Exceptions & Errors in PHP

Key concepts

exceptions, try catch-block, nested exceptions, error, error level, err_reporting, set_error_handler, trigger_error

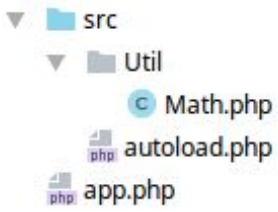
Alternatieve bronnen

http://php.net/manual/en/language.exceptions.php
http://php.net/manual/en/language.errors.php
https://phpro.org/tutorials/PHP-Exceptions.html
https://phpro.org/examples/PHP-Errors-As-Exceptions.html

1. Exceptions inleiding

Exceptions zijn grotendeels gelijkaardig met Java. Een try...catch block bevat 1 try gevolgd door 0, 1 of meerdere catch-blocken al dan niet gevolgd een finally-block.

Bijvoorbeeld de methode sum in de klasse Math werpt een Exception-obcject op als een van de argumenten in sum niet numerisch is. Let op de \ bij het opwerpen van Exception-object in de klasse Math. De code in Math.php is in namespace Util, Exception hoort bij de default-namespace van PHP.



src/Util/Math.php[1]

```
<?php namespace Util;
class Math
{
    public static function sum()
    {
        $arguments=func_get_args();
        $sum=0;
        foreach ($arguments as $argument){
            if ( is_numeric($argument) ){
                $sum+=$argument;
            } else {
                throw new \Exception("$argument is not numeric");
            }
        }
        return $sum;
    }
}
```

[1] Sinds PHP7 is het mogelijk om scalar als datatype voor een parameter van een function vast te leggen en kan er ook een return-type gespecifieerd worden. De functie sum kan herschreven worden als:

```
    public static function sum(float ...$numbers) : float {
        $sum=0;
        foreach ($numbers as $number){
            $sum += $number;
        }
        return $sum;
    }
```

Er wordt nu een TypeError gegenereerd als er het misse datatype binnenkomt.

app.php

```
<?php
require_once 'src/autoload.php';
use Util\Math;
try{
    $sum=Math::sum( 1, 2 );
    print($sum);
} catch ( Exception $exception ) {
    print( $exception->getMessage() );
}
```

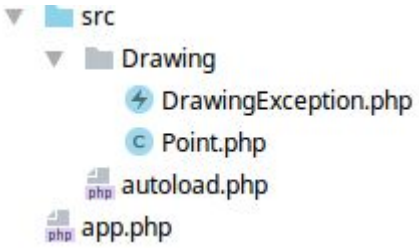
Een overzicht van de methodes die uitgevoerd kunnen worden op een Exception-object is terug te vinden op onderstaande link:
http://php.net/manual/en/language.exceptions.extending.php

2. Exceptions definiëren

Eigen Exceptions kunnen gedefinieerd worden afgeleid van de klasse Exception.

http://php.net/manual/en/language.exceptions.extending.php

In onderstaand voorbeeld wordt een DrawingException gedefinieerd.



src/Drawing/DrawingException.php

```
<?php namespace Drawing;

class DrawingException extends \Exception
{
    public function __construct($message, $code = 0,
                                \Exception $previous = null)
    {
        parent::__construct($message, $code, $previous);
    }
}
```

De exception $previous in de constructor zorgt ervoor dat de exceptions genest kunnen worden. In de constructor van de onderstaande klasse Point wordt de exception opgeworpen vanuit setX opgevangen en opnieuw opgeworpen als deel van een DrawingException. Bij het uitvoeren app.php wordt dus de informatie van zowel de exception opgeworpen uit setX als de exception opgeworpen uit de constructor getoond.

src/Drawing/Point.php

```
<?php namespace Drawing;

class Point
{
    private $x;
    private $y;
    const MAX_XY = 100;

    public function __construct($x, $y)
    {
        try {
            $this->setX($x);
        }catch (DrawingException $drawingException){
            throw new DrawingException("$x out of bounds".
               " in __construct",0,$drawingException);
        }
        try {
            $this->setY($y);
        }catch (DrawingException $drawingException){
            throw new DrawingException("$y out of bounds".
               " in __construct",0,$drawingException);
        }
    }

    public function setX($x)
    {
        if($x<0 || $x>self::MAX_XY ){
            throw new DrawingException("$x out of bounds".
               " in setX");
        }
        $this->x=$x;
    }

    public function setY($y)
    {
        if($y<0 || $y>self::MAX_XY ){
            throw new DrawingException("$y out of bounds".
              " in setY ");
        }
        $this->y=$y;
    }

    function __toString()
    {
        return "($this->x, $this->y)";
    }
}
```

