טיפול בשגיאות

המחלקה Exception

תקלות מתוארות באמצעות אובייקטים שנוצרים ממחלקות שיורשות מהמחלקה Exception או אפילו מהמחלקה Exception עצמה. כאשר מתרחשת תקלה האובייקט שנוצר כדי לתאר אותה נזרק (throw). באמצעות כתיבת בלוק try & catch ניתן לתפוס את האובייקט שנוצר לתיאור את התקלה. ניתן לכלול בקוד שלנו מספר catches שונים כדי לתפוס את התקלות השונות מהסוגים השונים או מספר בלוקים try & catch מכוננים אחד בתוך השני (יוסבר בהמשך).

כאשר מתרחשת תקלה ונוצר אובייקט (ממחלקה שיורשת מ-Exception או ממחלקה שיורשת מ-Exception) כדי לתאר אותה ואין קוד try & catch שמטפל בה אז הריצה של התכנית תיעצר.

כאשר יוצרים אובייקט ממחלקה שיורשת מ-Exception מומלץ לדאוג לכך שה-id וה-message של האובייקט החדש יאותחלו בערכים המתאימים. יתר המשתנים שקיימים באובייקט החדש כבר מאותחלים.

הדוגמא הבאה מציגה הגדרה של class חדש אשר יורש מ-Exception.

```
<?php
class RectangleException extends Exception
{
	var $rec_width;
	var $rec_height;
	var $value;
	public function RectangleException($id,$msg,$rec_w,$rec_h,$value)
	{
		$this->code = $id;
		$this->message = $msg;
		$this->rec_width = $rec_w;
		$this->rec_height = $rec_h;
		$this->value = $value;
	}
}
?>
```

הפקודה throw

באמצעות הפקודה throw ניתן לזרוק exception. המשמעות של זריקת exception היא שהאובייקט מסוג Exception שנזרק יכול להיתפס בבלוק try & catch שממוקם בפונקציה שבה ה-exceptino נזרק או בפונקציה אחרת שכוללת קריאה להפעלתה או בפונקציה אחרת שכוללת קריאה להפעלת האחרונה (וכו'). אם ה-exception שנזרק לא נתפס על ידי אף אחת מהפונקציות ששייכות לשרשרת ההפעלה של הקוד שממנו נזרק ה-exception אז התכנית תיקרוס.

דוגמת הקוד הבאה כוללת קטע קוד מתוך הגדרת המחלקה Rectangle ומציגה כיצד נזרק exception במקרה שבו מנסים לקבוע ערך לא חיובי בתור ערכו של width באובייקט מטיפוס Rectangle אשר מייצג מלבן מסויים.

```
class Rectangle
{
	private $width;
	private $height;
	function Rectangle($valW,$valH)
	{
		$this->setHeight($valH);
		$this->setWidth($valW);
	}
	...
	function setWidth($val)
	{
		if($val>0)
			$this->width = $val;
		else
		{
			throw new RectangleException(
				12,
				"width must be positive",
				$this->width,
				$this->height,
				$val);
		}
	}
	...
}
```

בלוק try & catch

את ה-exception (אובייקט מטיפוס המחלקה Exception או מטיפוס מחלקה אחרת שיורשת מ-Exception) ניתן לתפוס באמצעות בלוק try & catch.

המבנה של בלוק try & catch כולל את הבלוק try שבתוכו אנו כותבים את הפקודות שמהפעלתן עלול להיזרק exception, את משפט ה-catch מייד אחריו ואת בלוק ה-catch אשר יכלול את הפקודות שתתבצענה אם משפט ה-catchשמתייחס אליו אכן תפס exception.

```
try
{
	...
	...
	...
}
catch(Exception $e)
{
	...
}
```

המבנה של משפט ה-catch דומה למבנה של כותרת של מתודה. משפט ה-catch כולל הגדרה של פרמטר אחד מטיפוס Exception או מטיפוס מחלקה אחרת שיורשת מ-Exception. כאשר נזרק exception מאחת משורות הקוד שבבלוק ה-try אם הטיפוס המדוייק של ה-exception מתאים לטיפוס של הפרמטר במשפט ה-catch (כלומר, הטיפוסים זהים או שהטיפוס של ה-exception שנזרק יורש מהטיפוס שמצויין במשפט ה-catch) אז משפט ה-catch יצליח לתפוס את ה-exception והפקודות בבלוק ה-catch שמשוייך לאותו משפט catch שהצליח לתפוס את ה-exception יתבצע.

