Zend_Image - Dolf Schimmel

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

To do:

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

Zend Framework: Zend_Image Component Proposal

Proposed Component Name	Zend_Image
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Image
Proposers	Dolf Schimmel (Freeaqingme) David Caunt
Zend Liaison	TBD
Revision	1.0 - 1 Oct 2008: Initial Draft. 1.1 - 16 Oct 2008: Finished proposal 1.1 - 10 Jan 2009: Ready for community review 1.2 - 28 Jul 2009: Moved to ready for recommendation (wiki revision: 15)

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

Zend_image is a component used for handling images.

Marcin Lulek a.k.a. Ergo2 has written an image component on which this proposal and initial codebase is based. His component does however miss an object oriented interface, and therefore it has been decided to fully refactor it. As for there is already a suitable codebase (preview of its functionality can be found here ; http://webreactor.eu/image/), offers of components written by others are unnecessary (appreciated though).

2. References

- GD-library Wikipedia Entry
- ImageMagick

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** be able to resize images.
- This component **will** be able to rotate images.
- This component **will** be able to print a watermarks on images.
- This component **will** be able to draw; eclipses, arcs, lines and polygons
- This component **will** be able to convert images (I.E. from jpeg to bmp).
- This component **will** be able to apply several filters, as well as adjust contrast etc.

4. Dependencies on Other Framework Components

- GD or ImageMagick
- Zend_Exception

5. Theory of Operation

An instance of Zend_Image will have one image resource by default. The path of this image can be specified when constructing it. It can however also be loaded from a file or string, or be created by Zend_Image itself. Once a resource has been loaded, the actions on the image can be performed, like cropping, resizing, rotating, etc.

By default Zend_Image will support two adapters, namely Zend_Image_Adapter_Gd and Zend_Image_Adapter_ImageMagick.

6. Milestones / Tasks

- Milestone 1: Proposal finished (DONE)
- Milestone 2: Development of prototype started
- Milestone 3: Proposal approved
- Milestone 4: Working prototype checked into the incubator.
- Milestone 5: Unit tests exist, work, and are checked into SVN.
- Milestone 6: Documentation exists.

7. Class Index

- Zend_Image
- Zend_Image_Color
- Zend_Image_Point
- Zend_Image_Chain
- Zend_Image_Adapter_Abstract
- Zend_Image_Adapter_Gd
- Zend_Image_Adapter_ImageMagick
- Zend_Image_Action_DrawLine
- Zend_Image_Action_DrawEllipse
- Zend_Image_Action_DrawCircle
- Zend_Image_Action_DrawArch
- Zend_Image_Action_DrawText
- Zend_Image_Action_DrawPolygon
- Zend_Image_Action_ApplyFilter
- Zend_Image_Action_AdjustAlpha
- Zend_Image_Action_Resize
- Zend_Image_Action_Blend
- Zend_Image_Action_Rotate
- Zend_Image_Action_Brightness
- Zend_Image_Action_Contrast
- Zend_Image_Action_Colorize

- Zend_Image_Adapter_Gd_Action_DrawLine
- Zend_Image_Adapter_Gd_Action_DrawEllipse
- Zend_Image_Adapter_Gd_Action_DrawCircle

- Zend_Image_Adapter_Gd_Action_DrawArch
- Zend_Image_Adapter_Gd_Action_DrawText
- Zend_Image_Adapter_Gd_Action_DrawPolygon
- Zend_Image_Adapter_Gd_Action_ApplyFilter
- Zend_Image_Adapter_Gd_Action_AdjustAlpha
- Zend_Image_Adapter_Gd_Action_Resize
- Zend_Image_Adapter_Gd_Action_Blend
- Zend_Image_Adapter_Gd_Action_Rotate
- Zend_Image_Adapter_Gd_Action_Brightness
- Zend_Image_Adapter_Gd_Action_ActionInterface_Contrast
- Zend_Image_Adapter_Gd_Action_Colorize

- Zend_Image_Adapter_ImageMagick_Action_DrawLine
- Zend_Image_Adapter_ImageMagick_Action_DrawEllipse
- Zend_Image_Adapter_ImageMagick_Action_DrawCircle
- Zend_Image_Adapter_ImageMagick_Action_DrawArch
- Zend_Image_Adapter_ImageMagick_Action_DrawText
- Zend_Image_Adapter_ImageMagick_Action_DrawPolygon
- Zend_Image_Adapter_ImageMagick_Action_ApplyFilter
- Zend_Image_Adapter_ImageMagick_Action_AdjustAlpha
- Zend_Image_Adapter_ImageMagick_Action_Resize
- Zend_Image_Adapter_ImageMagick_Action_Blend
- Zend_Image_Adapter_ImageMagick_Action_Rotate
- Zend_Image_Adapter_ImageMagick_Action_Brightness
- Zend_Image_Adapter_ImageMagick_Action_Contrast
- Zend_Image_Adapter_ImageMagick_Action_Colorize

8. Use Cases

UC-01

```
$options = array('thickness' => 5,
'filled' => true,
'startX' => 10,
'startY' => 15,
'endX' => 50,
'endY' => 125);
$image = new Zend_Image('/path/to/image.png');
$image->draw(Zend_Image::LINE,$options);
$image->save();
```

UC-02

```
$line = new Zend_Image_Action_DrawLine();
$line->from(10,15)
->to(new Zend_Image_Point(50,125))
->setFilled(true);
$image = new Zend_Image('/path/to/image.png'); // Autodetect adapter to use
$image->draw($line);
$image->save();
```