app.php

```php
<?php
require_once 'src/autoload.php';

use Drawing\Point;
use Drawing\DrawingException;
try{
    $point=new Point(111,1);
}catch (Exception $exception){
    print( $exception);
}
```

```
/usr/bin/php /home/jan/PhpstormProjects/example/app.php
Drawing\DrawingException: 111 out of bounds in setX in /home/jan/PhpstormProjects/example/src/Drawing/Point.php:25
Stack trace:
#0 /home/jan/PhpstormProjects/example/src/Drawing/Point.php(12): Drawing\Point->setX(111)
#1 /home/jan/PhpstormProjects/example/app.php(7): Drawing\Point->__construct(111, 1)
#2 {main}

Next Drawing\DrawingException: 111 out of bounds in __construct in /home/jan/PhpstormProjects/example/src/Drawing/Point.php:14
Stack trace:
#0 /home/jan/PhpstormProjects/example/app.php(7): Drawing\Point->__construct(111, 1)
#1 {main}
Process finished with exit code 0
```

3. Errors

Errors kunnen niet behandeld worden in een try catch-block.

Bijvoorbeeld, de functie fopen genereert een error van type E_WARNING bij een niet bestaand bestand.

http://php.net/manual/en/function.fopen.php

app.php

```
<?php
$f=fopen('ikbestaniet.txt', 'r+');
```

```
/usr/bin/php /home/jan/PhpstormProjects/example/app.php
PHP Warning:  fopen(ikbestaniet.txt): failed to open stream: No such file or directory in /home/jan/PhpstormProjects/example/app.ph
```

Er zijn verschillende niveaus van errors gedefinieerd:

http://php.net/manual/en/errorfunc.constants.php

Afhankelijk van het niveau van de error wordt de verdere executie van het PHP-script al dan niet gestaakt.

In het configuratiebestand php.ini wordt geconfigureerd welke errors getoond moeten worden.

/etc/php/7.0/cli/php.ini

```
…
;   E_ALL & ~E_NOTICE & ~E_STRICT  (Show all errors, except for
notices and coding standards warnings.)
…
error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT
```

Dit gedrag kan overschreven worden via error_reporting
http://php.net/manual/en/function.error-reporting.php

app.php

```
<?php
error_reporting(E_ALL);
print($ikBestaNiet);
```

```
/usr/bin/php /home/jan/PhpstormProjects/example/app.php
PHP Notice:  Undefined variable: ikBestaNiet in /home/jan/PhpstormProjects/example/app.php on line 3
```

Het printen van de error-boodschap kan ook onderdrukt worden via een @-symbool.

app.php

```
<?php
error_reporting(E_ALL);
@ print($ikBestaNiet);
```

```
/usr/bin/php /home/jan/PhpstormProjects/example/app.php

Process finished with exit code 0
```

Via trigger_error kan een error gegenereerd worden.

http://php.net/manual/en/function.trigger-error.php

De onderstaande code genereert een E_USER_ERROR voor een niet bestaand bestand of voor een bestand dat niet leesbaar is. Deze error staakt de executie van app.php

src/Util/File.php

```
<?php namespace Util;

class File
{
    public static function readFile($fileName)
    {
        if (file_exists($fileName) && is_readable($fileName)){
            $file = fopen($fileName,'r');
            $contents = fread($file, filesize($fileName));
            fclose($file);
            return $contents;
        }else{
            trigger_error("$fileName not correct",E_USER_ERROR);
        }

    }
}
```

app.php

```
<?php
require 'src/autoload.php';
error_reporting(E_ALL);
use Util\File;
print(File::readFile('ikbestaniet.txt'));
print('done');
```

Ten slotte kan er ook een errorhandler gespecificeerd worden.

http://php.net/manual/en/function.set-error-handler.php

app.php

```
<?php
require 'src/autoload.php';
set_error_handler('errorHandler');
function errorHandler($errno, $errstr, $file, $line)
{
    print("$errstr\n");
    exit();
}
error_reporting(E_ALL);
use Util\File;
print(File::readFile('ikbestaniet.txt'));
print('done');
```

Er kan zelfs voor gezorgd worden dat errors omgezet worden naar exceptions.

app.php

```
<?php
require 'src/autoload.php';
set_error_handler('errorHandler');
function errorHandler($errno, $errstr, $file, $line)
{
     throw new Exception("$errstr");
}
error_reporting(E_ALL);
use Util\File;
try{
    print(File::readFile('ikbestaniet.txt'));
} catch (Exception $e){
     print($e);
}
print('done');
```