ניתן לרשום מספר משפטי catch עם בלוק catch נפרד לכל אחד מהם. במקרה כזה, משפט ה-catch הראשון שיצליח לתפוס את ה-exception הבלוק שמשוייך לו הוא שיתבצע וריצת התכנית תתעלם משאר משפטי ה-catch שבהמשך. התכנית תמשיך לרוץ משורת הקוד הראשונה שאחרי בלוק ה-catch האחרון.

```
try
{
	...
	...
	...
}
catch(ABException $eA)
{
	...
}
catch(MomaException $eB)
{
	...
}
catch(DongoException $eD)
{
	...
}
```

אם בפונקציה מסויימת נזרק exception באחת משורות הקוד שלה ואין בלוק try & class שתופס אותו אז הריצה תעבור לפונקציה שממנה הופעלה הפונקציה המסויימת. אם גם בה אין בלוק try & catch אשר מצליח לתפוס את ה-exception שנזרק אז הריצה תעבור לפונקציה הקודמת לה... וכך הלאה. אם באף אחת מהפונקציות אין בלוק try & catch שמצליח לתפוס את ה-exception אז התכנית תקרוס.

```
<?php
class RectangleException extends Exception
{
	var $rec_width;
	var $rec_height;
	var $value;
	public function RectangleException($id,$msg,$rec_w,$rec_h,$value)
	{
		$this->code = $id;
		$this->message = $msg;
		$this->rec_width = $rec_w;
		$this->rec_height = $rec_h;
```

```
            $this->value = $value;
      }
}

class Rectangle
{
      private $width;
      private $height;
      function Rectangle($valW,$valH)
      {
            $this->setHeight($valH);
            $this->setWidth($valW);
      }
      function area()
      {
            return $this->width * $this->height;
      }
      function setWidth($val)
      {
            if($val>0)
            {
                  $this->width = $val;
            }
            else
            {
                  throw new RectangleException(
                        12,
                        "width must be positive",
                        $this->width,
                        $this->height,
                        $val);
            }
      }
      function setHeight($val)
      {
            if($val>0)
            {
```

```
                $this->height = $val;
            }
            else
            {
                throw new RectangleException(
                    12,
                    "height must be positive",
                    $this->width,
                    $this->height,
                    $val);
            }
        }
}

try
{
    $ob = new Rectangle(-10,20);
}
catch(Exception $e)
{
    echo $e->getMessage();
}
?>
```

דוגמא זו ממחישה את האופן שבו פועל מנגנון הטיפול ב-exception. כיוון ששימוש במנגנון הטיפול ב-exceptions כרוך בפגיעה מסויימת בביצועים רצוי להימנע ממנו ובייחוד כשבבדיקה פשוטה אפשר להימנע מניסיון לייצור אובייקט Rectangle שה-width שלו איננו חיובי.

הפונקציה set_exception_handler

באמצעות פונקציה זו ניתן לקבוע פונקציה מסויימת אשר תופעל כל פעם שנזרק exception שאיננו מטופל באמצעות try & catch.
הפונקציה שקובעים בתור זו שתופעל צריכה להיות עם פרמטר אחד מטיפוס Exception.

לדוגמא, בהנחה שהגדרנו את הפונקציה myGeneralHandler באופן הבא:

```
function myGeneralHandler($e)
{
      ...
}
```

נוכל לכתוב את הקוד הבא:

```
set_exception_handler("myGeneralHandler");
```

כל פעם שייזרק exception אשר לא יטופל על ידי אף בלוק try & catch תופעל הפונקציה set_exception_handler וה-exception שנזרק יישלח אליה.

הדוגמא הבאה מציגה את אופן פעולתה של פונקציה זו.

```
<?php
function generalExceptionHandler($e)
{
      echo "<BR><B>General Error Message</B></BR>";
}
set_exception_handler("generalExceptionHandler");
echo "<BR>before...</BR>";
if(true) throw new Exception("MokoBoko Exception La La La");
echo "<BR>after...</BR>";
?>
```

הפלט שמתקבל:

```
before...
General Error Message
```