UC-03

```
$chain = new Zend_Image_Chain();
$line = new Zend_Image_Action_DrawLine();
$line->setcoords(10,15,50,125)->setFilled(true);
$chain->add($line);
$image = new Zend_Image('/path/to/image.png',Zend_Image::GD);
$image->perform($chain);
$image->save();
```

UC-04

```
// Create new image;
$image = new Zend_Image();
$image->create(50,20); // X, Y size;

file_put_contents('/path/to/image.png',$image->get()); //
$image->save('/path/to/image.png'); // Both lines do the same
```

UC-05

$options = array('thickness' => 5,
'filled' => true,
'startX' => 10,
'startY' => 15,
'endX' => 50,
'endY' => 125);
$image = new Zend_Image('/path/to/image.png'); // Autodetect adapter to use
$image->drawLine($options);
$image->save();

9. Class Skeletons

Some code (meant to get an impression of the API, not to get an impression of our coding skills) can be found here:
http://framework.zend.com/svn/framework/standard/branches/user/freak/ZendImage/

```
class Zend_Magic_ReadBrain extends Zend_Exception {}

/** Code is meant as a preliminary example of the API. Actual code may differ **/
class Zend_Image
{
    /**
     * Adapter: GD
     */
    const ADAPTER_GD = 'Gd';
    /**
     * Adapter: Imagemick
     */
    const ADAPTER_IMAGEMICK = 'Imagemick';

    /*
*
     * Names of Actions available
     */
    const LINE = 'DrawLine';
    const POLYGON = 'DrawPolygon';
    const ELLIPSE = 'DrawEllipse';
    const ARC = 'DrawArc';
    const ETC = 'MoreToAddHere';

    /**
     * Adapter set to use for image operation
     *
     * @var Zend_Image_Adapter
     */
    protected $_adapter = null;

    /**
     * Path of the location of the image
     */
    protected $_imagePath = null;

    /**
     * Loads the image, if a path is given
     *
     * @param string $path (Optional) Path to the image
     */
    public function __construct ($path = null)
    {
        if (! is_array($path)) {
            $this->setImagePath($path);
        }
    }
```

```
    /**
     * Sets the adapter to use
     * Currently only GD is available
     * ImageMagick will follow soon
     *
     * @param string   $adapter (Optional) The adapter to use
     * @param boolean  $check (Optional) If ture, check if the adapter is available
     * @throw Zend_Image_Exception When checked for availability of unavailable adapter
     */
    public function setAdapter ($adapter = null, $check = true)
    {
        if ($adapter) {
            $name = 'Zend_Image_Adapter_' . $adapter;
            Zend_Loader::loadClass($name);
            $this->_adapter = new $name();
        } else {
            /* No adapter was set. Attempt to detect. */
            $check = false;
            $this->setAdapter($this->_detectAdapter());
        }
        if ($check && ! $this->_adapter->isAvailable()) {
            require_once 'Zend/Image/Exception.php';
            throw new Zend_Image_Exception("Adapter '$adapter' is not available.");
        }
        $this->setImagePath();
    }

    /**
     * Detects the adapter to use.
     * Order: GD, ImageMick
     *
     * @return Available adapter
     */
    protected function _detectAdapter ()
    {
        if (function_exists('gd_info')) {
            return self::ADAPTER_GD;
        } elseif(false) {
            return self::ADAPTER_IMAGEMICK;
        }
        return null;
    }

    /**
     * Set the path of the image
     *
     * @param string   $path (Optional) The path of the image
     * @throw Zend_Image_Exception if path is set on nonexistent adapter
     */
    public function setImagePath ($path = null)
    {
        if (null !== $path) {
            $this->_imagePath = $path;
            if (null !== $this->_adapter) {
                $this->setImagePath();
            }
        } else {
            if (null === $this->_adapter) {
                require_once 'Zend/Image/Exception.php';
                throw new Zend_Image_Exception('Cannot set image path on an adapter that hasn\'t been
set.');
            } elseif (! file_exists($this->_imagePath)) {
                require_once 'Zend/Image/Exception.php';
                throw new Zend_Image_Exception('Image path does not exist.');
            }
            $this->_adapter->setPath($this->_imagePath);
        }
    }

    /**
```

```
 * Perform an action on the image
 * @param mixed $param1
 * @param array $options Options that will be parsed on to the action
 * @return Zend_Image
 * @todo: use plugin loader.
 */
public function apply ($param1, $param2 = null)
{
 if($param2 instanceof Zend_Image_Action_Abstract ) {
  $object = $param2;
 } elseif ($param1 instanceof Zend_Image_Action_Abstract) {
        $object = $param1;
    } else {
        $name = 'Zend_Image_Action_' . ucfirst($param1);
        Zend_Loader::loadClass($name);
        $object = new $name($param2);
    }

    if (! $this->_adapter) {
        $this->setAdapter();
    }
    $this->_adapter->apply($object);
    return $this;
}

public function __call($action,$arguments) {
 $this->apply($action,$arguments[0]);
}

/**
 * Get a string containing the image
 *
 * @param string $format (Optional) The format of the image to return
 * @return string The actual image
 */
public function render ($format = 'png')
{
    return $this->_adapter->getImage($format);
}

public function display($format = 'png',$sendHeader=true) {
 if($sendHeader) {
  header('Content-type: image/png');
 }

 echo $this->_adapter->getImage($format);
 return true;
}

/**
 * Get a string containing the image
 *
 * @return string The image
 */
public function __toString ()
```

```
    {
  return $this->render();
    }
```

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