
            //$a = array();
            for($r = 0; $r <= 255; $r += 51)
            {
                for($g = 0; $g <= 255; $g += 51)
                {
                    for($b = 0; $b <= 255; $b += 51)
                    {
                        $color = imagecolorallocate($image, $r, $g, $b);
                        //$a[$color] = array('r'=>$r,'g'=>$g,'b'=>$b);
                    }
                }
            }
```

```
            if($this->debug) echo "\n<br>-Captcha-Debug: Allocate 216 websafe
colors to image: (".imagecolorstotal($image).")";
            //return $a;
        }

        /** @private **/
        function random_color($min,$max)
        {
            srand((double)microtime() * 1000000);
            $this->r = intval(rand($min,$max));
            srand((double)microtime() * 1000000);
            $this->g = intval(rand($min,$max));
            srand((double)microtime() * 1000000);
            $this->b = intval(rand($min,$max));
            //echo " (".$this->r."-".$this->g."-".$this->b.") ";
        }

        /** @private **/
        function change_TTF()
        {
            if(is_array($this->TTF_RANGE))
            {
                srand((float)microtime() * 10000000);
                $key = array_rand($this->TTF_RANGE);
                $this->TTF_file = $this->TTF_folder.$this->TTF_RANGE[$key];
            }
            else
            {
                $this->TTF_file = $this->TTF_folder.$this->TTF_RANGE;
            }
            return $this->TTF_file;
        }

        /** @private **/
        function get_filename($public="")
```

```
        {
            if($public=="") $public=$this->public_key;
            return $this->tempfolder.$public.".jpg";
        }


        /** @private **/
        function get_gd_version()
        {
            static $gd_version_number = null;
            if($gd_version_number === null)
            {
               ob_start();
               phpinfo(8);
               $module_info = ob_get_contents();
               ob_end_clean();
               if(preg_match("/\bgd\s+version\b[^\d\n\r]+?([\d\.]+)/i",
$module_info, $matches))
               {
                   $gd_version_number = $matches[1];
               }
               else
               {
                   $gd_version_number = 0;
               }
            }
            return $gd_version_number;
        }

} // END CLASS bsp_CAPTCHA

?>
```

Author Bio

Eddy De Clercq has 20 years experience in computing. Currently he works at the Katholieke Universiteit Leuven (Belgium), the oldest university of the Low Countries and the largest Flemish university. Eddy makes part of the E-university team which creates mostly self services (web) applications.